Exploring the IC3 Algorithm to Improve the Siemens-Swansea Ladder Logic Verification Tool

Harry Bryant

Submitted to Swansea University in fulfilment
of the requirements for the Degree of Masters of Research

Swansea University
Prifysgol Abertawe

Department of Computer Science
Swansea University
January 2023

Improving the Verification of Railway Interlockings

Declaration

This work has not previously been accepted in substance for any degree and is not being concurrently submitted in candidature for any degree.

Signed: ..... (candidate)

Date: 14/01/2023 .....

Statement 1

This thesis is the result of my own investigations, except where otherwise stated. Other sources are acknowledged by footnotes giving explicit references. A bibliography is appended.

Signed: ..... (candidate)

Date: 14/01/2023 .....

Statement 2

I hereby give consent for my thesis, if accepted, to be available for electronic sharing.

Signed: ..... (candidate)

Date: 14/01/2023 .....

Statement 3

The University's ethical procedures have been followed and, where appropriate, that ethical approval has been granted.

Signed: ..... (candidate)

Date: 14/01/2023 .....

Abstract

Programming logic controllers (PLCs) [16] are widely used to control processes, as can be found in home appliances such as dishwashers and washing machines or in industrial applications where they control components in a production line or railway interlockings [10]. PLCs are often used to control safety critical systems. For instance, malfunctioning dishwashers and washing machines can flood kitchens and homes, malfunctioning robotic arm may lead to human injury or damage to the product, a malfunction of an interlocking computer could end in trains colliding. Therefore, there is a practical demand to verify PLCs as safe, more specifically, that their programs will remain in a set states which one considers to be safe.

In the context of railway control systems, Siemens Mobility has been working on verifying the control programs of interlocking computers [51]. These programs are written in Ladder Logic, one of the three specialised programming languages for PLCs that are introduced in the IEC standard 61131 [21]. Siemens Mobility has been working alongside the Swansea Railway Verification Group verifying Ladder Logic programs for these interlocking computers and developed the Ladder Logic Verifier [87, 51] based on the Inductive Verification and Bounded Model Checking techniques. The issue with Inductive Verification and, therefore, the Ladder Logic Verifier, is that it suffers from possibly returning False Positives [51] due to a method-inherent over-approximation of the state space. This leads to an ambiguity should the verification process return that a safety property is not fulfilled: this can either be due to a False Positive or due to a mistake in the program to be analysed. Experience in verification practice suggests that this is the case for 30% to 40% out of around 240 groups of Abstract Safety Properties to be checked by Siemens.

An Invariant can hopefully be used to prevent these over-approximation False Positives and allow the verification to pass, although they are labour intensive to find manually. This can be solved by applying the IC3 Algorithm [43, 44, 34]. In its current form the algorithm has been successful for the test bed of small Ladder Logic programs. However, it was proven to not scale well when tested on a Siemens industrial interlocking. This thesis presents developments in IC3 implementations that improve its efficiency so it can discover the Invariants and allow the industrial railway interlockings to pass under the Ladder Logic Verifier.

Acknowledgements

Firstly, I wish to thank Professor Markus Roggenbach for his guidance, patience, and support throughout the project that made it possible. I am very grateful for all that he has done and the opportunity given to me. I have enjoyed continuing to work with him on this project we started in July 2021 as an Undergraduate project. I also would like to thank his close friend Erwin R. Catesbeiana (Jr) for keeping us both on track. I would also like to thank my examiners Prof. Helen Treharne and Dr Phillip James. Their valuable time and comments have helped improve this thesis.

Secondly, I want to thank Siemens for co-creating, funding and supporting this project. As well as their time, the resources they have provided, and the opportunity to study a topic I have a great interest in.

Thank you to my family and close friends for their support during my time at Swansea University. I would especially like to thank Philip Bryant for helping with the proof reading of this thesis, and Kira Pugh, Sam Oliver, Victor Cai and Sadeer Beden for their encouragement during the project.

Finally, I would like to thank my second supervisor Dr Oliver Kullmann. I must also thank Liam O'Reilly for helping with the structure of the Aiger-fier, and Dr Xiuyi Fan who provided access to his machine for testing the Ladder Logic Verifier the implementations.

Table of Contents

CHAPTER 1

Introduction

Contents

Among its many products, Siemens Mobility offers railway interlockings. These are specialised computers, which are adapted to rail nodes written using bespoke software. In terms of computer science, railway interlockings are Program Logic Controllers (PLCs) [16]. Siemens writes their software for them in the programming language Ladder Logic [66]. Ladder Logic is one of the three specialised programming languages for PLCs introduced in the IEC standard 61131 [21]. These programs are often used to control safety critical systems, such as railway control systems, and also in Nuclear Power stations [93], where failures or malfunctions simply cannot occur because of the threat of serious injury or death, and the risk of damage to equipment, property and the environment. However, PLCs are also used in everyday items such as washing machines [53] or dishwashers [57]. While it may not be fatal if these appliances malfunction, they can still flood kitchens and homes. In all cases, there is a demand to ensure these PLC programs are safe and do not cause harm.

Siemens and the Swansea Rail Verification Group have developed the Ladder Logic Verifier [87, 51] which takes in a Ladder Logic PLC interlocking and verifies it for safety. Every interlocking developed has to adhere to a large set of properties that shall guarantee the program is safe in operation. It is an open research problem, how to formulate a complete set of properties guaranteeing safety. The Ladder Logic Verifier is an automated

tool that supports and optimises the testing process of interlocking software. It shall reduce the costs involved whilst increasing the quality. Currently, at Siemens the programming and testing dominates the time scale. By shortening the process overheads are reduced. The testing process takes about 25% of the cost of software development in interlocking projects. Overall, this is approximately 2000 hours of work in total, with approximately 500 hours for testing. Currently, Siemens takes less contracts than are on offer, because it cannot handle more volume due to a shortage of highly qualified experts required for testing. The hope is the Ladder Logic Verifier can reduce testing time from 500 to 100 hours. This is because standard testing by the test engineers needs only performed once. Namely after the interlocking has been verified to be correct through the Siemens' Ladder Logic Verifier.

James et al. write in [65] about the status of the Siemens Ladder Logic Verifier:

> As documented in our REF 2021 impact case study [83], there is now a fully functioning Ladder Logic Verifier running at Siemens Mobility. It has roughly 350 safety principles implemented, taken from various standards and developed from Siemens test objectives. This 2^{nd} prototype is fully integrated into the Siemens Mobility ecosystem of interlocking development tools and has been demonstrated in Siemens Mobility operational environment through the verification of about 10 interlockings with up to 12,000 rungs and 75,000 variables. Safety checking of one interlocking takes about two hours. The verification approach has uncovered mistakes in interlockings that Siemens Mobility deems non-detectable by testing.

In Siemens' Ladder Logic Verifier, the verification of Ladder Logic interlockings is performed using Inductive Verification [64] and Bounded Model Checking [55, 30, 54, 66]. However, the issue with Inductive Verification is that it can suffer from over-approximations of the state space, leading to a False Positive [22, 25] which causes the verification to fail. A False Positive is where a tool reports an error where there is none. When the formal tool detects the counter-example that caused the failure, which was due to missing constraints from the surrounding environment, and not because of design error, we refer to such a situation as a False Positive. In these Ladder Logic Programs there are states that are marked as unsafe and must be avoided for the program to be verified as safe. This is because the programs function on Boolean logic, and the extra unreachable states must be accounted for - some of these unreachable states could potentially be unsafe. A False Positive occurs when an unreachable safe state that can transition to an unsafe state is reached - in Inductive Verification all safe states are included no matter if there are reachable or unreachable. This results in an over-approximation as Inductive Verification deems that a safe state can reach an unsafe state no matter that it is unreachable. To resolve this issue, an Invariant, which acts as an extra pre-condition remaining always true, can hopefully be used. An example Ladder Logic Program can be used to demonstrate the process (Figure 1.1) where the Safety Property is $x \vee y$. When Inductive Verification is applied it fails because the safe

but unreachable state $x \wedge \neg y$ can transition to an unsafe state. To solve this, an Invariant can be built that excludes the state $x \wedge \neg y$ meaning the verification passes. However, finding Invariants manually is labour intensive because the state space is 2^n making it impossible for industrial interlockings.

Siemens currently applies Bounded Model Checking [55, 30, 54, 66] to those Ladder Logic Programs that fail Inductive Verification for a given Safety Property P. Bounded Model Checking, like Inductive Verification, utilises a SAT Solver [19] to determine if a counterexample trace is present. However, it is bounded so it limits how many transitions are considered when performing the verification process. This means that unless all states of the system are reached within this bounded number, the system may still be unsafe and the counterexample goes undiscovered. This is an example of a False Negative [22, 25] where a tool fails to report an error when there is one. If a validation tool is inaccurate, then it could suffer from False Positives or False Negatives. It should be said that Inductive Verification by comparison has False Negatives "built in".

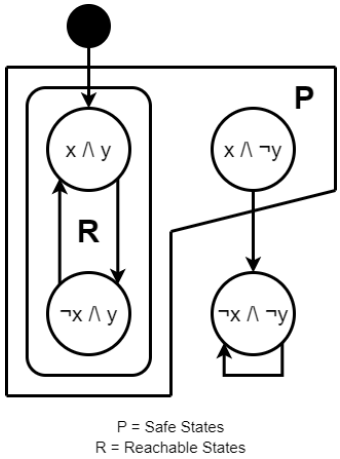


Figure 1.1: Automaton for a Two-Variable Ladder Logic Program

Theoretically, this issue of the undiscovered counterexamples can be solved using the IC3 (Incremental Construction of Inductive Clauses for Indubitable Correctness) algorithm [43, 44, 34] which produces a formula that will prevent the over-approximation False Positive problem in Inductive Verification when performed on an interlocking. The IC3 algorithm, developed by Aaron R. Bradley, takes a set of formulas that represent an automaton and produces a formula that prevents False Positives [22, 25]. It achieves this by allowing more states to hold under the Invariant formula in an iterative process. Starting with the Initial States and growing to include only the reachable safe states. A Java implementation was previously developed as an Undergraduate at Swansea University [48] and tested, and successfully produced Invariants that allowed the verification to hold for many small Ladder Logic Programs. The final implementation was also tested on a Siemens artifact to test its scale-ability, where it was proven to not scale well. This was due to the large number of states

involved in industrial-sized interlockings, and the amount of counterexamples that therefore must be removed. This research, as a Master of Research student, builds on the previous work in the hope to improve upon the IC3 algorithm, and its implementations, to fully aid the Ladder Logic Verifier's efficiency by applying SAT-Heuristics (such as branching), parallelism, and comparing existing implementations of the algorithm. Because of this, IC3 needs to be validated to ensure it works correctly and to determine whether it too suffers from False Positives and False Negatives. This project has been developed in collaboration with Siemens and co-created with its railway engineers.

The IC3 algorithm was chosen because it can find Invariants for sequential hardware circuits using an incremental method and since its introduction has had various adaptations for software model checking. The approach uses induction to form an Invariant for Inductive Strengthening [64] - making it ideal for use with Inductive Verification. Ladder Logic uses Propositional logic [81] with Boolean expressions and does not contain commands such as if-then-else statements and while-loops, making it closer to hardware than software. Alternatives to IC3 such as variations of Bounded Model Checking [55, 30, 54, 66] lack completeness because, for efficiency, the number of bounds used are too low for large programs. k-induction [91] solves this incompleteness, however, it too cannot find a convenient strengthening. Instead it builds a strengthening based on a characteristic of the transition system. Interpolation [71] extends k-induction, unrolling from a current formula containing all states, at most i steps from an initial state. The size of the unrolling can be smaller than with k-induction, and IC3 uses a similar principle to construct Invariants. Counterexample-Guided Abstraction Refinement (CEGAR) [61] is another widely used approach. Like IC3, CEGAR is iterative, constructing and refining abstractions that over-approximate the original system to determine if any errors occur. However, any abstract counterexamples found must be checked for reachability - if so, then the program is unsafe. Otherwise, the counterexample is excluded in the next iteration by building a finer abstraction.

1.1 Railway Verification

As railways and trains have become more advanced over time, so have the systems that control the networks. This has led to the rail industry becoming more and more computer controlled in order to increase safety and efficiency. As always with a safety critical system there is a practical need to verify the hardware and software is safe especially with the reduction of human input. Siemens Mobility has been developing modern-day signalling systems - a malfunction or an error in these systems could lead to an incorrect path or signal being set that may result in a derailment or crash. This could lead to serious injuries or deaths, along with damages to trains and infrastructure. These systems utilise PLCs [16] to create the interlockings that control the railway's infrastructure.

1.1.1 Interlockings

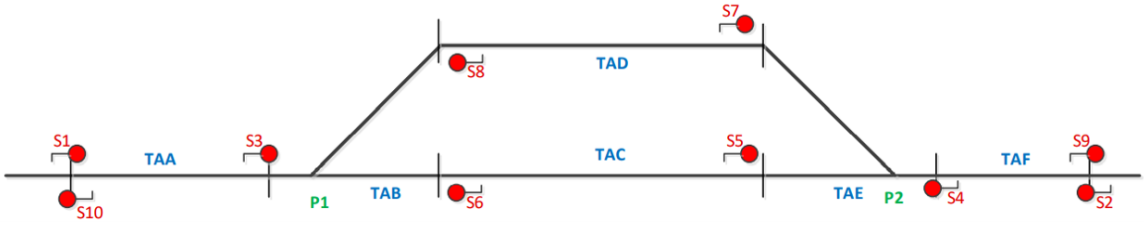


Figure 1.2: An Example Railway Interlocking by Siemens [51]

Railway interlockings (Figure 1.2) have come a long way over time. When railways first began, they were mechanically operated with levers to set the points and, originally, policemen were employed to communicate to train drivers about the status of the track ahead by using hand signals. As the speeds of the trains increased this was replaced by physical signals. However, the policemen had no contact with each other and would allow the next train along a section of track after a set period of time. The obvious flaw in this is there was no way of confirming whether a train had actually cleared the section. With the invention of the electronic telegraph, signalman in signal boxes (Figure 1.3) could communicate with the next box along the track to determine if the line was clear.

Figure 1.3: A mechanical signal box

As time went on the semaphore signals were replaced with colour light aspect signals. The mechanical levers were replaced by buttons and switches but it still required human interaction to control the trains. The signal boxes were replaced by signalling centres that control large sections of the railway remotely. However, there has been a desire to automatise these signalling systems. By doing this, it firstly removes any human error that could exist as a result of either setting the wrong path in the mechanical versions or overwriting a warning in the relay interlockings. Removing the human element also allows the system to work faster as there are no human inputs required. These interlockings are formed of Programmable Logic Controllers [16]. These are small computers that work with binary inputs and outputs, combined to form a larger network with timers to control a process. Siemens Mobility has developed a Ladder Logic [66] based interlocking "WESTRACE" [51]

[23] (Figure 1.4). This is a railway signalling system that has been deployed trackside across parts of the UK network. The goal with all the developments in railway control systems is to allow the trains to run faster and more frequently. While also ensuring the safety of the trains by adding extra levels of protection or by removing human control.

Figure 1.4: Westrace Trackside System developed by Siemens [23]

1.2 Verification Task

Currently, Siemens has the Ladder Logic Verifier [87, 51] used to verify their railway interlockings [10] for safety. In reality, though, there are many Safety Properties for a single Ladder Logic Program. Each Verification Task consists of the Ladder Logic [66] interlocking and a Boolean Propositional formula as a Safety Property P which determines what in the Ladder Logic Program's state space is considered to be safe. Discussions with a Siemens engineer revealed that there are approximately 240 Abstract Safety Properties, adhering to the design principles [82], written in First Order Logic which each interlocking must pass to be deemed safe. Each of these Abstract Safety Properties results in approximately 1-30 Verification Tasks. Therefore, a typical Ladder Logic Program will have to pass around 7,000 properties to be deemed safe. In its current form, the Verifier firstly performs Inductive Verification [64] which determines if the Verification Task is safe initially and also that a safe state transitions to another safe state. If this is the case, we say that the Verification Task is of Type A. If Inductive Verification is unsuccessful, then Bounded Model Checking is performed. Here, if a counterexample trace is found within the maximum bound, and therefore is proven to be unsafe, then the Verification Task is of Type B. However, if no counterexample trace is found within the maximum bound, then the Verification Task is of Type C and is deemed to be safe but only up until k transitions. A diagram of this approach can be seen below in Figure 1.5 which includes how IC3 can be included to classify those previously examples of Type C that were unable to be classified by the original Ladder Logic Verifier. The diagram has the three processes in this order because the runtimes of IC3 were unknown and potentially costly. For testing purposes though, all types of Verification Tasks will be tested with IC3 to form a comparison with the Ladder Logic Verifier to determine the correctness of the classifications. Those Verification Tasks classified as Types A and B should receive the same classification from IC3 to increase the trust in the implementations.

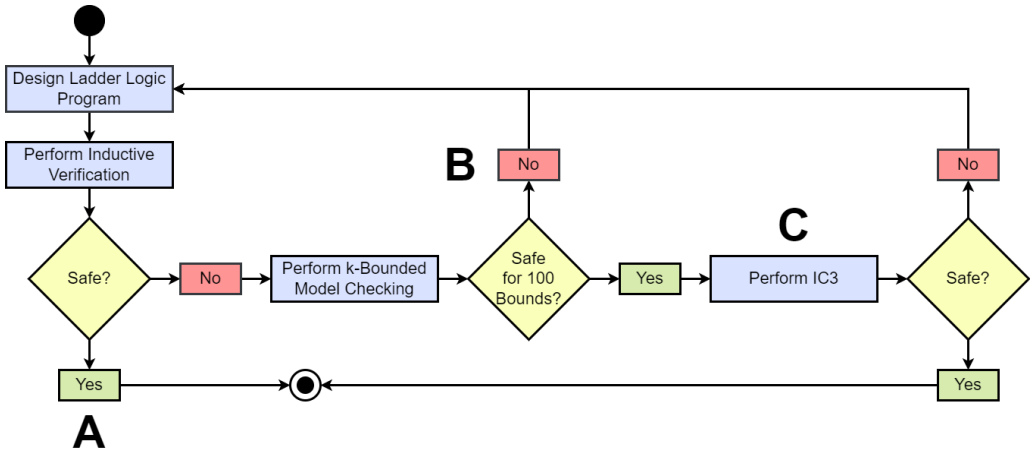


Figure 1.5: Proposed Verification Approach with IC3 to be performed by the Ladder Logic Verifier to resolve the current Decision Procedure

With the original Ladder Logic Verifier Inductive Verification would be performed first to determine which Verification Tasks are safe. Bounded Model Checking will then be run on those that previously fail, finding those that are unsafe. The issue is though that those that fall under Type C are effectively undecidable with this approach because Bounded Model Checking only checks up until a certain number of bounds which means that the error may have not been discovered. This is due to the incredibly large state space and the vast amount of transitions that are possible in order to control the railways. These undecidable tasks are then analysed by the engineers during the testing stage. However, if a fault is found then the interlocking will require redesigning and the testing process to be restarted. The idea was that IC3 [43, 44, 34] can decide these Verification Tasks by constructing an Invariant for those that failed Inductive Verification due to a False Positive [22, 25], and returning false, and potentially a counterexample, for when the Ladder Logic Program is not suitable for the Propositional Safety Property P. This way there are no undecidable tasks and it is less likely for testing to discover an error after verification has been completed.

We received three interlockings from Siemens in order to test the IC3 algorithm and how successful it is for Ladder Logic verification. These were classified by running Inductive Verification and then Bounded Model Checking with the bound set to 100 as suggested by Siemens (Figure 7.5). By running Bounded Model Checking with an increased number of bounds, it could result in more of Type B and less of Type C. Bounded Model Checking is computationally expensive which is why a relatively low bound is chosen. On average, those of Type A take a second, those of Type B take 55 seconds to be decided for 100 bounds, and Type C taking on average 130 seconds. The times for running Siemens' Ladder Logic Verifier come from Clausegen implementation provided. Each Abstract Safety Property has 1-30 tasks and assuming that they are 50% of Type A and 50% Type C, then that is an

average waiting time of around 30 seconds per task for the rail engineer. There are 240 of these properties, assuming that each has 30 properties with 50% Type A and 50% Type C it would result in a wait time of approximately 55 hours for each Ladder Logic Program. The idea is that IC3 cannot only efficiently decide those of Type C to reduce the waiting time for the engineers.

The motivation for this research is that there are enough Verification Tasks that are of Type C. While this percentage can potentially be reduced by applying Bounded Model Checking with an increased number of bounds, it adds to the wait time for the rail engineers. Crucially, Bounded Model Checking can only say for certain that a Verification Task is safe if the bound is large enough to include all the transitions which is not feasible for the industrial-sized interlockings with thousands of properties to test. This is because the runtimes become too great.

1.3 Aims and Contribution

The overall aim of this thesis is to show and explore the hypothesis that the IC3 algorithm [43, 44, 34] can help with Ladder Logic verification. The main result of this thesis is a well-tested approach that can solve the current flaw in the Ladder Logic Verifier [87, 51] by providing a method to efficiently decide the previously undecidable Verification Tasks when Bounded Model Checking was performed, having previously failed Inductive Verification due to False Positives [22, 25]. It builds on the work previously completed by the Siemens and the Swansea Rail Verification Group [67, 66, 92, 50, 64]. As stated previously, it was intended that IC3 would be run after Inductive Verification and Bounded Model Checking because the runtimes of IC3 were unknown. In reality IC3 could replace parts of the existing methods because the IC3 implementations are so much quicker by comparison. IC3 could in theory be run before Bounded Model Checking. For example, IC3 could be run first, and then Bounded Model Checking on those Verification Tasks that failed in order to obtain the counterexample - as this is computationally expensive. Both Inductive Verification and Bounded Model Checking utilise SAT-Sovlers [19] with Propositional formulas as does IC3. There are the limitations to both of these verification techniques and, because of this, there is a need for an extra form of verification. In order to utilise the IC3 implementations, the Ladder Logic provided by Siemens requires a translation similar to the Tseitin Transformation [73]. This ensures that the Propositional formulas [81, 76] for the program's transitions T and Safety Property P, defining the program's safe states, are in the AIGER format [38] primarily used for And-Inverter Graphs [12].

There were two IC3 implementations tested in this research - both will be compared to the existing Ladder Logic Verifier. The first, IC3ref [42], is from the algorithm's creator Aaron R. Bradley. The second, ABC [74, 75], is a refinement of the algorithm by Mishchenko. It should be said that ABC is significantly faster than IC3ref and would be used by Siemens going forward if successful. Therefore, this thesis could have been presented as a comparison of the applicability of the IC3ref and ABC implementations to railway in-

terlockings. Instead we stuck with the original question of whether IC3 would be suitable as the runtimes were unknown. We opted to give a historical account of the research because it seemed to better represent how insights have been acquired (Figure 1.6). With the research questions changing because of these insights. As ABC is significantly faster than IC3ref, a new toolchain in the Ladder Logic Verifier would built on ABC.

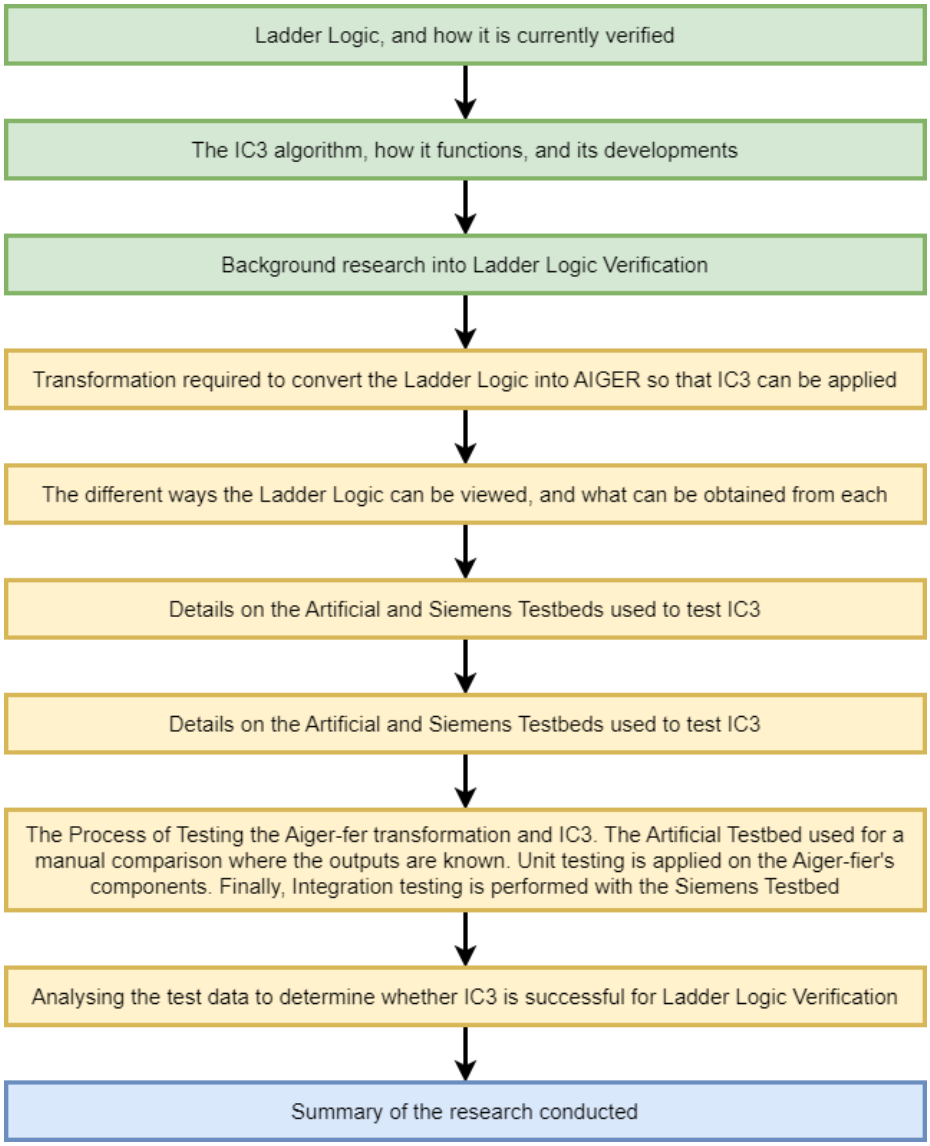


Figure 1.6: The workflow of this Thesis showing the Chapters build on each other in order to determine how suitable IC3 is for use in Ladder Ladder Logic Verification. The chapters of the Background are in green, with the Contribution in yellow and the Summary in blue

The later chapters of this thesis provide the theoretical framework behind the algorithm and its format and illustrate how it functions to form the Invariants for use in Inductive Verification. The work that has been completed can be split into the following main hypotheses:

1.3.1 Hypothesis 1: IC3 Algorithm can Efficiently Decide Type C

The most important result of this thesis is that the IC3 algorithm [43, 44, 34] can determine whether a Verification Task of Type C is safe or not. We explore the IC3 algorithm, developed by Aaron R. Bradley, which has been proven to be successful for hardware verification. The goal is for IC3 to step in for all examples that Bounded Model Checking is unable to find a counterexample trace for, after k bounds, and determine whether they are safe or not. This will be achieved either by producing a Ladder Logic Invariant via an Inductive Strengthening [64], or by returning that the Ladder Logic Program is unsuitable in its current form to remain safe.

1.3.2 Hypothesis 2: IC3 Algorithm runs "Fast" on Types A and B, "Slow" on Type C

It is important that the runtimes are measured and analysed for each Type of Verification Task in order to gain an understanding of how IC3 holds up against other verification techniques and what causes it to be computational expensive. It is expected for the types of tasks involved, that the average runtimes for IC3 are as follows:

- **Type A:** Cheap in runtime
- **Type B:** Cheap in runtime
- **Type C:** Expensive in runtime

Type A are those Verification Tasks that passed Inductive Verification which is cheap as it only requires two calls to a SAT-Solver. IC3 should match this in runtime because there is no False Positive over-approximation present. Therefore, the Propositional Safety Property will not need amending and IC3 should exit after a single iteration. Those tasks of Type B contain a counterexample trace and have already failed Inductive Verification but, unlike Bounded Model Checking, it is expected to be cheap for IC3. This is because IC3 performs checks early in the algorithm and at regular intervals so if the Ladder Logic Program is not suitable for the Propositional Safety Property, it will exit early without wasting resources. Although, this is dependent on the the Invariant property constructed discovering the error quickly. Finally, there are those of Type C. These either are of Type B and false will be returned relatively quickly. Or there could be a False Positive present in which case the algorithm could take many iterations to formulate an Invariant to ensure safety. Therefore, the runtime for these tasks would be expensive.

1.3.3 Hypothesis 3: IC3 Performs Better than the Ladder Logic Verifier in Terms of Runtime

The final main hypothesis for this thesis was to determine whether IC3 could actually replace the existing components of the Ladder Logic Verifier [87, 51]. For this to be the case the results from running the implementations must match both Inductive Verification and Bounded Model Checking in terms of correctness and runtime. All Verification Tasks that are deemed safe by Inductive Verification should also be safe under IC3 for correctness and in comparable time as Inductive Verification is relatively cheap as discussed. Again, for correctness, those tasks that fall into Type B should be deemed unsafe by IC3. However, the structure of the algorithm results in an inexpensive runtime compared to Bounded Model Checking. This is because the iterative process expands the Ladder Logic Invariant to include the states, rather than checking the result of each transition. IC3 can do this in a relatively small number of iterations to cover all the reachable states. By comparison Bounded Model Checking requires a enough bounds to cover all transitions.

1.4 Chapter Overview

The remaining chapters of this thesis are outlined as follows:

Chapter 2: Propositional Logic:

Chapter 3 focuses the process of developing and verifying Ladder Logic Programs. Starting with Propositional Logic, which is the basis of Ladder Logic using Boolean formulas, explaining the modelling approach and the syntax of Propositional Logic languages. The chapter then defines Ladder Logic's structure, its components and how the language can be used to construct Boolean formulas from Propositional Logic for Programmable Logic Controllers (PLCs). The premise of Safety Properties is introduced, used to ensure that the Ladder Logic Programs stay within a set of states considered to be safe. The chapter also explains the current process of verifying a Verification Task, consisting of a Ladder Logic Program and a Safety Property, is explained. The Ladder Logic Verifier currently performs Inductive Verification to determine if the property holds, however, it can suffer from False Positives where an unreachable state reaches an unsafe state. In Inductive Verification both the reachable and unreachable states which is why False Positives occur. If the condition does not hold under Inductive Verification then Bounded Model Checking is performed to see if a counterexample trace can be produced explaining how an unsafe state can be reached. If no trace can be produced then the program is said to be safe for k transitions. Finally, the chapter explains why there is a need to way to decide these Verification Tasks that are deemed safe for k transitions as a mistake may be present but just not yet discovered. To determine whether there was a mistake or they are indeed safe the IC3 algorithm can be used. This produces Invariants which can be used to block False Positives.

Chapter 3: IC3 Algorithm:

Chapter 3 explains how the IC3 functions in each phase of the algorithm and how the process of an Inductive Strengthening builds Ladder Logic Invariants in IC3 that can be used in Inductive Verification. This chapter also includes a run through of the algorithm on a small artificial Ladder Logic Program to demonstrate the stages involved in constructing a Ladder Logic Invariant. The file format AIGER, used by both implementations of IC3 that were tested, is introduced. The format is primarily used for And-Inverter graphs. The AIGER components and how the files can be constructed to represent logical statements are explained in this chapter. Also included is how the Safety Properties can be represented in AIGER in order to prove the unsafe states cannot be reached.

Chapter 4: Literature Review:

In Chapter 2, we review some existing work on the topic of Ladder Logic Verification, including how our work will extend and improve the process. This will also touch on the work in previous projects that have helped develop the Ladder Logic Verifier.

Chapter 5: Ladder Logic to AIGER Transformation:

We discuss the process of converting the Ladder Logic Programs, written in the TPTP (Thousands of Problems for Theorem Provers) format [8]. This allows the two IC3 implementations can be run. The chapter starts by describing how the Aiger-fier was refined to use a variation of the Tseitin Transformation rather than following it more closely and generating a new Latch for every variable in the Ladder Logic Program. The chapter also explains how the Aiger-fier applies De Morgan's laws to remove disjunctions from the Propositional formulas and how it recursively creates further conjunctions to leave the logical "Ands" with two components. It does this because AIGER does not accept disjunctions, and any conjunctions can only have two components. Next, the chapter explains how Input variables are dealt with. These are defined in AIGER to be 0 initially and then take the value of the Input for all further iterations as in the TPTP format. This is achieved using an Input and a Latch. All of this is contained inside the Aiger-fier. There is also the Safety Property Parser which converts the properties into a format similar to TPTP and converts any implications ready for the overall transformation with the Aiger-fier.

Chapter 6: How Different Effects are Visible in Different Formats:

Chapter 6 explains how a Ladder Logic Program looks across the different formats seen in this thesis. This firstly starts as Propositional formulas, before showing how it can be represented graphically as an automaton - which is useful in small examples for visualising counterexamples and Invariants. The next stage is to construct an And-Inverter graph from the Ladder Logic. Finally converting the gates used in the graph into AIGER components for the representations to be used by IC3.

Chapters 7: Ladder Logic Testbeds:

Chapter 7 details the two testbeds used in this thesis. The first is a set of artificial example Ladder Logic Programs that were small enough to be visualised as an automaton. Therefore the process of constructing a Ladder Logic Invariant could be performed by hand forming a comparison for the IC3 implementations. The second tested consists of railway interlockings provided by Siemens. These contain thousands of variables and are therefore too large to be visualised as an automaton or have a Ladder Logic Invariant found manually. Instead these are classified using the Ladder Logic Verifier into those Verification Tasks that pass Inductive Verification (Type A), those where Bounded Model Checking finds a counterexample trace after 100 bounds (Type B) or those where Bounded Model Checking was unable to find a counterexample trace after 100 bounds (Type C). This produces another comparison with IC3, as it should in theory be successful for those Verification Tasks of Type A but fail for Type B as it should be impossible to construct an Invariant.

Chapter 8: Correctness of the Aiger-fier:

In Chapter 8 the Aiger-fier is tested for correctness. It is assumed that because both the Ladder Logic Verifier and the IC3 implementations have been used elsewhere, there is more likely to be an error (if there is one) in the Aiger-fier which whilst has been tested has not be used in industry. In order to check the correctness of the Aiger-fier three checks have been conducted. The first uses the artificial testbed to establish whether IC3 can produce a Ladder Logic Invariant for a Verification Task where the Ladder Logic Program and Safety Property are known. Therefore, we know for a given property that an Invariant should be able to be produced and IC3 should match this. This also means that the Verification Task can be manipulated to force IC3 to fail. The second check is Unit testing of the stages in the transformation performed by the Aiger-fier. Finally, Integration testing takes place with the Siemens tested to see if IC3 can produce Invariants for those Verification Tasks of Type A but fail for those of Type B. If successful, these checks would therefore increase the confidence in how IC3 decides Type C.

Chapter 9: Hypotheses for IC3 Ladder Logic Verification:

Chapter 9 presents the results of the three hypotheses we had for IC3 Ladder Logic Verification. The first is that IC3 can efficiently decide those Verification Tasks of Type C, this is the case with a definative classification being quickly formed. The second checks to see if the algorithm will be slower when constructing a Ladder Logic Invariant for Type C than A and B because it will have to potentially spend more time iterating through to remove all the counterexamples. In reality though this was not the case as those Verification Tasks of Type A took on average the longest. The final hypothesis is to see whether IC3 could replace Inductive Verification or Bounded Model Checking because if it is shown to be quicker at classifying the Siemens testbed

- whilst being correct. It was found that both implementations of IC3 tested were quicker than than Inductive Verification and Bounded Model Checking.

Chapter 10: Summary and Future Work:

Finally, the last chapter reviews the work conducted in this Master of Research project and the success of using IC3 for Ladder Logic Verification. This leads onto potential areas that the algorithm could be improved either for efficiency, improving feedback to engineers about errors or printing the Ladder Logic Invariants allowing for Inductive Verification to be re-run and check that any False Positives are blocked. Currently, we know that if IC3 is successful then a Ladder Logic Invariant is produced that would prevent the False Positive. By adding the ability to print the Ladder Logic Invariant, it would provide extra reassurance.

Part I

Background

CHAPTER 2

Ladder Logic Verification

Contents

The railway interlockings by Siemens Mobility are developed in the language Ladder Logic [66] which uses Propositional Logic [81, 76] as its basis. These interlockings are a safety critical system in real-world operation where a failure or malfunction may result in serious injury or death, the loss or damage to equipment and property, or environmental harm. Because of this, there needs to be an assurance in the control systems that a crash is not possible. While certification still relies on testing, it is time consuming and therefore costly. Currently, Siemens use their Ladder Logic Verifier to determine whether the interlockings meet the required Safety Properties. This formal verification aims to confirm their accuracy and reliability, which provides trust in the safety critical system. However, there is a weakness in the current verification approach applied by Siemens (Figure 2.1) as it is unable to effectively decide all properties. Siemens would therefore like to improve their correctness assurances by amending the verification approach - with the hope that the IC3 algorithm [43, 44] will ensure all properties can be decided. The verification process is completed before the testing, because the testing process takes several weeks. Therefore, it is important for the verification results to be accurate to increase the trust in the interlockings so that they can be deployed on the railway network.

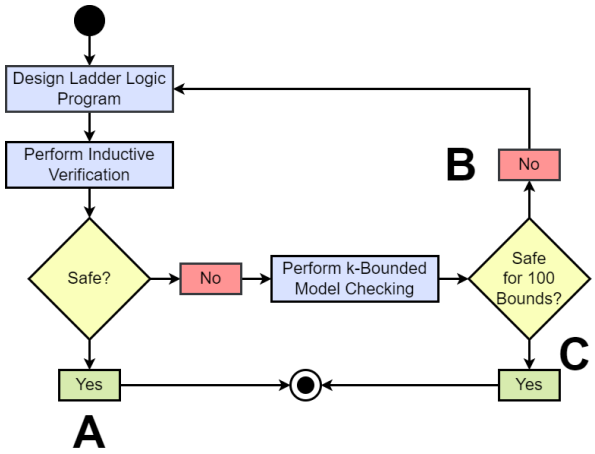


Figure 2.1: Existing Verification Approach performed by the Ladder Logic Verifier

Throughout this chapter, a simple test example of a railway interlocking will be used to demonstrate how the Ladder Logic is built from Propositional Logic and is then verified. The track plan chosen for this is the Dungeness Loop (Figure 2.2) on the Romney, Hythe and Dymchurch Railway. This example is a unique one to Siemens as there are not many cases where a train enters and exits without changing direction. The example has been abstracted for simplicity - more variables increase the state space exponentially. This model sees the trains enter from the left on the track segment $T0$, entering the loop on to track segment $T1$ after signal $S0$ gives permission, and exiting the loop back on to $T0$ when signal $S1$ is green. It should also be made clear that the model has been modified in order to expose the False Positive over-approximation problem that occurs [22].

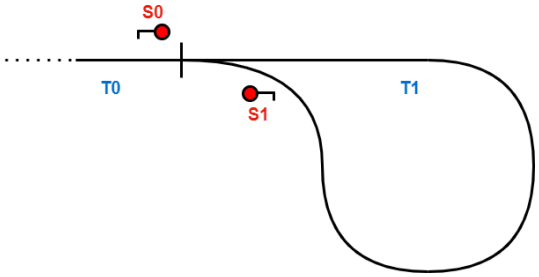


Figure 2.2: The Track plan of the Dungeness Loop

2.1 Propositional Logic

The basis for Ladder Logic [66] is Propositional Logic [81, 76], one of the oldest languages for formalised reasoning, that can be used to describe any circumstance using its logical statements. It is said that the world can be considered as the set of all true propositions

about it. For example, that the "sky is blue", or "there is no rain". Both completely random but these statements are true propositions about a given topic. These simple propositions can be combined by Boolean operators, or junctors, to form more complex statements: "if the sky is blue, then there is no rain". In software engineering, Propositional Logic can be used to define the behaviour of the software by using formulas to construct arguments that must be followed. Therefore allowing for the description of not only the actual world as it is, but also artifacts yet to be constructed.

2.1.1 Syntax and Semantics of Propositional Logic

The syntax and semantics of Propositional Logic and its languages have been defined in this section following section 2.2.2 of Formal Methods for Software Engineering Languages, Methods, Application Domains written by Markus Roggenbach et al. [84]:

Definition 2.1 (Syntax of propositional logic) Assume we are given a finite set of proposition symbols P. We define the following set of formulae:

- Every $p \in P$ is a formula
- $\bot$ is a formula
- If φ and ψ are formulae, then $(\varphi \Rightarrow \psi)$ is a formula

It can be the case that the set of proposition symbols may be empty. When this is the case the only formulae which can be built according to the definition are of the form $\bot$, $(\bot \Rightarrow \bot)$, $((\bot \Rightarrow \bot) \Rightarrow \bot)$, $(\bot \Rightarrow (\bot \Rightarrow \bot))$ and so on. This syntax is considered "minimalistic" because it only contains the constant $\bot$ and the junctor $\Rightarrow$. This allows easy proofs by induction [64, 86] on the structure of a formula. Usually, some more propositional junctors (or Boolean operators) are used to construct more complex formulas. Often it is the case that different symbols can be used to represent these junctions in different formalisms. For example, CASL [78], often used to represent Ladder Logic [66] allowing proofs to be performed, uses "false" and "true" instead of $\bot$ and $\top$ respectively. Implication is often written as $(\varphi \Rightarrow \psi)$, or $(\varphi \succ \psi)$. For negation $(\neg)$, the symbols ! and - are used, with CASL using "not". For disjunction $(\vee)$ various representations are common including "or", —, +, \/. With conjunction $(\wedge)$ representations can be "and", &, *, /\.

The syntax requires parenthesis around binary junctors in order parse formulas and maintain their meaning. The order of precedence for the operators $\neg < \wedge < \oplus < \vee < \Rightarrow < \Leftrightarrow$, this declares binary junctors to be left-associative and omits parentheses whenever appropriate - with $\Rightarrow$ being implication and $\Leftrightarrow$ being equivalence. In order to give a semantics (a meaning) to propositional formulae, each proposition symbol describes some statement which may or may not hold in the world defining the logic. A system or a set of states can therefore be modelled by setting certain propositions to true and false under certain scenarios.

Definition 2.2 (Propositional model) A Propositional model M is a function $P \Rightarrow \{true, false\}$ assigning a truth value to every proposition symbol.

If P contains n different proposition symbols, then there are 2^n different models. Given a model M, the truth assignment can be extended to arbitrary formulae. The validation relation $\models$ between a model M and a formula φ is defined by the following clauses:

- $M \models p$ if and only if $M(p) = true$
- $M \not\models \bot$
- $M \models (\varphi \Rightarrow \psi)$ if and only if $M \models \varphi$ implies $M \models \psi$.

That is, $M \models (\varphi \Rightarrow \psi)$ if and only if $M \not\models \varphi$ or $M \models \psi$. If $M \models \varphi$, then we say that M satisfies φ, or, equivalently, φ is valid in M. The model checking problem is defined as: Given a model M and a formula φ, determine whether $M \models \varphi$ holds. For propositional logic, this problem can be solved with a complexity which is linear compared to the length of the formula. The access time of the truth value of a proposition symbol in a model tends to be either constant, or again linear - this is dependent on the representation of the model. Model checking for propositional formulae involves parsing the formula according to its syntax, looking up the values of the proposition symbols in the model, and then assigning the truth value to compound formulae according to the above clauses [84, 29, 58, 70].

Definition 2.3 (Satisfiability, Validity) Given any formula φ, φ is satisfiable if there exists a model M such that $M \models \varphi$.

A formula φ which is valid in every model then it is universally valid, also known as logically true or a tautology, denoted by $\models \varphi$. Each universally valid formula is therefore satisfiable. It can also be said that if a formula φ is unsatisfiable, then $\neg\varphi$ must be universally valid. The satisfiability problem (SAT problem) [94] is to find out whether any given formula is satisfiable. SAT Solvers [19] are used to determine if there exists a model M such that $M \models \varphi$, and is why to prove a tautology with a SAT Solver the negation is used to prove that this is instead unsatisfiable. Similarly, there is the validity problem which determines whether a given formula is universally valid, however, any algorithm which solves the satisfiability problem also is a solution to the validity problem. In order to determine whether $\models \varphi$ holds, it is sufficient to find out whether $\neg\varphi$ is satisfiable or not. This is because if $\neg\varphi$ is satisfiable, then there is a model M such that $M \models \neg\varphi$, therefore, $M \not\models \varphi$ and φ cannot be universally valid. However, if $\neg\varphi$ is deemed unsatisfiable by the SAT Solver, then for all models M it is the case that $M \not\models \neg\varphi$, therefore, for all M we have $M \models \varphi$ and φ is universally valid. In summary, $\neg\varphi$ is satisfiable if and only if φ is not universally valid. Satisfiability can be generalised and relativised to sets of formulae: Let Γ be any set of formulae and φ be a formula. We say that Γ is satisfiable, if there is a model M satisfying all $\psi \in \Gamma$. Similarly, φ follows from Γ (written as $\Gamma \models \varphi$), if φ holds in every model M which satisfies all $\psi \in \Gamma$. Similar to above, it holds that $\Gamma \models \varphi$ if and only if $\Gamma \cup \{\neg\varphi\}$ is unsatisfiable [84, 29, 58, 70].

2.1.2 Rules of the Dungeness Loop written in Propositional Logic

There are four variables in the Dungeness Loop model (Figure 2.2): the signals $S0$ and $S1$, and the track segments $T0$ and $T1$. The rules of the model are:

- $S0 \Leftrightarrow (\neg S1 \wedge T0)$
  - The signal permitting entrance to the loop can only become green when the signal to exit ($S1$) is red, and a train is in the segment $T0$
- $S1 \Leftrightarrow (\neg S0 \wedge T1)$
  - The signal permitting exiting the loop can only become green when the signal to enter ($S0$) is red, and a train is in the segment $T1$
- $T0 \Leftrightarrow ((T0 \wedge \neg S0) \vee (T1 \wedge S1) \vee (\neg S0 \wedge \neg S1))$
  - The track segment outside of the loop will be occupied by a train if one of the following three conditions are met:
    - $T0$ already having a train and the signal $S0$ being red
    - $T1$ having a train and $S1$ being green, permitting the train to exit the loop into $T0$
    - Both signals being red - as there are no trains
- $T1 \Leftrightarrow ((T0 \wedge S0) \vee (T1 \wedge \neg S1))$
  - The track segment for the loop will be occupied by a train if one of the conditions are met:
    - $T0$ already having a train and the signal $S0$ being green, allowing the train to enter the loop into $T1$
    - $T1$ having a train and $S1$ being red

For this interlocking example, the Safety Property has been defined as $\neg(S0 \wedge S1)$, therefore, both signals should not be green at the same time.

2.2 Ladder Logic

Ladder Logic [66] is one of the three specialised programming languages for Programmable Logic Controllers (PLCs) [16, 41] introduced in the IEC standard 61131 [21]. It operates with, and is a subset of, Propositional Logic [76, 81] with the transitions equivalent to Boolean expressions. Siemens Mobility has been using Ladder Logic to develop its PLC interlockings. Ladder Logic is a graphical-based language made up of horizontal bars known as Rungs, giving it a ladder-like appearance. A PLC [16] is a loop that runs infinitely after

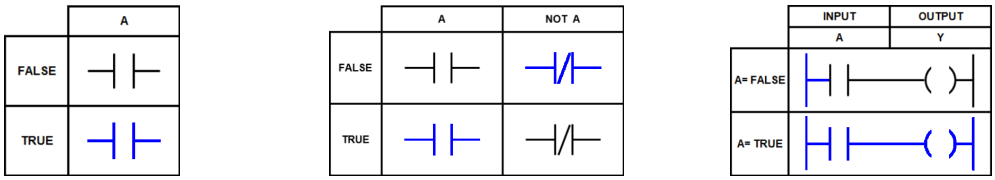
the system state of the Ladder Logic Program has been initialised. From this the PLC computes its next state State' which also includes some Output' values. This is achieved by taking the current values of the Boolean variables and the set of transitions T. These transitions T are rules that each determine when a Boolean variable is satisfied. After the new values are computed, the PLC writes Output' and updates the states in the system [59, 66, 85]. The value of a state takes the value of state' ready for the next transition. These primed states (') represent the values after a transition is made, which is why in the formal checks when T is present the right-hand side of the implication must be primed. This process runs iteratively, each time applying the current values of the variables to the rules defined by the transitions T to compute values for the primed variables that can be used in the next iteration. T is a Transition formula that allows the Ladder Logic Programs to function as a Transition System.

2.2.1 Ladder Logic Components

Ladder Logic [66] operates with Propositional formulas [76, 81] made up of variables containing contacts that represent 1 or 0 - or open and closed as they are known as by engineers. When a Ladder Logic Program is shown as a graphical diagram (Figure 2.5) it consists of two rails which are drawn as vertical lines running down the edges of the page. In a relay logic circuit [31] they represent the active and zero volt connections of the power supply where the power flow goes from the left-hand side to the right-hand side. Then there are the Rungs, which are the horizontal lines and connect the rails to the logic expressions. In the relay logic circuit they would represent the wires that connect the power supply to the switching and relay components. The rails and rungs are what give Ladder Logic its name.

External inputs can be included in Ladder Logic, there are control actions such as a push button being pressed. The outputs of the Ladder Logic are external devices that are being turned on and off, such as an electric motor. In terms of the railway interlockings, an input could be a track sensor which is activated when a train passes. This detection could then activate a level crossing so that the lights flash and the barriers are lowered making it safe for the train to pass. Inputs and outputs can be used to construct logical expressions as Open Contacts (Figure 2.3a) or Closed Contacts (Figure 2.3b). Each Rung represents a Boolean logical expression. These consist of a combination of inputs and outputs to formulate the desired control operations. Possible components on a rung can be:

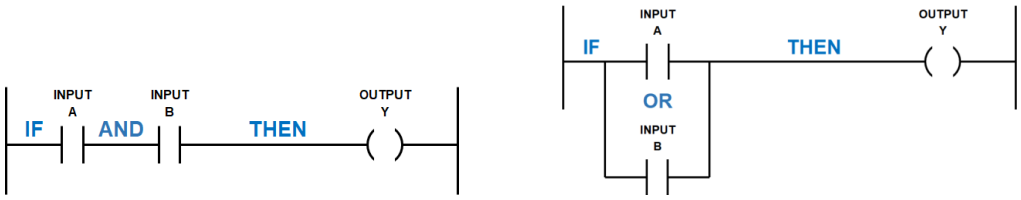
- **Coils:** Represent output values determined by the variables occurring previously as the coil must sit at the end of the rung (Figure 2.3c). The value of a coil can be used as either an output or internally in the next iteration.
- **Open Contacts:** Represent the value of an input or output variable (Figure 2.3a).
- **Closed Contacts:** Represent the negated value of an input or output variable (Figure 2.3b).



(a) Normally Open Contact (b) Normally Closed Contact (c) Coil

Figure 2.3: Main constructs of Ladder Logic [24]. The blue components in the diagrams are True, whilst the black components are False and do not help satisfy the coil

The language also has the ability to introduce timers to allow for repetition or to force the program to wait. For example, a timer could be used in the level crossing example to make the barriers lower after a few seconds of the lights flashing. The shape of each rung determines the value of a coil. Because Ladder Logic is a binary-based language, logical formulas can be built in order to produce more complex programs for the interlockings. Two contacts in series produce a conjunction (Figure 2.4a), while two contacts in parallel form a disjunction (Figure 2.4b). Ladder Logic [66] can be represented in Propositional Logic [76, 81], where the rules and arguments of the Ladder Logic are written as Propositional Formulas to reason with a statement. This produces a concrete and indisputable argument dictating exactly when a variable is considered true. There are many different operators that make up the Propositional Logic but, for Ladder Logic [66] the focus is only on four in particular: Equivalence, Implication, Conjunction, Disjunction and Negation [76, 81].



(a) Conjunction in Ladder Logic where Contacts are in series and both must be True in order to satisfy the coil

(b) Disjunction in Ladder Logic where Contacts are in parallel and at least one must be True in order to satisfy the coil

Figure 2.4: Conjunction and Disjunction in Ladder Logic [24]

Definition: A ladder logic formula ψ follows Propositional logic [76, 81], consisting of a finite set of input variables I, for user input, and a finite set of state variables C, the names of the system's states, with $I \cap C = \emptyset$:

$$\psi \equiv (c_1' \Leftrightarrow \psi_1) \wedge (c_2' \Leftrightarrow \psi_2) \wedge \cdots \wedge (c_n' \Leftrightarrow \psi_n))$$

where $n \geq 0$ and the ψ_i, $1 \geq i \geq n$, are Propositional Formulae, such that:

1. For all $1 \geq i \geq n : c'_i \in C'$.
2. For all $1 \geq i, j \geq n$: if $i \neq j$ then $c'_i \neq c'_j$.
3. For all $1 \geq i \geq n : vars(\psi_i) \subseteq I \cup \{c'_1, \cdots c'_{i-1}\} \cup \{c_i, \cdots c_n\}$.

The set of new states that the program moves into once making a transition is c'. These states are primed and are the values of variables after a transition. For example, if the variable x is set to the value of itself it can be written as $x' \Leftrightarrow x$ where the primed value gets the orginal value of x. After the transformation, the value of x takes the value of x' ready for the next transition in the iterative process. These conditions on the variables allow for $\Leftrightarrow$ in Ladder Logic to be read as a variable assignment in an imperative program. This definition was taken from Section 4.3.2 of Formal Methods for Software Engineering Languages, Methods, Application Domains [84].

2.2.2 Dungeness Loop expressed in Ladder Logic

The Dungeness Loop interlocking (Figure 2.2) can be written in Ladder Logic [66] (Figure 2.5). The four variables $S0$, $S1$, $T0$, $T1$ become Coils and each equivalence formula is converted using contacts in parallel for disjunctions and in series for conjunctions.

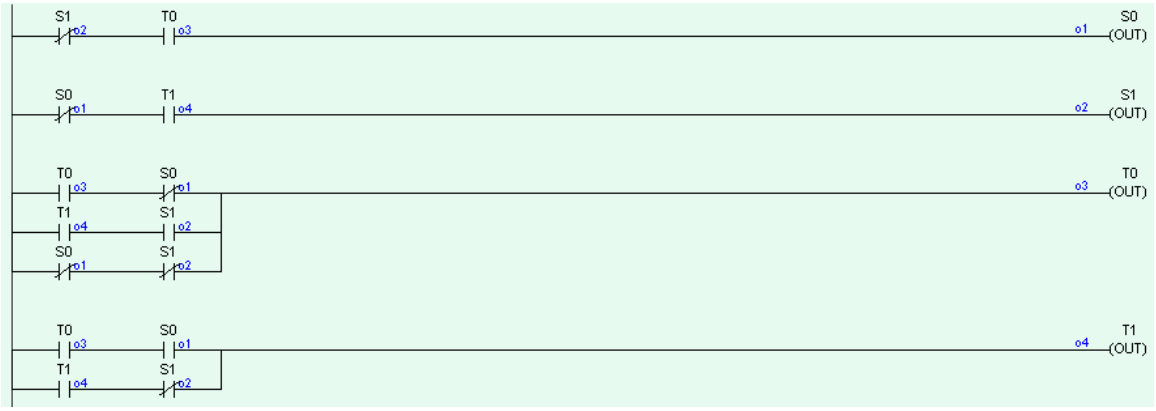
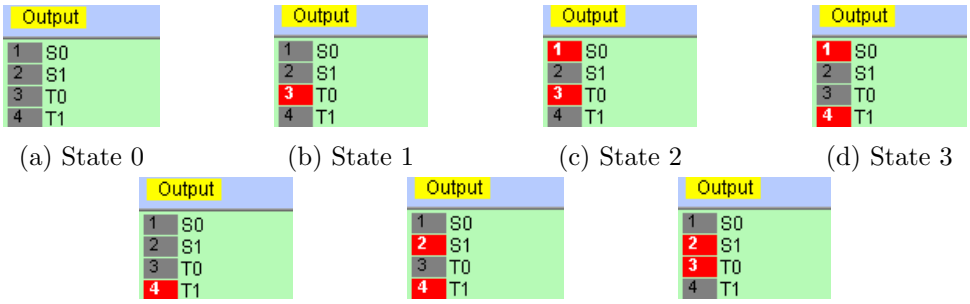


Figure 2.5: The Dungeness Loop interlocking written in Ladder Logic

At the start of the project, the task was to understand Ladder Logic and develop small programs. To do this the i-TRiLOGI software [14] was used to develop a number of example programs like the Dungeness Loop example - note that this program has no inputs. This program starts in a state where all four variables are false (Figure 2.6a) as there are no trains in the system. From here, the program transitions to a state where a train enters the interlocking from the left and is in track segment $T0$ (Figure 2.6b). Next, signal $S0$ turns green (Figure 2.6c) and the train can move from $T0$ into track segment $T1$ (Figure 2.6d). The signal $S0$ turns red (Figure 2.6e), which allows signal $S1$ to turn green (Figure 2.6f).

The train then moves from $T1$ into $T0$ (Figure 2.6g) and finally the program moves back into a state where a train is in $T0$ and both signals are red (Figure 2.6a). From this, the program can repeat the process iteratively. This process can be visualised as an automaton in Figure (Figure 2.7).



(a) State 0 (b) State 1 (c) State 2 (d) State 3

(e) State 4 (f) State 5 (g) State 6

Figure 2.6: States reached when running the Simulator on the Dungeness Loop Ladder Logic Example

2.2.3 Safety Properties for Ladder Logic Programs

In every program, there are certain states that must be avoided so nothing bad happens in the program's environment. These states can be considered unsafe. For example, in a railway interlocking, an unsafe state would be one where the signals allow two trains to travel towards each other on the same section of track. These unsafe states can be formalised by a Safety Property - a logical formula. For the examples developed, the Safety Property can be chosen as it is artificial - in the case of the Dungeness example it is $\neg(S0 \wedge S1)$ so that both signals cannot be green at once and allow two trains over the set of points. In real-world examples this would be dictated, for example, a railway point must be locked prior to a train passing over it.

All Ladder Logic programs [66] can be visualised with an automaton. These allow its states and transitions to be examined. Each automaton is a finite-state machine, accepting a string of symbols by following the transitions between the states. The two variable Ladder Logic example seen in Figure 1.1 shows the transition from the state $x \wedge y$ to the state $\neg x \wedge y$, and then back again. A transition can take place from the state $x \wedge \neg y$ to the state $\neg x \wedge \neg y$. Finally a loop exists from the state $\neg x \wedge \neg y$ to itself. The states $x \wedge \neg y$ and $\neg x \wedge \neg y$ have no transitions to them from the Initial State so are deemed to be unreachable. In the example there is one state $\neg x \wedge \neg y$ that is unsafe. This must not be reached in order for the program itself to be considered safe. In the Dungeness example (Figure 2.2), there are 4 states where both $S0$ and $S1$ are true. Whilst all 4 transition to safe states, no reachable safe state can transition to an unsafe state (Figure 2.6) meaning that the program will be safe in operation. The real-world interlockings are designed with this principle but obviously with an exponentially greater number of states. In the automatons, a small black circle with an

arrow pointing to a state depicts an Initial State I, in the Dungeness example it is the state where all variables are false. The number of states in a Ladder Logic Program is equal to 2^n. In reality, the Siemens interlockings have thousands of Safety Properties that must be proved to guarantee the safety of the railway.

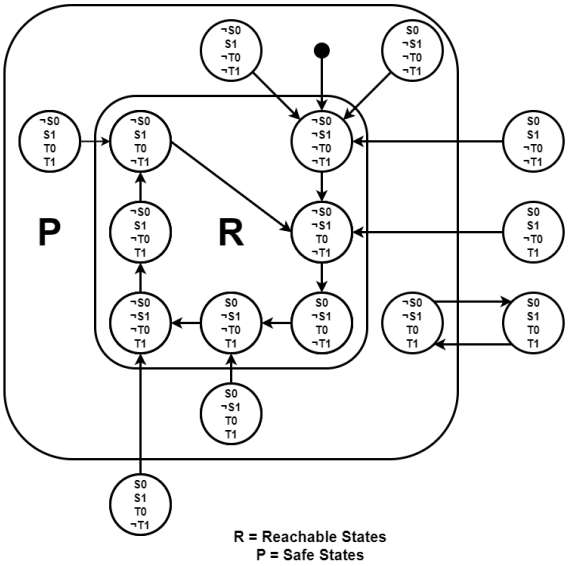


Figure 2.7: Automaton for the Dungeness Loop Ladder Logic Program

2.3 Approach to Ladder Logic Verification

The Ladder Logic interlockings need to be formally verified to determine if they meet the required safety conditions. Currently, Siemens use their Ladder Logic Verifier [87, 51] (Figure 2.1) which, first, performs Inductive Verification [64, 86] to determine if a Ladder Logic Program meets a safety condition. However, it suffers from False Positives [22] because the process also includes unreachable states. If a Ladder Logic Program fails Inductive Verification, Bounded Model Checking [55, 30, 54, 66] is performed which attempts to trace through the program to find the fault. However, it only does this for a pre-defined number of steps and therefore could potentially not reach the error. Siemens want to improve the accuracy of the Ladder Logic Verifier, and the hope is the IC3 algorithm [43, 44, 34] can find Invariants that will prevent False Positives. In this thesis, two versions of the algorithm will be tested, IC3ref by the algorithm's creator Bradley [42], and a refined version, ABC by Mishchenko [74, 75], to compare their efficiency.

2.3.1 Inductive Verification

Inductive Verification [64] is one of the techniques that can be performed to verify Ladder Logic [66] programs. A program is deemed safe if both of the following checks are met:

1. Show that all Initial States are safe.

2. Show that for all states s: If s is safe, then all successors of s are safe as well.

This approach is is efficient because it only requires the two calls to a theorem prover. Current industry standard is to have quality assurance by testing. Also, the authorities are happy with just tested systems.

2.3.1.1 False Positives

If Inductive Verification [64] is applied to the Dungeness Loop example from before (Figure 2.7), the first condition is met because the Initial State ($\neg S0 \wedge \neg S1 \wedge \neg T0 \wedge \neg T1$) is included in the set of safe states P (Figure 2.8). However, in Inductive Verification all states are considered regardless if they are reachable or not. The second condition states that all safe states must transition to a safe state. But as state ($\neg S0 \wedge \neg S1 \wedge T0 \wedge T1$) is safe and can transition to an unsafe state ($S0 \wedge S1 \wedge T0 \wedge T1$) this fails (Figure 2.8). Whilst the Ladder Logic Program has the property P that holds, it cannot be demonstrated with Inductive Verification because of this unreachable safe state that causes a False Positive.

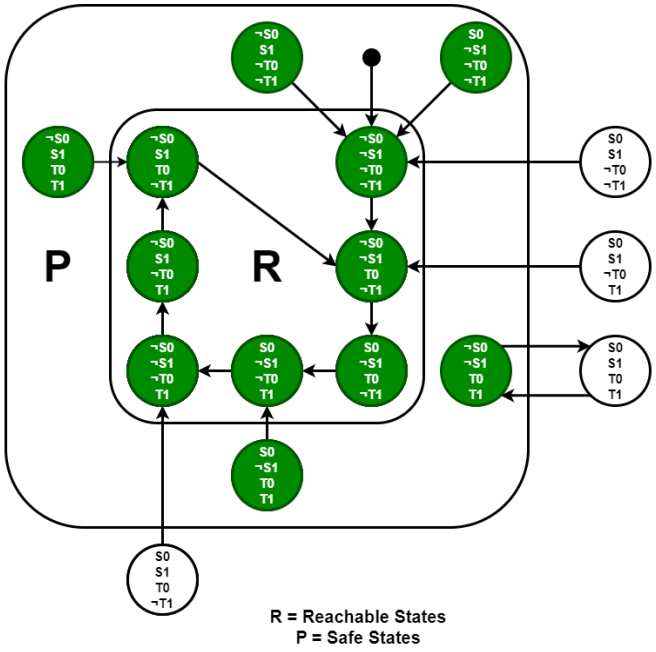


Figure 2.8: Two-Variable Ladder Logic Example's Safe States, marked in green. As there is a Safe State ($\neg S0 \wedge \neg S1 \wedge T0 \wedge T1$) that transitions to an unsafe state the second condition of Inductive Verification will fail - this is a False Positive as this state is unreachable in operation

In reality this program stays in the set of reachable states. Therefore, the transition that breaks Inductive Verification's [64] second condition, from $(\neg S0 \wedge \neg S1 \wedge T0 \wedge T1)$ to $(S0 \wedge S1 \wedge T0 \wedge T1)$, cannot take place in a system run. However, when the verification takes place, it is included creating this over-approximation False Positive [51]. It is important to proof and test that a computer system is not going to malfunction. This is especially important in the safety critical systems such as the railway control systems by Siemens.

2.3.1.2 Inductive Strengthening

A way to combat False Positives [51] is to use a Ladder Logic Invariant, a formula that acts as an extra precondition refining when a transition can take place, helping to Inductively Strengthen the program. The Invariant is therefore a formula that includes at least the reachable states, permitting all the required transitions to take place while still blocking the potential False Positives. With the Invariant, Inductive Verification's second clause becomes:

2. Show that for all states s: If s is safe and the Invariant holds in s, then all successors of s are safe as well.

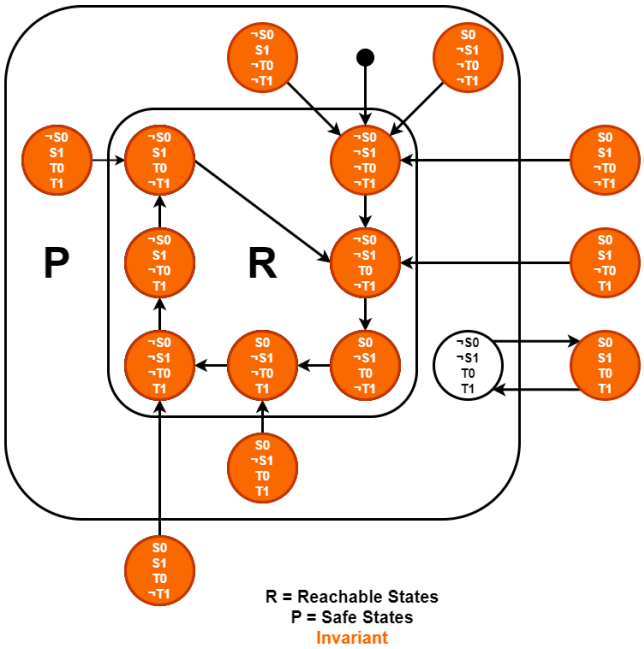


Figure 2.9: States that hold under the Invariant, marked in orange

For the Dungeness Loop example Ladder Logic program (Figure 2.5) (Figure 2.2) the Ladder Logic Invariant needs to block the transition from the state $(\neg S0 \wedge \neg S1 \wedge T0 \wedge T1)$

to the state ($S0 \wedge S1 \wedge T0 \wedge T1$) to prevent the False Positive. Therefore, the Ladder Logic Invariant (Figure 2.9) needs to include the reachable states (defined by the region R) and can also include any states that do not transition to an unsafe sate. Now when the verification is applied a transition can only take place when the state is safe (in the region P) and is one of the orange states (Figure 2.9). This prevents the False Positive and the program is verified to be safe.

A full description of Ladder Logic Invariant and proofs where they help Inductive Verification [64] to pass can be found in the Appendix (B).

2.3.2 Bounded Model Checking

In the Ladder Logic Verifier, Bounded Model Checking [55, 30, 54, 66] is performed on the Verification Tasks that fail Inductive Verification. Its purpose is to discover counterexamples in Ladder Logic Programs that cause them to be unsafe. A bound is placed on the number of transitions that can take place when performing the verification process. Siemens set this value to be 100 because, in practice, the counterexamples are sometimes found within the first 10 transitions. However, this of course only allows for 100 transitions to take place which in theory makes Bounded Model Checking incomplete. This is because only states reachable within this bounded number of transitions are considered, and for larger Ladder Logic Programs there are often more than 100 transitions. Therefore, unless all the reachable states are covered within 100 steps the system may still be unsafe. It can therefore be said that Bounded Model Checking will only guarantee safety up until k steps. Assuming the realistic cycle time of three seconds for an interlocking computer, this translates to a safety guarantee for the first five minutes after the system has been started.

The issue is that Siemens is dealing with Ladder Logic Programs that contain hundreds of variables and ultimately a vast number of transitions. Whilst in practise the Ladder Logic Verifier has shown that the majority of counterexamples have been found early. However, if there was no counterexample found after 100 bounds the program may still be unsafe. In this case in order to have trust in the system either a more bounds must be included, which is time consuming, or a new approach found.

CHAPTER 3

IC3 Algorithm

Contents

The process of manually finding Invariants for Ladder Logic Programs [66] is slow and labour-intensive. While it works for the small examples, for real-world interlockings, a manual process is infeasible. It is impossible to visualise their large state spaces - in contrast to our simple two-variable example (Figure 1.1). Note also, that each property to be verified might require a different Invariant. Instead, an algorithmic approach needs to be taken to automatically discover these Invariant formulas.

The algorithm chosen to achieve this is the IC3 (Incremental Construction of Inductive Clauses for Indubitable Correctness) algorithm [43, 44, 34] developed by Aaron R. Bradley in 2011 which finds Invariants for automata. IC3 was originally invented for the verification of sequential hardware circuits, however it has since had various adaptations for software model checking [30]. Ladder Logic [66] is based of the structure of Propositional Logic [76, 81] meaning it utilises Boolean expressions. Therefore, it does not contain commands such as if-then-else statements and while-loops, making it closer to hardware than software in this sense. This means the IC3 algorithm can be applied directly to Ladder Logic Programs because of the adaptions made to the Software-Model-Checking process.

In order to illustrate how the IC3 algorithm [43, 44, 34] can discover an IC3 Invariant that can be used as a Ladder Logic Invariant through the different phases, an example run through a small artificial Ladder Logic Program is in this section. This example has been chosen because, unlike the two-variable example (Figure 1.1), it requires all phases of the IC3 algorithm, including the counterexample generalisation. This example consists of four variables which allows for a state space which is still viewable as an automaton (Figure 3.1). This allows for a correctness check to be made to ensure that the algorithm includes only the desired states in its Invariant. The Ladder Logic Invariant formula would therefore be $(\neg a \wedge b) \vee (a \wedge \neg b) \vee (\neg a \wedge c \wedge \neg d) \vee (b \wedge c \wedge d)$ which allows all the reachable transitions while preventing the False Positive [51].

For the algorithm to take place, the Initial States I, Transition Rules T, and the Safety Property P must all be a CNF [80].

Initial States I**:** $a \wedge (\neg c \vee b) \wedge (\neg d \vee b) \wedge (\neg b \vee c) \wedge (\neg d \vee c) \wedge (\neg b \vee d) \wedge (\neg c \vee d)$.

Safety Property P**:** $(\neg d \vee c \vee \neg b \vee \neg a) \wedge (b \vee \neg d \vee a \vee \neg c) \wedge (b \vee c \vee a \vee d)$.

Transitions T**:** $(a' \Leftrightarrow (b \wedge c \wedge d) \vee (a \wedge \neg b \wedge \neg c) \vee (a \wedge \neg b \wedge d)) \wedge$
$(b' \Leftrightarrow (\neg a \wedge b) \vee (\neg a \wedge c \wedge \neg d)) \wedge$
$(c' \Leftrightarrow (\neg b \wedge c \wedge \neg d) \vee (\neg a \wedge b \wedge d) \vee (b \wedge c \wedge d) \vee (a \wedge \neg b \wedge \neg d)) \wedge$
$(d' \Leftrightarrow (\neg a \wedge b \wedge \neg c) \vee (\neg a \wedge b \wedge d) \vee (a \wedge c \wedge d))$

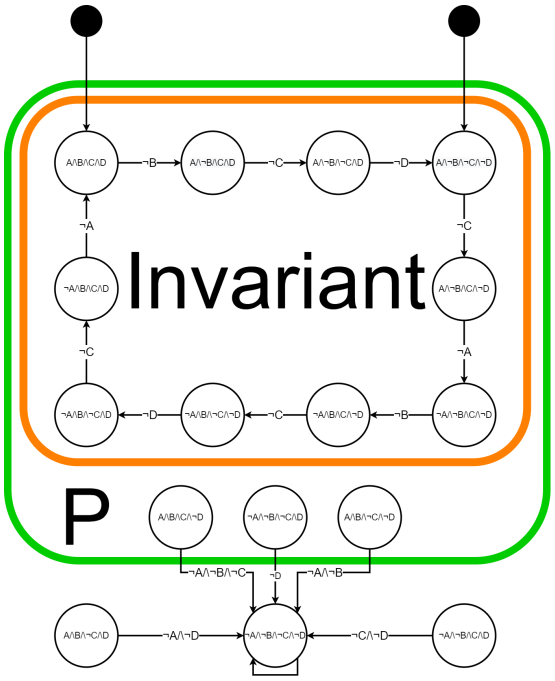


Figure 3.1: Automaton of a small artificial Ladder Logic Program that requires use of all stages of the IC3 algorithm, including the Erroneous Phase

3.1 How the Algorithm Works

The IC3 algorithm [43, 44, 34] is designed to produce a formula with an Inductive Strengthening that enables Ladder Logic Programs [66] to hold under Inductive Verification [64]. That being that the method has prove that the Safety Property is sufficient for the Ladder Logic Programs. Currently Ladder Logic Programs suffer from a problem when Inductive Verification is performed on an interlocking where an over-approximation results in a False Positive [51]. The IC3 algorithm, developed by Aaron R. Bradley, takes a set of formulas that represent an automaton and produces a formula that prevents the over-approximation.

The principle of Bradley's IC3 algorithm was investigated as an Undergraduate student at Swansea University [48]. In this work a Java implementation [46] was developed to construct Ladder Logic Invariants on a testbed of small artificial Ladder Logic Programs. This followed an overview of IC3 given in a presentation by Shoham Ben-David [34] utilising the SAT4j [35] SAT Solver [19]. This proved that IC3 was successful in finding Ladder Logic Invariants efficiently for smaller example programs but when it was tested on a larger industrial-scale example provided by Siemens Mobility, it struggled due to the large state space involved. This was mainly due to the fact a section of the algorithm was not included in the initial implementation which generalises discovered counter examples states to sets of states. Both the Undergraduate and Master of Research projects have been developed in collaboration with Siemens and co-created with their railway engineers. The aim is improve the efficiency of their Ladder Logic Verifier by finding Ladder Logic Invariant formulas automatically by refining the IC3 algorithm for use on an industrial scale.

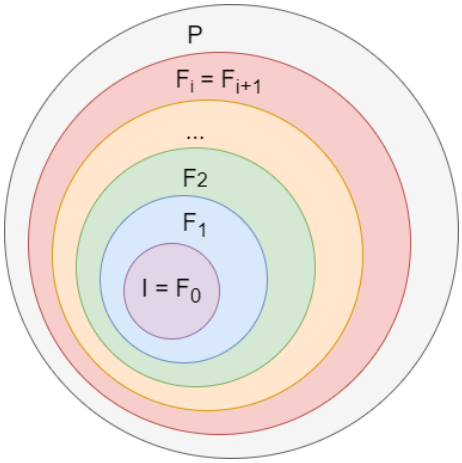


Figure 3.2: "Layers" of Ladder Logic Invariant Candidates $F_0, \ldots, F_i, F_{i+1}$

Ladder Logic Programs [66] can be converted into automata, which is what is needed to produce a Ladder Logic Invariant with the IC3 algorithm [43, 44, 34]. With an automaton, which shows the program's Initial States I and its transitions T, and a Safety Property P, the IC3 algorithm performs an incremental approach that produces a sequence $F_0, F_1, \ldots, F_h$ of

Ladder Logic Invariant Candidates for use to help Inductive Verification's [64] second clause hold and not suffer from False Positives [51]. Each increment, F_i, changes the number of states holding under the formula. The algorithm begins by only allowing transitions from the Initial States and then incrementally allowing more and more transitions. This continues until all intended reachable states are included in the set defined by F_h - where F_h is the outermost layer of the Ladder Logic Invariant Candidates (Figure 3.2). While importantly still ensuring that the program does not reach an unsafe state from a reachable state, or from an unreachable state to prevent the False Positive [22, 25]. A new layer is only built if there are additional safe states to include in a Ladder Logic Invariant Candidate by relaxing it further. The algorithm terminates when either:

- $F_i = F_{i+1}$, then F_i is an Invariant
- A new F_i cannot be produced, then the verification fails

For example, if $h = 2$ then the Ladder Logic Invariant Candidates F_0, F_1, F_2 have been developed with F_3 being equivalent to F_2 hence the termination of the algorithm. A run through of the algorithm can be found in Section 3.5 which illustrates how the layers are constructed and amended.

Input : I, T, P
Output: P holds / P does not hold
Result: An IC3 Invariant formula that with P will ensure a Ladder Logic Program remains in the set of safe states P for use as a Ladder Logic Invariant.

A pseudocode version of the IC3 Algorithm (Algorithm 1) (Algorithm 2) was developed following the presentation given by Shoham Ben-David [34] which explains the algorithm's steps. This includes the advanced generalisation approach that was missing in the implementation developed in the Undergraduate project [48, 46]. The first stage of the algorithm is the Initialisation Phase (Section 3.3) where checks are made to ensure the Initial States are safe and that they transition to another safe state. If this is not the case, then IC3 cannot be applied as there is a problem with the Safety Property P. Assuming these checks pass, the algorithm moves to the Iteration Phase (Section 3.4) where each iteration allows more states to hold under the Invariant formula . Each iteration further relaxes the Invariant formula, permitting more transitions to take place in the Ladder Logic Program.

The Iteration phase has two sub-phases. If the states that currently hold under the Ladder Logic Invariant Candidate formula keep the program in the safe states when a transition is made, it enters the Holding Phase. This where the algorithm continues to allow more states to hold by constructing a new layer (Figure 3.2) with a new Ladder Logic Invariant Candidate. However, if the current Ladder Logic Invariant Candidate formula is unable to keep the program in the safe states, it enters the Erroneous Phase. This check is made using a SAT Solver [19] which, when satisfiable, produces a counterexample.

This counterexample is the state that has an unsafe successor state and, if unreachable, is an example of the over-approximation that is resulting in a False Positive [51]. However, if it is a reachable state then the IC3 algorithm cannot be applied and the Verification Task in question is deemed to be of Type B. Assuming that the counterexample state is unreachable, then the algorithm attempts to form a generalisation so that appending the negative of this generalisation to the current Ladder Logic Invariant Candidates will block multiple counterexamples. If this is not possible, then the negative of this counterexample state is added to the Ladder Logic Invariant Candidates in order to block this unsafe state being reached. The Iteration Phase runs until the new formula is semantically equivalent to the previous. When this occurs, the new formula is the one that should be used as the extra parameter in Inductive Verification [64] to allow its second condition to pass.

3.2 Inductive Strengthening

The principle of the IC3 algorithm [43, 44, 34] involves creating a sequence of Ladder Logic Invariant Candidates $F_0, \ldots, F_h$. The algorithm constructs these Ladder Logic Invariant Candidates with each one building from the last. While maintaining that the Safety Property P of the program is adhered to by the set of states that hold under the Ladder Logic Invariant Candidates and they states they transition to. Any counterexamples states that break the Safety Property are removed from the Ladder Logic Invariant Candidate as part of the Iteration Phase. The goal is to produce an Inductive Strengthening of P, where $F_i = F_{i+1}$. To do this, four properties must be checked for each construction of a new Ladder Logic Invariant Candidate F_h to ensure that it has been correctly built as an IC3 Invariant, where $h \geq 0$:

IC3 Invariant(h):

IC3 Invariant 1. $I \Rightarrow F_0$ holds

- Do the Initial States hold for the initial iteration of the Inductive Assertion?

IC3 Invariant 2. $\forall(0 \leq i \leq h) : F_i \Rightarrow F_{i+1}$

- Are the states that hold under F_i a subset of the states that hold under F_{i+1}?

IC3 Invariant 3. $F_h \Rightarrow P$ holds

- Are the states that make F_h true safe?

IC3 Invariant 4. $\forall(0 \leq i \leq h) : F_i \wedge T \Rightarrow F_{i+1}$'

- If we go a step out of the states that make F_i true with the transitions defined by T, do these states fulfil F_{i+1}'?

Across all four properties, F_i is an inductive assertion describing the set of states inside the Ladder Logic Program. These four properties are in place to ensure that the newly built Ladder Logic Invariants are firstly meeting the Safety Property P and that they derive from the previous F_h as per Induction - with F_h being the latest Ladder Logic Invariant Candidate and the outermost layer of Invariants (Figure 3.2) such that $F_h \subseteq P$. These properties ensure that the Ladder Logic Invariants have been constructed correctly by the IC3 algorithm [43, 44, 34] as IC3 Invariant Candidates. This therefore guarantees that they are suitable for use on Ladder Logic programs as Ladder Logic Invariants when the verification is performed.

3.3 Initialisation Phase

The Initialisation Phase of the IC3 Algorithm (Algorithm 1) has two main purposes. The first is to ensure that the set of Initial States I is safe and that they transition to another safe state using the rules defined in T. This means the Ladder Logic Program is safe for 1-step. The second purpose of the Initialisation Phase is to construct the first two layers of Ladder Logic Invariant Candidates. F_0 which is set to the Initial States I and F_1 which is set to the Propositional Safety Property P conjoined with clauses from I that hold inductively [64, 86].

```
1  Initialisation Phase
2  if (¬ (I ⇒ P)) ∨ (¬ ((I ∧ T) ⇒ P′)) then
3  |  Output: "P does not hold"
4  else
5  |  F_0 := I
6  |  F_1 := P
7  |  C = Clauses(I)\Clauses(F_1)
8  |  for c ∈ C do
9  |  |  if (I ∧ T) ⇒ c′ then
10 |  |  |  F_1 := F_1 ∧ c
11 |  |  end
12 |  |  C := C\{c}
13 |  end
14 |  h := 1
15 end
```

Algorithm 1: Initialisation Phase of IC3 [43, 44, 34]

In the automaton (Figure 3.1) the two Initial States are both included in the set of safe states P, and both transition to a state also in the set. That means both these checks pass and the algorithm can continue. F_0 is set to I and F_1 to P. The next step is to see which of the clauses in the Initial States hold. The clauses are each disjunctive formula connected by a logical And. Therefore the clauses are the set:

$\{a, (\neg c \vee b), (\neg d \vee b), (\neg b \vee c), (\neg d \vee c), (\neg b \vee d), (\neg c \vee d)\}$

Each of these clauses, c, in the set C are checked to see if they hold after making one step with Transitions T from the Initial States (characterised by I), i.e., if $(I \wedge T) \Rightarrow c'$ is true - here c is primed because the check takes the values after a transition is made. If this is the case, then the clause is added to the Safety Property to form the second layer in the sequence of Ladder Logic Invariant Candidates F_1. If it does not hold then it the clause is not conjoined to F_1. When this process is applied to the four-variable example (Figure 3.1), F_1 becomes:

$(\neg d \vee c \vee \neg b \vee \neg a) \wedge (b \vee \neg d \vee a \vee \neg c) \wedge (b \vee c \vee a \vee d) \wedge a \wedge (\neg b \vee c) \wedge (\neg d \vee c) \wedge (\neg b \vee d)$

3.4 Iteration Phase

17 ***Iteration Phase***
18 **repeat**
19 **if** $((F_h \wedge T) \Rightarrow P')$ **then**
20 ***Case 1: Holding Phase***
21 $F_{h+1} = P \wedge \bigwedge_{c \in F_h, F_h \wedge T \Rightarrow c'}$
22 $h := h + 1$
23 **else**
24 ***Case 2: Erroneous Phase***
25 Let s be a counterexample to $F_h \wedge T \Rightarrow P'$
26 $j := -1$
27 Let j be the largest index with $F_j \wedge \neg s \wedge T \Rightarrow \neg s'$
28 **if** $j = -1$ **then**
29 Output: "P does not hold"
30 **else**
31 A Cube q is formed from s by dropping a literal l
32 **if** $\exists q\ ((q := s \setminus l) \wedge \neg(I \Rightarrow q) \wedge (F_j \wedge \neg q \wedge T \Rightarrow \neg q'))$ **then**
33 $\forall 0 \leq i \leq j : F_i := F_i \wedge \neg q$
34 **else**
35 $\forall 0 \leq i \leq j : F_i := F_i \wedge \neg s$
36 **end**
37 $h := j$
38 **end**
39 **end**
40 **until** $clauses(F_i) = clauses(F_{i+1})$ *[Syntactic Equivalence]*;

Algorithm 2: Iteration Phase of IC3 [43, 44, 34]

The Iteration Phase (Algorithm 2) aims to either construct new layers of Ladder Logic Invariant Candidates (Figure 3.2) via the Holding Phase so that after h iterations the final Invariant includes all the reachable transitions whilst ensuring that the Ladder Logic

Program remains safe. It can also amend the existing Ladder Logic Invariant Candidates via the Erroneous Phase if a counterexample is present and restricts the Ladder Logic Invariant Candidates to avoid the over-approximation. The algorithm goes into the Holding Phase when there is not a counterexample present, i.e., $F_h \wedge T \Rightarrow P$' holds. Otherwise the algorithm enters the Erroneous Phase.

In the IC3 pseudocode for the Iteration Phase (Algorithm 2), the first decision is to determine whether the Safety Property P holds when a transition is made from the states of states defined by the current Ladder Logic Invariant Candidate (Figure 3.3). If this is met then the program enters the Holding Phase. This creates a new Ladder Logic Invariant Candidate in the same way the Iteration Phase built the F_1, this is demonstrated in Section 3.4.1.

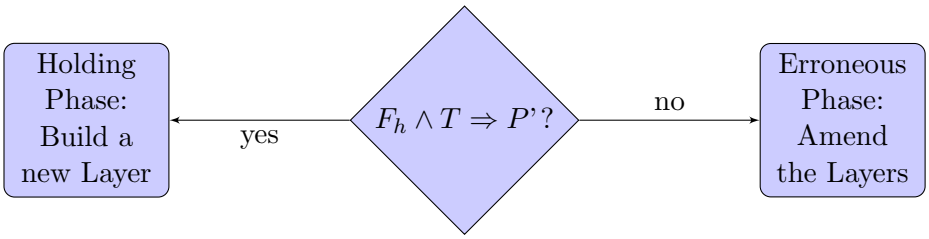


Figure 3.3: The decision procedure for the Iteration Phase. If after a transition from the states in the current Ladder Logic Invariant the program is safe then a new layer is built, otherwise the layers are amended to block a counterexample

Whereas if the check failed the Iteration Phase enters the the Erroneous Phase. The aim of this process is to remove the counterexample state s that has been included in the Ladder Logic Invariant Candidate and transfers to an unsafe state. The first stage is to determine the latest Ladder Logic Invariant Candidates where the negation of s holds. For this the check $F_j \wedge \neg s \wedge T \Rightarrow \neg s$' is used. Starting with F_0 and increasing until either the check fails, or F_h is reached. If this is successful then part of the Invariant can be saved by removing the counterexample state from this point. A Cube can be constructed that includes multiple counterexamples states to remove them all in a single iteration. This Cube is constructed by dropping a literal from s which relaxes the formula to include more states.

To ensure that only counterexample states are removed, the new Cube must also adhere to the following conditions. The first that it is not the case that that the Cube is not implied by the Initial States - therefore, the cube does not contain any of the Initial States. The second condition is that from the states that meet $F_j \wedge \neg q$, when a transition is made $\neg q$ will be preserved - ensuring that the counterexamples will not be reached again. If these conditions are met then the Cube is successful and $\neg q$ is appended to all current Ladder Logic Invariant Candidates. However, if no Cube can be constructed, then the negation of the single counterexample s is appended.

3.4.1 Holding Phase

The Holding Phase builds a new layer (Figure 3.2) such that more states can hold under the final Invariant. It does this in the same way that F_1 was constructed in the Initialisation Phase (Algorithm 1). This appends clauses from the previous Ladder Logic Invariant Candidate F_h that hold when a transition is made from F_h. This builds the new layer and the value of h (the number of layers) is incremented.

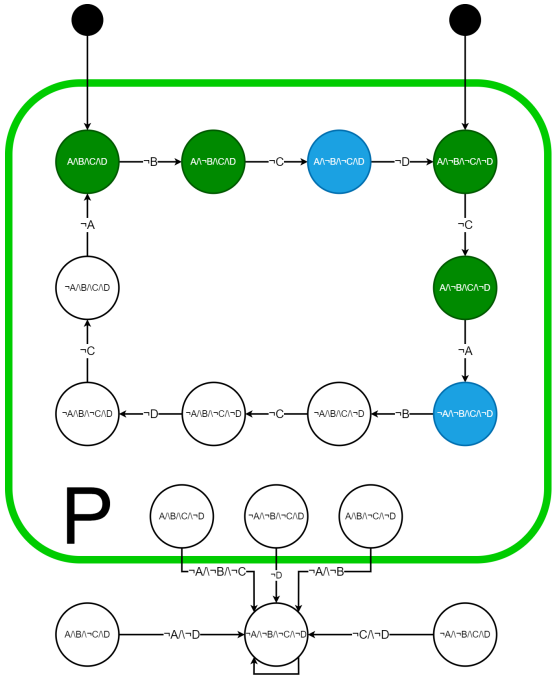


Figure 3.4: States that hold under F_1 and the states they transition to

When the example program (Figure 3.1) enters the Iteration Phase (Algorithm 2) the first part of this phase is to check that the newly created F_1 will not allow a transition to an unsafe state. This is done by seeing if $(F_1 \wedge T) \Rightarrow P$' holds. In this example F_1 only permits transitions to safe states so it can therefore enter the Holding Phase because of the states in F_1 (the green circles in Figure 3.4 transition to the blue states). Here, each clause c in F_1 is checked to see if it holds with the formula $(F_1 \wedge T) \Rightarrow c$'. If a clause holds, it is conjoined to P to form a new layer F_2 in the same way F_1 was constructed in the Initialisation Phase. The clauses are:

$\{(\neg d \vee c \vee \neg b \vee \neg a), (b \vee \neg d \vee a \vee \neg c), (b \vee c \vee a \vee d), a, (\neg b \vee c), (\neg d \vee c), (\neg b \vee d)\}$

After these checks, F_2 becomes:

$(\neg d \vee c \vee \neg b \vee \neg a) \wedge (b \vee \neg d \vee a \vee \neg c) \wedge (b \vee c \vee a \vee d) \wedge (\neg b \vee c) \wedge (\neg b \vee d)$

3.4.2 Erroneous Phase

If at the start of an iteration the check $(F_h \wedge T) \Rightarrow P$' fails and a counterexample s has been found in the latest Ladder Logic Invariant Candidate. When transitions are made the program will be able reach a state outside the safe region P from the state s that has been included in the Invariant. Therefore, the algorithm enters the Erroneous Phase to resolve this. This counterexample s is a safe state that will transition to an unsafe state. It must therefore be removed in order to prevent the False Positive [51] and, in turn, the second clause in Inductive Verification [64] from failing. The initial implementation developed [46] included a basic method to remove the counterexamples where the negation of s is appended to all Ladder Logic Invariant Candidates where $F_j \wedge \neg s \wedge T \Rightarrow \wedge s$' holds - meaning $\neg s$ is inductively relative to at at least I. This approach is logically correct but as the number of counterexamples increased can be huge a more scalable solution is required.

The generalisation method from Bradley (Algorithm 2) attempts to group multiple counterexamples together by forming a Cube of states that can be removed in a simple iteration. A Cube is formed by dropping a literal from s. There are checks in place though to firstly ensure a Cube q only contains states that are unreachable using $\neg I \Rightarrow q$ which ensures q is not inductive. The second check, $F_j \wedge \neg q \wedge T \Rightarrow \wedge \neg q$', ensures that the after a transition where q cannot be reached, q still is not reached. The negation of q is appended to all Ladder Logic Invariant Candidates where this is the case. If no cube was able to be formed then the original counterexample s is appended as before.

The example program (Figure 3.1) is now in its second iteration, the first step is to check that the new Ladder Logic Invariant Candidate F_2 will not allow a transition to an unsafe state ($(F_2 \wedge T) \Rightarrow P$'). However, this time the check is not a tautology and therefore it means an unsafe state can be reached from the states that hold under F_2. Those states that hold under F_2 (Figure 3.5a) and are reachable are shown in green and unreachable in red. The states they transition to are highlighted in blue. This unfortunately includes one of the unsafe states. This red state $\neg a \wedge \neg b \wedge \neg c \wedge d$ is causing this problem and is the counterexample s.

(a) F_2 and the states it transitions to

(b) The Cube built to block counterexamples

(c) Amended F_2

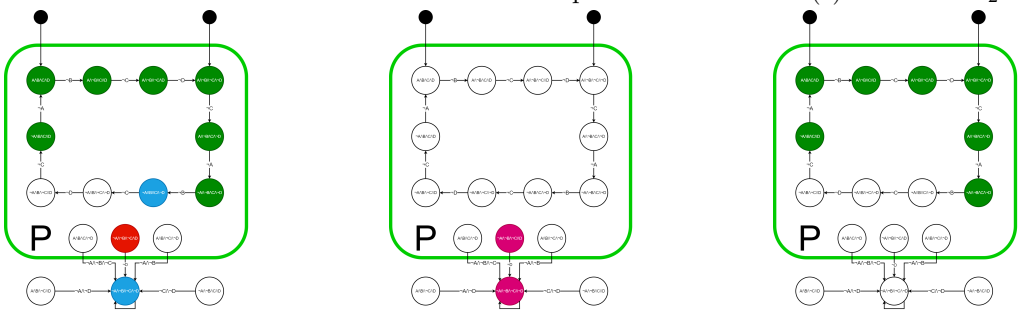


Figure 3.5: F_2 and the amendment process

The Erroneous Phase removes the state s by firstly finding the largest F_j where $\neg(F_j \wedge \neg s \wedge T \Rightarrow \neg s')$ - in this case it holds for all F_j, therefore $j := 2$. This is because the Ladder Logic Invariant Candidates F_0, F_1, and F_2 have been built. Next, a literal is dropped from s to form a Cube q, the literal d is dropped leaving $\neg a \wedge \neg b \wedge \neg c$. This cube needs to be checked to ensure that it is not inductive [64, 86], $\neg(I \Rightarrow \neg q)$, so that only contains unreachable states. This holds along with the second check $F_j \wedge \neg q \wedge T \Rightarrow \neg q$' which ensures the cube cannot reach a state where the cube holds again - which would result in the counterexample. As these two conditions are met the negation of the cube q is conjoined to F_0, F_1, F_2 which blocks the counterexample s including the unsafe state (Figure 3.5b) - q is an abstraction of s. Because the state that transitions to an unsafe state has been removed, the only transitions permitted are to a safe state (Figure 3.5c) and IC3 can continue to the Holding Phase. The new F_2 is:

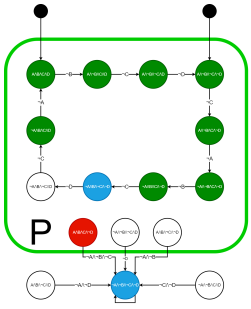
$$(\neg d \vee c \vee \neg b \vee \neg a) \wedge (b \vee \neg d \vee a \vee \neg c) \wedge (b \vee c \vee a \vee d) \wedge (\neg b \vee c) \wedge (\neg b \vee d) \wedge (\neg(\neg a \wedge \neg b \wedge \neg c))$$

3.5 Constructing an Invariant with IC3

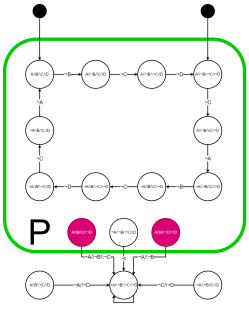
In the example Ladder Logic Program (Figure 3.1) the Initialisation Phase built F_0 and F_1 (Chapter 3.4). The Iteration Phase started with the Holding Phase where F_2 was constructed (Chapter 3.4.1). However, this was unsafe so it was refined using the Erroneous Phase in the second iteration (Chapter 3.4.2). F_2 does not cover all the reachable states, therefore, further iterations are required to construct more Ladder Logic Invariant Candidates such that F_h grows to include these states. Firstly, the Iteration Phase begins by checking that $(F_2 \wedge T) \Rightarrow P$' holds, therefore that a transition from the states in F_2 cannot reach an unsafe state. This holds therefore the Holding Phase checks which clauses c in F_2 to hold using $(F_2 \wedge T) \Rightarrow c$'. As before, if they hold, then they are conjoined to the P to form another Ladder Logic Invariant Candidate F_3:

$$(\neg d \vee c \vee \neg b \vee \neg a) \wedge (b \vee \neg d \vee a \vee \neg c) \wedge (b \vee c \vee a \vee d) \wedge (\neg b \vee c) \wedge (\neg(\neg a \wedge \neg b \wedge \neg c))$$

(a) F_3 and the states it transitions to



(b) The Cube built to block counterexamples



(c) Amended F_3

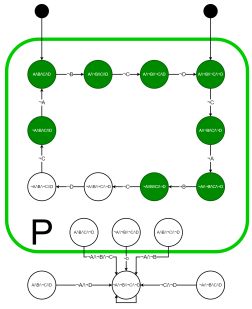


Figure 3.6: F_3 and the amendment process

However F_3 again includes a state that, while unreachable, can transition to an unsafe state. This state is marked in red in the automaton (Figure 3.6a) with the states that can be reached in blue. This means the formula $(F_3 \wedge T) \Rightarrow P'$ doesn't hold, so the program enters the Erroneous Phase. Here, the Cube $a \wedge b \wedge \neg d$ is formed by dropping the literal c which removes two counterexamples in one iteration (Figure 3.6b) by conjoining the negative of this cube q to each Ladder Logic Invariant Candidate to form $F_j \wedge \neg q$. This now blocks of these states from being reached (Figure 3.6c). A check is then made in the next iteration to ensure this new F_3 keeps the program safe, so the algorithm continues with the Ladder Logic Invariant Candidate:

$$(\neg d \vee c \vee \neg b \vee \neg a) \wedge (b \vee \neg d \vee a \vee \neg c) \wedge (b \vee c \vee a \vee d) \wedge (\neg b \vee c) \wedge (\neg(\neg a \wedge \neg b \wedge \neg c)) \wedge (\neg(a \wedge b \wedge \neg d)$$

This now allows for the program to enter the Holding Phase again to check whether the clauses in F_3 hold. This is the case so the Ladder Logic Invariant Candidate F_4 to be constructed and satisfies all of the reachable states. With the originally implemented IC3 method [46] there still would be a counter example to remove but this was dealt with already in the previous iteration.

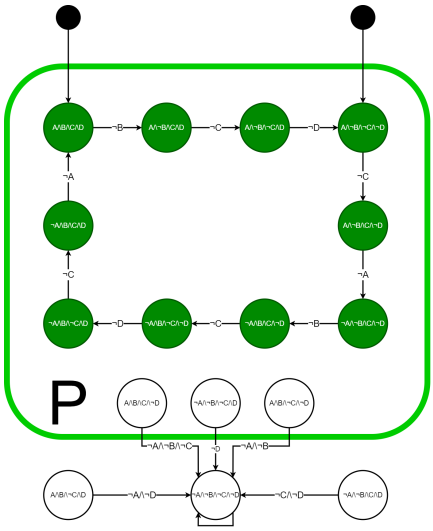


Figure 3.7: States that hold under F_4 and F_5

The program then starts another Iteration to produce F_5 via the Holding Phase. This checks all the clauses in F_4 to see if they hold, which they all do and are all added to P to from F_5. This means that $F_5 \Leftrightarrow F_4$ and the Iteration Phase can finish as the two Ladder Logic Invariant Candidates are equivalent. F_5 covers only the reachable states (Figure 3.7) meaning that the Invariant formula will only allow transitions between safe states and prevent a False Positive [51]. The Ladder Logic Invariant for use in Inductive Verification for this Verification Task is:

$$(\neg d \vee c \vee \neg b \vee \neg a) \wedge (b \vee \neg d \vee a \vee \neg c) \wedge (b \vee c \vee a \vee d) \wedge (\neg(\neg a \wedge \neg b \wedge \neg c \wedge d)) \wedge (\neg(a \wedge$$

$b \wedge c)) \wedge (\neg(a \wedge b \wedge \neg d))$

When this formula is combined with the safety property to form the CASL specification [78] below, the two clauses in Inductive Verification are met when it is applied with previous specifications (Figure 3.8). This is because the Initial States are in the safe region P. But now the unsafe states cannot be reached from the unreachable states that were causing the False Positive that previously resulted in the verification failing. With this Ladder Logic Invariant produced by the IC3 algorithm, the program can be verified to be safe using CASL [78] just the same as when the Invariant finding was performed manually.

In CASL the TRANSITION specification defines the variables used in the system, and their primed equivalents, along with the rules of the system described previously as T. The second specification, SAFEINTITIAL, checks whether the Initial States I are in the set of Safe States P. The STEPSAFE specification checks that any Safe State will reach another Safe State using the Transition rules T, however, for this example it fails due to a False Positive [22, 25]. With the Invariant this is blocked and therefore the specification STEPSAFEWITHINVARIANT passes.

spec TRANSITION =
  preds a, b, c, d : ()
  preds a', b', c', d' : ()
  • $a' \Leftrightarrow (b \wedge c \wedge d) \vee (a \wedge \neg\, b \wedge \neg\, c) \vee (a \wedge \neg\, b \wedge d)$
  • $b' \Leftrightarrow (\neg\, a \wedge b) \vee (\neg\, a \wedge c \wedge \neg\, d)$
  • c'
    $\Leftrightarrow (\neg\, b \wedge c \wedge \neg\, d) \vee (\neg\, a \wedge b \wedge d) \vee (b \wedge c \wedge d)$
      $\vee\, (a \wedge \neg\, b \wedge \neg\, d)$
  • $d' \Leftrightarrow (\neg\, a \wedge b \wedge \neg\, c) \vee (\neg\, a \wedge b \wedge d) \vee (a \wedge c \wedge d)$
end

spec SAFEINITIAL =
  TRANSITION
then • $(a \wedge b \wedge c \wedge d) \vee (a \wedge \neg\, b \wedge \neg\, c \wedge \neg\, d)$
then %implies
  • $(\neg\, d \vee c \vee \neg\, b \vee \neg\, a) \wedge (b \vee \neg\, d \vee a \vee \neg\, c) \wedge (b \vee c \vee a \vee d)$
    %(initial_safe)%
end

spec STEPSAFE =
  TRANSITION
then %implies
  • $(\neg\, d \vee c \vee \neg\, b \vee \neg\, a) \wedge (b \vee \neg\, d \vee a \vee \neg\, c) \wedge (b \vee c \vee a \vee d)$
    $\Rightarrow (\neg\, d' \vee c' \vee \neg\, b' \vee \neg\, a') \wedge (b' \vee \neg\, d' \vee a' \vee \neg\, c')$
      $\wedge\, (b' \vee c' \vee a' \vee d')$

%(step_safety)%

end

spec StepSafeWithInvariant =
Transition

then %implies

- $(\neg\ d \vee c \vee \neg\ b \vee \neg\ a) \wedge (b \vee \neg\ d \vee a \vee \neg\ c) \wedge (b \vee c \vee a \vee d)$
  $\wedge\ ((\neg\ d \vee c \vee \neg\ b \vee \neg\ a) \wedge (b \vee \neg\ d \vee a \vee \neg\ c)$
  $\wedge\ (b \vee c \vee a \vee d) \wedge \neg\ (\neg\ a \wedge \neg\ b \wedge d) \wedge \neg\ (a \wedge b \wedge \neg\ d))$
  $\Rightarrow (\neg\ d' \vee c' \vee \neg\ b' \vee \neg\ a') \wedge (b' \vee \neg\ d' \vee a' \vee \neg\ c')$
  $\wedge\ (b' \vee c' \vee a' \vee d')$

%(step_safety_with_invariant)%

end

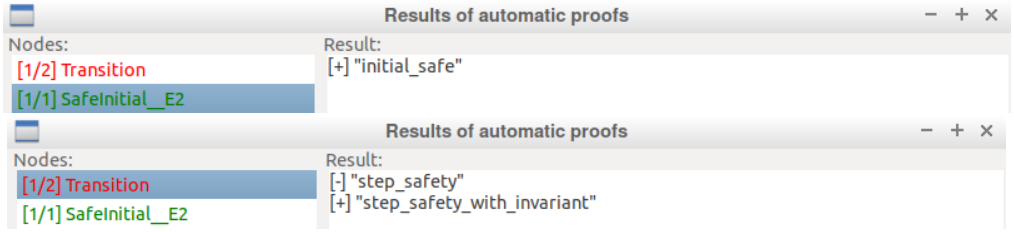


Figure 3.8: Four variable Ladder Logic Program passing Inductive Verification with Invariant. Without the Invariant, the second clause fails

A IC3 run-through of all artificial Ladder Logic Programs can be found in Appendix B.

3.6 Improved IC3

Inductive generalisation in Bradley's IC3 implementation makes it possible to deal with "huge state spaces", making IC3 more suitable for industrial use. The success is dependant on the ability to abstract counterexamples consisting of a single state into counterexamples consisting of a 'large' set of states, which itself hinges on the connectivity of the model's state graph and its encoding. The implementation developed as an Undergraduate [48, 46] removed the counterexamples, also known as Counterexamples to Induction [39], individually. This was fine for the smaller test examples but did not scale well. Bradley's implementation has the generalisation approach which makes it more scalable.

There have been refinements to IC3, in particular an approach by Zyad Hassan and Fabio Somenzi, working alongside Bradley, which improves the algorithm's generalisation [62]. The ABC implementation [74, 75] used in this research for Ladder Logic Verification follows this approach. This refinement finds Counterexamples To Generalisation (CTGs) [6] which enable IC3 to explore further from the error states to remove multiple counterexamples at once while still being backwards reachable - therefore reducing the number of

iterations required and therefore the runtime. This improvement to IC3 generalises Counterexamples to Induction [39] in the same way as Bradley's implementation. However, in Bradley's implementation if a Cube cannot be produced then the counterexample is removed individually. The refinement means that when a state is found that cannot be grouped it is dealt with to still allow Cubes to be formed.

3.6.1 Counterexamples to Generalisation

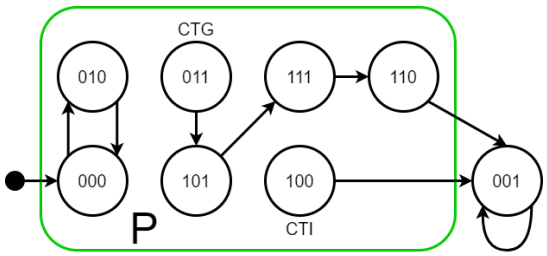


Figure 3.9: An example Ladder Logic Program that contains a counterexample that cannot be generalised [62]

The paper that presents this improved generalisation approach [62] contains an example automaton (Figure 3.9). In this automaton there are two reachable states: the Initial State 000, and then 010. All states are safe except for the state 001 - this is expressed by in Figure 3.9 by having a double circle for the unsafe state. If Inductive Verification [64] was performed it would fail because of two Counterexamples to Induction (CTI): states 100 and 110 as both transfer to 001 which is outside the safe region P. In an ideal scenario the IC3 algorithm [43, 44, 34] would look to generalise these two counterexamples to have a Cube that contains both states by dropping a literal - this would therefore prevent the need for an extra iteration. Taking these two states out from the final Invariant will block the False Positive. However, because the counterexample 110 has a predecessor the Cube must include the furthest predecessor to prevent this state from being reached. The set of states generalised by the cube should be {001, 101, 111, 110, 100}. It is therefore impossible to create a cube that generalises all states that lead to the unsafe state 001 because of the state 011. Dropping a single literal will not include all of the unreachable states in that chain, with 011 a Counterexample to Generalisation (CTG) as it is the furthest predecessor of a counterexample and blocks a Cube from being formed. Because of this the IC3 algorithm has to prove that both counterexamples are unreachable, and remove both separately to prevent a False Positive [51] instead of only the once if a cube could have been found. In general, Counterexamples to Generalisation are often deep backward reachable, where the state that prevents a cube from being formed is towards the beginning of a chain of transitions. Addressing them reduces the depth of the explicit backward search performed by IC3. In essence this means that the cubes are able to be as large as possible. IC3 currently uses induction [64, 86] in the dropping of literals to form cubes. Given a cube

q, the goal is to find an inductive subclause of $\neg q$. If the cube doesn't include any of the Initial State I then it is unreachable for $i + 1$ steps.

3.6.2 Addressing Counterexamples To Generalisation

When building a new cube q_{j+1} by dropping a literal, there is the check if the state s is a Counterexample to Generalisation, i.e., a reachable state is included in the cube. This would have as a consequence that a reachable transition is blocked. Therefore, in this case the cube would not be constructed. All counterexample states in the cube would have to be removed individually.

To solve this issue, first it must be established whether the negation of the cube ($\neg q$) is inductive and can be removed. If this is not the case s cannot be joined with q. Instead an attempt is made to block s by proving that s is inductive relatively to F_{i-1}.

If a cube q has a Counterexample to Generalisation that is not found to be inductive to any potential Invariant then the Counterexample to Generalisation is joined with q. For larger depths, an encountered Counterexample to Generalisation is one that interferes with the generalisation of a Counterexample to Generalisation Induced clause. A limit is therefore put in place to control the effort spent on addressing Counterexamples to Generalisation in this case. When this limit is exceeded Counterexamples to Generalisation are not examined any further and no joins take place. In this case the counterexample is removed individually in order to save time.

In the original IC3, if a safe state s has a transition to an unsafe state it must be dealt with to prevent a False Positive when the verification is performed. However, if s has a predecessor it too can reach the unsafe state. Hence, dropping literals to produce cubes reduces the number of states that need removing individually.

In the ABC implementation [74, 75] that follows the paper by Zyad Hassan, Aaron R. Bradley and Fabio Somenzi [62] all predecessors of large Cubes are considered as Counterexamples to Generalisation as they are less likely to be backwards reachable whilst being more likely to be further from the error state.

3.6.3 Comparison to Standard IC3

In the ABC [74, 75] implementation of IC3, the Cubes are expanded through ternary simulation [49], a process for detecting hazards, before being checked for inductiveness. The implementation addresses the generalisation weakness in the original IC3 by accelerating the exploration of the deep backward reachable states. The testing in the paper [62] confirmed that typically Counterexamples to Generalisation are found deeper than regular counterexamples to induction (the further the predecessors go the more likely they are to be a Counterexample to Generalisation). The new implementation produced stronger counterexample-induced clauses while reducing the depth of the explicit search of IC3 - which indicates a better performance. However, it can increase the convergence level mean-

ing that a state could be blocked that is not inductive. An excessive number of clauses derived to block Counterexamples to Generalisation often results in longer runtimes by comparison - as expected but still quicker than standard IC3.

3.7 The AIGER Format

Both implementations of IC3 tested utilise AIGER [38, 37], a format, library and set of utilities for And-Inverter graphs (AIGs) [12]. The name AIGER partially comes from an acronym of And-Inverter graphs and if pronounced in German sounds like the "Eiger" mountain in Switzerland. The AIGER format has an ASCII and a binary version. Bradley's IC3ref [42] can operate on either but Mishchenko's ABC [74, 75] requires the files to be in the binary format. The binary format is more restricted in comparison making it much more compact and easier to read making it a format used for benchmarks. An AIGER library developed by Armin Biere [36] has been used in this project which contains various tools to format the AIGER files. For example the "aigtoaig" utility to transform the files between the ASCII and Binary formats. The Ladder Logic files were translated to ASCII AIGER files using the Aiger-fier [45], developed in this project, and converted by "aigtoaig" utility into the Binary format for ABC. The Ladder Logic files need converting into AIGER, achieved using the Aiger-fier and its transformation developed in this thesis (Chapter 5).

3.7.1 And-Inverter Graphs

And-Inverter graphs (AIGs) [12] are directed, acyclic graphs that represent the logical structure of a circuit or a network. They consist of logical "Ands" and "Nots" (inverters) which gives the graphs their name. Research into simple circuits composed of simple gates, including AIGs, can be traced back to 1948 with Alan Turing's work on neural networks which described a randomised trainable network of NAND gates [89]. Conjunctions in an AIG consist of two-input nodes, this means that formulas are built recursively in the tree like structure (Figure 3.10) [13]. A negation is represented by a marker on an edge, and by using De Morgans Laws [1], logical "Ors" can be translated.

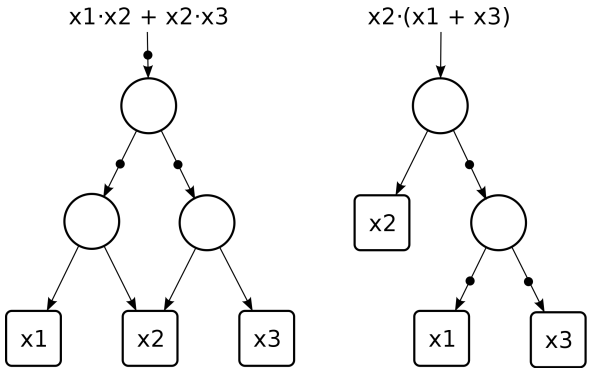


Figure 3.10: Examples of And-Inverter Graphs [12] $(x1 \wedge x2) \vee (x2 \wedge x3)$ and $x2 \wedge (x1 \vee x3)$

The translation from logic gates to an And-Inverter graph is inexpensive because it only requires that every gate be expressed as either an "And" gate or an inverter. AIGs have been used in Electronic Design Automation applications. Combining And-Inverter graphs with Boolean satisfiability has allowed them to be used in formal verification, including both model checking and equivalence checking - ABC [74, 75] is an example of this. ABC is an example that And-Inverter graphs can be used for logic synthesis, technology mapping, physical synthesis, and formal verification because of the simplicity and uniform structure of AIGs, allowing rewriting, simulation, mapping, placement, and verification to share the same data structure.

3.7.2 AIGER Components

The ASCII format of AIGER is easier to understand and develop. The Ladder Logic Programs were translated to the ASCII format for this reason so that the IC3 implementations could be ran. For ABC, the "aigtoaig" tool was used to convert the files to the binary format. AIGER is comprised of Inputs, Latches that store a result until reset by another signal and are initialised to zero unless stated, Outputs, and logical "Ands" comprised of two inputs. Each variable in AIGER is given a numerical value, even is true whilst odd is the negation. Each component given an ID that is a multiple of two, the value +1 is the negative ID of the component. For example, the variable a could have the ID 2, and $\neg a$ the ID 3. An AIGER file in ASCII format begins format identifier string "aag" (the identifier for the binary format is "aig" and 5 non-negative integers "M", "I", "L", "O", "A" separated by spaces where:

- **M:** The maximum variable index
- **I:** The number of input Variables
- **L:** The number of Latches
- **O:** The number of Outputs
- **A:** The number of logical "Ands"

A logical "And" of two inputs in the ASCII AIGER format is:

```
aag 3 2 0 1 1
2                          input 0
4                          input 1
6                          output 0
6 2 4                      AND gate 0      1 & 2
```

Figure 3.11: The conjunction of two Inputs written in the ASCII AIGER format

This is equivalent to $A \wedge B$. The inputs are given the IDs 2 and 4, in this case A is input 2 and B input 4. The literal representing the AND gate is 6 with variable index 3. This is the conjunction of inputs 2 and 4, or $A \wedge B$. The output of the AIGER file is the literal 6, the logical "And". The first number in the header, which denotes the maximal variable index is 3 as there are two Inputs and an AND gate requiring an ID. The Output shares the value of the conjunction as no more computation is required. The next number is 2 because of there are two Inputs. There are no Latches, but there is an Output and one AND gate.

```
aag 3 1 1 1 1
2                          input 0
4 5 1                      latch 0
7                          output 0        !(!1 & !2)
6 3 5                      AND gate 0      !1 & !2
```

Figure 3.12: The conjunction of an Input and a Latch written in the ASCII AIGER format using De Morgan's Laws

Above is a second example of AIGER this time representing a logical "Or" in the ASCII format (equivalent to $A \vee B$) and having only one Input with a Latch. This latch 4 is initialised to true and is set to the negation of this latch after each iteration - literal 5. Again, index 2 is A and 4 is B. De Morgan's Laws are used to represent an "Or" as a conjunction, with an AND gate being set to the negation of input 0 and latch 0. The output is the negation of this AND gate - therefore the formula is $\neg(\neg A \wedge \neg B)$. Again there 3 is the maximal variable index with one Input, Latch, Output and AND making up the header.

3.7.3 Safety Properties in AIGER

In order to use AIGER [38, 37] in Ladder Logic Verification there is a need to include a Safety Property P and whether a bad state can be reached. In AIGER this can be achieved by setting the output to be the unreachable states. That way when IC3 is performed if the unsafe states are true, then they have been reached and the program is unsafe. For example, the Two-Variable Ladder Logic Program (Figure 1.1) can be converted into the ASCII Aiger format as follows:

```
aag 4 0 2 0 2 1
2 6 1                          latch 0
4 4 1                          latch 1
8                              invariant constraint 0
6 3 4                          AND gate 0      !1 & 2
8 7 5                          AND gate 1      !1 & !2
```

Figure 3.13: The use of the Invariant Constraint in the ASCII AIGER format

Firstly, the header has changed. The maximum variable index is 4 from the two latches and two "Ands". There are no Inputs instead both variables x and y are Latches (2 and 4 respectively) because they both have a coil in the Ladder Logic [66] and their values are set after each transition - they are both initially true. There are no Outputs this time. There are two logical "Ands", the first is for deciding the value of x which is set to the result of $\neg x \wedge y$ - index 6 is set to the conjunction of the negation of 2 and 4. The other is for the formula that describes the unsafe state $\neg x \wedge \neg y$ - index 8 is set to the conjunction of 7 (as x is set to index 6) and 5 (y negated). The final 1 in the header indicates that there is one Invariant Constraint. The Invariant Constraint is the unsafe state. The idea is that if an unsafe state is reached then the program is unsafe and a Ladder Logic Invariant cannot be produced. In this case it is set to 8, therefore, if 8 is true then the unsafe state $\neg x \wedge \neg y$ has been reached. For IC3ref, the utility tool "aigmove" is run first to treat the Invariant Constraint as an output by rearranging the header. When "aigmove" is applied to the example in Figure 3.13, the header becomes "aag 4 0 2 1 2". The tool "aigreset" ensure all initial values are either 1 or 0. Both are these are from the AIGER Library [36]. ABC does not require any commands to be run to format the binary AIGER files.

```
aig 4 0 2 0 2 1
6 1                            input 0
4 1                            input 1
8                              invariant constraint 0
^B^A^A^B                       AND encodings
```

Figure 3.14: The Binary AIGER format

Above is an example of the binary AIGER file format used by ABC. This is also equivalent to the Two-Variable Ladder Logic Program (Figure 1.1). The binary AIGER format is considered a subset of the ASCII format, sharing the same header structure as with the ASCII format but place more restrictions on the order and also does not allow unused literals, it also more compact in comparison. After the header the list of inputs and all the current states of a latch can be omitted. Any AND gates are binary encoded.

CHAPTER 4

Literature Review

Contents

This section will provide a brief discussion on existing research on the topics covered by this thesis. For this, many case studies and approaches have been explored for verification of control systems for both liveness and safety properties. Here, we focus on areas of particular relevance to our project goals, and where potential problems have been overcome.

4.1 Railway Verification

Verifying and validating railway control systems are safe is vital in order for use in industry. In the UK, Network Rail have a vast set of standards [26] that define how the railway should be designed. Notably are NR/L2/SIG/11201/MOD B7 on the general principles for interlockings and NR/L2/SIG/11201/MOD B11 which focuses specifically on electronic interlocking guidelines. Siemens worked with Network Rail and Thameslink on the "Thameslink Central Core" project [18] that focused on upgrading the network in central London allowing for more trains through per hour using using European Train Control System (ETCS) Level 2 and In-cab signalling [27]. Verification of the proposed system specification was performed using a computer simulation with a proven model of the ETCS core functions to construct a proof that the system will always conform to safety properties. Network Rail's Systems Integration Laboratory was then used to model the rail environment and drive the relevant system inputs using the equipment used to be used in operation on the train and

trackside allowing for test simulations to be run. Following this, trains were run to test the system in operation.

Formal methods have been used elsewhere in railway verification. An example of this is the work by Iliasov, Taylor, Laibinis and Romanovsky on SafeCap [63] - a toolkit for modelling, simulation and formal verification of railway control systems. It allows signalling engineers to design interlockings relying on the provided domain specific language (SafeCap DSL), check their safety properties, and evaluate potential improvements using a combination of theorem proving, SMT solving and model checking. The verification process uses a SAT solver, external provers provided via the Why3 framework, and the ProB model checker. SafeCap was tested on railway data provided by Network Rail and returned no errors considered to be false positives. However, it was only tested on small test examples and therefore is still vulnerable to state explosion. The prover developed is not as powerful as many state-of-the-art provers, however, the advantage of it is that it could be customised to suit the needs.

4.2 Existing Ladder Logic Verifier developments

To give an insight into previous projects working on the Ladder Logic Verifier and how Siemens develop their railway interlockings. Firstly, the Master of Research thesis written by James in 2010 [66] focuses firstly on Ladder Logic in depth and how it affects railway signalling functions. It explores Propositional Logic and automatons, before explaining how Ladder Logic can be verified. In the thesis, it explains there is software developed that determines if a system is considered to be safe or not by applying Bounded Model Checking [55, 30, 66, 54] following on from the work with SAT-based model checking by Kanso [67]. This allows for a defined number of transitions (k) to take place and returns whether the program is guaranteed to be safe for at least k transitions. If unsafe then a counterexample trace is returned which identifies the transitions that lead to the unsafe states being reached which helps the refinement process as the engineers can amend the Ladder Logic to prevent this.

Werner's Master of Research thesis in 2017 [92], in collaboration with Siemens Railway Automation Ltd., explores the safety verification of ladder logic programs building on the development of a verification tool. The thesis explains how a generic format of Safety formulae for railway systems, that can be executed within a verification tool for a specific track plan, was produced using published standards and railway case studies. This also includes the translation process for these Safety formulae. The thesis finishes with an explanation of a Ladder Logic verification tool for potential use within Siemens. The implementation produced in this project was tested against a similar example to ensure that it functions correctly and efficiently. Both Werner's and James' theses have helped develop the Ladder Logic Verifier [87, 51] which suffers from the over-approximation problem where because Inductive Verification includes both reachable and unreachable states, any unreachable safe state that transitions to an unsafe state is included and the verification fails. Bounded

Model Checking can only determine if a program is unsafe by finding if the error can be found within k bounds. The current methods have no way to definitely confirm the over-approximation has occurred and it is not a genuine error because k is never equal to the number of transitions in the system due to runtime, which leaves the topic open. The aim of this project was to resolve this in an efficient manner.

We still want to utilise Inductive Verification [64], run prior to Bounded Model Checking, as this is incredibly efficient in terms of resources and runtime - requiring just two calls to a SAT-solver [19]. The first checking the initial states are safe and the second that any safe state will transition to another safe state - but only when the Invariant also holds. Bounded Model Checking and other verification approaches are more costly by comparison. Therefore, by applying a process that can find Ladder Logic Invariants that will prevent over-approximations it would still allow for Inductive Verification to be run to ensure the Ladder Logic Program is safe with respect to a given property.

4.3 Automated Ladder Logic Verification

Away from Siemens there have been developments in Ladder Logic [66] and PLC [16] verification running the approaches developed on small test programs similar the to the artificial testbed seen in this thesis. This section will give an overview of some of the work conducted. The first of which is the paper "Automated Deductive Verification for Ladder Programming" [56] by Cousineau, Mentré and Inoue in 2019. This presents a tool using Why3 [15] implemented for automating deductive verification. The goal was to develop a simple yet robust debugging tool for Ladder Logic development providing graphical feedback of failures - a continuation of the counterexample traces seen in Bounded Model Checking. Overall, the approach proved successful as it can be integrated into a Ladder IDE and provides details on errors. However, like with Siemens' Ladder Logic Verifier it too suffers from False Positives, where in reality a state cannot be reached but it cannot be distinguished. This we hope to resolve with IC3 and efficiently constructing Ladder Logic Invariants.

Elsewhere in the field of automatic verification of Ladder Logic Programs, the paper "Towards automatic verification of ladder logic programs" [95] by Zoubek, Roussel and Kwiatkowska in 2003 presents how exhaustive searching can be conducted to construct a model mechanically from the Ladder Logic Program. From this performing automatic verification against requirements using the model checker Uppaal [5]. This again proved successful, however, the approach was taken to "slice" the Ladder Logic to reduce the state space. The Ladder Logic Verifier has a similar slicing method where it removes redundant variables to reduce the code. The issue with slicing a Ladder Logic Program is firstly it takes time to perform, and secondly any counterexample traces returned are not the full trace because the transitions have been changed from the original.

On the topic of Bounded Model Checking, the paper "Bounded Model Checking of PLC ST Programs Using Rewriting Modulo SMT" [68] by Lee, Kim and Bae published in 2022 focuses on a Bounded Model Checking technique. This approach is for PLC Structured

Test programs based on rewriting-based semantics. It achieves this by implementing a way to rewrite Satisfiability Modulo Theory [32] formulas, giving the ability to reason with constrained terms with SMT variables. The approach follows the Bounded Model Checking method seen in the Ladder Logic Verifier where it is checking the reachability from the Initial State to any state that is unsafe. By setting the bound to be large enough it can therefore include all the reachable states. However, the results did not scale well as with Bounded Model Checking [55, 30, 66, 54] in the Ladder Logic Verifier due to the fact that no optimisation methods for rewriting modulo SMT were included. IC3 in general though is believed to be quicker than Bounded Model Checking, and is not hindered by only including up to a set number of transitions to reduce the runtime.

There is also the work published by researchers at Jadavpur University, Kolkata in 2008 which looked into PLC verification using symbolic model checking. This approach involved constructing Binary Decision Diagrams for the Ladder Logic Program and running a model checking to find errors. The method was proven to be efficient and validated by manipulating the Rungs. This while efficient still required three steps, one to translate and build the diagrams, one to check the Initial States and a final step to check the transitions. This is similar to Inductive Verification with IC3 in the sense that IC3 would be run first to produce a Ladder Logic Invariant and then Inductive Verification run after with it. The benefit of this approach would be that IC3 would also identify which are unsafe and therefore would be unable to construct an Invariant. IC3 would exist the iteration process early in this case.

Finally there is the 2022 paper "Automated formal analysis of temporal properties of Ladder programs" [33] by Belo Lourenço, Cousineau, Faissole, and Marché and Mentré and Inoue. This paper also focused on using Why3 [15] for deductive program verification, checking Safety Properties with automated theorem provers. This aims to either prove the Ladder Logic Program holds under the condition or return why there is a possible execution scenario where it does not conform to the timing chart the property is constructed from. The results showed this to be successful in both scenarios, although there is a question over the correctness in how the Loop-Invariants are constructed as it is not part of the trusted code base used. It may also be the case that as SMT solvers [32] are unable to solve the generated properties because the solvers are interrupted after a given time limit. As IC3 has no time limit this will not be problem as it cannot "time itself out". Overall though, the research in the field has identified ways to return counterexample traces as seen in the Ladder Logic Verifier. If proven to be successful at deciding Verification Tasks and to be efficient, then IC3 could be later adapted to include this in order to match existing Ladder Logic verification methods.

Part II

Contribution

CHAPTER 5

Efficient Ladder Logic to AIGER Transformation

Contents

Two tools are utilised in converting the Ladder Logic Programs into AIGER [38] for verification with IC3. The first is the Aiger-fier [45] which takes the Ladder Logic, in TPTP format [4, 8], and performs a variation of the Tseitin Transformation [73] with De Morgan's Laws [1] to produce an AIGER file comprising of only conjunctions and negations representing the Ladder Logic rules. The second tool formats Safety Properties by converting any Implications. This acts as a preprocessor, taking the Safety Condition files from Siemens and converts any implications. The Aiger-fier then builds a set of conjunctions and negations representing the unsafe states. If the output is true, the unsafe states are reached.

5.1 Amending the Aiger-fier for Efficiency and Accuracy

The Ladder Logic [66] to AIGER [38] transformation went through multiple refinements to fix bugs so that running IC3 run on the test examples converted into AIGER matched the known results. The transformation of the Ladder Logic was originally closer to the Tseitin Transformation [73] with each logical "And" and Negation being assigned a new Latch.

This was changed to only the coils being given Latches to make the transformation more efficient and reduce the number of arbitrary literals used (Figure 5.1). For example, with the original transformation a Latch that is initialised as true and then its negation from then on would require two Latches. One to set the Latch to true initially and then to the result of the second Latch. This second Latch would be set to false initially and then to the result of the first Latch from then on. This process was refined to remove the redundant Latch, having it set to true initially and then the negation of itself from then on.

```
aag 6 0 4 0 2 1
2 10 1
4 4 1
6 3 0
10 9 0
12
8 6 4
12 10 5
```

(a) Transformation using the Tseitin Transformation where each Latch and Negation are assigned a new Latch

```
aag 4 0 2 0 2 1
2 6 1
4 4 1
8
6 3 4
8 7 5
```

(b) Transformation using the refined transformation where only coils are given a Latch

Figure 5.1: Different ways of representing the Ladder Logic with Latches

The next thing to edit was the way the Safety Properties were passed in. These are in a different format to the Ladder Logic:

```
[group0]
#
Property with instances: r1: S6909(AM), r2: S6909(BM),
#
("S6909(AM).U_0") -> ("S6909(BM).N_0")
```

A second program, the Safety Property Parser [47] `https://github.com/HarryBryant99/SafetyPropertyParser`, was used to convert the Safety Property to be in the same format as the Ladder Logic [66]. This removes the implications from the formulas, and creates an equivalent formula using logical "Ors" and Negations.

The biggest confusion though came from the Input variables. In the Ladder Logic, the variables come with a number at the end in the forms: "variable_0", "variable_1", "variable_2". Originally it was assumed that they were all the same variable. But in reality, those of "_0" are Input values, "_1" are the results after one iteration, and "_2" are the results after two iterations which allow the result to be stored and passed through

iterations. The Aiger-fier was amended so that to include this. Because there are different versions of variables ("_0", "_1", "_2") not all variables have a coil. If a variable does not appear as a coil it is therefore an Input variable. The previous versions of the Aiger-fier transformation had one variable for all versions of a variable, therefore changing the logic of the program. After discussions it was determined that each version of a variable should be treated separately amending this issue.

Any variables that only exist as "_0" are inputs as they do not have a corresponding coil to reset their value after each iteration. In this case their values should be non-deterministic after an initial value. Originally, the inputs were incorrectly defined by latches, which if set to themselves will produce a constant. The second refinement had the inputs defined as an AIGER input. However, when Inductive Verification and Bounded Model Checking are performed the Input variables are all set initially to 0. In AIGER, the Inputs are arbitrary and therefore require a Latch to initialise them to 0. Therefore, more refinements were required: this time combining a Latch, which is given an initial value, and an AIGER input which is non-deterministic. The Latch is set to the value of Input after an iteration:

```
aag 2 1 1 1 0
2                          input 0
4 2 0                      latch for input 0
4                          output 0
```

With this Input/Latch approach combined with the refined Tseitin Transformation it allows the input variables to be set to 0 initially and then can be true or false from then on matching the way Siemens perform their verification. This approach was applied to all cases where a variable appeared only as _0 with no coil. This improved the accuracy of the IC3 verification drastically when performed on the Siemens Testbed. What this now means is if after reading in the Ladder Logic in the TPTP format there exists any variables that do not appear as a coil they are considered as an input. These inputs are defined as a Latch, which is set to false as per Siemens approach, and are set to the result of an AIGER input after each iteration allowing the variable to non-deterministically switch between true and false. All other variables that have a coil are defined as regular Latches, set to true or false initially as required and then are set to the result of computations after each iteration. The ID of the Latch is used in both cases in the rules that make up the transformation.

5.2 Refined Tseitin Transformation in the Aiger-fier

The Ladder Logic [66] to AIGER [38] transformation, implemented in the Aiger-fier [45], takes in a list of Ladder Logic Rungs, a set of Input Variables I where initial values are defined, and a Safety Property P. The Safety Property is included so that the same variable indexes can be used. The Aiger-fier returns the numerical version of an And-Inverter Graph consisting of a set of Inputs, a list of Latches, an Output, and a list of logical "Ands".

The Ladder Logic is passed in a set of Rungs, each consisting of a formula (f') that is equivalent to a variable (x') in the form of $x' \Leftrightarrow f'$. In the transformation the variable x' becomes a Latch, with the formula f' being broken down. The formulas can be a single literal or a Negation, an end node, where no further work is required. Logical "Ors" need converting into conjunctions through the use of De Morgan's Laws [1]. Any logical "Ands" need to have two inputs so therefore a recursive approached is taken to break any larger conjunctions down into further conjunctions. Both of these transformations are required because the AIGER format [38], used to represent And-Inverter-Graphs [12], only has the ability to use Negations and logical "Ands" consisting of two literals. For all variations of the transformation, if any new Inputs discovered recursively are added to the set of Inputs. The transformation takes in a set of Rungs with each one being an equivalence. It also takes a set of pairs consisting of variable names allowing for the initial values to be set for these variables. Finally, it takes in a Safety Property. It returns a AIGER file made up of Inputs, Latches, an Output and a set of logical "Ands".

All parts of the transformation were tested to ensure that the formulas built in AIGER are equivalent to the originals. To ensure this, firstly the transformations were performed by hand so that the results of the Aiger-fier could be compared. For each version of the transformation the results were compared with the originals using Hets [77] to ensure that their equivalence is a tautology. Once satisfied that the transformation worked for all known cases the Aiger-fier was tested on the small Ladder Logic examples, with the resulting AIGER files drawn as And-Inverter Graphs [12] to confirm the programs were equivalent.

5.2.1 Transformation of Logical "Ands" where both children are not End Nodes

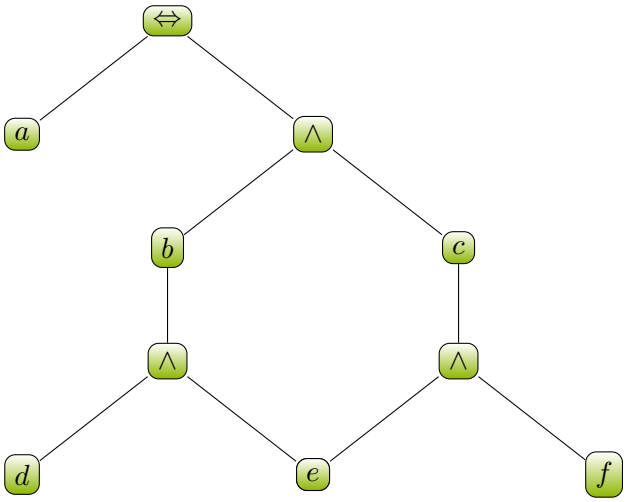


Figure 5.2: Boolean Expression Tree [13] of the expression $a \Leftrightarrow (b \wedge c)$ where b and c are further expressions and are not end nodes

If the top operator in the formula (the right-hand side of the equivalence) is a logical "And" and both children are not end nodes (not variables and are either conjunctions or disjunctions) then the transformation recurses on both child nodes to create further conjunctions. The original Latch a is then set to a new logical "And", and this is the conjunction of the two logical "Ands" created for the child nodes. For example, for the formula seen in Figure 5.2 where b and c are children of a conjunction, and are both logical "Ands" themselves. b is equivalent to $d \wedge e$, whilst c is equivalent to $e \wedge f$. In Figure 5.2 the node e is therefore a child of both c and d and is not redefined on the second occurrence. If those was the case then both versions of e would have to be updated with the same value after each iteration to maintain consistency and for large examples adds unnecessary components. Therefore, a check is made to see a variable already exists prior to it being stored. For this example the following AIGER components would be built:

```
Latch 0: a, set to value of And 0
And 0: the conjunction of And 1 and And 2
And 1: the conjunction of d and e
And 2: the conjunction of e and f
```

- $Latch\,0 \Leftrightarrow And\,0$
- $And\,0 \Leftrightarrow (And\,1 \wedge And\,2)$
- $And\,1 \Leftrightarrow (d \wedge e)$
- $And\,2 \Leftrightarrow (e \wedge f)$

Above is how the AIGER components would look when represented as a Boolean formula to demonstrate how that when the transformation takes place, the logic and satisfying arguments remain the same. These have been laid out in bullet points for simplicity, with each bullet point joined by a conjunction.

For this example the original formula was $a \Leftrightarrow ((d \wedge e) \wedge (e \wedge f))$. The nodes in the Tree can only have two children for a conjunction because this is the constraint set by AIGER. Therefore the formula becomes $(a \Leftrightarrow (b \wedge c)) \wedge (b \Leftrightarrow (d \wedge e)) \wedge (c \Leftrightarrow (e \wedge f))$. This is why the AIGER file produced has the three "Ands". The variable a has become a Latch in AIGER that is set to the value of the formula $(b \wedge c)$, with b and c also equivalent to conjunctions. Despite adding extra variables, the two formulas are logically equivalent. This process has been repeated for all varations of the transformation.

5.2.2 Transformation of Logical "Ors" where both children are not End Nodes

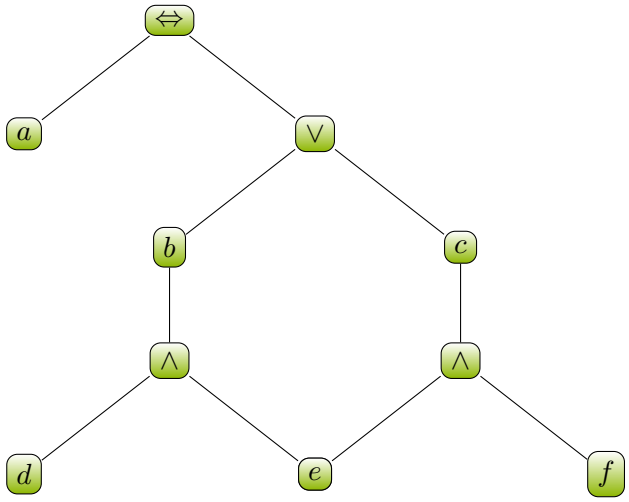


Figure 5.3: Boolean Expression Tree [13] of the expression $a \Leftrightarrow (b \vee c)$ where b and c are further expressions and are not end nodes

If the operand is a logical "Or" with both children not end nodes. Firstly, the two child nodes are investigated recursively. However, because it was an "Or" it must be converted into an "And" using De Morgan's Law [1]. Any double negations present are removed. A Latch is created which is set to negation of this new "And". With the "And" being the conjunction of the negation of the left child and the negation of the right child:

```
Latch 0: a, set to value of ¬(And 0)
And 0: the conjunction of ¬(And 1) and ¬(And 2)
And 1: the conjunction of d and e
And 2: the conjunction of e and f
```

- $Latch\,0 \Leftrightarrow \neg And\,0$
- $And\,0 \Leftrightarrow (\neg And\,1 \wedge \neg And\,2)$
- $And\,1 \Leftrightarrow (d \wedge e)$
- $And\,2 \Leftrightarrow (e \wedge f)$

5.2.3 Transformation of a Logical "And" with two End Nodes

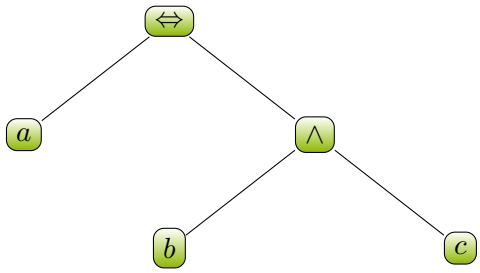


Figure 5.4: Boolean Expression Tree [13] of $a \Leftrightarrow (b \wedge c)$ where b and c are end nodes

When the operator is an "And" and it's the case that both sides of the operator are end nodes, meaning that they are variables, the two arguments are added to the list of Input Variables, if they are not already there, and the Latch is set to $arg_0 \wedge arg_1$.

```
Latch 0: a, set to value of And 0
And 0: the conjunction of b and c
```

- $Latch\,0 \Leftrightarrow And\,0$
- $And\,0 \Leftrightarrow (b \wedge c)$

5.2.4 Transformation of a Logical "Or" with two End Nodes

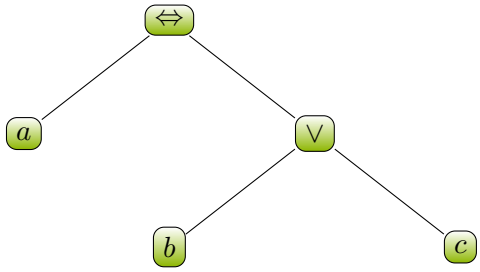


Figure 5.5: Boolean Expression Tree [13] of $a \Leftrightarrow (b \vee c)$ where b and c are end nodes

When the operator is an "Or" and again children are end nodes it must be converted into an "And". Again, the "Or" is converted to a conjunction and any double negations are removed. The two arguments are added to the list of Input Variables if they are not already there. While the new Latch is added to the set of Latches. The transformation produces:

```
Latch 0: a, set to value of ¬(And 0)
And 0: the conjunction of ¬b and ¬c
```

- $Latch\,0 \Leftrightarrow \neg And\,0$
- $And\,0 \Leftrightarrow (\neg b \wedge \neg c)$

5.2.5 Transformation of a Mixed Logical "And"

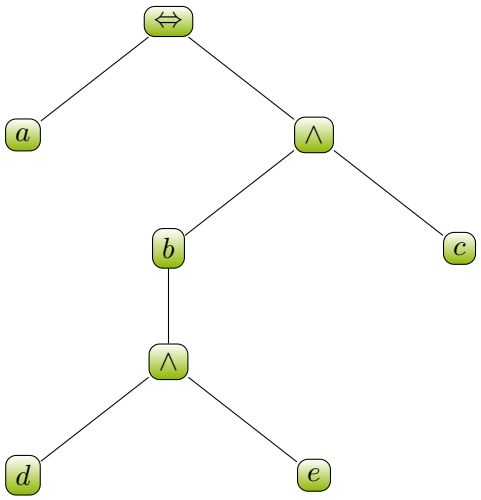


Figure 5.6: Boolean Expression Tree [13] of the expression $a \Leftrightarrow (b \wedge c)$ where b is not an end node, but c is an end node

It can also be the case that the operator is a logical "And" with one child is not an end node and the other is an end node. Firstly, the end node is added to the list of Input Variables if not already present. The non end node explored recursively. The original conjunction is set to the result of this recursion and the end node. With a Latch set to this conjunction:

```
Latch 0: a, set to value of And 0
And 0: the conjunction of And 1 and c
And 1: the conjunction of d and e
```

- $Latch\,0 \Leftrightarrow And\,0$
- $And\,0 \Leftrightarrow (And\,1 \wedge c)$
- $And\,1 \Leftrightarrow (d \wedge e)$

5.2.6 Transformation of a Mixed Logical "Or"

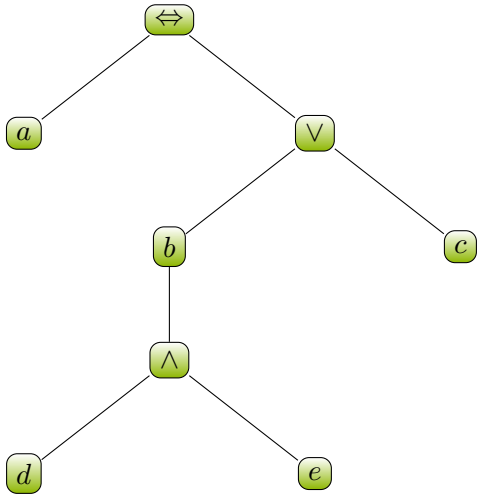


Figure 5.7: Boolean Expression Tree [13] of the expression $a \Leftrightarrow (b \vee c)$ where b is not an end node, but c is an end node

As before with the logical "Ors" it must be converted to an "And" with De Morgan's Laws. The original Latch n is set to this conjunction which is negation of the end node and conjoined with the negated result of recursively exploring the non end node:

```
Latch 0: a, set to value of ¬(And 0)
And 0: the conjunction of ¬(And 1) and ¬c
And 1: the conjunction of d and e
```

- $Latch\,0 \Leftrightarrow \neg And\,0$
- $And\,0 \Leftrightarrow (\neg And\,1 \wedge \neg c)$
- $And\,1 \Leftrightarrow (d \wedge e)$

5.3 Initialising the Input Variables

In the Ladder Logic Verifier [87, 51] all variables are initially set to false. In the AIGER format the Inputs cannot be constrained, an Input is a variable that doesn't appear as a Coil (Figure 2.3c) in the Ladder Logic Program. However, AIGER allows for Latches that can be set to true or false initially. Therefore, to represent an Input in from the Ladder Logic Program they are defined as a Latch. This Latch can be set as true or false (1 or 0) initially and then after an iteration the value will be set a regular AIGER input that is unconstrained. For example, one Input would be defined as:

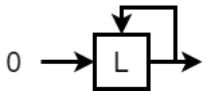
```
2       input 0 - unconstrained
4 2 1   latch 0 - set to 0 originally, then to input 0 after one iteration
```

Various checks have been developed to ensure that this process does indeed constrain an Input to be 0 initially and from then on be non-deterministic to match the Ladder Logic Verifier. These checks were developed as And-Inverter Graphs initially and then converted into the corresponding AIGER files so that the values could be manually traced and compared to what IC3 produces. For example, if the the check was to see whether a Latch initialised to false would remain false if set to itself the output of the AIGER file would be the Latch. This Latch would remain false and the output never true with the And-Inverter Graph operating as a cycle, overwriting the value of the Latch to false each time. Therefore, IC3 should return 0 (stating that an Invariant could be found) because the Latch is never true. If this is the case, then the check is correct.

5.3.1 Initialising a Latch

When a Latch is defined in AIGER that is set to 0 initially and is set to itself after one iteration, the output from IC3ref [42] is 0 meaning that the unsafe state was not reached and an Invariant could be found. Therefore, the Latch will always be false. Here the unsafe state is where the Latch is true. However, when the Latch is initialised to 1 IC3ref returns a 1 stating that the unsafe state was reached.

And-Inverter Graph	AIGER	Report

	`aag 1 0 1 0 0 1`
	`2 2 0  latch 0`
	`2  output 0`
		0

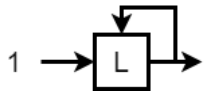
	`aag 1 0 1 0 0 1`
	`2 2 1  latch 0`
	`2  output 0`
		1

Table 5.1: Running initialised Latches with IC3ref, returning that the Latch remains false when initialised to 0 and true when 1. $Output \Leftrightarrow Latch$, $Latch \Leftrightarrow Latch$

5.3.2 Values can be dependant on Inputs

To show that the Inputs can make a difference a conjunction was set up between an Input and a Latch which is set to a constant value. The Input is non-deterministic, so it has the choice of being true or false. When the Latch is false the "And" is never true and therefore a 0 is returned by IC3ref as the unsafe state is never reached. However, when the constant

Latch is set to true the unsafe state can be reached and therefore 1 is returned. Here the unsafe state is the result of the conjunction.

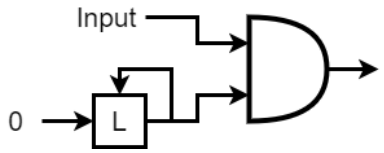
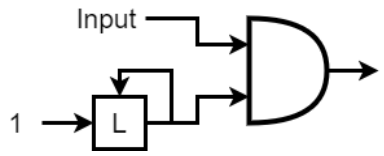
And-Inverter Graph	AIGER	Report
	aag 3 1 1 0 1 1 2 input 0 4 4 0 latch 0 6 output 0 6 2 4 and 0	0
	aag 3 1 1 0 1 1 2 input 0 4 4 1 latch 0 6 output 0 6 2 4 and 0	1

Table 5.2: Running a logical "And" of an Input conjoined with a Latch that is false and another that is true. $Output \Leftrightarrow Input \wedge Latch$, $Latch \Leftrightarrow Latch$

5.3.3 Inputs are Non-Deterministic

To show that the Input/Latch combination still allows for non-determinism an And-Inverter [12] was built that is the conjunction of two Latches that are both initialised to 0. These Latches are then set to the value of an AIGER Input after each iteration. When run with IC3ref [42] it returned a 1 because the unsafe state (the conjunction being true) was reached. This would be after at least one iteration because initially the conjunction would be false as both Latches are false, but would occur once the Input was true.

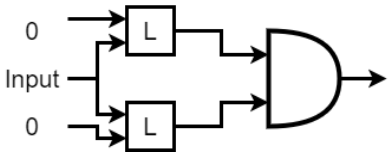
And-Inverter Graph	AIGER	Report
	aag 4 1 2 0 1 1 2 input 0 4 2 0 latch 0 6 2 0 latch 1 8 output 0 8 4 6 and 0	1

Table 5.3: Running an AIGER file consisting of the conjunction two Latches that are false initially, but after each iteration are set to the value of an input.
$Output \Leftrightarrow Latch_0 \wedge Latch_1$, $Latch_0 \Leftrightarrow Input$, $Latch_1 \Leftrightarrow Input$

5.3.4 Inverters

In order to show that the value of a Latch is updated after each iteration a pair of inverters were set up. It has already been shown that a Latch can be constant if needed (Section 5.3.1) but for the Ladder Logic Programs they need to be able to change. In the examples below the first Latch is set to false initially and then the negation of itself from then on. The output of this AIGER file is the Latch itself. When run, IC3 produces 1 stating that the Latch is true. Initially it was false, therefore the value of the Latch must have changed in order for IC3 to claim an Invariant cannot be produced. The second Latch is the opposite, starting as true and switching to false. This again fails IC3 because the Latch is true in the first iteration even though it then becomes false. If a constant Latch was set up that remained false throughout then IC3 would pass.

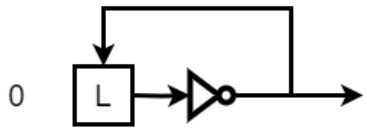
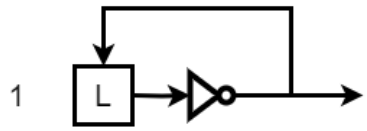
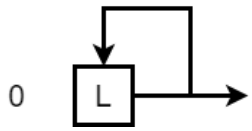
And-Inverter Graph	AIGER	Report
	aag 1 0 1 0 0 1 2 3 0 latch 0 2 output 0	1
	aag 1 0 1 0 0 1 2 3 1 latch 0 2 output 0	1
	aag 1 0 1 0 0 1 2 2 0 latch 0 2 output 0	0

Table 5.4: Running inverting Latches to show that their values change and reach an unsafe state after different iterations. A non-inverting Latch is also run to demonstrate that a constantly false Latch will not fail IC3. The first two tests are equivalent to $Output \Leftrightarrow \neg Latch$ and the Latch equivalent to $\neg Latch$. The final test is equivalent to $Output \Leftrightarrow Latch$ with $Latch \Leftrightarrow Latch$

5.3.5 Latch/Input Combination is False Initially and then Non-Deterministic from then on

In order to prove that the Latches are initially 0 and then can take the value of an input from then on an And-Inverter Graph was built that would reach the unsafe state when the input becomes true. The top half of the disjunction is a Latch that remains false. The bottom half of the disjunction is also false initially but then takes the value of the non-deterministic

input. Therefore, the disjunction is only satisfied when the input value is true. When run IC3 returns 1 confirming this.

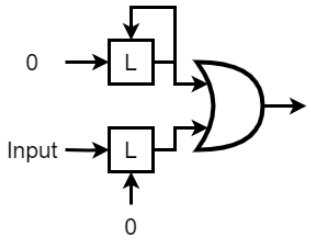
And-Inverter Graph	AIGER	Report
	aag 4 1 2 0 1 1 2 4 4 0 6 2 0 9 8 5 7	1

Table 5.5: An And-Inverter Graph that will not reach the unsafe state initially but will in the iterations after when the input is true. $Output \Leftrightarrow Latch_0 \wedge Latch_1$, $Latch_0 \Leftrightarrow Latch_0$, $Latch_1 \Leftrightarrow Input$

5.4 Aiger-fier Overview

The Aiger-fier [45] takes a Ladder Logic Program and converts it into the AIGER [38] format by performing the translation seen in this chapter. It also takes in any Initial Variables that need their value setting which allows Latches or inputs to be initialised as true when required. The Safety Property is also read in, having been formatted by the Safety Property Parser if necessary, so that the complete And-Inverter graph [12] can be constructed. It takes in a set of Inputs consisting of a Ladder Logic Program, a set of Initial values I, and a Safety Property P and produces an equivalent file in the AIGER format [38] made up of Inputs, Latches, an Output and logical "Ands". The program consists of over 30 Java classes, with approximately 14,300 lines of code. Each component has been tested as part of the Unit Testing [20] as seen in Section 8.2. Every process has a class that performs JUnit tests [2] to ensure that they function correctly. These were developed to check that all variations of formulas in the TPTP format are dealt with and processed as intended. The Aiger-fier can be found online in GitHub repository at `https://github.com/HarryBryant99/Aiger-fier` which also includes examples of the transformation.

The Ladder Logic Program is passed in the TPTP format [4, 8] with each Rung consisting of an equivalence that are stored in Tree Structures. Each variable in the TPTP file begins with "v", these are processed and are later replaced with a non-negative even value for AIGER. The parsing and numbering stages were heavily tested to ensure that each unique variable is correctly read and given a unique number for AIGER. The Safety Properties are read in a similar fashion. These variables should exist in the Ladder Logic and therefore do not require a new number. However, any new variables found are assigned the next available non-negative even value. The processing stage applies De Morgan's Laws

[1] to remove the Disjunctions from the Trees and replace them with Negations and Conjunctions. The second part is to remove any double negations for efficiency. This helps the final stage of the process where the Conjunctions become AIGER "Ands" and variables Latches because there is no need to process double Negations as both stages are recursive.

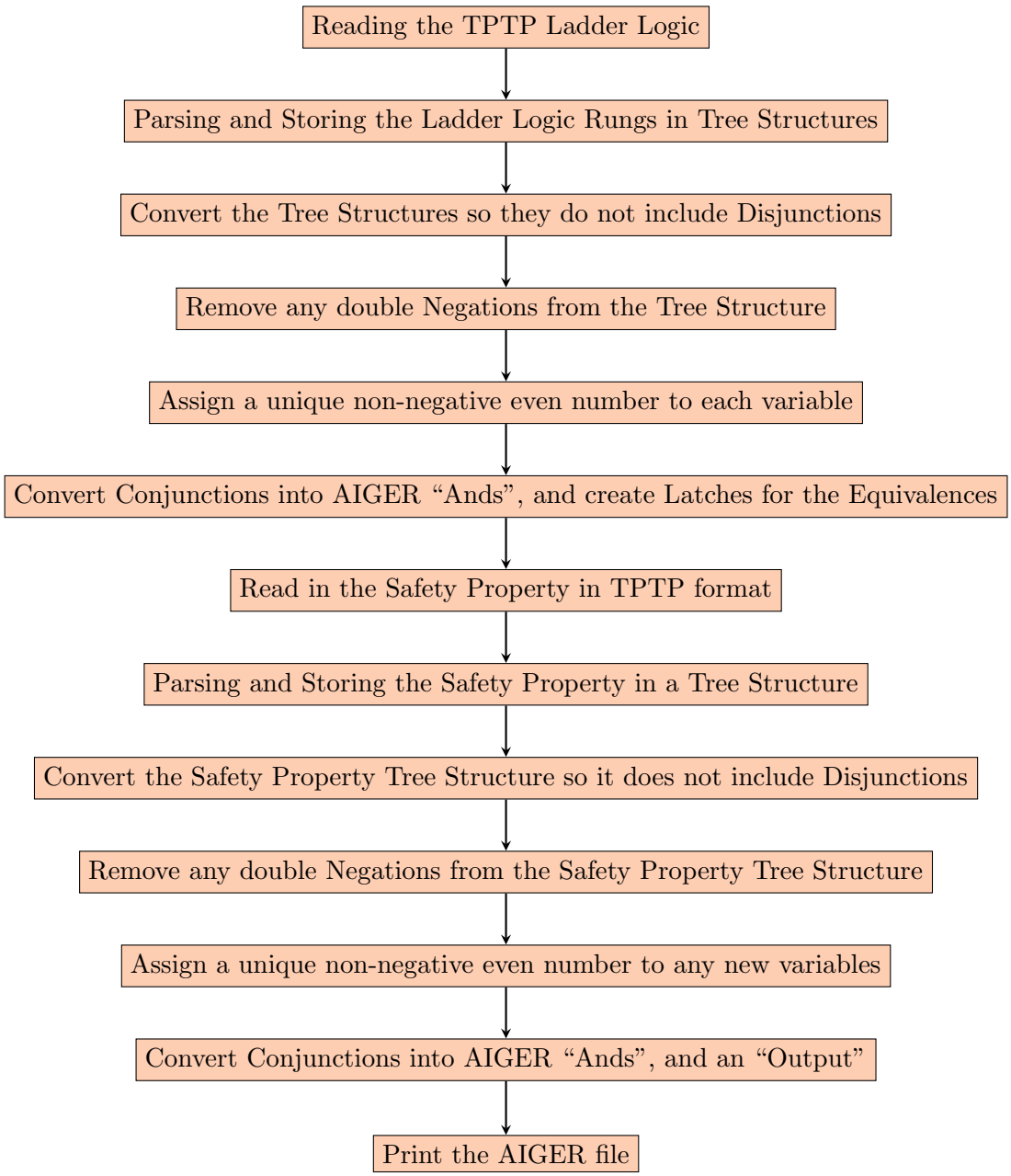


Figure 5.8: The process taken by the Aiger-fier to take in the Ladder Logic Program and a Safety Property and produce an AIGER file

The Safety Properties are considered to be the outputs of the AIGER file, however, are dealt with in the same way as in the Rungs as seen in Figure 5.8. The idea is that the formula is parsed in to the Aiger-fier in the same TPTP format so that the same parsers that read the Ladder Logic can be used and the formula can be stored in a tree. For example, the formula $a \vee b$ would be written in the TPTP format and stored a the tree (Figure 5.9) with the disjunction being the parent node.

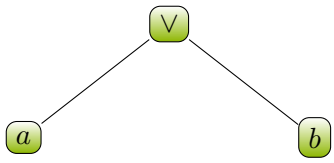


Figure 5.9: Boolean Expression Tree [13] of the expression $a \vee b$

As with the Ladder Logic, any disjunctions are converted into conjunctions using De Morgan's Laws. Any double negations are also removed at this stage. After this the formula becomes $\neg(\neg a \wedge \neg b)$. The Aiger-fier then converts the Ladder Logic to Aiger components. the same process is applied to the tree storing the Safety Property. Each conjunction becomes an Aiger "And" and the variables are replaced with their numeric values. For instance if the variable a was the first variable in the Ladder Logic it would be given the value 2. Therefore, any a in the Safety Property becomes 2, and any $\neg a$ becomes 3. The process is repeated for b and its value. This process preserves the satisfiability. If it is the case that a variable in the Safety Property is not in the Ladder Logic, it is defined as a new Input as in Section 5.3.

The final stage is to create the Aiger "Output". This is the numerical value of the conjunction built. Because the Safety Properties in Aiger are negated, this Output is the positive value because the formula is already negated. This is done so that if the output is true, then unsafe state are reached. The entire process was tested with Unit Testing and the small examples to ensure that the Safety Properties were correctly translated into Aiger. The small examples made it is easy to determine what the result of IC3 should be. For example, by manipulating the properties the translation should result in IC3 failing if the property now makes the program unsafe.

5.5 Safety Property Parser Overview

The Safety Property Parser takes a Safety Property P in the form (("A_0") -> ("B_1")) and converts it to be in format similar to the Ladder Logic Rungs in the TPTP format [4]: fof(ax,axiom, $(\sim (vA\_0)|(vB\_1))$ which allows the Tokeniser from before to be used again. This proves removes the implications, replacing it with a disjunction using the principle that the formula $a \Rightarrow b$ is equivalent to the formula $\neg a \vee b$. Again, Unit testing was conducted. The Safety Property Parser is also available on GitHub at `https://github.`

`com/HarryBryant99/SafetyPropertyParser` and has around 1,300 lines of Java code. The Safety Property Parser opperates in a similar way to the Aiger-fier, reading in the condition and storing it in a Tree Structure. It too rearranges the formulas, this time taking an Implication and producing a Disjunction - also matching the TPTP format so that the Aiger-fier's parsers can be reused. These processes were also tested using JUnit to ensure accurate translations. The Safety Property Parser acts as a extra step for the Aiger-fier, formatting the Safety Properties for it where required. For example, the properties provided by Siemens all needed translating whereas for the small artificial examples this step could be skipped.

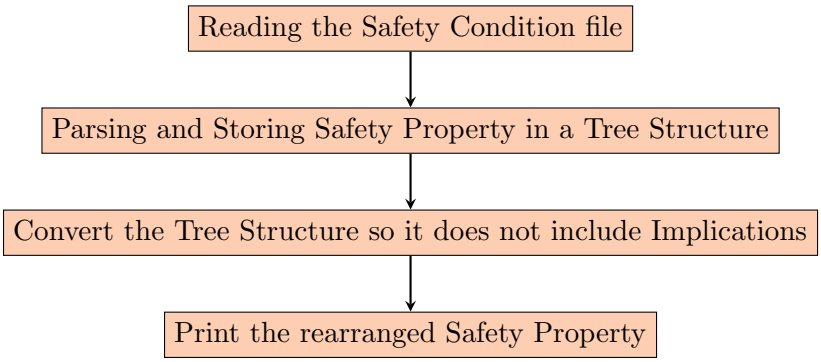


Figure 5.10: The process taken by the Safety Property Parser which takes a Safety Condition file and converts it to be without any Implications

CHAPTER 6

How Different Effects are Visible in Different Formats

Contents

Ladder Logic Programs are made up of a set of Propositional Formulae called Rungs (Figure 2.5) that describe the rules of the system and ultimately the transitions between the states. There are also Safety Properties that the Ladder Logic Programs must adhere to at all times. We are verifying that a property holds throughout a program's execution. Ladder Logic Programs can be represented in Propositional Logic [81, 76]. IC3 [43, 44, 34] requires the interlockings to be in the AIGER format [38] used for And-Inverter Graphs [12]. In this the Propositional Variables are replaced with Latches in AIGER that also store Boolean values across iterations. This chapter explain how the Ladder Logic can be represented across the different formats.

6.1 Ladder Logic Formulae in Propositional Logic

Every Rung in a Ladder Logic Program is a Propositional Formulae [81, 76] computing the value a Coil will be set to after each transition. This value is computed using the values of Inputs and values of Coils that were computed in the previous iteration. The definition from Phillip James [66] describes the principles that each Propositional Formulae in Ladder Logic must follow:

Definition 6.1 (Ladder Logic Propositional Formulae) A Ladder Logic [66] formula ψ_P (relative to a set of Input Variables I and a set of state variables C) is a propositional formula:
$\psi_P \equiv ((c'_1 \Leftrightarrow \psi_1) \wedge (c'_2 \Leftrightarrow \psi_2) \wedge \cdots \wedge (c'_n \Leftrightarrow \psi_n)$
for some $n \geq 0$, such that:

- for all $1 \leq i \leq n : c'_i \in C'$. Each c_i is an output coil (Figure 2.3c) and can be considered an output in the Ladder Logic
- for all $1 \leq i, j \leq n$: if $i \neq j \rightarrow c'_i \neq c'_j$. All coils are uniquely defined
- for all $1 \leq i \leq n : vars(\psi_i) \subseteq I \cup \{c'_1, \cdots, c'_{i-1}\} \cup \{c_i, \cdots, c_n\}$. All variables occurring on a Rung must have been previously defined either as Inputs or Coils

In the Ladder Logic [66] there are three phenomena:

1. Set of Inputs I
   - $\exists i \in ((i \in I) \wedge (i \notin C) \wedge (i \notin C'))$. A variable is an input if it does not appear as a coil.
2. Inputs are non-deterministic in choosing their next value.
   - An Input can be True or False after each transition allowing the Ladder Logic Program to display different behaviours on different executions to simulate the real-world sensors.
3. Initialisation of Inputs
   - Inputs in the Ladder Logic Program constrained for the first iteration. They are initially set to false at the beginning of the verification approach unless specified.

The example Ladder Logic Program seen previously in Figure 1.1 can be expressed as a Ladder Logic Program in TPTP format [4] used by Siemens:

```
fof(ax,axiom, vX <=> (~ vX & vY))).
fof(ax,axiom, vY <=> vY).
```

This describes the transition relation for this program. The variable x transitions to the value of $\neg x \vee y$ whilst y remains the same. In addition to this the values of the Initial States would be stated - in this case both x and y are true. The Propositional Logic allows for the Ladder Logic rules to be visualised by creating a truth table.

6.2 Automata Interpretation of the Ladder Logic

All Ladder Logic Programs can be represented as an Automaton, which helps in manually verification process, as described by Philip James [66]:

Definition 6.2 (Automaton) Given a Ladder Logic Propositional formula ψ_P over V, define an automaton
$A(\psi_P) = (S, I_s, \rightarrow)$ where

- $S = \{\mu | \mu : I \cup C \Rightarrow \{0, 1\}\}$ is the set of states
- $\mu \rightarrow \mu' if f \mu; \mu' \models \psi_P$ defines the transitions between the states
- $I_s = \{\mu' | \exists \mu : \mu \models \neg I, \mu; \mu' \models \psi_P\}$ gives the set of Initial States, where $\neg I$ expands to $\bigwedge_{i \in I} \neg i$ for all $i \in I$.

The two-variable Ladder Logic Program can be represented as an automaton (Figure 6.1), which also includes the Safety Property $x \vee y$, allowing the transitions to be visualised. False Positives [51] can also be visualised and Ladder Logic Invariant to be formed by simply marking which states the property holds in. The benefit of visualising a Ladder Logic Program as an automaton is that the system's transitions can be traced allowing the operation to be followed. Therefore, it is easy to see which are the unsafe states and where the False Positives can occur. It also therefore allows for Invariants to be checked before the verification is performed, adding an extra layer of validation to them and the processes for constructing them.

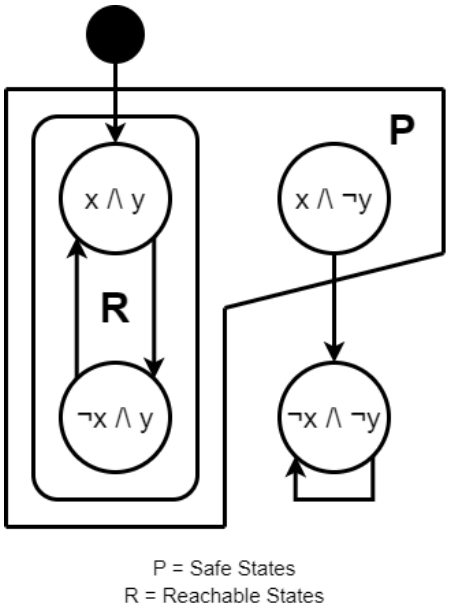


Figure 6.1: Automaton for the Two-Variable Ladder Logic Program

6.3 And-Inverter Graph Interpretation of the Ladder Logic

The IC3 implementations [43, 44, 34] IC3ref [42] and [74, 75] both utilise the AIGER file format [38] for And-Inverter graphs [12]. Ladder Logic Program can be translated into AIGER by using the transformation seen in Section 5.2 which first removes any disjunctions before recursively building conjunctions with two components. After this the numerical AIGER format can be constructed. The two-variable Ladder Logic Program becomes:

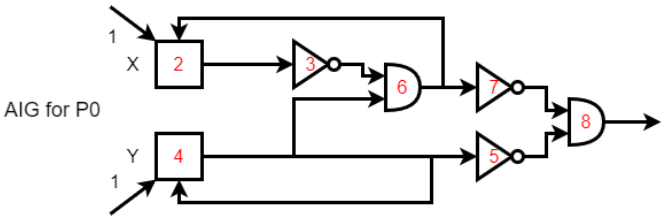


Figure 6.2: And-Inverter Graph for the two-variable Ladder Logic Program, with the Safety Property P0: $x \vee y$

Definition 6.3 (And-Inverter Graph) Given a Ladder Logic Program L, $L = \{S, I_s, \rightarrow\}$, define an And-Inverter Graph where

- Inputs are defined and are initially set to 0 - unless otherwise stated. This is achieved by constructing a Latch, defined to be initially 0 or 1, which is set to the result of a separate Input after each iteration
- Initial States declared by setting the initial values of the Latches
- Non-determinism. Inputs have a free choice giving the states in μ multiple paths

Definition 6.4 (Bubble Flow Diagram) And-Inverter Graphs are cyclic and can be shown as Bubble Flow Diagrams (Figure 6.3) $\{I, Init, L, L', P\}$ where:

- I = Input Variables that are non-deterministic
- $Init$ = The set of initial values of the Inputs and Latches
- L = A set of Latches with values before a transition has been made
- L' = A set of Latches with values after a transition. A Latch $l \in L'$ is set to a combination of latches from the set L.
- P = A Safety Property. Treated as an Output in AIGER [12] and negated so that we can prove that the unsafe states can not be reached.

After a transition, the Latches in the set L are set to the values in the set L' to create the cyclic graph. Values of the Latches (L) go into the Bubble and after a transition it produces the set of values L'. The Output of an And-Inverter graph in Ladder Logic Verification is set to the negation of the Safety Property P, which is set to values from L.

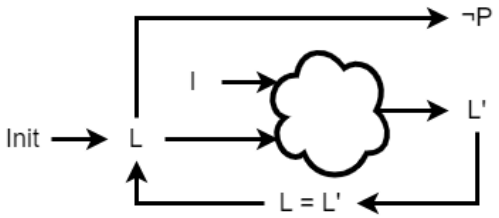


Figure 6.3: And-Inverter Graph as expressed as a "Bubble Flow Diagram" which shows how the variables are cyclic

Both And-Inverter Graphs and the Bubble Flow Diagrams allow for the values of Latches and Conjunctions to be traced through the system across each iteration as they function as a cycle. The And-Inverter Graphs can be compared to the Automaton versions to ensure that the AIGER files function with the same rules as in the original Propositional Logic by passing the values through each iteration and comparing which states are reached. The Bubble Flow Diagrams, developed in this thesis, show how the values update after each iteration where some operations take place. For this purpose it is considered to be a Black Box method where given a set of Latch values and Inputs at the start, the Bubble will produce new values for the Latches using the rules defined. This cycle continues for as long is necessary. Both the graphs and diagrams show how the Unsafe States are considered the Output. The idea being that if $\neg P$ is reached at any point in the execution then the program is unsafe. Because the two formats allow the program to be traced, for the small examples they can be checked to see if or how the program will fail.

6.4 AIGER Equivalences of the Propositional Formulae

Finally, there is the numerical representation of an And-Inverter Graph [12], AIGER format [38], used by the IC3 implementations. In the previous section the And-Inverter Graph (Figure 6.2) had the numerical values included in each component, it looks as follows:

```
aag 4 0 2 0 2 1
2 6 1                    latch 0
4 4 1                    latch 1
8                        invariant constraint 0
6 3 4                    AND gate 0      !1 & 2
8 7 5                    AND gate 1      !1 & !2
```

The AIGER file consists of:

- Two Latches (2 and 4), both set to be true initially
- An Output (8) set to the negation of the Safety Property, if reached the verification fails. This is computed with a conjunction.
- Two logical "Ands" (6 and 8):
  - One to compute the value of x after each transition. This is $\neg x \wedge y$, and therefore conjunction of the negation of value of Latch 0 and the value of Latch 1.
  - The second is to compute whether an unsafe state has been reached. The unsafe state in this in this example is $\neg x \wedge \neg y$, and therefore conjunction of the negation of value x is set to and the negation value of Latch 1.

AIGER files can also contain Inputs that are non deterministic, these are defined using an Input and Latch combination as described in Section 5.3. As these are numerical versions of And-Inverter Graphs they can be checked by constructing a graph. This can confirm that the file operates correctly by tracing the program or comparing the gates with the Propositional Logic. AIGER files can also be manipulated to confirm the properties seen in the previous formats are adhered to. For example, a Ladder Logic Program can be defined in Propositional Logic, visualised as an Automaton, traced as an And-Inverter Graph, and then using the AIGER file confirm that an Invariant could be found using IC3. By swapping the Safety Property, such that Safe States are now Unsafe, it can also confirm that one cannot be found in this case.

CHAPTER 7

Ladder Logic Testbeds

Contents

In order to determine whether the IC3 algorithm is suitable for Ladder Logic Verification, two testbeds were set up. The first consisting of small programs that have been verified manually such that IC3's results can be compared. These give confidence in the results of the second testbed, made up of Siemens Interlockings, to test whether the algorithm scales.

7.1 Artificial Ladder Logic Testbed

To ensure that the IC3 algorithm [43, 44, 34] is functioning correctly a number of artificial Ladder Logic Programs have been designed. These programs contain a small number of states which makes them ideal for testing because they can be easily visualised as an automaton, as seen in Figure 1.1. This allows Ladder Logic Invariant Candidates to be found manually by examining the automaton. Due to their size they can be manipulated to force certain scenarios, and can be tweaked to make them unsafe. When running IC3, they are useful for seeing how the state space changes as each Ladder Logic Invariant Candidate is produced, which allows for the counterexamples to be visualised and therefore the removal procedure checked. The final Ladder Logic Invariant can be visualised along with the proof to ensure everything is safe while still permitting all the required transitions.

7.1.1 Artificial Ladder Logic Interlockings

The purpose of the artificial Ladder Logic Programs is to test that the IC3 algorithm [43, 44, 34] successfully produces a Ladder Logic Invariant that will allow Inductive Verification to pass by preventing any False Positives. These examples are all small enough so they can be visualised by an automaton. The examples include Example 1 which is the two-variable program seen previously (Figure 1.1), and Example 3 which was used to demonstrate IC3 (Figure 3.1). All the artificial examples have had an Ladder Logic Invariant found manually, which will be compared to the one discovered by performing IC3 manually. By running these manual experiments it allow the result of the IC3 implementations to be checked to ensure that the result is consistent.

Artificial Program	Number of Variables	Number of States	Number of Rungs
Example 1	2	4	2
Example 2	3	8	3
Example 3	4	16	4
Example 4	2	4	4
Example 5	3	8	4
Level Crossing	6	64	6
MRes Example	4	16	4
CTG Example [62]	3	8	3

Table 7.1: Artificial Ladder Logic Programs

7.1.2 Classifying the Artificial Testbed with CASL and SPASS

The artificial Ladder Logic Programs have been designed to include a False Positive [51] which will result in the second condition of Inductive Verification [64] failing. In order to verify if a Ladder Logic Program is safe, it must be converted to a Propositional Formula [76, 81]. This can be written in CASL [78] with specifications to represent the program's transitions and Safety Properties.

Figure 7.1: Ladder Logic for the Two-Variable Program: Example 1

The two-variable program Example 1 (Figure 1.1) can be used to demonstrate this. In this, the program starts in the state where x and y are both true, and it transitions between

this and the state $\neg x \wedge y$. There are also transitions for the unreachable states: $x \wedge \neg y$ to $\neg x \wedge \neg y$ and $\neg x \wedge \neg y$ to $\neg x \wedge \neg y$. CASL Propositional Logic specifications of the Ladder Logic can be written [81, 76] allowing for verification. The Transition specification, which defines the variables and Transitions T for this two-variable program, is as follows:

spec TRANSITION =
preds x, y, x', y' : ()
• $x' \Leftrightarrow \neg\ x \wedge y$
• $y' \Leftrightarrow y$

This specification states what variables are used, along with their prime equivalents. In CASL, when a transition is made, the program moves from an unprimed state to a prime state. The process returns back to the unprimed states and can perform further transitions. Therefore, the language can be used to represent Inductive Verification [64]. Using the definition in Section 4.3.4 of Formal Methods for Software Engineering Languages, Methods, Application Domains [84], it can be done with Ψ representing the Ladder Logic Program, φ the safety property formulated over the unprimed variables (the set of states in pre-transition), and φ' the safety property formulated over the primed variables (the set of states post-transition). Allowing the CASL version to be encoded as follows:

Definition:

spec InitialStatesAreSafe = Ψ **then** • $\bigwedge_{c \in C} \neg c$ **then %implies** • φ' **end**

spec TransitionsAreSafe = Ψ **then %implies** • $\varphi \Rightarrow \varphi'$ **end**

In order to perform Inductive Verification [64] on CASL specifications [78], the broker tool Hets [77] can be used. This allows for the specifications to be translated into the input of a proof tool. In CASL, axioms are used so the specification has formulae for the Inductive Verification to test against and state the relations between the operation and predicate symbols. Each axiom determines the sentence of the CASL specification by defining the program's basic specifications and operations. The theorem prover used to disprove these proof obligations that can come from using **%implies** is SPASS [3] inside the Hets Tool [77]. The CASL Consistency Checker [69] uses calculus.

Inductive Verification [64] requires more than just the Transition specification, defining all the variables and rules used, to meet its two clauses. A specification is therefore needed to prove that the Initial State is safe in order to prove the first requirement of Inductive Verification, called SafeInitial. A second specification, StepSafe, is then needed for Inductive Verification's second condition to prove that, from a safe state, the program will move to another safe state. These specifications are combined with the Transition specification from previously to perform the proof and will look as follows:

spec SafeInitial = Transition
then • $x \wedge y$
then %**implies** • $x \vee y$ %(initial_safe)%

spec StepSafe = Transition
then %**implies** • $x \vee y \Rightarrow x' \vee y'$ %(step_safety)%

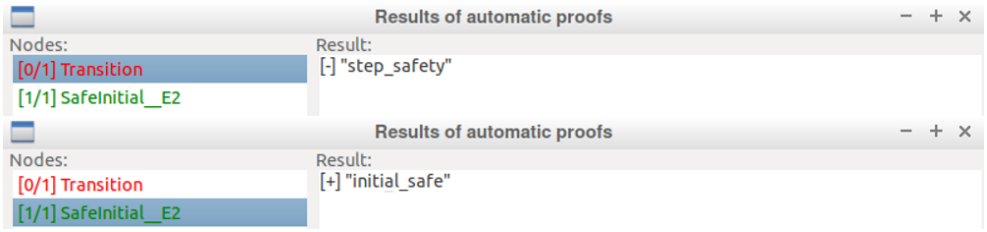


Figure 7.2: Initial CASL specification failing after the Hets proof

However, as expected these specifications fail Inductive Verification [64] (Figure 7.2) because while unreachable, there is a move from a safe state ($x \wedge \neg y$) to an unsafe state ($\neg x \wedge \neg y$). This False Positive [51] breaks Inductive Verification's second condition. However, an invariant can be used to fix this. When the formula $y \vee \neg x$ is added to the Safety Property, it will block this False Positive and allow the verification to pass as no state (reachable or unreachable) will be able to transition to an unsafe state. This Invariant can be added as a parameter to a new specification StepSafeWithInvariant. Now a transition can only take place if both the Safety Property ($x \vee y$) and the Ladder Logic Invariant hold. The new specifications are:

spec StepSafeWithInvariant = Transition
then %**implies** • $(x \vee y) \wedge (y \vee \neg x) \Rightarrow x' \vee y'$ %(step_safety_with_invariant)%

When these specifications are combined with the ones seen earlier, and run through the Hets tool [77], the previous StepSafe specification fails but the new specification with the Ladder Logic Invariant passes, removing the False Positive issue from before (Figure 7.3). This approach has been applied to larger examples to demonstrate how these Ladder Logic Invariant formulas solve the over-approximation that occurs.

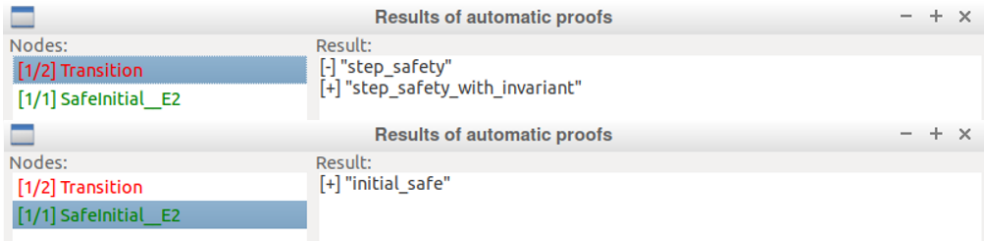


Figure 7.3: CASL specification now passing the Hets proof because of the Ladder Logic Invariant that blocks the False Positive

All the artificial Ladder Logic Programs have been checked in this way (Table 7.2) to see whether they are safe when Inductive Verification is performed. All but the Level Crossing Example failed Inductive Verification because a False Positive is present - the Level Crossing was safe. A Ladder Logic Invariant was found for all these examples that previously failed, allowing for Inductive Verification to pass when performed with the Ladder Logic Invariant.

Artificial Program	**Inductive Verification Passes?**	**Ladder Logic Invariant can be found?**	**Inductive Verification Passes with Invariant?**
Example 1	✗	✓	✓
Example 2	✗	✓	✓
Example 3	✗	✓	✓
Example 4	✗	✓	✓
Example 5	✗	✓	✓
Level Crossing	✓	N/A	N/A
MRes Example	✗	✓	✓
CTG Example [62]	✗	✓	✓

Table 7.2: Artificial Ladder Logic Programs and whether Inductive Verification was successful with and without a Ladder Logic Invariant

7.2 Siemens Railway Interlocking Testbed

Whilst the artificial Ladder Logic Programs are suitable for checking for correctness, in order to prove the IC3 algorithm improves the efficiency of the Ladder Logic Verifier [87, 51] it needs to be scalable. With the artificial programs, it has proven IC3 can successfully find a Ladder Logic Invariant. A testbed of interlockings from Siemens will be used to prove it can also be applied on an industrial scale. These are too large to visualise as an automata so cannot be manually verified. However, the existing Ladder Logic Verifier [87, 51] can be run on these interlockings to determine which Verification Tasks are known to pass and fail, allowing for a comparison with IC3.

7.2.1 Structure of the Siemens Railway Interlocking Testbed

In order to test the efficiency and the correctness of IC3 [43, 44, 34] on an industrial scale, test interlockings were obtained from Siemens Mobility. The goal is to test both the original IC3 [42], developed by the creator Bradley, and the improved ABC IC3 implementation [74, 75] to determine how effective the IC3 algorithm is at verifying Ladder Logic Programs. The algorithm has been proven to be successful for hardware verification and testing on the small artificial examples proved a Ladder Logic Invariant can be found that will allow Inductive Verification [64] to pass. The next step was to ensure this is still the case for the industrial-scale railway interlockings [10] from Siemens (Table 7.3) to determine that IC3 is improving the efficiency of its Ladder Logic Verifier [87, 51].

Interlocking	**Description**	**Safety Properties**
DesignX	Real-world interlocking on the North Wales Coastline	48 out of 300 "Generic Safety Properties" [79] groups developed from a Siemens Internal Document containing properties relevant to what each group is protecting
Interlocking A	Siemens' internal artificial track plan and interlocking computer designs developed with a view to challenge interlocking designs	5 properties adapted to 1278 propositional formulas for Interlocking A
Interlocking B	Siemens' internal artificial track plan and interlocking computer designs developed with a view to challenge interlocking designs	5 properties adapted to 636 propositional formulas for Interlocking B

Table 7.3: The Siemens Railway Interlockings

Three industrial interlockings [10] were provided by Siemens, details on each interlocking can be found in Table 7.4. Two are an artificial track plan, developed by Siemens for testing and to challenge how Siemens develop their interlockings. Interlocking A is a terminus station with sidings and a branch. Interlocking B is another terminus station but this time with just a single line. The Safety Properties for these are grouped into five categories:

- Route set and locked
- Train detection indicates clear
- Indicators proved
- All signals on
- Approach Locking

There are also properties for test Interlockings A and B used by Siemens for testing purposes in the development of the Ladder Logic Verifier [87, 51] included in the testbed. The final interlocking, DesignX, is a real-world interlocking from the North Wales Coastline and is considerably larger than the test Interlockings. In total there are 300 "Generic Safety Properties" (GSPs) [79] that categorise the Propositional Safety Property formulas. For DesignX, we received 48 of these, with each group containing between 1-30 formulas. Our testbed for this interlocking consisted of 714 Propositional Safety Property formulas.

Interlocking	**Rungs**	**Variables**	**Propositional Safety Formulae**
DesignX	11922	74857	714
Interlocking A	8460	53165	1278
Interlocking B	8460	53165	636

Table 7.4: Summary of the Siemens Railway Interlockings

Each railway interlocking has a different number of Propositional Safety Property formulas [81, 76]. This is due to the nature of each track plan consisting of varying numbers of points, signals and routes that trains can take. For example, a small station with a passing loop on a single track line would only have a set of points at each end and few signals. Whereas a larger station with multiple platforms and sidings would have more points and therefore more signals to control the routes. Every signal in a track plan will have its own propositional formula for safety and can be written in first order logic as shown in Figure 7.4 which describes how movement authorities are assigned.

$$\forall(s \in Signal)$$
$$(not(proceed(s)) \wedge (proceed'(s)) \Rightarrow (approachLocked(s))$$

Figure 7.4: An example of a generic formula
(From Werner's MRes thesis [92])

After discussions with Siemens it was discovered it is usual to represent the Propositional Safety Properties of interest in its negated form ($\neg P$) during the testing process with the Ladder Logic Verifier [87, 51] as proven by James [66]. In the artificial Ladder Logic Programs the properties were always designed to be the set of Safe States, with the idea being that the verification approaches would take these Safety Properties and prove whether the example was indeed safe. However, the Siemens approach is to formalise the Unsafe States, although their verification tools are still proving that the Interlockings are safe. A reason for doing this is that with 2^n number of states if the Unsafe State is simply a few states it is more efficient (and easier to read) to represent just these states with the property, rather than representing all the other states that are safe. Therefore, it means when running IC3, the Safety Properties also need to be negated to match their approach.

7.2.2 Classifying the Testbed with the Ladder Logic Verifier

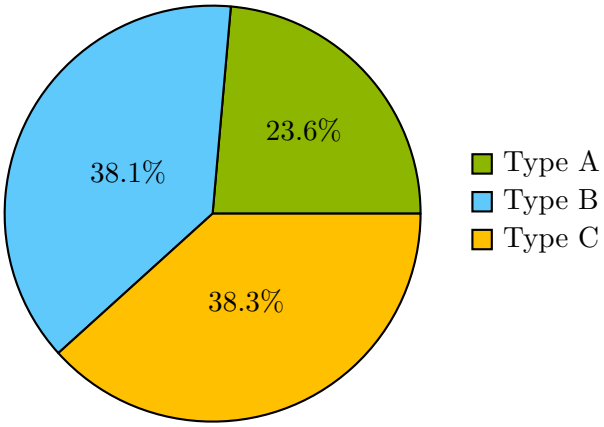


Figure 7.5: Distribution of the 2628 Verification Tasks across the three types after running the Ladder Logic Verifier

The Verification Tasks from the Siemens data each needed a label assigned so groups of each type could be formed. To achieve this, each task was tested using the tool Clausegen, developed by Phillip James, with Inductive Verification [64] initially and then Bounded Model Checking [55, 30, 54, 66]. The Ladder Logic Verifier [87, 51] uses Clausegen to generate files for Inductive Verification and Bounded Model Checking that can be checked with Z3 [72, 40]. Overall, there are 2628 Verification Tasks (Figure 7.5). 714 are from the real-world interlocking DesignX, however not all of the Abstract Safety Properties are represented. There are 1278 from the Interlocking A interlocking and 636 from Interlocking B. The Verification Tasks for the artificial test interlockings came from top level properties from a Masters project by Tom Werner [92] along with additional test properties used by Siemens in the development of the Ladder Logic Verifier [66, 51]. It must be mentioned that the interlockings provided by Siemens are early versions prior to being applying the Ladder Logic Verifier - in a state where only testing has been applied. These have since been refined before use in controlling trains. It is also the case that a number of properties are bound to fail verification as they concern the boundary of the network. In these cases successful verification can only be achieved by collaboration between two different interlockings - as we consider here only single interlockings in isolation, they cannot be proven to hold.

The Testbed is classified into Types A, B and C (Figure 7.5) using the current verification approach (Figure 2.1). Of the 2628 Verification Tasks, 23.6% are of Type A meaning they pass Inductive Verification [64]. 38.1% are of Type B which means they fail Inductive Verification and Bounded Model Checking [55, 30, 54, 66] finds a counterexample. Finally, the remaining 38.3% are of Type C. These are the examples of most interest because they are the ones that fail Inductive Verification, but crucially Bounded Model Checking has deemed safe as Bounded Model Checking was unable to find a counterexample after 100

bounds. We chose 100 bounds after correspondence with Siemens, these tasks are therefore undecided because a counterexample could still be present but is undiscovered. This is where IC3 [43, 44, 34] is being investigated to see if it can evaluate Type C. A breakdown of the labels can be found in Table 7.5, with further analysis found in Sections 7.2.2.1 and 7.2.2.2. All 2628 Verification Tasks were run with Inductive Verification initially, and only those that failed would then be run with Bounded Model Checking. Running Inductive Verification resulted in 2008 Verification Tasks that failed the check (Type B and Type C). Appendix C contains a classification for all the Verification Tasks.

Interlocking	**Verification Tasks**	**Type A**		**Type B**		**Type C**	
DesignX	714	351	49.2%	138	19.3%	225	31.5%
Interlocking - A	1278	269	21.1%	228	17.8%	781	61.1%
Interlocking - B	636	0	0.0%	636	100.0%	0	0.0%
Total	2628	620	23.6%	1002	38.1%	1006	38.3%

Table 7.5: Summary of the Siemens Testbed across the three verification types

7.2.2.1 Running Inductive Verification on the Testbed

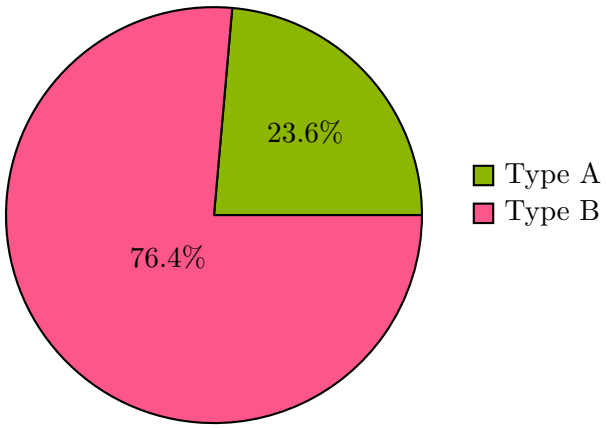


Figure 7.6: Distribution of the 2628 Verification Tasks from the Siemens Testbed after Inductive Verification

The distribution of the Verification Tasks across the three types was produced using Inductive Verification [64] initially to establish which tasks are safe, Type A, and which fail either due to a mistake (Type B) or are of Type C where they either contain a False Positive or the mistake has not been found. This approach only involves two calls to a SAT-Solver, [19], one to check whether the Initial States I are safe, and the other to check whether all safe states transition to another safe state. Because there are only two calls to a Sat-Solver it makes Inductive Verification quick in terms of runtime. Those Verification Tasks of Type

A are those that have passed Inductive Verification because both calls to the SAT-Solver are unsatisfiable, these made up 23.6% of the testbed. Both Type B and Type C have failed Inductive Verification which means at least one of the calls to the SAT-Solver is satisfiable. These tasks that have failed are those that should be run with Bounded Model Checking [55, 30, 66, 54].

7.2.2.2 Running Bounded Model Checking on the Testbed

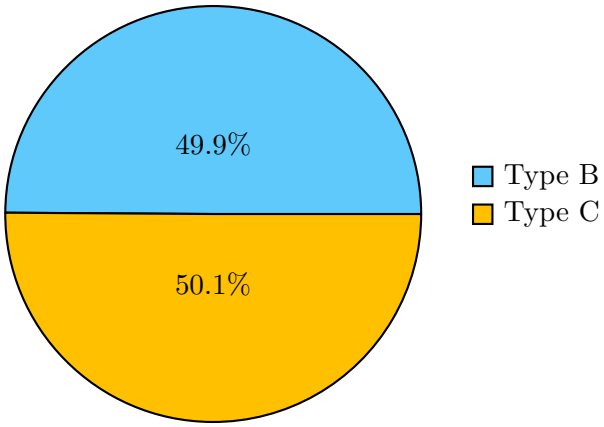


Figure 7.7: Distribution of the 2008 Verification Tasks that previously failed Inductive Verification after Bounded Model Checking was performed

Bounded Model Checking [55, 30, 66, 54] determines if a counterexample is present in a Verification Task. The counterexample must however be found within k bounds. In this case 100 bounds were chosen to match Siemens. Those Verification Tasks that passed Inductive Verification (Type A) do not need to be run with Bounded Model Checking because they have already been proven to be safe. With Bounded Model Checking, safety can only be ensured for k bounds, and it is computationally more expensive than Inductive Verification and is the reason why that is performed first. 100 bounds are chosen to reduce the overall runtime of Bounded Model Checking and, in practice, the majority of the counterexamples are found within a small number of bounds. However, it only guarantees safety up to 100 transitions, which is nothing compared to the number of transitions in the Ladder Logic Programs. The assumption here being that with 2^n number of states and the interlockings having thousands of variables, it results in more transitions than can be feasibly checked using Bounded Model Checking. The 2008 tasks that failed Inductive Verification were run with Bounded Model Checking (Figure 7.7) resulting in 49.9% being Type B where a counterexample is found within the 100 bounds. The remaining 51.1% (1006) were Type C where a potential counterexample is yet to be discovered and are considered undecidable.

7.2.2.3 Why so many Verification Tasks are unsafe

It is worth noting the interlockings provided by Siemens are early versions of the Ladder Logic Programs before they had been tested on the Ladder Logic Verifier [87, 51] containing bugs that have since been fixed for real-world operation. Sometimes Siemens engineers write Ladder Logic Programs so they are optimised for runtime. The cost of the optimisation are spurious variables which are wrong for one cycle, however, correct for all following cycles. However, it is unclear to what extent the full effect of this error can be, it could change a set of points by mistake. The boundaries of the controlled region of the track plan can also lead to mistakes where they are correct with assumptions made on the environment. As we make no assumptions on the environment, many mistakes could be found. The Siemens Engineers have since refined the interlocking and the way they are developed in order to ensure safety before use in the real-world. This has been achieved partially through Bounded Model Checking's [55, 30, 66, 54] ability to find and visualise counterexamples which provides an insight into where and why the Ladder Logic Program fails.

Three interlockings were provided by Siemens (Table 7.4). One of these was a real-world interlocking located on the North Wales Coastline and the other two came from the fictional track plans. However, running Inductive Verification [64] for the entire testbed only produced 620 Verification Tasks that were deemed safe. Bounded Model Checking [55, 30, 66, 54] found a counterexample was present for 1002 Verification Tasks, with the remaining 1006 tasks that were classified as Type C. Even if all of these failed Inductive Verification because a False Positive was present and IC3 finds an Ladder Logic Invariant, deeming them safe, it only results in 1002 of the 2628 Verification Tasks being unsafe - 38%. However, potentially, not all of those classified as Type C will have failed Inductive Verification because of a False Positive, and there is a counterexample that was not discovered in 100 bounds, so this percentage could drop depending on the interlocking. By breaking it down per interlocking, and assuming again that all of Type C are safe, Interlocking A is unsafe for 18% of its Safety Properties. For Interlocking B, all 100% of the Safety Properties would fail because of counterexamples - although this is just a test interlocking. However, for DesignX (an early version of a real-world interlocking) 19% of the Safety Properties will fail, but again this percentage could be larger. This 19% fail rate is an issue because this railway interlocking is a Safety Critical System that needs to be verified for safety. However, this interlocking has since been developed further by Siemens.

7.2.2.4 The stability of Type C

In order to demonstrate the need for an alternative to Bounded Model Checking to prove Ladder Logic Programs are safe, Figures 7.8 and 7.9 show how increasing the number of bounds does not immediately result in a drop in the number of Verification Tasks of Type C, whilst the runtime of increases exponentially. This increase is ultimately why adding to the bounds is not viable in practice because, in order to include all the transitions available in the Ladder Logic Program, the number of bounds becomes impractical with the resulting runtime. As with the previous testing with the Ladder Logic Verifier, a version

of its Clausegen implementation has been used to obtain the classifications and runtimes. Originally, Bounded Model Checking [55, 30, 54, 66] was performed with 100 bounds to decide the Siemens Testbed which classified 1006 Verification Tasks of Type C that are considered undecidable. In theory, by increasing the number of bounds would reduce this if counterexamples are present but this comes with a cost. When Bounded Model Checking was performed with 100 bounds it took on average 132 seconds to decide Type C. Increasing this to 200 bounds took on average 444 seconds, but still returned all 1006 tasks as Type C (Figure 7.8). Increasing the number of bounds to 300 also returned all of the sample Verification Tasks as Type C, this time taking on average 1042 seconds. A sample of 120 tasks was chosen because of this. These large runtimes are why Siemens choose to perform Bounded Model Checking with 100 bounds because, in reality, many of the counterexamples are found early. However, it leaves the process incomplete as the Ladder Logic Program is only safe up to 100 transitions. Between 100 and 300 bounds it can be seen that the percentage of the Verification Tasks categorised as Type C remains the same (Figure 7.8). It means that any counterexample present can only be discovered after at least 301 transitions - and that the task is safe for 300 transitions. The issue with this is, in reality, 300 transitions is not very long in real time and there is a need to have completeness in the verification of a Safety Critical System - otherwise there is little point in verifying for safety. This is where IC3 comes in as it can decide Type C by producing a Ladder Logic Invariant.

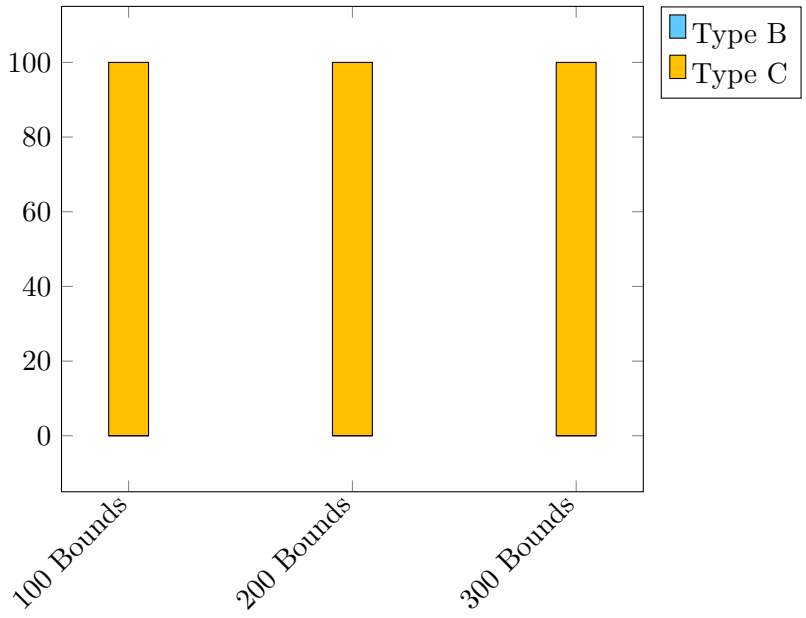


Figure 7.8: Percentages of Type B and Type C after Bounded Model Checking was performed with k bounds on a sample of 120 Verification Tasks previously of Type C. 300 bounds was chosen to be the upper limit because of the large runtimes

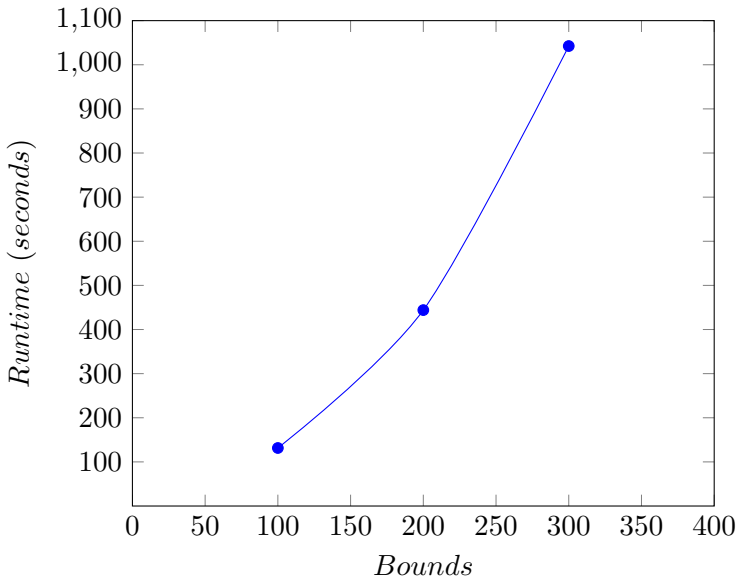


Figure 7.9: Runtimes for performing Bounded Model Checking on the 1006 Verification Tasks of Type C with 100 and 200 bounds, with a sample of 120 Verification Tasks of Type C performed with 300 bounds

CHAPTER 8

Testing the Aiger-fier

Contents

In order to prove that IC3 [43, 44, 34] is suitable for Ladder Logic Verification it must be correct so that the results produced by IC3ref [42] and ABC [74, 75] can be trusted. As both IC3 implementations have been developed elsewhere it is assumed they have been tested in depth and therefore are correct by use. This can be supported by testing the solutions by manually converting the artificial Ladder Logic Testbed (Table 7.2) into AIGER - the file format used by both IC3 implementations. As the examples are small, they can be visualised as And-Inverter Graphs [12] to check that what is produced is correct and can be manipulated to produce different results from IC3. For example, if a Ladder Logic Program passed IC3, inverting the Safety Property P should result in it failing. This is also useful for checking the process of converting the Ladder Logic [66] functions correctly.

The second tool used for the process of Ladder Logic Verification with IC3 is the Aiger-fier [45] which takes the Ladder Logic Program with P and produces an equivalent And-Inverter Graph. The manual translations can be constructed in the same way as the Aiger-fier and can be checked by visualising the corresponding And-Inverter Graph. Unit tests [20] can be used to ensure all the components that make up the Aiger-fier are translating the Ladder Logic correctly. Finally, the results from IC3 are compared with the results produced by the Ladder Logic Verifier [87, 51] to determine if there are any misclassifications from the Aiger-fier. It is still possible there is a mistake in the Ladder Logic Verifier, but as this has been proven correct by use, it is more likely that the Aiger-fier is at fault.

8.1 Manual experiments with the Artificial Testbed

The Artificial Testbed will be used increase trust in the classification of the Siemens Testbed.

8.1.1 Testing IC3

Artificial Program	IV Passes with Invariant?	IC3Ref Passes?	ABC Passes?
Example 1	✓	✓	✓
Example 2	✓	✓	✓
Example 3	✓	✓	✓
Example 4	✓	✓	✓
Example 5	✓	✓	✓
Level Crossing	✓	✓	✓
MRes Example	✓	✓	✓
CTG Example [62]	✓	✓	✓

Table 8.1: Running IC3ref [42] and ABC [74, 75] on the Artificial Testbed to confirm that the results from IC3 match the results of building an Invariant manually

To check that the Aiger-fier, and IC3, functions as expected the artificial examples can be used as it has been proven a Ladder Logic Invariant allows Inductive Verification [64] to pass (Table 8.1). These Invariants were found by examining the automatons. Next manually performing IC3 to ensure the results match. The IC3 implementations should then also state an Invariant can be found. Editing the Safety Property P can force the program to be unsafe where a Ladder Logic Invariant cannot be found. IC3 should fail for these examples. For all Ladder Logic Programs in the Artificial Testbed both IC3ref [42] and ABC [74, 75] found a Ladder Logic Invariant (Figure 8.1a) (Figure 8.1c). This was expected as manually found Ladder Logic Invariants blocked the False Positive in Inductive Verification.

Artificial Program	IC3Ref		ABC	
	P	$\neg P$	P	$\neg P$
Example 1	✓	✗	✓	✗
Example 2	✓	✗	✓	✗
Example 3	✓	✗	✓	✗
Example 4	✓	✗	✓	✗
Example 5	✓	✗	✓	✗
Level Crossing	✓	✗	✓	✗
MRes Example	✓	✗	✓	✗
CTG Example [62]	✓	✗	✓	✗

Table 8.2: Correctness from running the IC3 implementations on the Artificial Testbed. A Ladder Logic Invariant can be found for the Safety Property P, negating it prevents this

It was important to test that IC3 implementations would state a Ladder Logic Program is unsafe and that a Ladder Logic Invariants cannot be produced in this case. To do this each example in the Testbed was run again with both IC3 implementations, however, this time the Safety Property P is inverted, therefore making all safe states unsafe. Table 8.2 shows that for all P a Ladder Logic Invariant could be produced and the programs are safe, but for $\neg P$ it fails (Figure 8.1b) (Figure 8.1d).

```
ctg.aag
0, IC3 says a Ladder Logic Invariant can be found

example1.aag
0, IC3 says a Ladder Logic Invariant can be found
```

(a) Testing two of the Ladder Logic Programs with their Safety Properties P on IC3ref

```
ctg_notP.aag
1, IC3 says the Ladder Logic Program is unsafe

example1_notP.aag
1, IC3 says the Ladder Logic Program is unsafe
```

(b) Testing two of the Ladder Logic Programs with the negative of their Safety Properties P on IC3ref to force IC3 to fail

```
ctg.aig
Invariant F[1] : 2 clauses with 2 flops (out of 3) (cex = 0, ave = 2.67)
Verification of invariant with 2 clauses was successful.  Time =     0.00 sec
Property proved.  Time =     0.01 sec

example1.aig
Invariant F[1] : 1 clauses with 1 flops (out of 2) (cex = 0, ave = 1.00)
Verification of invariant with 1 clauses was successful.  Time =     0.00 sec
Property proved.  Time =     0.01 sec
```

(c) Testing two of the Ladder Logic Programs with their Safety Properties P on ABC

```
ctg_notP.aig
Output 0 of miter "ctg_notP" was asserted in frame 0.  Time =     0.01 sec

example1_notP.aig
Output 0 of miter "example1_notP" was asserted in frame 0.  Time =     0.01 sec
```

(d) Testing two of the Ladder Logic Programs with the negative of their Safety Properties P on ABC to force IC3 to fail and no Invariant produced

Figure 8.1: Running IC3 on the Artificial Testbed to show how IC3 finds a Ladder Logic Invariant for their Safety Properties P. However, when the Safety Property is negated, making the safe states unsafe, IC3 fails and an Invariant cannot be found

8.1.2 Experiments based on handcoded And-Inverter Graphs

In order to test the IC3 implementations (Table 8.2) And-Inverter graphs [12] can be drawn from the AIGER [38] files and then traced to ensure the Ladder Logic Programs are cor-

rectly translated. The idea being that for the small examples the Safety Properties can be manipulated to force a failure and IC3 should match these results for the corresponding AIGER files. For example, the two-variable example (Figure 1.1) can be converted as seen in Figure 6.2. Both IC3ref [42] and ABC [74, 75] state that a Ladder Logic Invariant can be produced for this graph and therefore the program is safe.

The Safety Property x can also be used (Figure 8.2) which is not safe because the program alternates between the states $x \wedge y$ and $\neg x \wedge y$. In this case the graph has the output 7 which is reached and ultimately causes IC3 to fail because it cannot include all the reachable transitions in an Invariant without becoming unsafe. This can be checked by running Inductive Verification, then tracing the execution using the automaton or the And-Inverter Graph to show that an unsafe state can be reached and therefore a Ladder Logic Invariant cannot be found. This is indeed the case for both IC3ref and ABC with both failing.

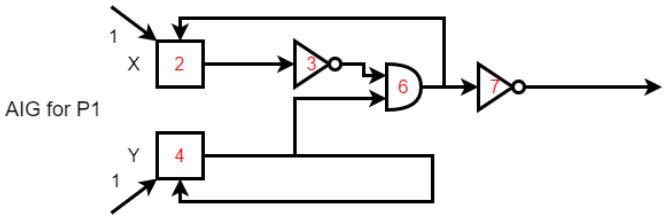


Figure 8.2: And-Inverter Graph Example 1, with the Safety Property P1: x

Another Safety Property $\neg y$ (Figure 8.3) will also result in IC3 failing because the program is not safe. The IC3 implementations again match this which, in turn, also provides trust in the method of converting the Ladder Logic Programs into AIGER.

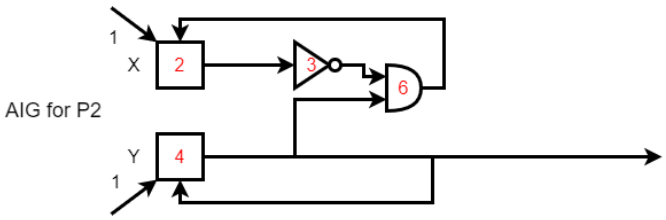


Figure 8.3: And-Inverter Graph Example 1, with the Safety Property P2: $\neg y$

8.2 Unit Testing with Transformation Components

Each component of the Aiger-fier [45] has a set of Unit Tests [20] that can be used to ensure they are producing correct results. Therefore each component of the program has been broken down and tested separately. The full set of test cases can be found in the GitHub repository (`https://github.com/HarryBryant99/Aiger-fier`).

```
@Test
public void test1(){
    String data = "fof(ax,axiom, vA <=> vB)";

    Ladder sourceL = new Ladder();
    sourceL.addRung(new Rung(new Equivalence(new Proposition( name: "vA"),new Proposition( name: "vB"))));

    // TODO: Calculate real expected result
    Aig expectedAig = new Aig();
    expectedAig.addComponent(new Latch( id: 2, computed: 4, original: 0));
    expectedAig.addComponent(new Latch( id: 4, computed: 4, original: 0));

    AigerLadderTransformation tt = new AigerLadderTransformation( initalVariableValues: null);
    assertEquals(expectedAig, tt.convertLadder(sourceL));
}
```

Figure 8.4: Example JUnit [2] test to check the components of the Aiger-fier function correctly using a manually computed result to be compared to the result of the program

The checks are a set of JUnit tests [2] (Figure 8.4) which take source data and uses program to produce an output. This output is compared to manually translating the source data. The principle being that the manual translations are small enough so that is easy to verify them as correct so there is the trust in the comparison. The example in Figure 8.4 the formula $vA \Leftrightarrow vB$ is being passed and translated into AIGER. The manual translation produces two Latches, one for vA which is set to the Latch vB - which is in turn set to itself. The Aiger-fier produces the same result therefore the test passes. This process has been repeated for all aspects of the parsing, translating, and printing processes. Each unit has been tested as part of a form of Integration Testing [28] in order to build up the program and ensure that the components together work correctly. Further testing methods could be applied in order to increase the confidence in the Aiger-fier. This could focus more on the performance of the Aiger-fier and ensuring that the translation is correct as the Ladder Logic Programs increase in size. To do this larger examples could be designed that are still simple enough to still be validated manually, however, this is time consuming to do so.

8.2.1 File Parser

There are multiple processes involved in reading in the components of φ. Each has its own individual process but for both the Ladder Logic and the Safety Property the Parser and Tokeniser classes are utilised to break up the expressions into trees [13].

8.2.1.1 Input Propositions

The class “InputPropositions” takes a file consisting of a list of variable and value pairs, placing them in a hashmap. The Aiger-fier has been coded in Java and uses the predefined data types where possible. This hashmap specifies the initial values of the Inputs and Latches from the Ladder Logic Program. The Input file can be left blank, any variables not included will be set initially to 0. The table below is an example of how two variables in the Ladder Logic Program can be initialised to true.

Input : List $[I]$
Output: Hashmap (I')

Input	Output
$vX = 1$ $vY = 1$	{vX=1, vY=1}

8.2.1.2 Ladder Parser

The "LadderParser" class is used to read in the Ladder Logic Program. It takes each element of the list of the form $fof(ax, axiom, v' <=> e)$ and initially trims it to $v' <=> e$ where v' is the coil variable and e (a Propositional Formula [81, 76]). This allows for the Parser and Tokeniser to break it down further into a tree [13]. This tree is stored as a *Rung* and the collection of Rungs read form the Ladder. The table below shows one of the Unit test cases for how a formula in the TPTP format is read in by the first stage in of the Aiger-fier transformation to be stored as an a Rung, this is stored in an Array List.

Input : List $[x' \Leftrightarrow f']$
Output: List $[Rung]$

Input	Output
$\{fof(ax, axiom, vX <=> (\sim vX \& vY))\}$	$Ladder\{rungs =$ $[Rung\{(vX <=> (\sim vX \& vY))\}]\}$

8.2.1.3 Safety Condition Parser

The "SafetyConditionParser" class reads in the Safety Property P and it also needs trimming from the form $fof(ax, axiom, P)$ and to P. From this the Parser and Tokeniser can be used to break it down further into another tree [13]. Below shows an example test case, the result is stored in an Array List because it enabled the same parsers to be used for simplicity. The Saftey Properties are formulated by the Safety Property Parser [47] which converts the formulas into a format similar to the TPTP format [4]. The idea behind this is that it removes any implications from the formulae, leaving them in a state where the parsers can be reused.

Input : P
Output: List $[SafetyCondition]$

Input	Output
$fof(ax, axiom, (vX \mid vY))$	$SafetyCondition\{conditions =$ $[\sim (vX \mid vY)]\}$

8.2.1.4 Parser

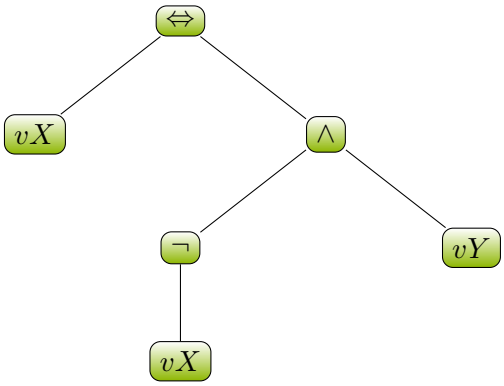


Figure 8.5: Boolean Expression Tree [13] of the expression $vX <=> (\sim vX \& vY)$

The "Parser" is used to read in both the Ladder Logic and the Safety Property and relies on the "Tokeniser". The purpose of the Parser is to break down an expression e to form a tree representation t such that $e \equiv t$. The expression $vX <=> (\sim vX \& vY)$ can be seen below in Figure 8.5. The Parser begins by evaluating the left-hand side of the equation calling the Tokeniser to read each character to form propositions. It then checks to see if there are any conjuctions, disjunctions or negations present on the left-hand side. Once complete, the Parser checks to see if an equivalence is present as all the Ladder Logic expressions should be a variable followed by an equivalence. If found, then the Parser repeats the process used for the left-hand side on the right-hand side to build the final expression. When a new component is discovered, a new instance of that component is instantiated. The process creates a tree-like structure (Figure 8.5) [13] allowing each component of the expression to be accessed recursively. Note, in the table below the outputs of the test cases are stored in a tree structure as seen in Figure 8.5 but for simplicity the bracketed expression is shown.

Input : Expression e
Output: Boolean Expression Tree e

Input	**Output**
va	va
$\sim va$	$\sim (va)$
$va \& vb$	$(va \& vb)$
$va \mid vb$	$(va \mid vb)$
$va <=> vb$	$(va <=> vb)$
(va)	va

$va\|vb$	$(va\|vb)$
$\sim\sim va$	$\sim(\sim(va))$
$va\&vb\&vc$	$((va\&vb)\&vc)$
$(vX <=> (\sim vX\&vY))$	$(vX <=> (\sim(vX)\&vY))$

8.2.1.5 Tokeniser

The Tokeniser is used by the Parser to read in the characters that make up the Ladder Logic. It has methods in place to read a character at a time and then to determine whether the sequence is legal. Once it has read in all the variables and operators it returns the final list containing the components of the expression as strings. The test cases below show how the components of the formulas are broken down by the Tokeniser into the Strings.

Input : String s
Output: List[String] l

Input	**Output**
$vavbvcvd$	$\{va, vb, vc, vd\}$
$va\|\& <=> \&vb \sim vc()$	$\{va, \|, \&, <=>, \&, vb, \sim, vc, ()\}$

8.2.2 De Morgan Transformation

In order to convert the Ladder Logic data into the AIGER format it needs translating - a reminder that the formulas do not contain any implications. The AIGER format [38] only permits Latches, a circuit which holds a value until reset, Negations and logical Ands - each "And" is the conjunction of two values. Importantly, there are also no logical "Ors", therefore the translation needs to do the following:

1. Use De Morgan's Laws [1] to convert any "Ors" to "Ands"
2. Remove any double negations (for efficiency)
3. Split any conjunctions with more than two clauses into sub-conjunctions

To remove any disjunctions each Boolean Expression Tree contains a variant of the method "cloneWithoutDisjunctions" which takes an expression e, in the tree format, and returns e' so that $e \equiv e'$ with the disjunctions converted. Any double negations are then removed by the method "cloneRemovingDoubleNegation" which takes e' and returns e'', e' containing no disjunctions whilst e'' has no disjunctions and no double negations. In order

to complete the transformation the method "splitEquivalence" breaks down each expression into the left and right-hand side of the equivalence. The method "splitExpression" breaks down the right-hand side where required. The result from "splitExpression" is a list of "Ands" (when further breakdown is necessary) and a final result. A latch is built with the proposition from the left-hand side and its value the final result produced by "splitExpression". A set of Unit test cases have been included for each section showing how the formulas are manipulated ready for the final translation into AIGER [38].

8.2.2.1 Clone Without Disjunctions

In order to represent the Ladder Logic in AIGER [38] all the disjunctions need to be converted into conjunctions using De Morgan's Laws [1]. The method duplicates the Boolean Expression Tree e and edits the new version e' so that it doesn't contain disjunctions.

Input : Boolean Expression Tree e
Output: Boolean Expression Tree e'

Input	Output
va	va
$\sim (va)$	$\sim (va)$
$(va \& vb)$	$(va \& vb)$
$(va \mid vb)$	$(\sim (\sim (va) \& \sim (vb)))$
$((va \& vb) \& vc)$	$((va \& vb) \& vc)$
$((va \mid vb) \mid vc)$	$(\sim ((\sim (va) \& \sim (vb)) \& \sim (vc)))$

8.2.2.2 Removing Double Negations

The next stage of the transformation is to remove any double negations, this is to make the Boolean Expression Trees more efficient and ease the transformation into AIGER. This method recursively works through the levels of e' to limit the negations to, at most, one where appropriate to form e''.

Input : Boolean Expression Tree e'
Output: Boolean Expression Tree e''

Input	Output
va	va
$\sim (va)$	$\sim (va)$
$\sim (\sim (va))$	va
$\sim (\sim (\sim (va)))$	$\sim (va)$
$\sim (\sim (\sim (\sim (va))))$	va

8.2.2.3 Split Equivalence

To complete the transformation ready for the final conversion into AIGER, the "splitEquivalence" method takes the Boolean Expression Trees (containing no disjunctions or double negations) and produces a Transition Relation containing Transitions. Each Transition is composed of an equivalence and then a set of conjunctions. Inside "splitEquivalence" it calls "splitExpression" which takes the right-hand side of the equivalence to break down any conjunctions with more than two clauses into multiple conjunctions - as the AIGER format can only accept conjunctions with two clauses. This whole process roughly follows the Tseitin Transformation [73]. This method produces a final result which is either a single proposition (meaning no conjunction was present) or the top conjunction in the Boolean Expression Tree - the result can also be negated. This final result is used to create the equivalence such that $left\ hand\ side \Leftrightarrow final\ result$. The result of this transformation is a Transition Relation. This is a data type that stores an equivalence and a list of conjunctions. This allows for the any conjunctions to be broken down recursively to contain only two variables. The equivalence is the left-hand of the original equivalence (the output of the Rung) set to the top-most conjunction, or a variable if no conjunction exists.

Input : Boolean Expression Tree e''
Output: Transition Relation t

Input	**Output**
$(vA <=> vB)$	$TransitionRelation\{transitions = [Transition\{equiv = (vA <=> vB), conjunctions = []\}]\}$
$(vA <=>\sim (vB))$	$TransitionRelation\{transitions = [Transition\{equiv = (vA <=>\sim (vB)), conjunctions = []\}]\}$
$(vA <=> (vB\&vC))$	$TransitionRelation\{transitions = [Transition\{equiv = (vA <=> gen\_0), conjunctions = [gen\_0 : (vB\&vC)]\}]\}$
$(vA <=> (vB\|vC))$	$TransitionRelation\{transitions = [Transition\{equiv = (vA <=>\sim (gen\_0)), conjunctions = [gen\_0 : (\sim (vB)\& \sim (vC))]\}]\}$

$(vA <=> (\sim (vB) \mid \sim (vC)))$	$TransitionRelation\{transitions = [Transition\{equiv = (vA <=> \sim (gen\_0)), conjunctions = [gen\_0 : (vB\&vC)]\}]\}$
$(vA <=> ((vB\&vC)\&vD))$	$TransitionRelation\{transitions = [Transition\{equiv = (vA <=> gen\_0), conjunctions = [gen\_0 : (gen\_1\&vD), gen\_1 : (vB\&vC)]\}]\}$

8.2.3 Safety Condition Transformation

The Safety Conditions are transformed in a similar way to the Ladder Logic (8.2.2) instead using the class "SafetyConditionTransformation". This again takes in a Boolean Expression Tree, replaces the disjunctions, removes any double negations, and splits the conjunctions. The difference here is there are no equivalences involved in the expression. The result of this transformation is stored as a "Safety Condition", which contains an Array List of conditions. The list allows for the Aiger-fier to combine multiple conditions in future.

Input : Boolean Expression Tree e''
Output: Safety Condition s

Input	**Output**
vB	$SafetyCondition\{conditions [vB]\}$
$\sim (vB)$	$SafetyCondition\{conditions [\sim (vB)]\}$
$(vB\&vC)$	$SafetyCondition\{conditions [sc\_0 : (vB\&vC), sc\_0]\}$
$(vB \mid vC)$	$SafetyCondition\{conditions [sc\_0 : (\sim (vB)\& \sim (vC)), \sim (sc\_0)]\}$
$(\sim (vB) \mid \sim (vC))$	$SafetyCondition\{conditions [sc\_0 : (vB\&vC), \sim (sc\_0)]\}$
$((vB\&vC)\&vD)$	$SafetyCondition\{conditions [sc\_0 : (vB\&vE), sc\_1 : (sc\_0\&vD), \sim (sc\_1)]\}$

8.2.4 AIGER Transition Relation Transformation

Once the Ladder Logic and the Safety Property have been converted, the AIGER transformation can take place. There are two components to this, one for the Ladder Logic and one for the Safety Property. The Ladder Logic translation builds a latch and a set of "Ands" for each Transition in the set. There are inputs to deal with, any of the variables that are not a coil need an input defining along with a latch that is set to this input. For example, if the Ladder Logic [66] only consisted of $x \Leftrightarrow y$, then y would be an input variable as it does not appear on the right-hand side of an equivalence. The Safety Condition Translation also builds a set of "Ands" but this time produces an output instead of a Latch.

8.2.4.1 Ladder Logic Transition Relation to AIGER Transformation

Take a transition relation consisting of conjunctions and equivalences and produce a set of Latches, Inputs and logical Ands in the AIGER format [38] stored in the date type "Aig". Each component is given a numberical ID with each new ID being a the next available multiple of 2, the negated version is the ID+1. Therefore, the formula $a \Rightarrow b$ gets converted to a Latch with the ID 2 (a), and set to the value of Latch 4 (b) - which is set to itself.

Input : Transition Relation t
Output: Aig a

Input	**Output**
$TransitionRelation\{transitions = [Transition\{equiv = (vA <=> vB), conjunctions = []\}]\}$	$Aig\{aigerComponents = [Latch\{2\ 4\ 0\}, Input\{6\}, Latch\{4\ 6\ 0\}]\}$
$TransitionRelation\{transitions = [Transition\{equiv = (vA <=>\sim (vB)), conjunctions = []\}]\}$	$Aig\{aigerComponents = [Latch\{2\ 5\ 0\}, Input\{6\}, Latch\{4\ 6\ 0\}]\}$
$TransitionRelation\{transitions = [Transition\{equiv = (vA <=> gen\_0), conjunctions = [gen\_0 : (vB\&vC)]\}]\}$	$Aig\{aigerComponents = [And\{2\ 4\ 6\}, Latch\{8\ 2\ 0\}, Input\{10\}, Latch\{4\ 10\ 0\}, Input\{12\}, Latch\{6\ 12\ 0\}]\}$
$TransitionRelation\{transitions = [Transition\{equiv = (vA <=>\sim (gen\_0)), conjunctions = [gen\_0 : (\sim (vB)\& \sim (vC))]\}]\}$	$Aig\{aigerComponents = [And\{2\ 5\ 7\}, Latch\{8\ 3\ 0\}, Input\{10\}, Latch\{4\ 10\ 0\}, Input\{12\}, Latch\{6\ 12\ 0\}]\}$

$TransitionRelation\{transitions = [Transition\{equiv = (vA <=> \sim (gen\_0)), conjunctions = [gen\_0 : (vB \& vC)]\}]\}$	$Aig\{aigerComponents = [And\{2\ 4\ 6\}, Latch\{8\ 3\ 0\}, Input\{10\}, Latch\{4\ 4\ 0\}, Input\{10\}, Latch\{6\ 6\ 0\}]\}$
$TransitionRelation\{transitions = [Transition\{equiv = (vA <=> gen\_0), conjunctions = [gen\_0 : (gen\_1 \& vD), gen\_1 : (vB \& vC)]\}]\}$	$Aig\{aigerComponents = [And\{2\ 4\ 6\}, And\{4\ 8\ 10\}, Latch\{12\ 3\ 0\}, Input\{10\}, Latch\{8\ 8\ 0\}, Input\{10\}, Latch\{6\ 6\ 0\}, Input\{10\}, Latch\{10\ 10\ 0\}]\}$

8.2.4.2 Safety Condition to AIGER Transformation

Take a Safety Property consisting of conjunctions and equivalences and produce a set of logical "Ands" and an Output in the AIGER format [38]. As with the Ladder Logic, the variables are renamed to be numerical in AIGER. The variables should already exist as an AIGER ID, if this is the case then the Safety Property variables take their corresponding value from the hashmap. However, if the variables do not already exist then they are given a new AIGER ID value.

Input : Safety Condition s
Output: Aig a

Input	**Output**
$SafetyCondition\{conditions[vB]\}$	$Aig\{aigerComponents = [Output\{2\}$
$SafetyCondition\{conditions[\sim (vB)]\}$	$Aig\{aigerComponents = [Output\{3\}]\}$
$SafetyCondition\{conditions [sc\_0 : (vB \& vC), sc\_0]\}$	$Aig\{aigerComponents = [And\{2\ 4\ 6\}, Output\{2\}]\}$
$SafetyCondition\{conditions [sc\_0 : (\sim (vB) \& \sim (vC)), \sim (sc\_0)]\}$	$Aig\{aigerComponents = [And\{2\ 5\ 7\}, Output\{3\}]\}$
$SafetyCondition\{conditions [sc\_0 : (vB \& vC), \sim (sc\_0)]\}$	$Aig\{aigerComponents = [And\{2\ 5\ 7\}, Output\{3\}]\}$
$SafetyCondition\{conditions [sc\_0 : (vB \& vE), sc\_1 : (sc\_0 \& vD), \sim (sc\_1)]\}$	$Aig\{aigerComponents = [And\{2\ 5\ 7\}, And\{4\ 8\ 10\}, Output\{3\}]\}$

8.2.5 PrintAiger

The last component of the Aiger-fier is to print the newly-created Aig [12]. The first line of the file is the header which states the maximum variable index along with the number of Inputs, Latches, Outputs and "Ands". The values are calculated by the tracking the maximum variable index found from the components used in the file. The Inputs, Latches and "Ands" are sorted into separate Array Lists - the sizes of each are used for the header. There is no Outputs in these files, instead one Invariant property with a "1" at the end of the header. A new file is created with this header at the top, followed by the Inputs, then the Latches, then the Output and then any "Ands" from each Array List with one component per line.

Input : Aig a
Output: aiger.aag

Input	Output
$Aig\{aigerComponents = [And\{6\ 3\ 4\}, Latch\{2\ 6\ 1\}, Latch\{4\ 4\ 1\}, And\{8\ 3\ 5\}, Output\{8\}]\}$	aag 4 0 2 0 2 1 2 6 1 4 4 1 8 8 7 5 6 3 4

8.3 Integration Testing with the Siemens Testbed

The purpose of the research is to see if the IC3 algorithm [43, 44, 34] can classify those Verification Tasks of Type C. Whilst this has to be efficient too, the focus here will be firstly how the two IC3 implementations (IC3ref [42] and ABC [74, 75]) classify Type C, and then how they classify what is already known in Type A and Type B. We can safely assume that the IC3 algorithm and its implementations are correct. Whilst it has been proven to be correct with the Artificial Testbed (Table 8.2) it needs to be accurate with its classifications for the large real-world interlockings, so the Siemens engineers can trust its classification of Type C. For example, Type A and Type B have been decided by the Ladder Logic Verifier [87, 51] and therefore the implementations should have the same classifications to provide trust. To do this, those Verification Tasks of Type A and Type B can be run with both IC3ref and ABC. It it assumed the Ladder Logic Verifier is correct because of its usage in industry and therefore its classifications are trusted. Therefore, both IC3 implementations should match Types A and B correctly in order to trust the classification of Type C.

Version	IV ✓, and IC3 ✓	IV ✓, but IC3 ✗	BMC ✗, and IC3 ✗	BMC ✗, but IC3 ✓
IC3ref	590	30	675	327
ABC	590	30	675	327
Percentage	36.4%	1.8%	41.6%	20.2%

Table 8.9: Accuracy of the IC3 implementations on the pre-classified Verification Tasks. For complete correctness, all 100% should be in the cases either where Inductive Verification passes and IC3 holds, or Bounded Model Checking finds a counterexample and IC3 cannot find a Ladder Logic Invariant

Both IC3ref [42] and ABC [74, 75] produced the same results for the 1662 Verification Tasks of Types A and B. There are four types of classification in the correctness check (Table 8.9). The first is where Inductive Verification [64] passes and IC3 deems the program is safe, 36% of the data fell into this category. The second is where Inductive Verification has again passed, and the Verification Tasks is safe, yet IC3 has failed stating that it is unsafe. 30 Verification Tasks fell into this class, or 1.8% of the data, and these are known as Type A1 in the overall classification. The next is where Bounded Model Checking [55, 30, 66, 54] finds a counterexample and IC3 states that the Ladder Logic Program is unsafe with the Safety Property P which consisted of 42% of the data. The final category is where again Bounded Model Checking finds a counterexample but IC3 says a Ladder Logic Invariant can be found that will keep the program safe. 327 Verification Tasks fell into this class, or 20% of the data, and these are known as Type B1 in the overall classification.

From the testing, it can be said that for the Verification Tasks where the classification is known, either the program is safe for a given Safety Property or a counterexample is found and it is unsafe. This is because Inductive Verification can only determine if a Verification Task is safe, whilst Bounded Model Checking can only determine if a Verification Task is unsafe - and that is why Type C is not included. Overall, out of the 1662 Verification Tasks that were considered for the correctness check, 1265 were correctly matched to the classification from the Ladder Logic Verifier [87, 51]. A full breakdown of the testing with IC3 can be found in Appendix C. It is assumed the classifications made by the Ladder Logic Verifier have been proven to be correct and used in Industry. There were 357 Verification Tasks that failed the comparison check.

327 of these are Type B1. A counterexample was found within 100 bounds for these Verification Tasks, however, IC3 has said that a Ladder Logic Invariant can be found. The remaining 30, of Type A1, are where the trusted Ladder Logic Verifier has found a counterexample within 100 bounds yet both IC3 implementations have deemed these Verification Tasks to be safe and that a Ladder Logic Invariant can be produced - which makes Type A1 an issue. In AIGER there is a need to define the Inputs as Latches to ensure that they are initialised to 0. The Latch is given the value of the Input after each iteration and is initially 0 to achieve the desired effect. The unrolling process in IC3 and

Clausegen differs slightly. In Clausegen, which uses the TPTP file format [4], all variables have a version for each iteration - "_0" initially, "_1" after iteration 1, "_2" after iteration 2 and so on. In IC3 the same process is replicated but the passing of the values across variables in the unrolling. The difference is that Clausegen deals with the variables in a top down approach whereas IC3 in a bottom up. This is logically the same but further work would be needed to check whether when running the Ladder Logic Program it unrolls identically with each transition T.

The likelihood is there is a small error in the Aiger-fier and the translation process from Ladder Logic [66] to AIGER [12] because it was developed specifically for this project and, while tested thoroughly, it has not gone through rigorous industrial examination. The IC3 implementations have been proven successful for hardware verification so are assumed to be correct. Further work to print the Ladder Logic Invariant would allow for Inductive Verification to be performed again to see if the Ladder Logic Program is indeed safe and therefore determine whether the error is in Bounded Model Checking. However, it is assumed that through testing and its usage in industry, this is not the case.

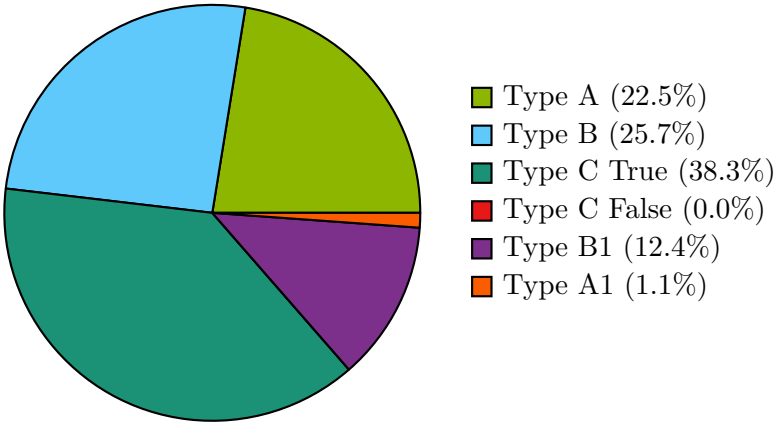


Figure 8.6: Overall distribution of the Verification Tasks with IC3

When IC3 was run on all the Verification Tasks that were previously classified by Inductive Verification and Bounded Model Checking, both IC3ref and ABC classified the Verification Tasks (Figure 8.6) as 22.5% Type A, 25.7% Type B, 38.3% Type C True, 0.0% Type C False, 12.4% Type B1 and 1.1% as Type A1. Type C True are those Verification Tasks originally classified as Type C where a Ladder Logic Invariant was found that allows the verification to pass. Conversely, any Verification Tasks originally of Type C where a Ladder Logic Invariant could not be built are defined as Type C False. Those classified as Type B1 are where Bounded Model Checking has found a counterexample, however, IC3 returns the Verification Task is safe which results in a correctness issue. This is potentially due to a failure on. Type A1 is also a correctness issue because Inductive Verification has claimed the Verification Task are safe yet IC3 says they are unsafe.

Further investigations and discussions with Siemens is required to deal with these 357 classifications of Type B1 and Type A1. Looking into those Verification Tasks of Type B1 where Bounded Model Checking found a counterexample yet IC3 deemed safe as an Invariant could be formed the traces can be evaluated. Bounded Model Checking produces a counterexample trace which we hope to analyse with Siemens to establish whether the it is "real". When Siemens perform the verification they make assumptions and it could also be the case that they concern the boundaries where multiple interlockings have to be verified together. Examining the counterexamples also allows for comparison with IC3 and how it deals with the error. For those Verification Tasks of Type A1, where IC3 deems them unsafe yet they pass Inductive Verification, we need to establish with Siemens what the correct encoding is for the Ladder Logic Programs. There were 30 Verification Tasks of Type A1 of which 21 of them had variables in the Safety Property that were not included in the Ladder Logic. This only seems to be the case for those of Type A1. In the Aiger-fier transformation this simply creates a new Latch that will remain constant. It is unclear how the Ladder Logic Verifier deals with variables in the Safety Property that are not in the Ladder Logic so discussions will be needed to establish which is the correct encoding. Without knowing how they are dealt with, this syntactic problem reduces are trust in the Ladder Logic Verifier. The remaining 9 Verification Tasks were of the case where the Safety Property had an implication where the left-hand side contained two versions of the same variable - for example a_0 and a_1. Further investigations on this will be needed to ensure that the way the inputs are dealt with by the Aiger-fier is correct with their non-determinism and how Safety Properties with two versions of a variable should be represented.

If any of these are indeed the case, it would reduce the trust in the classification from the Ladder Logic Verifier but seeing as this has been proven to be correct and used in Industry, it is less likely to be the case. The Ladder Logic [66] to AIGER [38] transformation was deemed to be correct with the testing performed using the small examples and their variations, however, there could be an error in the transformation as it scales. Although, the testing stated that the rules of the Ladder Logic TPTP [4, 8] files are dealt with correctly. It could be the case there is an error in the translation of the Safety Property where a rule was missed. The IC3 implementations could also be at fault - printing the Invariants would allow a proof to be performed to check its validity. However, the goal of this Integration Testing was to increase the trust in the classification of Type C with IC3, which for the overall Testbed has a success rate of 86.5%.

Interlocking	**Verification Tasks**	**Type A**	**Type B**	**Type CT**	**Type CF**	**Type B1**	**Type A1**
DesignX	714	47%	2%	32%	0%	17%	2%
Interlocking A	1278	20%	2%	61%	0%	16%	1%
Interlocking B	636	0%	100%	0%	0%	0%	0%
Total	2628	22.5%	25.7%	38.3%	0.0%	12.4%	1.1%

Table 8.10: Summary of the Siemens Testbed when IC3 is performed

CHAPTER 9

Hypotheses for IC3 Ladder Logic Verification

Contents

The overall goal of this research is to determine whether the IC3 algorithm [43, 44, 34] could decide the Verification Tasks of Type C efficiently. If successful, it would fill the gap in the Ladder Logic Verifier [87, 51] where if Inductive Verification [64] fails, Bounded Model Checking [55, 30, 66, 54] can only guarantee safety up to k bounds. IC3 has been initially tested on all tasks that Bounded Model Checking was unable to find a counterexample trace for after 100 bounds. Either producing a Ladder Logic Invariant via an Inductive Strengthening [64], or by returning that an Invariant cannot be produced and the program is unsafe. A further two hypotheses are explored, firstly the speed of IC3 across the various verification types (Figure 2.1). The second is whether IC3 could replace Inductive Verification or Bounded Model Checking. For this to be the case, IC3 must not only beat the existing methods in terms of runtime, but be accurate and correctly classify Verification Tasks.

9.1 Hypothesis 1: IC3 Algorithm can Efficiently Decide Type C

The current verification problem (Figure 2.1) is that Bounded Model Checking [55, 30, 66, 54] can only verify for safety up to a certain point. The idea is that IC3 can take all of these undecidable Verification Tasks of Type C and determine if they are indeed safe, or whether they are unsafe and the counterexample is yet to be found. Two versions of IC3 are being investigated: IC3ref [42] and ABC IC3 [74, 75] which improves the efficiency of the algorithm.

9.1.1 Deciding Type C

Among the 2828 Verification Tasks, there were 1006 Verification Tasks classified as Type C by Inductive Verification [64] and Bounded Model Checking [55, 30, 66, 54]. All of these 1006 tasks have been classified as "true" (Type C True) by both IC3ref and ABC (Table 9.1). Verification Tasks of Type C have failed Inductive Verification, Bounded Model Checking was unable to find a counterexample after 100 bounds and IC3 has deemed the tasks to be safe as a Ladder Logic Invariant can be produced. This Ladder Logic Invariant is one that would allow Inductive Verification to pass by removing the False Positive [51] that previously existed. None of the tasks were classified as Type C False where again Inductive Verification [64] fails, and Bounded Model Checking [55, 30, 66, 54] again would have been unable to find a counterexample after 100 bounds. However, this time IC3 would have been unable to produce a Ladder Logic Invariant that would permit all the reachable transitions. This is because some are unsafe and would not be included in the Ladder Logic Invariant - therefore the program is unsafe. Given enough bounds a counterexample would be found by Bounded Model Checking. A full breakdown of IC3 running IC3 on the tasks, which includes their classifications from the Ladder Logic Verifier, can be found in Appendix C. Overall, it can be said that this Hypothesis is true, as IC3 was able to be run on the 1006 Verification Tasks and can return a concrete decision. In principle, we could try to use the Invariant to independently establish these results through an Inductive Strengthening. However, we have not implemented the required output functionality yet. In future, the ability to print the Invariant, translated back into a Boolean formula, could be included to achieve this. With regards to Bounded Model Checking, this result shows that even by increasing the bound we cannot obtain a counterexample trace. These traces could be added into IC3 in future by identifying states that are reachable and unsafe - preventing the Invariant. The trace could then be formed using the transitions (T) and working back.

Implementation	**Type C True**		**Type C False**	
IC3ref	1006	100.0%	0.0	0.0%
ABC	1006	100.0%	0.0	0.0%

Table 9.1: Classification of the 1006 Verification Tasks of Type C from the Siemens Testbed by the IC3 implementations

9.1.2 Runtimes for Deciding Type C

IC3ref [42] and ABC [74, 75] produced identical classifications for those Verification Tasks of Type C. Determining all as Type C True because a Ladder Logic Invariant can be found. Bradley's IC3ref implementation includes the counterexample generalisation feature in the IC3 algorithm that was missing in the implementation developed previously when this was an Undergraduate project [46]. This generalisation approach helps make IC3 scalable for use on industrial-sized Ladder Logic Programs as previously, while it was successful for the small artificial testbed, after five days it was still processing. Mishchenko's ABC [74, 75] improves on this further by addressing any counterexamples that prevent generalisation to still allow large Cubes to be formed. The IC3 element of ABC is contained as part of the tool which allows for synthesis and verification of binary sequential logic circuits for hardware designs making it very efficient. In general, by representing the Boolean Logic as And-Inverter Graphs it results in efficient interfacing with CNF-based SAT Solvers for Boolean reasoning.

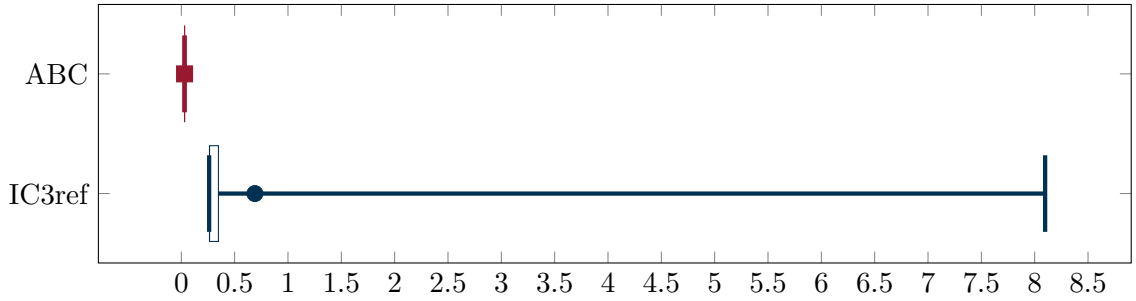


Figure 9.1: Runtimes in seconds of deciding the 1006 Verification Tasks of Type C from the Siemens Testbed with IC3ref and ABC

Clarifying Type C took IC3ref on average 0.690 seconds, whilst ABC took just 0.031 seconds as shown in the boxplot graph Figure 9.1. This graph, like the others that come later in the chapter, have the average runtimes as data points along with the upper and lower quartiles (the box), the median runtimes (the line in the box) and finally the minimum and maximum runtimes (the left and right whiskers). The times for ABC are so insignificant compared to IC3ref making the extra information indistinguishable from the average so a table is provided for clarity (Table 9.2).

Implementation	Avg	Min	LQ	Median	LQ	Max
IC3ref	0.690	0.261	0.264	0.265	0.348	8.098
ABC	0.031	0.030	0.030	0.031	0.031	0.031

Table 9.2: Average, Minimum, Lower Quartile, Median, Upper Quartile, and Maximum Runtimes in seconds of classifying the Verification Tasks of Type C from the Siemens Testbed with the IC3 implementations

9.1.2.1 Runtimes for Deciding Type C with IC3ref

When Bradley's IC3 implementation [42] was run on the 1006 Verification Tasks that were previously determined to be Type C (Figure 9.2), those that were then deemed to be safe because a Ladder Logic Invariant was found took on average 0.690 seconds, with a median of 0.265 seconds (Table 9.3). These tasks have been classified as Type C True, as they were originally Type C where they failed Inductive Verification, but Bounded Model Checking was unable to find a counterexample in 100 bounds. Now IC3 has found a Ladder Logic Invariant that will allow the verification to pass. IC3ref did not classify any Verification Tasks as Type C False.

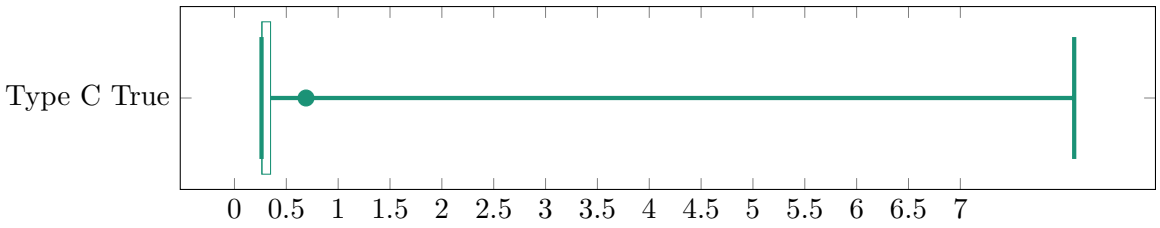


Figure 9.2: Runtimes in seconds in classifying the 1006 Verification Tasks of Type C as Type C True and Type C False with IC3ref

Type	Avg	Min	LQ	Median	LQ	Max
Type C True	0.690	0.261	0.264	0.265	0.348	8.098

Table 9.3: Average, Minimum, Lower Quartile, Median, Upper Quartile, and Maximum Runtimes in seconds of classifying Type C True and Type C False with IC3ref

9.1.2.2 Runtimes for Deciding Type C with ABC

The runtimes from running the 1006 Verification Tasks determined to be Type C with ABC [74, 75] is misleading at first (Figure 9.3), this is due to the small runtimes (Table 9.4). Again it was the case that all Verification Tasks were classed as Type C True because ABC was able to find a Ladder Logic Invariant. This took on average only 0.0305 seconds, which is 23 times quicker the original implementation from Bradley. The ABC implementation includes the ability to deal with Counterexamples to Generalisation but there is also the factor that IC3ref is calling Minisat [9], whereas ABC uses its in built solvers. The different solvers and how they react with the rest of the program can contribute to different runtimes.

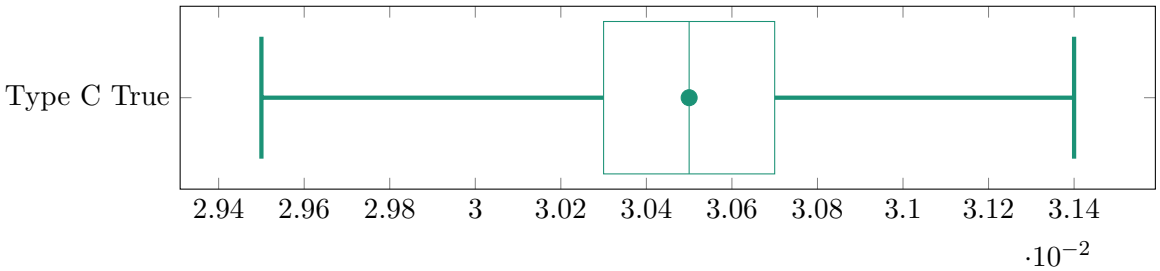


Figure 9.3: Runtimes in hundredth seconds for classifying the 1006 Verification Tasks of Type C as Type C True and Type C False with ABC

Type	Avg	Min	LQ	Median	LQ	Max
Type C True	0.0305	0.0295	0.0303	0.0305	0.0307	0.0314

Table 9.4: Average, Minimum, Lower Quartile, Median, Upper Quartile, and Maximum Runtimes in seconds for classifying Type C True and Type C False with IC3ref

9.2 Hypothesis 2: IC3 Algorithm runs "Fast" on Types A and B, "Slow" on Type C

Before running the IC3 implementations it was expected that constructing a Ladder Logic Invariant that prevents a False Positive [51] would be more computationally expensive compared to either returning the Safety Property, as it was already sufficient in keeping the Ladder Logic Program safe, or stating that a Ladder Logic Invariant cannot be produced as the program is unsafe. This is down to the assumption that for Types A and B, IC3 will exit the Iteration Phase early. For Bradley's IC3ref [42] returning that a Verification Task was unsafe (Type B) was the fastest. Finding a Ladder Logic Invariant that prevents False Positives (found in Type C) does cause an increase in runtime. There was also an increase in classifying Type A and was slowest overall which went against the expectations. Whereas for ABC [74, 75] the increase is runtime between Type B and Types A and C was just 2 ten-thousandths of a second, so it can be disregarded (Figure 9.5).

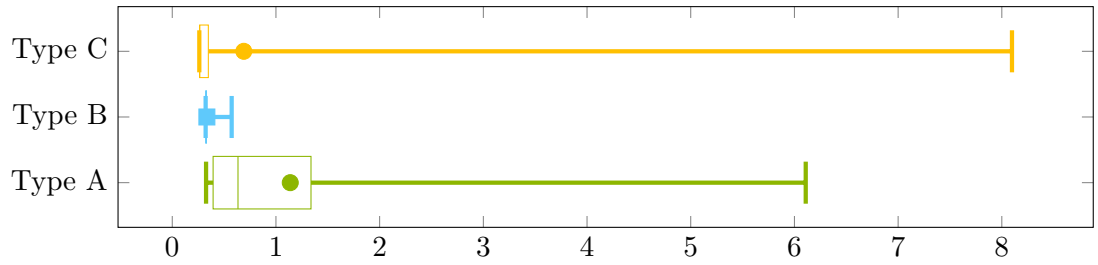


Figure 9.4: Runtimes in seconds in classifying the 2628 Verification Tasks from the Siemens Testbed with IC3ref

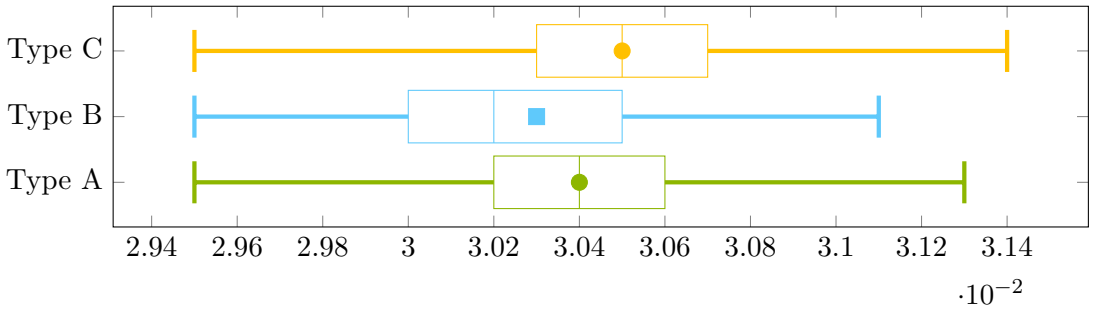


Figure 9.5: Runtimes in hundredth seconds in classifying the 2628 Verification Tasks from the Siemens Testbed with ABC

Type A Fast:
For those Verification Tasks that passed Inductive Verification [64] it was anticipated that IC3 would decide these tasks quickly because the Safety Property P includes only reachable states and therefore the Ladder Logic Programs is safe. Therefore, the IC3 algorithm would not need to amend the Safety Property and would exit after a few iterations - returning a Ladder Logic Invariant which is close to the Safety Property. This was not the case (Figure 9.4) with IC3ref [42] deciding Type A in 1.138 seconds on average - which whilst fast is slower than the runtimes for Types B and C. This increase might be because it is the opposite to what was expected and the algorithm requires more iterations to reduce the Ladder Logic Invariant Candidate down to just the Safety Propety P. Because the Invariant originally includes the clauses for the Initial States it requires relaxing to include all reachable safe states. ABC [74, 75] took 0.030 seconds on average (Figure 9.5) which is practically the same as classifiying the other two types.

Type B Fast:
For those Verification Tasks where Bounded Model Checking found a counterexample in 100 bounds, it was also expected the IC3 algorithm would decide these tasks quickly because the counterexample is discovered within a relatively small number of iterations. Therefore, IC3 would determine that a Ladder Logic Invariant would be unsuitable because it would be unable to keep the program safe, whilst allowing all the safe transitions. In theory, once the Invariant has grown enough to include the reachable unsafe state the algorithm will fail and terminate. This is because this state makes the program unsafe but cannot be removed as a counterexample state as it is reachable. It was indeed fast for both iterations with ABC [74, 75] taking on average 0.030 seconds (Figure 9.5), slightly slower then with Type A. For IC3ref [42] it took on average 0.334 seconds (Figure 9.4).

Type C Slow:
Finally, it was expected that for Verification Tasks of Type C the runtimes would be slowest. This is because many iterations in theory will be required to either block the counterexamples and present a False Positive [51] by producing a Ladder Logic Invariant,

or to determine that the Safety Property is not sufficient and a Ladder Logic Invariant cannot be produced. For IC3ref [42] this was the case taking on average 0.690 seconds. However, ABC [74, 75] took 0.031 seconds (Figure 9.5), which was effectively the same time taken to classify Types A and B.

Hypothesis two went against the expectations requiring further investigations to understand the runtimes, and how many iterations were taken by the IC3 algorithm to make a decision. By adding the ability to print the Invariants after each iteration it allows for the number of states that hold under each Ladder Logic Invariant Candidate to be revealed, any bugs to be tracked, and to see how quickly the sets grow to include all the reachable states. In the example of ABC, it could be the case that the majority of the reachable states are included after a single iteration making it very efficient. It is also worth noting how many False Positives are present, the fewer there are, the quicker the runtime as less time is spent amending the Ladder Logic Invariant Candidates.

9.3 Hypothesis 3: IC3 Performs Better than the Ladder Logic Verifier in Terms of Runtime

The final hypothesis was to see if IC3 could replace Inductive Verification [64] or Bounded Model Checking [55, 30, 66, 54] when it comes to runtime. For this to be the case IC3ref [42], ABC [74, 75] and the Aiger-fier [45] all have to be correct in classifying the known Verification Tasks which gave an accuracy of 86.5% for both implementations. If IC3 is faster and correct then in theory it could replace the existing methods. Further investigation into these tasks of Type B1 and Type A1, and then why they were deemed safe by IC3 will hopefully reveal the fault. In this section, the focus will be solely on the runtimes.

Type	**Ladder Logic Verifier**		**IC3ref**		**ABC**	
	Average	**Median**	**Average**	**Median**	**Average**	**Median**
Type A	1.285	1.416	1.138	0.634	0.030	0.030
Type B	54.921	57.734	0.334	0.326	0.030	0.030
Type C	132.694	64.913	0.690	0.265	0.031	0.031

Table 9.5: Runtimes of classifying the Siemens Testbed in seconds

A full breakdown of the runtimes taken to decide each type of Verification Tasks with the Ladder Logic Verifier [87, 51], IC3ref [42] and ABC [74, 75] can be found in Table 9.5. The times to compute Types B and C are the times taken for Inductive Verification combined with Bounded Model Checking when running the Ladder Logic Verifier. Whereas only IC3ref or ABC are run. IC3ref uses Minisat [9], a small but efficient SAT Solver, to perform the algorithm's checks. ABC uses its own system to perform the algorithm using the PDR (Property Driven Reachability) [88, 60] - also known as IC3. The PDR package is built into the ABC system as one of the many packages for synthesis and verification

of binary sequential logic circuits. By comparison, Inductive Verification and Bounded Model Checking are built into the Clausegen implementation of the Ladder Logic Verifier and perform their checks using Z3 [72, 40]. The use of different SAT Solvers and how they interact with the rest of the program are contributing factors to why there is such a difference in runtimes between the three methods, especially between IC3ref and ABC.

For those Verification Tasks that the Safety Property P holds for the Ladder Logic Program without the need for a Ladder Logic Invariant (Type A) it takes Inductive Verification [64] on average 1.285 seconds (Figure 9.6). IC3ref [42] takes 1.138 seconds (Figure 9.8) and ABC [74, 75] just 0.030 seconds (Figure 9.9) - 42 times faster than Inductive Verification.

Type B took on average 1.176 seconds with Inductive Verification (Figure 9.6) and 53.774 seconds with Bounded Model Checking (Figure 9.7), therefore 54.921 seconds in total. This increase compared to Inductive Verification is why it is seen as computationally expensive, and why only 100 bounds are chosen by Siemens but leads to the incompleteness problem. In comparison, it takes 0.334 seconds with IC3ref (Figure 9.8), and just 0.030 seconds with ABC (Figure 9.9) - 1,813 times faster than the Ladder Logic Verifier. However, when the Ladder Logic Verifier performs Bounded Model Checking it also takes the time to compute a counterexample trace that helps the engineers fix the error. Whereas in its current form IC3 produces a “yes/no” response. If IC3 was expanded to include the counterexample traces it would increase the runtimes.

Inductive Verification failed on average after 1.201 seconds for Type C (Figure 9.6), and Bounded Model Checking completed 100 bounds without finding a counterexample in 131.483 seconds (Figure 9.7). Making the total average 132.694 seconds. IC3ref took 0.690 seconds on average to decide the 1006 Type Cs (Figure 9.8) with ABC taking 0.031 seconds (Figure 9.9) - 4,351 times faster than the Ladder Logic Verifier. Unlike Bounded Model Checking the IC3 implementations can provide completeness and therefore determine whether those of Type C are safe. It took IC3ref 1.111 seconds to decide the misclassified Type B1, whilst ABC took 0.031 seconds. Finally, Type A1 took IC3ref 0.463 whilst ABC again took 0.031 seconds. Further work will be conducted produce a correct classification. As with Type B, the Ladder Logic Verifier will attempt to produce a counterexample trace which slows the approach. If IC3 had the same feature it would affect the runtimes - although the scale of this increase is unknown until implemented.

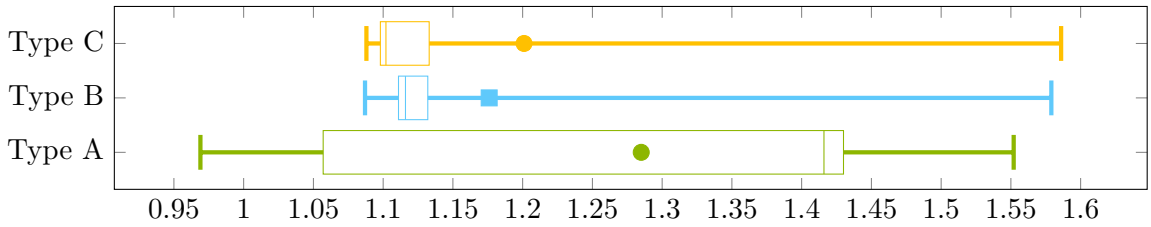


Figure 9.6: Runtimes in seconds for classifying each type of Verification Task with Inductive Verification for the Siemens Testbed

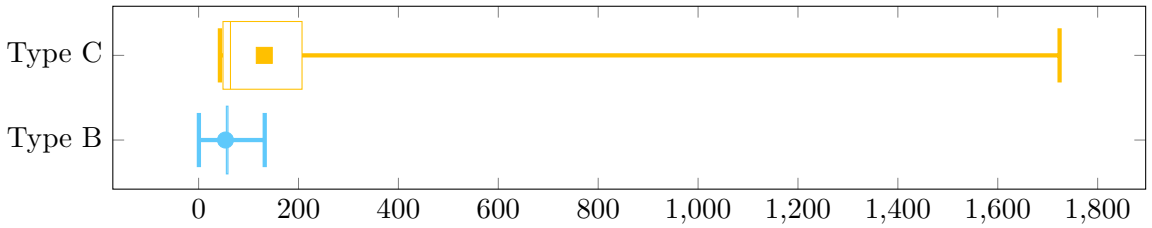


Figure 9.7: Runtimes in seconds for classifying each type of Verification Task with Bounded Model Checking for the Siemens Testbed

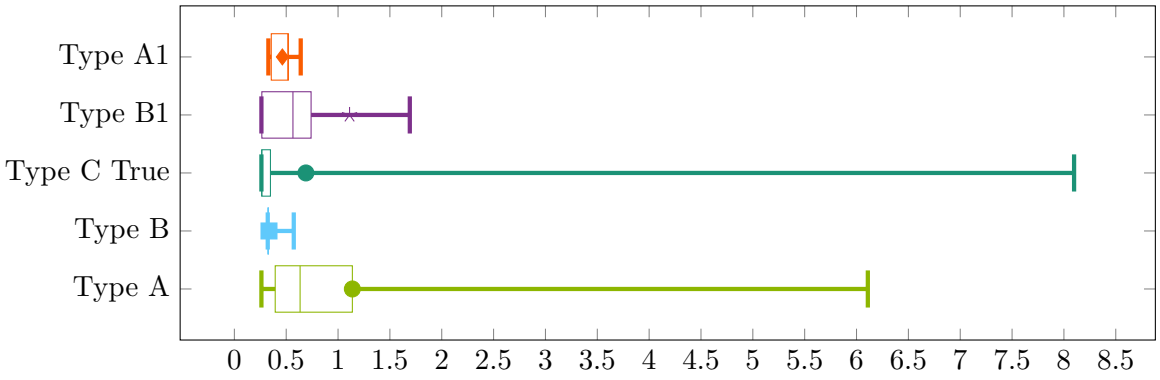


Figure 9.8: Runtimes in seconds for classifying each type of Verification Task with IC3ref for the Siemens Testbed

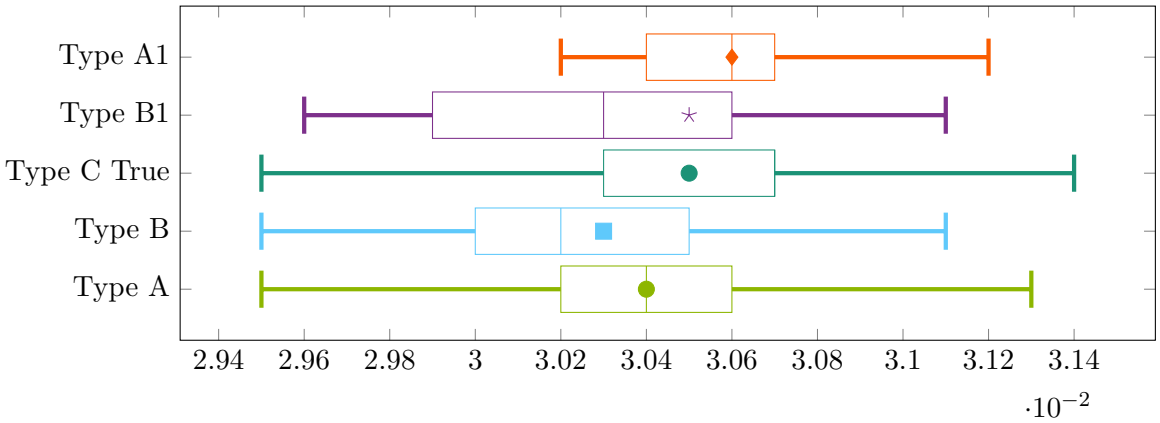


Figure 9.9: Runtimes in hundredth seconds for classifying each type of Verification Task with ABC for the Siemens Testbed

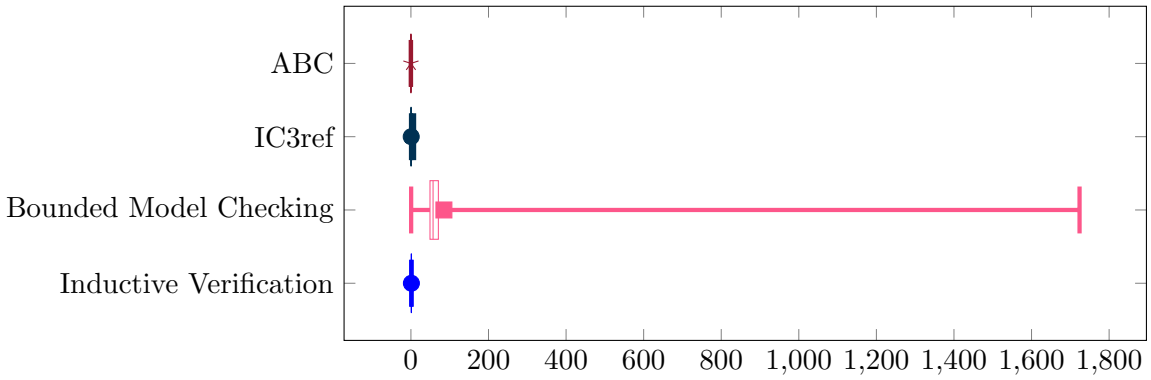


Figure 9.10: Runtimes in seconds of classifying each type of Verification Task from the Siemens testbed with all methods, showing how expensive in terms of runtime Bounded Model Checking is by comparison

From the runtimes (Table 9.5) IC3 is slightly quicker then Inductive Verification. When Inductive Verification runs with Z3 [72, 40] it returns countersatisfiable if the check passes or satisfiable if it fails. Whilst IC3ref just prints 0 if successful and 1 if not, ABC has its own command line interface providing more detailed responses. When a Ladder Logic Invariant can be found, because the program previously failed due to a False Positive [51], the number of the final Ladder Logic Invariant Candidate F_h is printed along with a statement that the verification of this Invariant was successful (Figure 9.11a). This is number of the final Invariant, e.g., the number of layers produced. The time is also printed amongst other information. When the Ladder Logic Invariant cannot be found though, ABC again prints the time but it also states that the unsafe state has been reached and what frame that led to it. A frame is the equivalent of one step in the Ladder Logic Program. This can determine whether an error is located early or late in the program. What this means is that IC3 provides the same or more information than Inductive Verification whilst being quicker and correct. Therefore, IC3 could replace Inductive Verification. These outputs were the same for running ABC with the Siemens Verification Tasks (Figure 9.11c). The difference with these is that the number of iterations needed to produce the Invariant can be larger due to the number of states involved, therefore more layers of Invariants are built (Figure 3.2).

Both IC3 implementations are significantly faster than Bounded Model Checking (Table 9.5). However, the benefit of Bounded Model Checking is that it returns detailed information from Z3 when a counterexample is found assisting the engineers in addressing the faults in the Ladder Logic Program. This is currently missing from IC3. However, IC3 in its current form could be performed first, and the computationally expensive Bounded Model Checking then run after the Verification Tasks that failed, to provide details on the counterexamples. It is hoped that future work could develop IC3 further to allow for the counterexample traces to be printed so the implementations could fully replace Bounded

Model Checking too. The Ladder Logic Invariant could also be printed when it can be produced by IC3. By doing this, it would increase trust in the algorithm and allow for Inductive Verification to be run with the Invariant to prove it does indeed show the program is safe. This process was employed to show the Ladder Logic Invariants were successful in verifying the manual Artificial Testbed where the Ladder Logic Invariants could be manually found.

```
abc 01> read_aiger correctness/P/example1.aig
abc 02> pdr
Invariant F[1] : 1 clauses with 1 flops (out of 2) (cex = 0, ave = 1.00)
Verification of invariant with 1 clauses was successful.  Time =     0.00 sec
Property proved.  Time =     0.01 sec
```

(a) ABC's output when a Ladder Logic Invariant can be produced

```
abc 02> read_aiger correctness/¬P/example1_notP.aig
abc 03> pdr
Output 0 of miter "correctness/¬P/example1_notP" was asserted in frame 0.  Time =     0.01 sec
```

(b) ABC's output when a Ladder Logic Invariant cannot be produced

```
abc 03> pdr
Invariant F[18] : 1 clauses with 1 flops (out of 20960) (cex = 0, ave = 15.77)
Verification of invariant with 1 clauses was successful.  Time =     0.00 sec
Property proved.  Time =     0.08 sec
```

(c) ABC's output when a Ladder Logic Invariant can be produced for a Siemens Verification Task

Figure 9.11: ABC's command line interface

There is also the possibility that different Safety Properties will have an influence on the runtimes. This in theory could be a more of an issue for the Ladder Logic Verifier where Bounded Model Checking is computationally expensive. In order to test this theory the Ladder Logic Verifier, IC3ref, and ABC runtimes of the Verification Tasks for DesignX have been grouped into their corresponding categories recieved from Siemens to form comparisons. The reason for choosing DesignX is firstly that it is the largest of the interlockings, and secondly features the most number of categories of Safety Property with 47. A full breakdown of the average runtimes per category for each Type of Verification Task when running the Ladder Logic Verifier, IC3ref, and ABC can be found in Appendix C.3. A condensed version of the table (Table 9.6) shows the maximum average, minumum average, the range of averages, and Standard Deviations per category of Safety Property for DesignX. For simplicity, Types A and A1 have been combined in the tables, along with Types B and B1, allowing for a direct comparison with the Ladder Logic Verifier's results. The runtimes for the ABC are far quicker than with IC3ref, which are then in turn quicker than the Ladder Logic Verifier which was why the Standard Deviation was included.

	LL Verifier			IC3ref			ABC		
	A	**B**	**C**	**A+A1**	**B+B1**	**C**	**A+A1**	**B+B1**	**C**
Range	0.13	51.77	407.62	2.897	5.780	3.831	0.0010	0.0011	0.0007
Standard Deviation	0.05	11.72	96.00	1.059	1.615	1.662	0.0002	0.0003	0.0002
Max	1.54	123.74	474.28	3.294	6.179	4.226	0.0309	0.3010	0.0308
Min	1.41	71.97	66.66	0.397	0.400	0.396	0.0299	0.0299	0.0301

Table 9.6: Runtimes of classifying the Siemens Testbed in seconds

For Verification Tasks of Type A the Standard Deviation highest for IC3ref, with the ranges staying within a few seconds. For Type B the Standard Deviation and ranges for IC3ref and ABC increased but are nothing when compared to that of the Ladder Logic Verifier. The values for IC3ref and ABC stayed roughly the same for Type C, whereas the values for the Ladder Logic Verifier again increased drastically. This data therefore shows that for the Ladder Logic Verifier when Bounded Model Checking is applied to different categories of Safety Property there is an effect on the runtime with some tasks being decided a lot faster than others. Whereas with the IC3 implementations this is not a problem.

Part III

Conclusion

CHAPTER 10

Summary and Future Work

Contents

This Master of Research project has been aimed at improving the original verification approach (Figure 2.1) for Ladder Logic Programs by Siemens Mobility and their Ladder Logic Verifier [87, 51] through the use of the IC3 algorithm [43, 44, 34]. This chapter will provide an overview of the work conducted and the success of the algorithm, along with potential direction for future work in order to further improve Ladder Logic Verification with IC3.

10.1 Summary

The goal of this thesis was to prove the IC3 algorithm [43, 44, 34] could efficiently decide those Verification Tasks of Type C. These have failed Inductive Verification [64] but Bounded Model Checking [55, 30, 66, 54] deemed them safe after 100 bounds. However, the problem is the railway interlockings developed by Siemens are so large that this leaves the verification incomplete as it only covers a fraction of the transitions possible. In order to prove they are completely safe, the number of bounds must be increased, therefore adding to the already incredibly large runtime - on average 85 seconds for 100 bounds. This is an issue because a typical interlocking has approximately has 7,000 Verification Tasks. If they all take 85 seconds it equates to roughly 165 hours to perform the verification which is costly for Siemens. The result from this is still considered undecided because if Inductive Verification failed it could have been due to an over-approximation, or an error not yet found.

IC3 was demonstrated to solve the over-approximation problem that results in a False Positive [22, 25] when Inductive Verification is performed. This was done initially through the use of small artificial Ladder Logic Programs where they could be verified manually as they are small enough to be visualised as an Automaton. The solution is to construct a Ladder Logic Invariant (Ladder Logic Invariant) that permits all the reachable transitions in the program to take place whilst crucially blocking any unreachable states from transitioning to an unsafe state - a False Positive. It was demonstrated in Table 8.1 that with these Ladder Logic Invariants the verification was successful (Figure 3.8) using the artificial examples by blocking the over-approximation. This could be validated using automatons (Figure 3.7) which allow the Invariant to be mapped allowing for the states covered to be visualised - if successful it will be all the reachable safe states. Further proofs of how IC3 can produce successful Invariants can be found in Appendix B. Using the small examples also showed that when a program is unsafe IC3 would be unable to produce a Ladder Logic Invariant (Table 8.2) - making it appropriate for use on the undecidable Verification Tasks.

The next stage was to test how effective Ladder Logic Verification with IC3 is on an industrial scale. To do this we were given three interlockings from Siemens, one of which is an early development of an interlocking in current use on the North Wales Coastline. Each interlocking came with a set of Safety Properties which, when combined with the relevant interlocking, formed a testbed of 2628 Verification Tasks - all were tested with IC3. These were classified using the Ladder Logic Verifier [87, 51] to create a comparison to IC3, firstly, by running Inductive Verification [64] to determine which Verification Tasks are safe (Type A). All those that failed, because they are either unsafe or contain a False Positive, were run with Bounded Model Checking [55, 30, 66, 54] to find counterexamples and determine whether a Verification Tasks is unsafe (Type B). Bounded Model Checking was run with bounds set to 100 to match Siemens. If no counterexample was found, then it can be said the Verification Tasks are confirmed to be safe at least for 100 transitions (Type C). However, in this case, it could still be unsafe and the counterexample is yet to be found. This is where IC3 comes in to determine these undecidable Verification Tasks by either constructing a Ladder Logic Invariant that allows the verification to pass by blocking the False Positive, or by deeming the program is unsafe. IC3 deemed all Verification Tasks previously classified as Type C from the three interlockings in the Siemens Tested to be safe by constructing Ladder Logic Invariants. Taking on average 0.690 seconds with Bradley's IC3ref [42] and just 0.031 seconds with the improved ABC implementation [74, 75] (Table 9.5).

To increase trust in this classification, and to see if IC3 could replace the existing methods because of its incredibly quick runtime, those Verification Tasks previously classified by the Ladder Logic Verifier as Types A and B were also run with the IC3 implementations (Figure 7.5). The idea being that IC3 should determine those of Type A are safe and those of Type B are unsafe with it unable to construct a Ladder Logic Invariant - helping to prove the correctness of the approach. However, there were some misclassifications with 12.4% being of Type B1 where Bounded Model Checking found a counterexample yet IC3 deemed

the Verification Task safe, and 1.1% of Type A1 where Inductive Verification passed but IC3 says an Invariant cannot be produced as the Verification Task is unsafe (Figure 8.6). Because the Ladder Logic Verifier is used in Industry it is less likely to contain the error. This is because it has been through a rigorous testing process, therefore, it is less likely a major error will have gotten through to this version. Again, IC3 has been successfully used in hardware verification and will have been heavily tested in order to have trust in the results it gives. Instead, the error is likely to be in the transformation from Ladder Logic [66] into AIGER [38] using the Aiger-fier [45] and the Safety Property Parser [47]. This can either be because of an error in the transformation or a mistake in the understanding of how the Ladder Logic Programs unroll and convert correctly for each Safety Property in AIGER. For example, whilst thoroughly tested, the Aiger-fier has not had the team of testers spending 100s of hours debugging the program which the Ladder Logic Verifier received.

Work was also completed to prove the correctness of the algorithm using formal methods, along with analysing the transformation with all the sections being subject to Unit Testing [20]. The aim of the testing and use of formal methods increase trust in the transformation. It also helped to reduce the number of errors, and therefore possibilities as to why there is this misclassification when compared to the Ladder Logic Verifier. The greater the trust in the current transformation being correct, the more likely the error is due to the understanding being incorrect. Therefore, further testing and discussions with Siemens are required to ensure that transformation can be refined to fully represent the Ladder Logic in AIGER correctly. Those Verification Tasks will also be examined with Siemens in order to establish what has caused the misclassification.

Overall, we set out to determine whether the IC3 algorithm could be used in Ladder Logic Verification on a large scale. IC3 can decide those Verification Tasks that were previously incomplete in a fraction of the time, reducing it from around two minutes to under a second whilst being complete. This gives a definitive decision on the safety of all Verification Tasks without having to sacrifice the number of transitions to save on the runtime. IC3 can determine all Verification Tasks considerably faster than the original approach using Inductive Verification and Bounded Model Checking, although there is still a question mark over the misclassification for which further discussions with Siemens will take place.

10.2 Possible Future Work

The IC3 algorithm [43, 44, 34] has already seen developments since the original implementation from Bradley [42] with the ability to address counterexamples to generalisation in Mishchenko's ABC implementation [74, 75]. In order to continue improving the IC3 algorithm, and Ladder Logic Verification as a whole, the following could be explored:

10.2.1 Printing Ladder Logic Invariants from IC3

The first is to return the Ladder Logic Invariants produced by the IC3 implementations. Currently, the ABC implementation prints that the verification is successful inside the ABC environment. However, we cannot trust this completely because we do not know what is being performed in this process. Bradley's original implementation does not include anything like this. If the implementations were extended so the Ladder Logic Invariants are returned, then Inductive Verification could be run again with the Invariant. If it has been correctly constructed, then the verification will pass, increasing the trust. This process was previously applied to the artificial examples and proved successful. By printing the Ladder Logic Invariants, they could be compared to the ones manually produced for these small examples that were proven to be correct. If they matched, it would also increase the trust in the algorithm and the transformation process.

10.2.2 Returning Counterexample Traces from IC3

To further increase the confidence in IC3 and the transformation, the counterexample trace feature from Bounded Model Checking [55, 30, 66, 54] could be implemented in IC3. When Bounded Model Checking determines a Verification Task is unsafe, it returns a counterexample trace which consists of all transitions taken to reach this state that fail. This, therefore, allows the Siemens engineers to locate the problem and amend the Rungs of the Ladder Logic Program so the Propositional formulas are correct. In its current form, IC3 lacks such a feature, although ABC does state after how many transitions the unsafe state is reached. However, the state space in these interlockings is so large that, in reality, it is irrelevant. IC3 is significantly quicker than Bounded Model Checking, and is complete, therefore if the counterexamples could be returned in a similar way it could be fully replaced by IC3 - providing the correctness issue is resolved. If the counterexample traces, consisting of why the Verification Task fails and is unsafe, can be returned they could be used to understand what is causing the misclassification. This could be achieved by checking the counterexamples produced by IC3 against known counterexamples from the Ladder Logic Verifier. If the result is the same then it increases trust in IC3, otherwise it helps to point to the problem.

10.2.3 Investigations into the Correctness Issues found in the Integration Testing

In its current form the process of verifying Ladder Logic Programs with IC3 cannot be fully trusted because when classifying the testbed from Siemens it did not produce identical results for Types A and B when compared to the results from the Ladder Logic Verifier. By adding the ability to print the Ladder Logic Invariants and for the counterexamples to be returned, it would increase the trust in the process. That would enable the verification to be performed with the Invariant, if one could be successfully built, or provide extra understanding into why the Ladder Logic Program is unsafe. These two features allow for analysis after running IC3 to increase the confidence in the classification or expose errors.

The current misclassification makes up 13.5% of the Siemens testbed. If broken down into just the Correctness check, out of those Verification Tasks classed as Types A and B by the Ladder Logic Verifier, 22% were incorrectly classified by IC3. From this it can be said that IC3 correctly classifies most Verification Tasks, assuming that the Ladder Logic Verifier is correct. This means the error is less likely to be in the transformation of the Ladder Logic Rungs as otherwise, in theory, many more would have been misclassified. Therefore, the error is likely to be in the transformation of the Safety Properties as these change each time. Whilst there was testing during the development, there may still be errors in the transformations. A second and more extended test suite could be developed to try to fill any gaps present from the first test suite. In order to achieve this larger examples of Ladder Logic would be needed so the results of the Aiger-fier can be compared on a larger scale by creating a manual comparison. The test suite should ideally be developed with the Siemens engineers in order to establish whether the AIGER files correctly match the Ladder Logic files and our understanding is correct.

The next stage is to work out whether the errors are due to a transformation error, mistake from Siemens or miscommunication. Those Verification Tasks of Types D and E have certain characteristics that, firstly, have been shown to be translated correctly with testing and, in some cases, solely rely on input variables. There is a potential difference in how the Safety Properties are unrolled because AIGER re-initialises all inputs whereas in the TPTP format, used by both Inductive Verification and Bounded Model Checking, it seems the values are passed on. What this means is a version of the Safety Property is stored for each bound, and the conjunction of these properties are checked (Figure 10.1). Whereas the AIGER files for IC3 only store the overall property. In theory in execution these methods should be equivalent with the values of the Safety Property being passed along with Bounded Model Checking and staying consistent with the value of the Safety Property in IC3. However, it could also be the case where two Safety Property versions have different values and potentially is why in same cases Bounded Model Checking fails and IC3 passes. To establish exactly what is correct, further discussions with Siemens will be needed, as with all other cases that have resulted in misclassifications to determine where the faults lie (if in fact there are any). Analysing the counterexample traces produced by Bounded Model Checking in Type B1 is also required to establish whether they are "fake" and hence why IC3 can produce a Ladder Logic Invariant.

```
fof(ax,conjecture,
(~ (~ vS6035_BM__U_0 & vS6035_BM__U_1) |  (vL5214_LOD_P__1 & vL5214__LODP_REL1Z_1)) &

(~ (~ vS6035_BM__U_1 & vS6035_BM__U_2) |  (vL5214_LOD_P__2 & vL5214__LODP_REL1Z_2)) &

(~ (~ vS6035_BM__U_2 & vS6035_BM__U_3) |  (vL5214_LOD_P__3 & vL5214__LODP_REL1Z_3)) &

(~ (~ vS6035_BM__U_3 & vS6035_BM__U_4) |  (vL5214_LOD_P__4 & vL5214__LODP_REL1Z_4)) &

(~ (~ vS6035_BM__U_4 & vS6035_BM__U_5) |  (vL5214_LOD_P__5 & vL5214__LODP_REL1Z_5)) &
```

Figure 10.1: An example of a formatted Safety Property file used in Bounded Model Checking which is a conjunction of the Safety Property after every bound

10.2.4 Further Improving IC3's Efficiency

Refinements have already been made to improve the IC3 algorithm seen in the ABC implementation with its ability to deal with counterexamples that cannot be grouped into a Cube. There are ways though to further improve the algorithm to decrease the runtimes further. Although in reality, with ABC taking just a few thousandths of a second this may not be required if ABC can be proven correct. A potential approach would be to apply multi-threading to IC3 following Chaki and Karimi [52] allowing for concurrent execution of the algorithm. The aim of applying both of these techniques would be to decrease the number of iterations required by the algorithm, therefore further improving the runtime.

Glossary

Cube A group of states formed by dropping one literal. Used by the advanced version of IC3 to remove a group of counterexamples in a single iteration. . 38, 40–42, 45, 46, 113, 130

I The set of starting states of a Ladder Logic Program. . 26, 32–37, 40, 41, 43, 46, 59, 69, 74, 87, 98, 132

IC3 Invariant The IC3 algorithm [43, 44, 34] can be used to automatically generate Invariants for use in Ladder Logic verification process. To ensure that the Ladder Logic Invariant candidates (Ladder Logic Invariant Candidates) built are correct and they must be proven as IC3 Invariants (IC3-Inv) with the four IC3-Inv conditions that make up the algorithm. It can therefore be said that an $IC3 - Inv \equiv (IC3 - Inv1 \wedge IC3 - Inv2 \wedge IC3 - Inv3 \wedge IC3 - Inv4)$, otherwise it will not hold and function correctly. . 32, 34, 35

IC3 Invariant Candidate A potential IC3 Invariant built in the sequence $F_0, \ldots, F_h$. . 36

Ladder Logic Invariant Ladder Logic Invariants are used in verification. They are formulas that act as an extra pre-condition refining when a transition T can take place, Inductively Strengthening the program. The Loop Invariant is therefore a formula that includes at least the reachable states, permitting all the required transitions to take place while still blocking the potential False Positives [51]. With a Ladder Logic Invariant, Inductive Verification's [64] second clause becomes:

Show that the following holds for all states s: If s is safe and the invariant holds in s, then all successors of s are safe as well.

The final Ladder Logic Invariant is the one in the sequence Ladder Logic Invariant Candidates Ladder Logic Invariant Candidates $F_0, \ldots, F_h$ that prevents a False Positive problem in the verification process whilst still permitting the safe transitions. . 10–14, 28, 29, 32–34, 36, 38, 42, 43, 50, 53, 54, 75, 79, 80, 82–84, 89, 90, 94–96, 107, 108, 111–118, 120, 121, 126, 128, 129, 132, 145, 147

Ladder Logic Invariant Candidate A potential Ladder Logic Invariant built in the sequence $F_0, \ldots, F_h$. . 34–42, 79, 116, 117, 120, 131

Ladder Logic Program A program written in Ladder Logic [66] made up of a set of Rungs representing a set of states. The starting state(s) of the program are given by I, and the transitions between states by T. . 2, 3, 6–8, 10–13, 22, 26, 27, 29, 31–34, 36, 37, 41, 44, 45, 48–50, 53, 54, 57, 65, 68, 69, 73–76, 78–81, 83–85, 88–90, 93–98, 107–109, 113, 115, 116, 118, 120, 125–128, 131–133, 146, 147

P A Boolean Propositional formula dictating a set of states deemed to be safe. . 3, 6–8, 32–43, 45, 49, 59, 69, 71, 77, 93–95, 98, 107, 116, 118, 133

Rung A "Rung" in the ladder represents a Propositional rule [17] consisting of an equivalence. . 21, 22, 54, 59, 60, 69, 71, 73, 74, 98, 102, 128, 129, 132, 133

T The set of potential movements between states in a Ladder Logic Program. . 8, 22, 32–38, 40, 43, 81, 108, 112, 132

TPTP The TPTP (Thousands of Problems for Theorem Provers) is a library of test problems for Automated Theorem Proving systems. The language allows for ATP problems and ATP solutions, while being flexible and extensible, making the files easy to process. . 12, 57

Type A A Verification Task holds under Inductive Verification and is proven to be safe. . 6–8, 10, 13, 86–88, 106, 108, 115–118, 122, 126

Type A1 Inductive Verification deems the Verification Task to be safe as with Type A, however, IC3 returns that the Verification Task is unsafe as a Ladder Logic Invariant cannot be found. . 107–109, 117, 118, 127

Type B The Verification Task has failed Inductive Verification and a counterexample trace is found within the maximum bound under Bounded Model Checking and proven to be unsafe. . 6, 7, 10, 11, 13, 35, 86–88, 90, 106, 108, 115–118, 122, 126

Type B1 Bounded Model Checking has found a counterexample as with Type B, however, IC3 returns that the Verification Task is safe as a Ladder Logic Invariant could be found. . 107–109, 117, 118, 126, 129

Type C Inductive Verification fails. No counterexample trace is found for a Verification Task within the maximum bound under Bounded Model Checking. Unsure if it is safe or the counterexample is yet to be discovered. . 6–8, 10, 13, 86–91, 106–109, 111–118, 122, 125, 126, 133

Type C False No counterexample trace is found for a Verification Task within the maximum bound under Bounded Model Checking and is classified as Type C. IC3 returns false and states the Ladder Logic Program is unsafe. Therefore, a counterexample is present. . 108, 112, 114, 115

Type C True No counterexample trace is found for a Verification Task within the maximum bound under Bounded Model Checking and is classified as Type C. IC3 returns true and states the Ladder Logic Program is safe. Therefore, a False Positive is present. . 108, 112–115

Verification Task A set of Ladder Logic [66] Rungs with a Safety Property P. . 6–8, 10, 11, 13, 29, 42, 54, 83, 86–91, 106–109, 111–122, 125–129, 132, 133, 148

Bibliography

[1] De Morgan's Laws. Brilliant.org. https://brilliant.org/wiki/de-morgans-laws/. Online; accessed 8. November. 2021.

[2] Junit 5. https://junit.org/junit5/. Online; accessed 20 February 2022.

[3] Max-Planck-Institut für Informatik: SPASS Workbench. https://www.mpi-inf.mpg.de/departments/automation-of-logic/software/spass-workbench/. Online; accessed 18. Sep. 2020.

[4] TPTP Format for Derivations. http://tptp.cs.miami.edu/TPTP/QuickGuide/Derivations.html. Online; accessed 8. September. 2022.

[5] UPPAAL. https://uppaal.org/. Online; accessed 18. April. 2022.

[6] Generalization. https://en.wikipedia.org/wiki/Generalization, 10 2001. Online; accessed 21. January. 2022.

[7] Secure Shell. https://en.wikipedia.org/wiki/Secure_Shell, 11 2001. Online; accessed 20 February 2022.

[8] The TPTP Problem Library for Automated Theorem Proving. https://www.tptp.org/, Mar 2001. Online; accessed 25. May. 2023.

[9] MiniSat Page. http://minisat.se/MiniSat.html, 2003. Online; accessed 24. May. 2020.

[10] Interlocking. https://en.wikipedia.org/wiki/Interlocking, 2004. Online; accessed 12. October. 2022.

[11] PuTTY. https://en.wikipedia.org/wiki/PuTTY, 4 2004. Online; accessed 20 February 2022.

[12] And-inverter graph. https://en.wikipedia.org/wiki/And-inverter_graph, 2006. Online; accessed 12. July. 2022.

[13] Binary expression tree. https://en.wikipedia.org/wiki/Binary_expression_tree, 2010. Online; accessed 08. September. 2022.

[14] TRiLOGI — Triangle Research International. https://www.triplc.com/trilogi.htm, April 2011. Online; accessed 8. Jul. 2020.

[15] Why3. https://why3.lri.fr/, 2012. Online; accessed 12. April. 2022.

[16] Engineering Essentials: What Is a Programmable Logic Controller? https://www.machinedesign.com/learning-resources/engineering-essentials/article/21834250/engineering-essentials-what-is-a-programmable-logic-controller, Jun 2015. Online; accessed 28. Oct. 2020.

[17] PLC Ladder Logic Programming Tutorial (Basics) — PLC Academy. https://www.plcacademy.com/ladder-logic-tutorial/, May 2015. Online; accessed 12. Jun. 2020.

[18] Thameslink signalling update. https://www.railengineer.co.uk/thameslink-signalling-update/, Mar 2017. Online; accessed 29. May. 2023.

[19] Satisfiability and SAT solvers. https://cse.buffalo.edu/~erdem/cse331/support/sat-solver/index.html, 2018. Online; accessed 11. April. 2021.

[20] What Is Unit Testing? https://smartbear.com/learn/automated-testing/what-is-unit-testing/, 11 2018. Online; accessed 28. November. 2022.

[21] IEC 61131. https://en.wikipedia.org/wiki/IEC_61131, July 2020. Online; accessed 17. Oct. 2020.

[22] Vulnerability Scans and False Positives. https://www.itgovernance.co.uk/blog/vulnerability-scans-false-positives-the-importance-of-sanitising-input, Aug 2020. Online; accessed 20. April. 2023.

[23] Interlocking System. https://www.mobility.siemens.com/uk/en/portfolio/rail-solutions/automation/interlocking-systems.html/, 2021. Online; accessed 09. April. 2021.

[24] Ladder Logic Basics. https://ladderlogicworld.com/ladder-logic-basics/, 02 2021. Online; accessed 10. November. 2022.

[25] Classification: True vs. False and Positive vs. Negative. https://developers.google.com/machine-learning/crash-course/classification/true-false-positive-negative, Jul 2022. Online; accessed 20. April. 2023.

[26] Catalogue of Network Rail Standards. https://www.networkrail.co.uk/wp-content/uploads/2022/03/Catalogue-of-NR-Standards-Issue-123.pdf, Mar 2023. Online; accessed 29. May. 2023.

[27] European Train Control System (ETCS). https://www.mobility.siemens.com/global/en/portfolio/rail/automation/automatic-train-control/european-train-control-system.html, April 2023. Online; accessed 29. May. 2023.

[28] Awati, R. integration testing or integration and testing (I&T). https://www.techtarget.com/searchsoftwarequality/definition/integration-testing#:~:text=Integration%20testing%20%2D%2D%20also%20known,tested%20as%20a%20combined%20entity., 2022. Online; accessed 10. July. 2023.

[29] Bacelar Almeida, C. Validity Checking, 2010. Online; accessed 05. June. 2023.

[30] Baier, C., and Katoen, J.-P. *Principles of Model Checking (Representation and Mind Series)*. The MIT Press, 2008.

[31] Baradi, D., and Xavier, J. Relay logic programming explained. pp. 1–11.

[32] Barrett, C., and Tinelli, C. *Satisfiability Modulo Theories*. 05 2018, pp. 305–343.

[33] Belo Lourenço, C., Cousineau, D., Faissole, F., Marché, C., Mentré, D., and Inoue, H. Automated formal analysis of temporal properties of Ladder programs. *Int. J. Softw. Tools Technol. Transf. 24*, 6 (dec 2022), 977–997.

[34] Ben-David, S. Symbolic Model Checking: The IC3 Algorithm.

[35] Berre, D. SAT4J. http://sat4j.org/. Online; accessed 5. March. 2021.

[36] Biere, A. aiger. https://github.com/arminbiere/aiger. Online; accessed 10. October. 2021.

[37] Biere, A. The aiger and-inverter graph (aig) format version 20070427.

[38] Biere, A., Heljanko, K., and Wieringa, S. Aiger 1.9 and beyond. FMV Technical Reports ; 11/2, Linz : Johannes Kepler Universität Linz, Intitut für Formale Modelle und Verifikation.

[39] Birgmeier, J., Bradley, A. R., and Weissenbacher, G. Counterexample to induction-guided abstraction-refinement (ctigar). In *Computer Aided Verification* (Cham, 2014), A. Biere and R. Bloem, Eds., Springer International Publishing, pp. 831–848.

[40] Bjorner, N. Z3Prover/z3. https://github.com/Z3Prover/z3. Online; accessed 26. Oct. 2020".

[41] Bolton, W. *Programmable Logic Controllers*. 12 2009, pp. 1–19.

[42] Bradley, A. R. IC3ref. https://github.com/arbrad/IC3ref. Online; accessed 10. October. 2021.

[43] Bradley, A. R. Sat-based model checking without unrolling. In *Verification, Model Checking, and Abstract Interpretation* (2011), R. Jhala and D. Schmidt, Eds., Springer Berlin Heidelberg, pp. 70–87.

[44] Bradley, A. R. Understanding IC3. In *Theory and Applications of Satisfiability Testing – SAT 2012* (2012), A. Cimatti and R. Sebastiani, Eds., Springer Berlin Heidelberg, pp. 1–14.

[45] Bryant, H. Aiger-fier. `https://github.com/HarryBryant99/Aiger-fier`. Online; accessed 20. May. 2022.

[46] Bryant, H. IC3. `https://github.com/HarryBryant99/IC3`. Online; accessed 28. March. 2021.

[47] Bryant, H. SafetyPropertyParser. `https://github.com/HarryBryant99/SafetyPropertyParser`. Online; accessed 20. May. 2022.

[48] Bryant, H. Making Siemens' LL-Verifier More Efficient by Constructing Invariants. Undergraduate Dissertation.

[49] Brzozowski, J. A., and Seger, C.-J. H. *Ternary Simulation*. Springer New York, New York, NY, 1995, pp. 113–141.

[50] Chadwick, S., James, P., Roggenbach, M., and Werner, T. Formal Methods for Industrial Interlocking Verification. In *2018 International Conference on Intelligent Rail Transportation (ICIRT)* (2018), pp. 1–5.

[51] Chadwick, S., Werner, T., and Siemens Mobility. Formal Methods From Theory towards Practice, 2020.

[52] Chaki, S., and Karimi, D. Model Checking with Multi-threaded IC3 Portfolios. In *Proceedings of the 17th International Conference on Verification, Model Checking and Abstract Interpretation* (2016), Springer, pp. 517–535.

[53] Chidrewar, S., and Redekar, V. *PLC Based Washing Machine*, vol. 5. IRJET, 2018.

[54] Clarke, E. Bounded Model Checking with SAT/SMT.

[55] Clarke, E., Grumberg, O., and Peled, D. *Model Checking*. 01 2001.

[56] Cousineau, D., Mentré , D., and Inoue, H. Automated deductive verification for ladder programming. *Electronic Proceedings in Theoretical Computer Science 310* (dec 2019), 7–12.

[57] Dinesh, P., Gopinathan, K., Dhamodharan, M., Devarajan, J., Imayavaramban, N., and Kumar, P. A. *PLC BASED AUTOMATIC HYGIENIC DISH WASHING MACHINE*. International journal of scientific research and innovations X11, 2018.

[58] Franco, J., and Martin, J. A history of satisfiability. *Frontiers in Artificial Intelligence and Applications 185* (01 2009).

[59] Gruner, S., Kumar, A., Maibaum, T., and Roggenbach, M. *On the Construction of Engineering Handbooks with an Illustration from the Railway Safety Domain*, 1 ed. 2191-5768. Springer, 2020.

[60] Gurfinkel, A. Unbounded Model Checking: IC3 and PDR. `https://ece.uwaterloo.ca/~agurfink/ece750t29f18/assets/pdf/05_IC3_PDR.pdf`, 2018. Online; accessed 11. June. 2023.

[61] Hajdu, A., and Micskei, Z. Efficient strategies for cegar-based model checking. *Journal of Automated Reasoning 64* (08 2020).

[62] Hassan, Z., Bradley, A., and Somenzi, F. Better generalization in ic3. pp. 157–164.

[63] Iliasov, A., Taylor, D., Laibinis, L., and Romanovsky, A. Formal verification of signalling programs with safecap. In *Computer Safety, Reliability, and Security* (Cham, 2018), B. Gallina, A. Skavhaug, and F. Bitsch, Eds., Springer International Publishing, pp. 91–106.

[64] James, P., Lawrence, A., Moller, F., Roggenbach, M., Seisenberger, M., Setzer, A., Kanso, K., Chadwick, S., and Swansea Railway Verification Group. *Verification of Solid State Interlocking Programs*, vol. 8368. Springer-Verlag London, March 2014.

[65] James, P., Moller, F., and Roggenbach, M. Software Model Checking of Interlocking Programs. In *Applicable Formal Methods for Safe Industrial Products – Essays Dedicated to Jan Peleska on the Occasion of His 65th Birthday.*, A. E. Haxthausen, W.-l. Huang, and M. Roggenbach, Eds., LNCS 14165. Springer, To appear.

[66] James, P., and Swansea University. School of Physical Sciences. Computer Science. *SAT- Based Model Checking and Its Applications to Train Control Systems*. Swansea University, 2010.

[67] Kanso, K., Moller, F., and Setzer, A. Verification of safety properties in railway interlocking systems defined with ladder logic. In *AVOCS08* (2008), M. Calder and A. Miller, Eds., Glasgow University.

[68] Lee, J., Kim, S., and Bae, K. Bounded Model Checking of PLC ST Programs using Rewriting Modulo SMT. *Proceedings of the 8th ACM SIGPLAN International Workshop on Formal Techniques for Safety-Critical Systems* (2022).

[69] Lüth, C., Roggenbach, M., and Schröder, L. CCC - The CASL Consistency Checker. vol. 3423, pp. 94–105.

[70] Manolios, P. Boolean-Logic-Notes. `https://www.ccs.neu.edu/home/harshrc/courses/cs2800-fall2010/Boolean-Logic-Notes.pdf`, 2009. Online; accessed 05. June. 2023.

[71] McMillan, K. L. Interpolation and sat-based model checking. In *Computer Aided Verification* (Berlin, Heidelberg, 2003), W. A. Hunt and F. Somenzi, Eds., Springer Berlin Heidelberg, pp. 1–13.

[72] Microsoft Research. Z3 Theorem Prover, 2020.

[73] Minea, M. Conjunctive Normal Form. Tseitin Transform.

[74] Mishchenko, A. abc. `https://github.com/berkeley-abc/abc`. Online; accessed 21. October. 2021.

[75] Mishchenko, A. ABC: A system for Sequential Synthesis and Verification. `https://people.eecs.berkeley.edu/~alanmi/abc/`. Online; accessed 07. November. 2021.

[76] Moller, F., and Struth, G. *Modelling Computing Systems*. Springer, 2013.

[77] Mossakowski, T., Maeder, C., and Lüttich, K. The Heterogeneous Tool Set. In *Proc. 13th Intl. Conf. Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2007)* (2007), O. Grumberg and M. Huth, Eds., vol. 4424 of *Lect. Notes Comp. Sci.*, Springer, pp. 519–522.

[78] Mosses, P. D. *CASL Reference Manual — The Complete Documentation of the Common Algebraic Specification Language*, vol. 2960 of *Lect. Notes Comp. Sci.* Springer, 2004.

[79] Network Rail. Signalling principles handbook: Interlocking principles (former railway group standard gk/rt0060) - nr/l2/sig/30009/gkrt0060.

[80] Pfahringer, B. *Conjunctive Normal Form*. Springer US, Boston, MA, 2010, pp. 209–210.

[81] Pilling, G., Chattopadhyay, A., Guha Mazumder, K., Pan, W., Mendrin, M., Celestian, B., Ball, R., Khim, J., and Kau, A. Propositional Logic. `https://brilliant.org/wiki/propositional-logic/`, 2021. Online; accessed 10. April. 2021.

[82] Rail, N. Signalling Principles Handbook: Interlocking Principles (Former Railway Group Standard GK/RT0060) - NR/L2/SIG/30009/GKRT0060Signalling Principles Handbook: Interlocking Principles (Former Railway Group Standard GK/RT0060) - NR/L2/SIG/30009/GKRT0060.

[83] Improving performance, safety and software development of railway signalling. `https://results2021.ref.ac.uk/impact/a117e4ed-a960-4dc6-8e13-8c98d8ea5aef?page=1`.

[84] Roggenbach, M., Cerone, A., Schlingloff, B.-H., Schneider, G., and Shaikh, S. A. *Formal Methods for Software Engineering Languages, Methods, Application Domains*, 1 ed. 1862-4499. Springer International Publishing, 2021.

[85] Romanov, V. PLC Programming — How to Read Ladder Logic & Ladder Diagrams. `https://www.solisplc.com/tutorials/how-to-read-ladder-logic`, 2023. Online; accessed 05. June. 2023.

[86] Sheeran, M., Singh, S., and Stålmarck, G. Checking safety properties using induction and a sat-solver. In *Formal Methods in Computer-Aided Design* (Berlin, Heidelberg, 2000), W. A. Hunt and S. D. Johnson, Eds., Springer Berlin Heidelberg, pp. 127–144.

[87] Siemens Mobility UK, and Swansea Railway Verification Group. LL-Verifier, 2019. Siemens front-end combined with a Swansea back-end, using Z3 as a proof engine.

[88] Suda, M. Property directed reachability for automated planning. *Proceedings of the International Conference on Automated Planning and Scheduling 24* (2014), 540–541.

[89] Turing, A. 395Intelligent Machinery (1948). In *The Essential Turing*. Oxford University Press, 09 2004.

[90] University of California Berkeley. Berkeley Logic Interchange Format (BLIF).

[91] Wahl Thomas. The k-Induction Principle.

[92] Werner, T. Safety Verification of Ladder Logic Programs for Railway Interlockings.

[93] Yoo, J., Cha, S., Son, H. S., Kim, C. H., and Lee, J.-S. Plc-based safety critical software development for nuclear power plants. In *Computer Safety, Reliability, and Security* (Berlin, Heidelberg, 2004), M. Heisel, P. Liggesmeyer, and S. Wittmann, Eds., Springer Berlin Heidelberg, pp. 155–165.

[94] Zhong, P., Martonosi, M., and Malik, S. *Boolean Satisfiability*. 12 2008, pp. 613–636.

[95] Zoubek, B., Roussel, J.-M., and Kwiatkowska, M. Towards automatic verification of ladder logic programs.

Part IV

Appendix

APPENDIX A

Challenges faced when using Standard Implementations of IC3

Installing the two IC3 [43, 44, 34] implementations and understanding the AIGER was not as straight-forward as hoped. The two IC3 [43, 44, 34] implementations chosen to be tested were firstly IC3ref [42] by Bradley that follows the original algorithm. The second is the ABC implementation [74, 75] by Alan Mishchenko that follows a paper by Zyad Hassan and Fabio Somenzi working alongside Bradley to improve the algorithm's generalisation [62]. Both implementations attempt to find a Ladder Logic Invariant that will block a False Positive when Inductive Verification is performed. The implementations both use the AIGER file format [38] as an Input and return true if the algorithm can successfully construct a Ladder Logic Invariant. It is also beneficial to do this because the implementations have been developed for Linux environments. There were also issues in understanding the AIGER file format [38] so that the transformation from Ladder Logic [66] to AIGER was correct. The transformation, developed in this thesis, had to be refined on multiple occasions to ensure that for example the Input variables were dealt with correctly.

A.1 Machines for the Transations and Testing

Two machines were used in this project. The first is a high-performance Linux machine which was used to run all of experiments with the Ladder Logic Verifier and IC3 implmentations because of its large amount of cores and memory. The second is a Windows Machine used to perform the translations from Ladder Logic to AIGER as this was relatively inexpensive by comparison.

A.1.1 Linux Machine

Both these variants of IC3 have been installed and run successfully on a high-performance Linux machine running Ubuntu version 18.04.1, released in July 2018. This machine has been accessed remotely using PuTTY [11] and SSH [7]. It has 128 2194.443 MHz AMD Ryzen Threadripper 3990X 64-Core Processors using x86_74 architecture, running 2 threads per core, and a cache size of 512 KB.

A.1.2 Windows Machine

The two implementations were initially to be installed on a Windows Machine running 64-bit Windows 10. This machine has an Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz and 8 GB of RAM. However, because of the drastic advantage in performance in remote accessing the Linux machine it was chosen that the testing should be conducted on that as it is more suited to the large industrial scale interlockings. The Windows machine though was used for translating the Ladder Logic Program into AIGER.

A.2 IC3ref by Bradley

The first implementation installed was the original development by Bradley [42]. This implementation takes in files in the Aiger format [38], in both as a ASCII .aag file and a binary .aig file. The C++ code can be found on GitHub (`https://github.com/arbrad/IC3ref`) and provides documentation on the setup. However, as the last development was in 2015 extra commands are required in order to get the make files running.

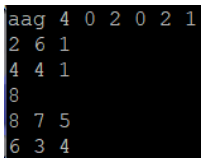
A.2.1 Installing IC3ref

IC3ref utilises the Sat Solver [19] MiniSat [9] to perform the various checks that make up the IC3 algorithm [43, 44, 34]. The latest version of MiniSat is used in IC3ref and can be found here: `https://github.com/niklasso/minisat`. This needs to be unpacked, and then the make file ran so that the Solver.h file is found in $ic3ref/minisat/minisat/core/Solver.h$. When running the make file the command might need an extra flag: *make CXXFLAGS="-fpermissive"*. This is an extra flag to give to the C++ compiler that allows the latest versions of gcc to be run on the make file. For the making of IC3ref itself, the same *-fpermissive* flag was needed for the Windows version and the command became *make CFLAGS="-fpermissive"*, however this was not required for the Linux version.

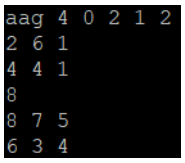
IC3ref works on AIGER files, therefore the AIGER libraries have to be installed from `http://fmv.jku.at/aiger/` with version 1.9.4. This also has to be unpacked so that the file aiger.c is at $ic3ref/aiger/aiger.c$. However, the make file does not need to be ran for AIGER. The final stage of the setup is to run IC3ref's make file. This process creates an IC3 executable which can perform IC3.

A.2.2 Running IC3ref

Contact was made with Armin Biere, at Johannes Kepler University Linz, who has worked on the development of AIGER and published many of the versions including the AIGER library [36]. The correspondence established that because of the age in Bradley's implementation it might not function correctly because the latest versions of the AIGER format can handle also non-zero initialised latches. This allows the latches to either start as false (0), or true (1). Because of this it was suggested that utility programs "aigmove" and "aigreset" [36] were run prior to IC3. The program "aigmove" treats non-primary outputs as primary outputs, allowing the Safety Property to be treated as an output (Figure A.1), meaning that if the output is true the unsafe states have be reached because in AIGER verification the negation of the Safety Property is used. Whilst "aigreset" normalises constant reset either to 0 or 1. The examples from Siemens have latches that are all initialised to 0, therefore this will not have affected the overall results. To run IC3, the command now becomes ./aigmove file.aag—./aigreset —./IC3. Correspondence with Martina Seidl at Johannes Kepler University Linz, who has also worked on the development of Aiger [38] and its uses, it was established that the output is 1 if the unsafe state is reached. Therefore, when the IC3ref implementation returns a 1, it is returning false stating that the IC3 algorithm cannot produce a Ladder Logic Invariant. And a 0 is when a successful Ladder Logic Invariant is found and therefore the unsafe states were not reached.



(a) Two variable Ladder Logic Program (Figure 1.1) written in AIGER using the Invariant Property to define the Unsafe States



(b) Running "aigmove" on the Two variable Ladder Logic Program (Figure 1.1) AIGER file, moving the unsafe state property to an output

Figure A.1: An example AIGER file before and after running "aigmove"

A.3 ABC by Mishchenko

The second implementation installed on the Linux machine is the ABC [74, 75] version of IC3 [43, 44, 34] developed by Alan Mischenko. This follows the work on improving the way the IC3 algorithm generalises counterexample states [62] to make it more efficient and therefore in theory more scalable. The C code for this can also be found on GitHub (`https://github.com/berkeley-abc/abc`) and provides documentation but like IC3ref requires further steps in order to get it running successfully. The difference this time is while ABC still takes in Aiger files, it only accepts the Binary .aig files and not the ASCII .aag files. However, it does also accept Berkeley Logic Interchange Format (BLIF) file [90].

A.3.1 Installing ABC

The package *libreadline-dev* had to be installed onto the linux machine. This was not included in the documentation on GitHub but is required in order for the make file to run. This can be done via the command: sudo apt-get install -y libreadline-dev. After which the code can be installed from the GitHub repository, and the make file ran with no flags required. By running, the command "make libaba.a" it compiles ABC as a static library.

A.3.2 Running ABC

ABC is able to perform various model checking tasks, a full list of the ABC commands can be found at `https://people.eecs.berkeley.edu/~alanmi/abc/` but this does not include the command for IC3. Because ABC only accepts the Binary AIGER files the ASCII files must be converted. For this it was suggested by Armin Biere that the program "aigtoaig" could be used to achieve this. This converts the files to the binary format as seen in Figure 3.14.

Originally, the "dprove" command was being ran which performs unbounded sequential equivalence checking - not IC3. After investigating the files in the GitHub repository it was found in a comment in one of the files in the proof folder that the "pdr" command is the one used for IC3:

// Abc_Print(1, "Running PDR by Niklas Een (aka IC3 by Aaron Bradley) with these parameters:\n");

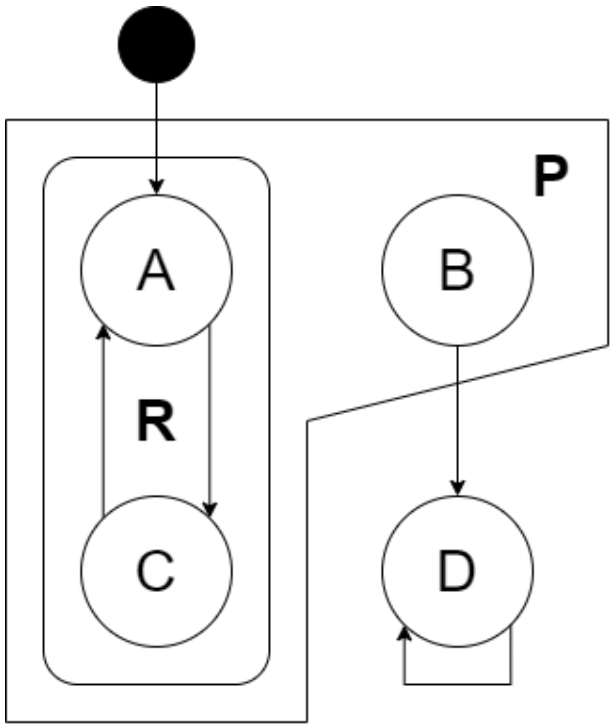
Therefore, the demo program was edited so that to performt the IC3 commands. Firstly using the "read_aiger" command followed by "pdr". A script was developed to run this for all Verification Tasks - scripts were also written to run Inductive Verification, Bounded Model Checking, and IC3ref in bulk.

APPENDIX B

Manual Ladder Logic Verification

B.1 Artificial Ladder Logic Examples

B.1.1 Two Variable Ladder Logic Example



P = Safe States

R = Reachable States

spec TRANSITION =
  preds x, y, x', y' : ()
  • $x' \Leftrightarrow \neg\, x \wedge y$
  • $y' \Leftrightarrow y$
end

spec TRANSITIONTEST =
  TRANSITION
then %implies
  • $x \wedge y \Rightarrow \neg\, x' \wedge y'$ %(t1)%
  • $\neg\, x \wedge y \Rightarrow x' \wedge y'$ %(t2)%
  • $x \wedge \neg\, y \Rightarrow \neg\, x' \wedge \neg\, y$ %(t3)%
end

spec SAFEINITIAL =
  TRANSITION
then • $x \wedge y$
then %implies
  • $x \vee y$ %(initial_safe)%
end

spec STEPSAFE =
  TRANSITION
then %implies
  • $x \vee y \Rightarrow x' \vee y'$ %(step_safety)%
end

spec STEPSAFEWITHINVARIANT =
  TRANSITION
then %implies
  • $(x \vee y) \wedge (y \vee \neg\, x) \Rightarrow x' \vee y'$
%(step_safety_with_invariant)%
end

spec INVARIANTHOLDSINITIALLY =
  TRANSITION
then • $x \wedge y$
then %implies
  • $y \vee \neg\, x$ %(invariant_initially)%
end

spec INVARIANTSTEP =

TRANSITION
then %implies
• $y \vee \neg\, x \Rightarrow y' \vee \neg\, x'$ %(Invariant_step)%
end

IC3 run through:

Part I: the formulae how we read them off the automaton and their CNFs

---

```
Initialisation Formula:

  I =  x /\ y

in CNF:

  CNF_CASL(I) = x /\ y

Transition Formula:

  T =       x' <=> not x /\ y
        /\  y' <=> y

  CNF_CASL(T) =    (x \/ not y \/ x')
                /\ (not x' \/ not x)
                /\ (not x' \/ y)
                /\ (not y' \/ y)
                /\ (not y \/ y')

Safety property:

  P  =  x \/ y
  P' = x' \/ y'

  CNF_CASL(P) = x \/ y
  CNF_CASL(P') = x' \/ y'


Checking with Hets that T and CNF_CASL(T) are equivalent:
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

spec Check1 =
  pred x,y,x',y': ()
```

```
then %implies
 axiom
              (       (x' <=> not x /\ y)

              )
           /\
              (
                 (y' <=> y)
              )

  <=>      (          (x \/ not y \/ x')
                   /\ (not x' \/ not x)
                   /\ (not x' \/ y)
                   /\ (not y' \/ y)
                   /\ (not y \/ y')

           )

end

[successfully proved with HETS interface]


Part II:
------------------------------------


once more the formulae (now all in CNF):

  I =  x /\ y

  T =      (x \/ not y \/ x')
        /\ (not x' \/ not x)
        /\ (not x' \/ y)
        /\ (not y' \/ y)
        /\ (not y \/ y')

  P = x \/ y

  P' = x' \/ y'

a) Check I=>P?
```

```
spec Check1 =
  pred x,y: ()
  axiom  x /\ y
then %implies
 axiom x \/ y
end

[successfully proved with HETS interface]
```

b) Check I/\T=>P‘?

```
spec Check2 =
  pred x,y,x',y': ()
  axiom  x /\ y
  axiom      (x \/ not y \/ x')
       /\ (not x' \/ not x)
       /\ (not x' \/ y)
       /\ (not y' \/ y)
       /\ (not y \/ y')
then %implies
 axiom x' \/ y'
end

[successfully proved with HETS interface]
```

c) Set F1 :=P

```
  F1 = (x \/ y)
```

d) For every clause c in clauses(I), if c not−in clauses(F1),
check:
    I /\ T => c‘?
    If it does, set F1 := F1 /\ c

```
clauses(I) = {x, y}
clauses(F1) = { (x \/ y)}
```

d−1) c = x

```
check if I /\ T => x':

spec Check3 =
  pred x,y,x',y': ()
  axiom  x /\ y
  axiom       (x \/ not y \/ x')
       /\ (not x' \/ not x)
       /\ (not x' \/ y)
       /\ (not y' \/ y)
       /\ (not y \/ y')
then %implies
 axiom x'
end

[disproved with Hets]

Thus, we do not add x' to F1.

d-2) c= y

spec Check4 =
  pred x,y,x',y': ()
  axiom  x /\ y
  axiom       (x \/ not y \/ x')
       /\ (not x' \/ not x)
       /\ (not x' \/ y)
       /\ (not y' \/ y)
       /\ (not y \/ y')
then %implies
 axiom y'
end

[proved with Hets]

Thus, we add y to F1, and obtain:

   F1 =      (x \/ y)
        /\   y



Part III:
```

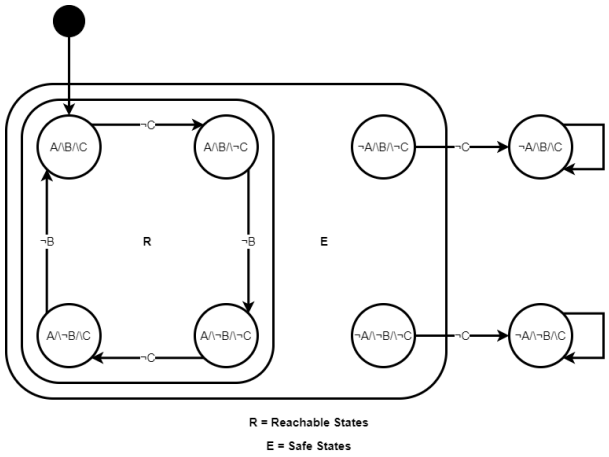
```
a) Check: Is it the case that Fk /\ T => P'?
b) If it is, then
 b-1) set Fk+1 := P
 b-2) for every clause c in Fk , check if Fk/\T=>c'?
      If it is, set Fk+1 := Fk+1 /\ c
c) If Fk =Fk+1: done


k=1:
++++

a) do we have F1 /\ T => P' ?

spec Check4 =
  pred x,y,x',y': ()
  axiom       (x \/ y)
        /\  y
  axiom     (x \/ not y \/ x')
       /\ (not x' \/ not x)
       /\ (not x' \/ y)
       /\ (not y' \/ y)
       /\ (not y \/ y')

then %implies
 axiom  x' \/ y'
end

[proved with Hets]

b-1) F2 = x \/ y

b-2) clauses(F1) = { (x \/ y) , y }

    let c = (x \/ y)
    check F1 /\ T => c'

spec Check4 =
  pred x,y,x',y': ()
  axiom       (x \/ y)
        /\  y
```

```
  axiom     (x \/ not y \/ x')
        /\ (not x' \/ not x)
        /\ (not x' \/ y)
        /\ (not y' \/ y)
        /\ (not y \/ y')
then %implies
 axiom  x' \/ y'
end

[successfully proved with HETS]

thus F2 = (x \/ y) /\ (x \/ y)


    let c = y
    check F1 /\ T => c'

spec Check4 =
  pred x,y,x',y': ()
  axiom       (x \/ y)
        /\  y
  axiom     (x \/ not y \/ x')
        /\ (not x' \/ not x)
        /\ (not x' \/ y)
        /\ (not y' \/ y)
        /\ (not y \/ y')
then %implies
 axiom  y'
end

[successfully proved with HETS]


thus F2 = (x \/ y) /\ (x \/ y) /\ y
        = (x \/ y) /\ y
        = F1

i.e. F2 = F1.

The algorithms terminates and we have proven that P holds.
```

```
Short version of the algorithm:
++++++++++++++++++++++++++++++

Used formulae:
--------------

  I =  x /\ y

  T =     (x \/ not y \/ x')
       /\ (not x' \/ not x)
       /\ (not x' \/ y)
       /\ (not y' \/ y)
       /\ (not y \/ y')

  P = x \/ y

  P' = x' \/ y'

Initialisation phase:
---------------------------------

a) Check I=>P?
      This property holds.
b) Check I/\T=>P'?
      Thus property holds.
c) Set F1 :=P
      F1 = (x \/ y).
d) For every clause c in clauses(I), if c not-in clauses(F1),
check:
    I /\ T => c'?
    If it does, set F1 := F1 /\ c

    Here: clauses(I) = {x, y}
          clauses(F1) = { (x \/ y)}

   for c = x, we obtain:  I /\ T => x' does not hold.
   for c = y, we obtain:  I /\ T => y' holds.

   Thus:  F1 =     (x \/ y)
               /\  y
```

```
Iteration phase for k=1:
------------------------

a) Check: Is it the case that F1 /\ T => P'?
          This property holds.

b) If it is, then

 b-1) set Fk+1 := P

    F2 = x \/ y

 b-2) for every clause c in F1 ,
check if F1/\T=>c'?
      If it is, set F2 := F2 /\ c

     clauses(F1) = { (x \/ y) , y }


    let c = (x \/ y):

        F1 /\ T => c' holds.
        Thus: F2 = (x \/ y) /\ (x \/ y)

    let c = y

        F1 /\ T => c' holds.
        Thus F2 = (x \/ y) /\ (x \/ y) /\ y
                = (x \/ y) /\ y

c) As F1 = F2, we are done and verification was successful.
```

B.1.2 Three Variable Ladder Logic Example



spec TRANSITION =
preds a : ();
b : ();
c : ()
preds a' : ();
b' : ();
c' : ()
• $a' \Leftrightarrow a$
• $b' \Leftrightarrow (a \wedge c) \vee (\neg\ a \wedge b)$
• $c' \Leftrightarrow \neg\ a \vee \neg\ b$
end

spec TRANSITIONTEST =
TRANSITION
then %implies
• $a \wedge b \wedge c \Rightarrow a' \wedge b' \wedge \neg\ c'$ %(t1)%
• $a \wedge b \wedge \neg\ c \Rightarrow a' \wedge \neg\ b' \wedge \neg\ c'$ %(t2)%
• $a \wedge \neg\ b \wedge \neg\ c \Rightarrow a' \wedge \neg\ b' \wedge c'$ %(t3)%
• $a \wedge \neg\ b \wedge c \Rightarrow a' \wedge b' \wedge c'$ %(t4)%
• $\neg\ a \wedge b \wedge \neg\ c \Rightarrow \neg\ a' \wedge b' \wedge c'$ %(t5)%
• $\neg\ a \wedge b \wedge c \Rightarrow \neg\ a' \wedge b' \wedge c'$ %(t6)%
• $\neg\ a \wedge \neg\ b \wedge \neg\ c \Rightarrow \neg\ a' \wedge \neg\ b' \wedge c'$ %(t7)%
• $\neg\ a \wedge \neg\ b \wedge c \Rightarrow \neg\ a' \wedge \neg\ b' \wedge c'$ %(t8)%
end

spec SAFEINITIAL =
TRANSITION

then • $a \wedge b \wedge c$
then %**implies**
• $a \vee \neg\ c$ %(initial_safe)%
end

spec StepSafe =
Transition
then %**implies**
• $a \vee \neg\ c \Rightarrow a' \vee \neg\ c'$ %(step_safety)%
end

spec StepSafeWithInvariant =
Transition
then %**implies**
• $(a \vee \neg\ c) \wedge (a \vee c) \Rightarrow a' \vee \neg\ c'$
%(step_safety_with_invariant)%
end

spec InvariantHoldsInitially =
Transition
then • $a \wedge b \wedge c$
then %**implies**
• $a \vee c$ %(invariant_initially)%
end

spec InvariantStep =
Transition
then %**implies**
• $a \vee c \Rightarrow a' \vee c'$ %(Invariant_step)%
end

IC3 run through:

```
Part I: the formulae how we read them off the automaton and their CNFs
-----------------------------------------------------------------------

Initialisation Formula:

  I =  a /\ b /\ c
in CNF:

  CNF_CASL(I) = a /\ b /\ c
```

```
Transition Formula:

  T =   a' <=> a
        /\ b' <=> ((a /\ c) \/ (not a /\ b))
        /\ c' <=> (not a \/ not b)

  CNF_CASL(T) = (not a \/ a')
        /\ (not a' \/ a)
        /\ (not a \/ not c \/ b')
        /\ (a \/ not b \/ b')
        /\ (b \/ a \/ not b')
        /\ (not a \/ c \/ not b')
        /\ (b \/ c \/ not b')
        /\ (a \/ c')
        /\ (b \/ c')
        /\ (not c' \/ not a \/ not b)

Safety property:

  P  =  a \/ not c
  P' = a' \/ not c'

  CNF_CASL(P) = a \/ not c
  CNF_CASL(P') = a' \/ not c'


Checking with Hets that T and CNF_CASL(T) are equivalent:
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

spec Check1 =
  pred a,b,c,a',b',c': ()
then %implies
 axiom
            (
                (a' <=> a)
            )
        /\
            (
                (b' <=> ((a /\ c) \/ (not a /\ b)))
            )
        /\
            (
```

```
                (c' <=> (not a \/ not b))
            )

  <=>     (          (not a \/ a')
                /\ (not a' \/ a)
                /\ (not a \/ not c \/ b')
                /\ (a \/ not b \/ b')
                /\ (b \/ a \/ not b')
                /\ (not a \/ c \/ not b')
                /\ (b \/ c \/ not b')
                /\ (a \/ c')
                /\ (b \/ c')
                /\ (not c' \/ not a \/ not b)
          )

end

[successfully proved with HETS interface]
```

Part II:

---

once more the formulae (now all in CNF):

```
  I =  a /\ b /\ c

  T =     (not a \/ a')
        /\ (not a' \/ a)
        /\ (not a \/ not c \/ b')
        /\ (a \/ not b \/ b')
        /\ (b \/ a \/ not b')
        /\ (not a \/ c \/ not b')
        /\ (b \/ c \/ not b')
        /\ (a \/ c')
        /\ (b \/ c')
        /\ (not c' \/ not a \/ not b)

  P = a \/ not c

  P' = a' \/ not c'
```

a) Check I=>P?

```
spec Check2 =
  pred a,b,c: ()
  axiom  a /\ b /\ c
then %implies
 axiom a \/ not c
end
```

[successfully proved with HETS interface]

b) Check I/\T=>P‘?

```
spec Check3 =
  pred a,b,c,a',b',c': ()
  axiom  a /\ b /\ c
  axiom       (not a \/ a')
        /\ (not a' \/ a)
        /\ (not a \/ not c \/ b')
        /\ (a \/ not b \/ b')
        /\ (b \/ a \/ not b')
        /\ (not a \/ c \/ not b')
        /\ (b \/ c \/ not b')
        /\ (a \/ c')
        /\ (b \/ c')
        /\ (not c' \/ not a \/ not b)
then %implies
 axiom a' \/ not c'
end
```

[successfully proved with HETS interface]

c) Set F1 :=P

```
  F1 = (a \/ not c)
```

d) For every clause c in clauses(I), if c not–in clauses(F1), check:
    I /\ T => c‘?
    If it does, set F1 := F1 /\ c

```
clauses(I) = {a, b, c}
clauses(F1) = { (a \/ not c)}

d-1) c = a

check if I /\ T => a':

spec Check4 =
  pred a,b,c,a',b',c': ()
  axiom  a /\ b /\ c
  axiom       (not a \/ a')
        /\ (not a' \/ a)
        /\ (not a \/ not c \/ b')
        /\ (a \/ not b \/ b')
        /\ (b \/ a \/ not b')
        /\ (not a \/ c \/ not b')
        /\ (b \/ c \/ not b')
        /\ (a \/ c')
        /\ (b \/ c')
        /\ (not c' \/ not a \/ not b)
then %implies
 axiom a'
end

[proved with Hets]

Thus, we add a to F1, and obtain:

   F1 =     (a \/ not c)
        /\ a

d-2) c= b

spec Check5 =
  pred a,b,c,a',b',c': ()
  axiom  a /\ b /\ c
  axiom       (not a \/ a')
        /\ (not a' \/ a)
        /\ (not a \/ not c \/ b')
        /\ (a \/ not b \/ b')
        /\ (b \/ a \/ not b')
```

```
        /\ (not a \/ c \/ not b')
        /\ (b \/ c \/ not b')
        /\ (a \/ c')
        /\ (b \/ c')
        /\ (not c' \/ not a \/ not b)
then %implies
 axiom b'
end
```

Thus, we add b to F1, and obtain:

```
   F1 =     (a \/ not c)
        /\ a
        /\ b
```

d–3) c= c

```
spec Check6 =
  pred a,b,c,a',b',c': ()
  axiom  a /\ b /\ c
  axiom      (not a \/ a')
        /\ (not a' \/ a)
        /\ (not a \/ not c \/ b')
        /\ (a \/ not b \/ b')
        /\ (b \/ a \/ not b')
        /\ (not a \/ c \/ not b')
        /\ (b \/ c \/ not b')
        /\ (a \/ c')
        /\ (b \/ c')
        /\ (not c' \/ not a \/ not b)
then %implies
 axiom c'
end
```

[disproved with Hets]

Thus, we do not add c' to F1.

Part III:

---

a) Check: Is it the case that Fk /\ T => P'?

```
b) If it is, then
 b-1) set Fk+1 := P
 b-2) for every clause c in Fk , check if Fk/\T=>c'?
      If it is, set Fk+1 := Fk+1 /\ c
c) If Fk =Fk+1: done


k=1:
++++

a) do we have F1 /\ T => P' ?

spec Check7 =
  pred a,b,c,a',b',c': ()
  axiom      (a \/ not c)
       /\ a /\ b
  axiom    (not a \/ a')
       /\ (not a' \/ a)
       /\ (not a \/ not c \/ b')
       /\ (a \/ not b \/ b')
       /\ (b \/ a \/ not b')
       /\ (not a \/ c \/ not b')
       /\ (b \/ c \/ not b')
       /\ (a \/ c')
       /\ (b \/ c')
       /\ (not c' \/ not a \/ not b)

then %implies
 axiom  a' \/ not c'
end

[proved with Hets]

b-1) F2 = a \/ not c

b-2) clauses(F1) = { (a \/ not c) , a , b }

    let c = (a \/ c)
    check F1 /\ T => c'

spec Check8 =
  pred a,b,c,a',b',c': ()
```

```
  axiom  (a \/ not c)
       /\ a /\ b
  axiom       (not a \/ a')
       /\ (not a' \/ a)
       /\ (not a \/ not c \/ b')
       /\ (a \/ not b \/ b')
       /\ (b \/ a \/ not b')
       /\ (not a \/ c \/ not b')
       /\ (b \/ c \/ not b')
       /\ (a \/ c')
       /\ (b \/ c')
       /\ (not c' \/ not a \/ not b)
then %implies
 axiom a' \/ not c'
end

[successfully proved with HETS]

thus F2 = (a \/ not c)
       /\ (a \/ not c)


    let c = a
    check F1 /\ T => c'

spec Check9 =
  pred a,b,c,a',b',c': ()
  axiom  (a \/ not c)
       /\ a /\ b
  axiom       (not a \/ a')
       /\ (not a' \/ a)
       /\ (not a \/ not c \/ b')
       /\ (a \/ not b \/ b')
       /\ (b \/ a \/ not b')
       /\ (not a \/ c \/ not b')
       /\ (b \/ c \/ not b')
       /\ (a \/ c')
       /\ (b \/ c')
       /\ (not c' \/ not a \/ not b)
then %implies
 axiom a'
end
```

```
[successfully proved with HETS]

thus F2 = (a \/ not c)
        /\ (a \/ not c)
        /\ a


    let c = b
    check F1 /\ T => c'

spec Check10 =
  pred a,b,c,a',b',c': ()
  axiom  (a \/ not c)
        /\ a /\ b
  axiom      (not a \/ a')
        /\ (not a' \/ a)
        /\ (not a \/ not c \/ b')
        /\ (a \/ not b \/ b')
        /\ (b \/ a \/ not b')
        /\ (not a \/ c \/ not b')
        /\ (b \/ c \/ not b')
        /\ (a \/ c')
        /\ (b \/ c')
        /\ (not c' \/ not a \/ not b)
then %implies
 axiom b'
end

[disproved with Hets]

Thus, we do not add b' to F2.


c) thus F2 = (a \/ not c) /\ (a \/ not c) /\ a
        = (a \/ not c) /\ a
        != F1

i.e. F2 = F1.

Part IV: Repeat part 3 until Fk =Fk+1:
```

```
a) Check: Is it the case that Fk /\ T => P'?
b) If it is, then
 b-1) set Fk+1 := P
 b-2) for every clause c in Fk , check if Fk/\T=>c'?
      If it is, set Fk+1 := Fk+1 /\ c
c) If Fk =Fk+1: done


k=1:
++++

a) do we have F2 /\ T => P' ?

spec Check11 =
  pred a,b,c,a',b',c': ()
  axiom       (a \/ not c)
        /\ a
  axiom     (not a \/ a')
        /\ (not a' \/ a)
        /\ (not a \/ not c \/ b')
        /\ (a \/ not b \/ b')
        /\ (b \/ a \/ not b')
        /\ (not a \/ c \/ not b')
        /\ (b \/ c \/ not b')
        /\ (a \/ c')
        /\ (b \/ c')
        /\ (not c' \/ not a \/ not b)

then %implies
 axiom  a' \/ not c'
end

[proved with Hets]

b-1) F3 = a \/ not c

b-2) clauses(F2) = { (a \/ not c) , a }

     let c = (a \/ c)
     check F2 /\ T => c'
```

```
spec Check12 =
  pred a,b,c,a',b',c': ()
  axiom  (a \/ not c)
       /\ a
  axiom      (not a \/ a')
       /\ (not a' \/ a)
       /\ (not a \/ not c \/ b')
       /\ (a \/ not b \/ b')
       /\ (b \/ a \/ not b')
       /\ (not a \/ c \/ not b')
       /\ (b \/ c \/ not b')
       /\ (a \/ c')
       /\ (b \/ c')
       /\ (not c' \/ not a \/ not b)
then %implies
 axiom a' \/ not c'
end

[successfully proved with HETS]

thus F3 = (a \/ not c) /\ a
       /\ (a \/ not c)


    let c = a
    check F1 /\ T => c'

spec Check13 =
  pred a,b,c,a',b',c': ()
  axiom  (a \/ not c)
       /\ a
  axiom      (not a \/ a')
       /\ (not a' \/ a)
       /\ (not a \/ not c \/ b')
       /\ (a \/ not b \/ b')
       /\ (b \/ a \/ not b')
       /\ (not a \/ c \/ not b')
       /\ (b \/ c \/ not b')
       /\ (a \/ c')
       /\ (b \/ c')
       /\ (not c' \/ not a \/ not b)
then %implies
```

```
 axiom a'
end

[successfully proved with HETS]

thus F3 = (a \/ not c) /\ a
        /\ (a \/ not c)
        /\ a


c) thus F3 = (a \/ not c) /\ a /\ (a \/ not c) /\ a
        = (a \/ not c) /\ a
        = F2

i.e. F3 = F2.

The algorithms terminates and we have proven that P holds.
```

```
Short version of the algorithm:
++++++++++++++++++++++++++++++++

Used formulae:
--------------

  I =  a /\ b /\ c

  T =     (not a \/ a')
        /\ (not a' \/ a)
        /\ (not a \/ not c \/ b')
        /\ (a \/ not b \/ b')
        /\ (b \/ a \/ not b')
        /\ (not a \/ c \/ not b')
        /\ (b \/ c \/ not b')
        /\ (a \/ c')
        /\ (b \/ c')
        /\ (not c' \/ not a \/ not b)

  P = a \/ not c

  P' = a' \/ not c'
```

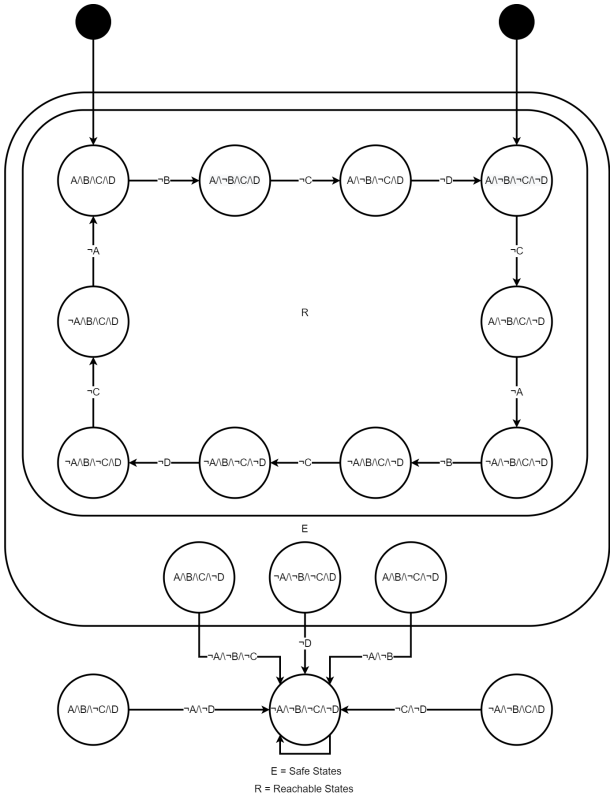
```
Initialisation phase:
---------------------

a) Check I=>P?
      This property holds.
b) Check I/\T=>P'?
      Thus property holds.
c) Set F1 :=P
      F1 = (a \/ not c).
d) For every clause c in clauses(I), if c not-in clauses(F1),
check:
    I /\ T => c'?
    If it does, set F1 := F1 /\ c

    Here: clauses(I) = {a, b, c}
          clauses(F1) = {a \/ not c}

   for c = a, we obtain:  I /\ T => a' holds.
   for c = b, we obtain:  I /\ T => b' holds.
   for c = c, we obtain:  I /\ T => c' does not hold.

   Thus:  F1 = (a \/ not c) /\ a /\ b

Iteration phase for k=1:
------------------------

Iteration 1:

a) Check: Is it the case that F1 /\ T => P'?
          This property holds.

b) If it is, then

 b-1) set Fk+1 := P

    F2 = a \/ not c

 b-2) for every clause c in F1 , check if F1/\T=>c'?
      If it is, set F2 := F2 /\ c

     clauses(F1) = { (a \/ not c) , a , b }
```

```
    let c = (a \/ not c):

        F1 /\ T => c' holds.
        Thus: F2 = (a \/ not c) /\ (a \/ not c)

    let c = a

        F1 /\ T => c' holds.
        Thus F2 = (a \/ not c) /\ (a \/ not c) /\ a

    let c = b

        F1 /\ T => c' doesn't hold.
        Thus F2 = (a \/ not c) /\ (a \/ not c) /\ a
                = (a \/ not c) /\ a

c) As F1 != F2, itertion phase must continue.

Iteration 2:

a) Check: Is it the case that F2 /\ T => P'?
          This property holds.

b) If it is, then

 b-1) set Fk+1 := P

    F3 = a \/ not c

 b-2) for every clause c in F2 , check if F2/\T=>c'?
      If it is, set F3 := F3 /\ c

     clauses(F2) = { (a \/ not c) , a }


    let c = (a \/ not c):

        F2 /\ T => c' holds.
        Thus: F3 = (a \/ not c) /\ (a \/ not c)
```

```
let c = a

    F2 /\ T => c' holds.
    Thus F3 = (a \/ not c) /\ (a \/ not c) /\ a
            = (a \/ not c) /\ a
```

c) As F2 = F3, we are done and verification was successful.

B.1.3 Four Variable Ladder Logic Example



spec TRANSITION =
preds a : ();
b : ();
c : ();
d : ()
preds a' : ();
b' : ();
c' : ();
d' : ()

- $a' \Leftrightarrow (b \wedge c \wedge d) \vee (a \wedge \neg\, b \wedge \neg\, c) \vee (a \wedge \neg\, b \wedge d)$
- $b' \Leftrightarrow (\neg\, a \wedge b) \vee (\neg\, a \wedge c \wedge \neg\, d)$
- c'
  $\Leftrightarrow (\neg\, b \wedge c \wedge \neg\, d) \vee (\neg\, a \wedge b \wedge d) \vee (b \wedge c \wedge d)$
  $\vee\ (a \wedge \neg\, b \wedge \neg\, d)$
- $d' \Leftrightarrow (\neg\, a \wedge b \wedge \neg\, c) \vee (\neg\, a \wedge b \wedge d) \vee (a \wedge c \wedge d)$

end

spec TransitionTest =
TRANSITION
then %implies

- $a \wedge b \wedge c \wedge d \Rightarrow a' \wedge \neg\, b' \wedge c' \wedge d'$ %(t1)%
- $a \wedge \neg\, b \wedge c \wedge d \Rightarrow a' \wedge \neg\, b' \wedge \neg\, c' \wedge d'$ %(t2)%
- $a \wedge \neg\, b \wedge \neg\, c \wedge d \Rightarrow a' \wedge \neg\, b' \wedge \neg\, c' \wedge \neg\, d'$ %(t3)%
- $a \wedge \neg\, b \wedge \neg\, c \wedge \neg\, d \Rightarrow a' \wedge \neg\, b' \wedge c' \wedge \neg\, d'$ %(t4)%
- $a \wedge \neg\, b \wedge c \wedge \neg\, d \Rightarrow \neg\, a' \wedge \neg\, b' \wedge c' \wedge \neg\, d'$ %(t5)%
- $\neg\, a \wedge \neg\, b \wedge c \wedge \neg\, d \Rightarrow \neg\, a' \wedge b' \wedge c' \wedge \neg\, d'$ %(t6)%
- $\neg\, a \wedge b \wedge c \wedge \neg\, d \Rightarrow \neg\, a' \wedge b' \wedge \neg\, c' \wedge \neg\, d'$ %(t7)%
- $\neg\, a \wedge b \wedge \neg\, c \wedge \neg\, d \Rightarrow \neg\, a' \wedge b' \wedge \neg\, c' \wedge d'$ %(t8)%
- $\neg\, a \wedge b \wedge \neg\, c \wedge d \Rightarrow \neg\, a' \wedge b' \wedge c' \wedge d'$ %(t9)%
- $\neg\, a \wedge b \wedge c \wedge d \Rightarrow a' \wedge b' \wedge c' \wedge d'$ %(t10)%
- $a \wedge b \wedge \neg\, c \wedge d \Rightarrow \neg\, a' \wedge \neg\, b' \wedge \neg\, c' \wedge \neg\, d'$ %(t11)%
- $a \wedge b \wedge c \wedge \neg\, d \Rightarrow \neg\, a' \wedge \neg\, b' \wedge \neg\, c' \wedge \neg\, d'$ %(t12)%
- $\neg\, a \wedge \neg\, b \wedge \neg\, c \wedge d \Rightarrow \neg\, a' \wedge \neg\, b' \wedge \neg\, c' \wedge \neg\, d'$ %(t13)%
- $a \wedge b \wedge \neg\, c \wedge \neg\, d \Rightarrow \neg\, a' \wedge \neg\, b' \wedge \neg\, c' \wedge \neg\, d'$ %(t14)%
- $\neg\, a \wedge \neg\, b \wedge c \wedge d \Rightarrow \neg\, a' \wedge \neg\, b' \wedge \neg\, c' \wedge \neg\, d'$ %(t15)%
- $\neg\, a \wedge \neg\, b \wedge \neg\, c \wedge \neg\, d \Rightarrow \neg\, a' \wedge \neg\, b' \wedge \neg\, c' \wedge \neg\, d'$ %(t16)%

end

spec SafeInitial =
TRANSITION
then $\bullet\ (a \wedge b \wedge c \wedge d) \vee (a \wedge \neg\, b \wedge \neg\, c \wedge \neg\, d)$
then %implies

- $(\neg\, d \vee c \vee \neg\, b \vee \neg\, a) \wedge (b \vee \neg\, d \vee a \vee \neg\, c) \wedge (b \vee c \vee a \vee d)$
  %(initial_safe)%

end

spec StepSafe =
TRANSITION
then %implies

- $(\neg\, d \vee c \vee \neg\, b \vee \neg\, a) \wedge (b \vee \neg\, d \vee a \vee \neg\, c) \wedge (b \vee c \vee a \vee d)$
  $\Rightarrow (\neg\, d' \vee c' \vee \neg\, b' \vee \neg\, a') \wedge (b' \vee \neg\, d' \vee a' \vee \neg\, c')$

$\wedge\ (b' \vee c' \vee a' \vee d')$

%(step_safety)%

end

spec StepSafeWithInvariant =
Transition
then %implies
• $(\neg\ d \vee c \vee \neg\ b \vee \neg\ a) \wedge (b \vee \neg\ d \vee a \vee \neg\ c) \wedge (b \vee c \vee a \vee d)$
$\wedge\ ((\neg\ d \vee c \vee \neg\ b \vee \neg\ a) \wedge (b \vee \neg\ d \vee a \vee \neg\ c)$
$\wedge\ (b \vee c \vee a \vee d) \wedge \neg\ (\neg\ a \wedge \neg\ b \wedge d) \wedge \neg\ (a \wedge b \wedge \neg\ d))$
$\Rightarrow (\neg\ d' \vee c' \vee \neg\ b' \vee \neg\ a') \wedge (b' \vee \neg\ d' \vee a' \vee \neg\ c')$
$\wedge\ (b' \vee c' \vee a' \vee d')$

%(step_safety_with_invariant)%

end

spec InvariantHoldsInitially =
Transition
then • $(a \wedge b \wedge c \wedge d) \vee (a \wedge \neg\ b \wedge \neg\ c \wedge \neg\ d)$
then %implies
• $(\neg\ d \vee c \vee \neg\ b \vee \neg\ a) \wedge (b \vee \neg\ d \vee a \vee \neg\ c) \wedge (b \vee c \vee a \vee d)$
$\wedge \neg\ (\neg\ a \wedge \neg\ b \wedge d) \wedge \neg\ (a \wedge b \wedge \neg\ d)$

%(invariant_initially)%

end

spec InvariantStep =
Transition
then %implies
• $(\neg\ d \vee c \vee \neg\ b \vee \neg\ a) \wedge (b \vee \neg\ d \vee a \vee \neg\ c) \wedge (b \vee c \vee a \vee d)$
$\wedge \neg\ (\neg\ a \wedge \neg\ b \wedge d) \wedge \neg\ (a \wedge b \wedge \neg\ d)$
$\Rightarrow (\neg\ d' \vee c' \vee \neg\ b' \vee \neg\ a') \wedge (b' \vee \neg\ d' \vee a' \vee \neg\ c')$
$\wedge\ (b' \vee c' \vee a' \vee d') \wedge \neg\ (\neg\ a' \wedge \neg\ b' \wedge d')$
$\wedge \neg\ (a' \wedge b' \wedge \neg\ d')$

%(Invariant_step)%

end

IC3 run through:

CNF converter:

---

https://www.erpelstolz.at/gateway/formular-uk-zentral.html

[note that I changed the CNF converter, when I used the previous one

```
  http://formal.cs.utah.edu:8080/pbl/PBL.php,

I could not establish the result that T and CNF(T) are equivalent.]


Hets:
-----

   http://rest.hets.eu

==========================

Part I: the formulae how we read them off the automaton and their CNFs
----------------------------------------------------------------------

Initialisation Formula:

  I =  (a /\ b /\ c /\ d) \/ (a /\ not b /\ not c /\ not d)
in CNF:

  CNF_CASL(I) = a /\ (not c \/ b) /\ (not d \/ b) /\ (not b \/ c)
/\ (not d \/ c) /\ (not b \/ d) /\ (not c \/ d)

Transition Formula:

  T =   (a' <=> (b /\ c /\ d) \/ (a /\ not b /\ not c)
\/ (a /\ not b /\ d))
        /\ (b' <=> (not a /\ b)
\/ (not a /\ c /\ not d))
        /\ (c' <=> (not b /\ c /\ not d) \/ (not a /\ b /\ d)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not d))
        /\ (d' <=> (not a /\ b /\ not c) \/ (not a /\ b /\ d)
\/ (a /\ c /\ d))

  CNF_CASL(T) = (not b \/ not c \/ not d \/ a')
        /\ (not a \/ b \/ c \/ a')
        /\ (not a \/ b \/ not d \/ a')
        /\ (a \/ b \/ not a')
        /\ (not c \/ b \/ a \/ not a')
```

```
/\ (a \/ c \/ not a')
/\ (not b \/ c \/ a \/ not a')
/\ (a \/ d \/ not a')
/\ (not b \/ d \/ a \/ not a')
/\ (not c \/ d \/ a \/ not a')
/\ (a \/ c \/ not b \/ not a')
/\ (not b \/ c \/ not a')
/\ (a \/ d \/ not b \/ not a')
/\ (not b \/ d \/ not a')
/\ (not c \/ d \/ not b \/ not a')
/\ (a \/ b \/ d \/ not a')
/\ (not c \/ b \/ d \/ not a')
/\ (a \/ c \/ d \/ not a')
/\ (not b \/ c \/ d \/ not a')
/\ (not c \/ d \/ not a')
/\ (a \/ not b \/ b')
/\ (a \/ not c \/ d \/ b')
/\ (not a \/ not b')
/\ (c \/ not a \/ not b')
/\ (not d \/ not a \/ not b')
/\ (not a \/ b \/ not b')
/\ (c \/ b \/ not b')
/\ (not d \/ b \/ not b')
/\ (b \/ not c \/ c')
/\ (d \/ c')
/\ (a \/ not b \/ not d \/ c')
/\ (not b \/ not c \/ not d \/ c')
/\ (not a \/ b \/ d \/ c')
/\ (c \/ not d \/ b \/ a \/ not c')
/\ (not b \/ not d \/ not a \/ c \/ not c')
/\ (c \/ not d \/ not a \/ not b \/ not c')
/\ (c \/ not d \/ not a \/ b \/ not c')
/\ (c \/ not d \/ b \/ not c')
/\ (c \/ not d \/ not a \/ not c')
/\ (a \/ not b \/ c \/ d')
/\ (a \/ not b \/ not d \/ d')
/\ (not a \/ not c \/ not d \/ d')
/\ (b \/ a \/ not d')
/\ (d \/ b \/ a \/ not d')
/\ (b \/ not c \/ a \/ not d')
/\ (d \/ not c \/ a \/ not d')
/\ (not a \/ c \/ not d')
```

```
        /\ (b \/ not a \/ c \/ not d')
        /\ (d \/ not a \/ c \/ not d')
        /\ (b \/ c \/ not d')
        /\ (d \/ b \/ c \/ not d')
        /\ (not a \/ d \/ not d')
        /\ (b \/ not  a \/ d \/ not d')
        /\ (b \/ d \/ not d')
        /\ (not a \/ not c \/ d \/ not d')
        /\ (b \/ not c \/ d \/ not d')
        /\ (d \/ not c \/ not d')

Safety property:

  P  = (c /\ not d) \/ (b /\ not d) \/ (b /\ c) \/ (a /\ not b)
\/ (not a /\ not c /\ d)
  P' = (c' /\ not d') \/ (b' /\ not d') \/ (b' /\ c')
\/ (a' /\ not b') \/ (not a' /\ not c' /\ d')

  CNF_CASL(P)  = (not d \/ c \/ not b \/ not a)
/\ (b \/ not d \/ a \/ not c)
/\ (b \/ c \/ a \/ d)
  CNF_CASL(P') = (not d' \/ c' \/ not b' \/ not a')
/\ (b' \/ not d' \/ a' \/ not c')
/\ (b' \/ c' \/ a' \/ d')


Checking with Hets that T and CNF_CASL(T) are equivalent:
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

spec Check1 =
  pred a,b,c,d,a',b',c',d': ()
then %implies
 axiom
(
            (a' <=> (b /\ c /\ d) \/ (a /\ not b /\ not c)
\/ (a /\ not b /\ d))
        /\ (b' <=> (not a /\ b) \/ (not a /\ c /\ not d))
        /\ (c' <=> (not b /\ c /\ not d) \/ (not a /\ b /\ d)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not d))
        /\ (d' <=> (not a /\ b /\ not c) \/ (not a /\ b /\ d)
\/ (a /\ c /\ d))
)
```

```
<=>     (
        (not b \/ not c \/ not d \/ a')
        /\ (not a \/ b \/ c \/ a')
        /\ (not a \/ b \/ not d \/ a')
        /\ (a \/ b \/ not a')
        /\ (not c \/ b \/ a \/ not a')
        /\ (a \/ c \/ not a')
        /\ (not b \/ c \/ a \/ not a')
        /\ (a \/ d \/ not a')
        /\ (not b \/ d \/ a \/ not a')
        /\ (not c \/ d \/ a \/ not a')
        /\ (a \/ c \/ not b \/ not a')
        /\ (not b \/ c \/ not a')
        /\ (a \/ d \/ not b \/ not a')
        /\ (not b \/ d \/ not a')
        /\ (not c \/ d \/ not b \/ not a')
        /\ (a \/ b \/ d \/ not a')
        /\ (not c \/ b \/ d \/ not a')
        /\ (a \/ c \/ d \/ not a')
        /\ (not b \/ c \/ d \/ not a')
        /\ (not c \/ d \/ not a')
        /\ (a \/ not b \/ b')
        /\ (a \/ not c \/ d \/ b')
        /\ (not a \/ not b')
        /\ (c \/ not a \/ not b')
        /\ (not d \/ not a \/ not b')
        /\ (not a \/ b \/ not b')
        /\ (c \/ b \/ not b')
        /\ (not d \/ b \/ not b')
        /\ (b \/ not c \/ d \/ c')
        /\ (a \/ not b \/ not d \/ c')
        /\ (not b \/ not c \/ not d \/ c')
        /\ (not a \/ b \/ d \/ c')
        /\ (b \/ c \/ a \/ not c')
        /\ (d \/ c \/ b \/ a \/ not c')
        /\ (b \/ not d \/ a \/ not c')
        /\ (d \/ not b \/ c \/ a \/ not c')
        /\ (d \/ c \/ a \/ not c')
        /\ (b \/ not d \/ c \/ a \/ not c')
        /\ (d \/ not b \/ a \/ not c')
        /\ (b \/ c \/ d \/ a \/ not c')
        /\ (not a \/ not b \/ c \/ not c')
```

```
        /\ (d \/ not b \/ c \/ not c')
        /\ (not a \/ c \/ not b \/ not c')
        /\ (d \/ c \/ not b \/ not c')
        /\ (not a \/ not d \/ c \/ not b \/ not c')
        /\ (not a \/ not b \/ d \/ not c')
        /\ (d \/ not b \/ not c')
        /\ (not a \/ c \/ d \/ not b \/ not c')
        /\ (not a \/ c \/ b \/ not d \/ not c')
        /\ (b \/ c \/ not d \/ not c')
        /\ (not a \/ not d \/ b \/ not c')
        /\ (b \/ not d \/ not c')
        /\ (not a \/ not b \/ c \/ not d \/ not c')
        /\ (not a \/ c \/ not d \/ not c')
        /\ (not a \/ not d \/ c \/ not c')
        /\ (b \/ not d \/ c \/ not c')
        /\ (a \/ not b \/ c \/ d')
        /\ (a \/ not b \/ not d \/ d')
        /\ (not a \/ not c \/ not d \/ d')
        /\ (b \/ a \/ not d')
        /\ (d \/ b \/ a \/ not d')
        /\ (b \/ not c \/ a \/ not d')
        /\ (d \/ not c \/ a \/ not d')
        /\ (not a \/ c \/ not d')
        /\ (b \/ not a \/ c \/ not d')
        /\ (d \/ not a \/ c \/ not d')
        /\ (b \/ c \/ not d')
        /\ (d \/ b \/ c \/ not d')
        /\ (not a \/ d \/ not d')
        /\ (b \/ not a \/ d \/ not d')
        /\ (b \/ d \/ not d')
        /\ (not a \/ not c \/ d \/ not d')
        /\ (b \/ not c \/ d \/ not d')
        /\ (d \/ not c \/ not d')
         )
end

[successfully proved with HETS interface]


Part II: Page 11 from the slides:
```

```
once more the formulae (now all in CNF):

  I =  a /\ (not c \/ b) /\ (not d \/ b) /\ (not b \/ c)
/\ (not d \/ c) /\ (not b \/ d) /\ (not c \/ d)

  T =   (
        (a' <=> (b /\ c /\ d) \/ (a /\ not b /\ not c)
\/ (a /\ not b /\ d))
        /\ (b' <=> (not a /\ b) \/ (not a /\ c /\ not d))
        /\ (c' <=> (not b /\ c /\ not d) \/ (not a /\ b /\ d)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not d))
        /\ (d' <=> (not a /\ b /\ not c) \/ (not a /\ b /\ d)
\/ (a /\ c /\ d))
        )

  P = (not d \/ c \/ not b \/ not a)
/\ (b \/ not d \/ a \/ not c)
/\ (b \/ c \/ a \/ d)

  P' = (not d' \/ c' \/ not b' \/ not a')
/\ (b' \/ not d' \/ a' \/ not c')
/\ (b' \/ c' \/ a' \/ d')

a) Check I=>P?

spec Check2 =
  pred a,b,c,d: ()
  axiom  a /\ (not c \/ b) /\ (not d \/ b) /\ (not b \/ c)
/\ (not d \/ c) /\ (not b \/ d) /\ (not c \/ d)
then %implies
 axiom (not d \/ c \/ not b \/ not a)
/\ (b \/ not d \/ a \/ not c)
/\ (b \/ c \/ a \/ d)
end

[successfully proved with HETS interface]


b) Check I /\ T => P'?

spec Check3 =
  pred a,b,c,d,a',b',c',d': ()
```

```
  axiom  a /\ (not c \/ b) /\ (not d \/ b) /\ (not b \/ c)
/\ (not d \/ c) /\ (not b \/ d) /\ (not c \/ d)
  axiom      (
        (a' <=> (b /\ c /\ d) \/ (a /\ not b /\ not c)
\/ (a /\ not b /\ d))
        /\ (b' <=> (not a /\ b) \/ (not a /\ c /\ not d))
        /\ (c' <=> (not b /\ c /\ not d) \/ (not a /\ b /\ d)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not d))
        /\ (d' <=> (not a /\ b /\ not c) \/ (not a /\ b /\ d)
\/ (a /\ c /\ d))
        )
then %implies
 axiom (not d' \/ c' \/ not b' \/ not a')
/\ (b' \/ not d' \/ a' \/ not c')
/\ (b' \/ c' \/ a' \/ d')
end


c) Set F1 :=P

  F1 = (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)

d) For every clause c in clauses(I), if c not-in clauses(F1),
check:
    I /\ T => c'?
    If it does, set F1 := F1 /\ c


clauses(I) = {a,
        (not c \/ b),
        (not d \/ b),
        (not b \/ c),
        (not d \/ c),
        (not b \/ d),
        (not c \/ d)}

clauses(F1) = {(not d \/ c \/ not b \/ not a),
               (b \/ not d \/ a \/ not c),
               (b \/ c \/ a \/ d)}
```

```
d-1) c = a

check if I /\ T => a':

spec Check4 =
  pred a,b,c,d,a',b',c',d': ()
  axiom  a /\ (not c \/ b) /\ (not d \/ b) /\ (not b \/ c)
/\ (not d \/ c) /\ (not b \/ d) /\ (not c \/ d)
  axiom      (
        (a' <=> (b /\ c /\ d) \/ (a /\ not b /\ not c)
\/ (a /\ not b /\ d))
        /\ (b' <=> (not a /\ b) \/ (not a /\ c /\ not d))
        /\ (c' <=> (not b /\ c /\ not d) \/ (not a /\ b /\ d)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not d))
        /\ (d' <=> (not a /\ b /\ not c) \/ (not a /\ b /\ d)
\/ (a /\ c /\ d))
        )
then %implies
 axiom a'
end

[proved with Hets]

Thus, we add b to F1, and obtain:

   F1 = (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ a


d-2) c = (not c \/ b)

check if I /\ T => (not c' \/ b'):

spec Check5 =
  pred a,b,c,d,a',b',c',d': ()
  axiom  a /\ (not c \/ b) /\ (not d \/ b) /\ (not b \/ c)
/\ (not d \/ c) /\ (not b \/ d) /\ (not c \/ d)
  axiom      (
        (a' <=> (b /\ c /\ d) \/ (a /\ not b /\ not c)
\/ (a /\ not b /\ d))
```

```
        /\ (b' <=> (not a /\ b) \/ (not a /\ c /\ not d))
        /\ (c' <=> (not b /\ c /\ not d) \/ (not a /\ b /\ d)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not d))
        /\ (d' <=> (not a /\ b /\ not c) \/ (not a /\ b /\ d)
\/ (a /\ c /\ d))
        )
then %implies
 axiom (not c' \/ b')
end

[disproved with Hets]

Thus, we do not add (not c \/ b) to F1


d-3) c = (not d \/ b)

check if I /\ T => (not d' \/ b'):

spec Check6 =
  pred a,b,c,d,a',b',c',d': ()
  axiom  a /\ (not c \/ b) /\ (not d \/ b) /\ (not b \/ c)
/\ (not d \/ c) /\ (not b \/ d) /\ (not c \/ d)
  axiom       (
        (a' <=> (b /\ c /\ d) \/ (a /\ not b /\ not c)
\/ (a /\ not b /\ d))
        /\ (b' <=> (not a /\ b) \/ (not a /\ c /\ not d))
        /\ (c' <=> (not b /\ c /\ not d) \/ (not a /\ b /\ d)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not d))
        /\ (d' <=> (not a /\ b /\ not c) \/ (not a /\ b /\ d)
\/ (a /\ c /\ d))
        )
then %implies
 axiom (not d' \/ b')
end

[disproved with Hets]

Thus, we do not add (not d \/ b) to F1


d-4) c = (not b \/ c)
```

```
check if I /\ T => (not b' \/ c'):

spec Check7 =
  pred a,b,c,d,a',b',c',d': ()
  axiom  a /\ (not c \/ b) /\ (not d \/ b) /\ (not b \/ c)
/\ (not d \/ c) /\ (not b \/ d) /\ (not c \/ d)
  axiom      (
        (a' <=> (b /\ c /\ d) \/ (a /\ not b /\ not c)
\/ (a /\ not b /\ d))
        /\ (b' <=> (not a /\ b) \/ (not a /\ c /\ not d))
        /\ (c' <=> (not b /\ c /\ not d) \/ (not a /\ b /\ d)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not d))
        /\ (d' <=> (not a /\ b /\ not c) \/ (not a /\ b /\ d)
\/ (a /\ c /\ d))
        )
then %implies
 axiom (not b' \/ c')
end

[proved with Hets]

Thus, we add b to F1, and obtain:

   F1 = (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ a
        /\ (not b \/ c)


d-5) c = (not d \/ c)

check if I /\ T => (not d' \/ c'):

spec Check8 =
  pred a,b,c,d,a',b',c',d': ()
  axiom  a /\ (not c \/ b) /\ (not d \/ b) /\ (not b \/ c)
/\ (not d \/ c) /\ (not b \/ d) /\ (not c \/ d)
  axiom      (
        (a' <=> (b /\ c /\ d) \/ (a /\ not b /\ not c)
\/ (a /\ not b /\ d))
```

```
        /\ (b' <=> (not a /\ b) \/ (not a /\ c /\ not d))
        /\ (c' <=> (not b /\ c /\ not d) \/ (not a /\ b /\ d)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not d))
        /\ (d' <=> (not a /\ b /\ not c) \/ (not a /\ b /\ d)
\/ (a /\ c /\ d))
        )
then %implies
 axiom (not d' \/ c')
end

[proved with Hets]

Thus, we add b to F1, and obtain:

   F1 = (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ a
        /\ (not b \/ c)
        /\ (not d \/ c)


d-6) c = (not b \/ d)

check if I /\ T => (not b' \/ d'):

spec Check9 =
  pred a,b,c,d,a',b',c',d': ()
  axiom  a /\ (not c \/ b) /\ (not d \/ b) /\ (not b \/ c)
/\ (not d \/ c) /\ (not b \/ d) /\ (not c \/ d)
  axiom       (
        (a' <=> (b /\ c /\ d) \/ (a /\ not b /\ not c)
\/ (a /\ not b /\ d))
        /\ (b' <=> (not a /\ b) \/ (not a /\ c /\ not d))
        /\ (c' <=> (not b /\ c /\ not d) \/ (not a /\ b /\ d)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not d))
        /\ (d' <=> (not a /\ b /\ not c) \/ (not a /\ b /\ d)
\/ (a /\ c /\ d))
        )
then %implies
 axiom (not b' \/ d')
end
```

```
[proved with Hets]

Thus, we add b to F1, and obtain:

   F1 = (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ a
        /\ (not b \/ c)
        /\ (not d \/ c)
        /\ (not b \/ d)


d-7) c = (not c \/ d)

check if I /\ T => (not c' \/ d'):

spec Check10 =
  pred a,b,c,d,a',b',c',d': ()
  axiom  a /\ (not c \/ b) /\ (not d \/ b) /\ (not b \/ c)
/\ (not d \/ c) /\ (not b \/ d) /\ (not c \/ d)
  axiom      (
        (a' <=> (b /\ c /\ d) \/ (a /\ not b /\ not c)
\/ (a /\ not b /\ d))
        /\ (b' <=> (not a /\ b) \/ (not a /\ c /\ not d))
        /\ (c' <=> (not b /\ c /\ not d) \/ (not a /\ b /\ d)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not d))
        /\ (d' <=> (not a /\ b /\ not c) \/ (not a /\ b /\ d)
\/ (a /\ c /\ d))
        )
then %implies
 axiom (not c' \/ d')
end

[disproved with Hets]

Thus, we do not add (not c \/ d) to F1


Part III: Page 12 from the slides:
```

```
a) Check: Is it the case that Fk /\ T => P'?
b) If it is, then
 b-1) set Fk+1 := P
 b-2) for every clause c in Fk , check if Fk/\T=>c'?
      If it is, set Fk+1 := Fk+1 /\ c
c) If Fk =Fk+1: done


k=1:
++++

a) do we have F1 /\ T => P' ?

spec Check11 =
  pred a,b,c,d,a',b',c',d': ()
  axiom (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ a
        /\ (not b \/ c)
        /\ (not d \/ c)
        /\ (not b \/ d)
  axiom    (
        (a' <=> (b /\ c /\ d) \/ (a /\ not b /\ not c)
\/ (a /\ not b /\ d))
        /\ (b' <=> (not a /\ b) \/ (not a /\ c /\ not d))
        /\ (c' <=> (not b /\ c /\ not d) \/ (not a /\ b /\ d)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not d))
        /\ (d' <=> (not a /\ b /\ not c) \/ (not a /\ b /\ d)
\/ (a /\ c /\ d))
        )
then %implies
 axiom  (not d' \/ c' \/ not b' \/ not a')
/\ (b' \/ not d' \/ a' \/ not c')
/\ (b' \/ c' \/ a' \/ d')
end

[proved with Hets]


b-1) F2 = (not d \/ c \/ not b \/ not a)
```

```
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)

b-2) clauses(F1) = {(not d \/ c \/ not b \/ not a),
        (b \/ not d \/ a \/ not c),
        (b \/ c \/ a \/ d),
        a,
        (not b \/ c),
        (not d \/ c),
        (not b \/ d)}


    let c = (not d \/ c \/ not b \/ not a)
    check F1 /\ T => c'

spec Check12 =
  pred a,b,c,d,a',b',c',d': ()
  axiom (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ a
        /\ (not b \/ c)
        /\ (not d \/ c)
        /\ (not b \/ d)
  axiom      (
        (a' <=> (b /\ c /\ d) \/ (a /\ not b /\ not c)
\/ (a /\ not b /\ d))
        /\ (b' <=> (not a /\ b) \/ (not a /\ c /\ not d))
        /\ (c' <=> (not b /\ c /\ not d) \/ (not a /\ b /\ d)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not d))
        /\ (d' <=> (not a /\ b /\ not c) \/ (not a /\ b /\ d)
\/ (a /\ c /\ d))
        )
then %implies
 axiom (not d' \/ c' \/ not b' \/ not a')
end

[successfully proved with HETS]

thus F2 = (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
```

```
        /\ (not d \/ c \/ not b \/ not a)


    let c = (b \/ not d \/ a \/ not c)
    check F1 /\ T => c'

spec Check13 =
  pred a,b,c,d,a',b',c',d': ()
  axiom (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ a
        /\ (not b \/ c)
        /\ (not d \/ c)
        /\ (not b \/ d)
  axiom      (
        (a' <=> (b /\ c /\ d) \/ (a /\ not b /\ not c)
\/ (a /\ not b /\ d))
        /\ (b' <=> (not a /\ b) \/ (not a /\ c /\ not d))
        /\ (c' <=> (not b /\ c /\ not d) \/ (not a /\ b /\ d)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not d))
        /\ (d' <=> (not a /\ b /\ not c) \/ (not a /\ b /\ d)
\/ (a /\ c /\ d))
        )
then %implies
 axiom (b' \/ not d' \/ a' \/ not c')
end

[successfully proved with HETS]

thus F2 = (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)


    let c = (b \/ c \/ a \/ d)
    check F1 /\ T => c'

spec Check14 =
  pred a,b,c,d,a',b',c',d': ()
```

```
  axiom (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ a
        /\ (not b \/ c)
        /\ (not d \/ c)
        /\ (not b \/ d)
  axiom       (
        (a' <=> (b /\ c /\ d) \/ (a /\ not b /\ not c)
\/ (a /\ not b /\ d))
        /\ (b' <=> (not a /\ b) \/ (not a /\ c /\ not d))
        /\ (c' <=> (not b /\ c /\ not d) \/ (not a /\ b /\ d)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not d))
        /\ (d' <=> (not a /\ b /\ not c) \/ (not a /\ b /\ d)
\/ (a /\ c /\ d))
        )
then %implies
 axiom (b' \/ c' \/ a' \/ d')
end

[successfully proved with HETS]

thus F2 = (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)


    let c = a
    check F1 /\ T => c'

spec Check15 =
  pred a,b,c,d,a',b',c',d': ()
  axiom (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ a
        /\ (not b \/ c)
        /\ (not d \/ c)
        /\ (not b \/ d)
```

```
  axiom       (
        (a' <=> (b /\ c /\ d) \/ (a /\ not b /\ not c)
\/ (a /\ not b /\ d))
        /\ (b' <=> (not a /\ b) \/ (not a /\ c /\ not d))
        /\ (c' <=> (not b /\ c /\ not d) \/ (not a /\ b /\ d)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not d))
        /\ (d' <=> (not a /\ b /\ not c) \/ (not a /\ b /\ d)
\/ (a /\ c /\ d))
        )
then %implies
 axiom a'
end

[disproved with Hets]

Thus, we do not add a to F2


    let c = (not b \/ c)
    check F1 /\ T => c'

spec Check16 =
  pred a,b,c,d,a',b',c',d': ()
  axiom (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ a
        /\ (not b \/ c)
        /\ (not d \/ c)
        /\ (not b \/ d)
  axiom       (
        (a' <=> (b /\ c /\ d) \/ (a /\ not b /\ not c)
\/ (a /\ not b /\ d))
        /\ (b' <=> (not a /\ b) \/ (not a /\ c /\ not d))
        /\ (c' <=> (not b /\ c /\ not d) \/ (not a /\ b /\ d)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not d))
        /\ (d' <=> (not a /\ b /\ not c) \/ (not a /\ b /\ d)
\/ (a /\ c /\ d))
        )
then %implies
 axiom (not b' \/ c')
end
```

```
[successfully proved with HETS]

thus F2 = (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ (not b \/ c)


    let c = (not d \/ c)
    check F1 /\ T => c'

spec Check17 =
  pred a,b,c,d,a',b',c',d': ()
  axiom (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ a
        /\ (not b \/ c)
        /\ (not d \/ c)
        /\ (not b \/ d)
  axiom       (
        (a' <=> (b /\ c /\ d) \/ (a /\ not b /\ not c)
\/ (a /\ not b /\ d))
        /\ (b' <=> (not a /\ b) \/ (not a /\ c /\ not d))
        /\ (c' <=> (not b /\ c /\ not d) \/ (not a /\ b /\ d)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not d))
        /\ (d' <=> (not a /\ b /\ not c) \/ (not a /\ b /\ d)
\/ (a /\ c /\ d))
        )
then %implies
 axiom (not d' \/ c')
end

[disproved with Hets]

Thus, we do not add (not d \/ c) to F2
```

```
    let c = (not b \/ d)
    check F1 /\ T => c'

spec Check18 =
  pred a,b,c,d,a',b',c',d': ()
  axiom (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ a
        /\ (not b \/ c)
        /\ (not d \/ c)
        /\ (not b \/ d)
  axiom      (
        (a' <=> (b /\ c /\ d) \/ (a /\ not b /\ not c)
\/ (a /\ not b /\ d))
        /\ (b' <=> (not a /\ b) \/ (not a /\ c /\ not d))
        /\ (c' <=> (not b /\ c /\ not d) \/ (not a /\ b /\ d)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not d))
        /\ (d' <=> (not a /\ b /\ not c) \/ (not a /\ b /\ d)
\/ (a /\ c /\ d))
        )
then %implies
 axiom (not b' \/ d')
end

[successfully proved with HETS]

thus F2 = (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ (not b \/ c)
        /\ (not b \/ d)


c) thus F2 = (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ (not b \/ c)
        /\ (not b \/ d)
```

```
        != F1

i.e. F2 = F1.

Part IV: Repeat part 3 until Fk =Fk+1:
-------------------------------------


a) Check: Is it the case that Fk /\ T => P'?
b) If it is, then
 b-1) set Fk+1 := P
 b-2) for every clause c $\in$ Fk , check if Fk/\T=>c'?
      If it is, set Fk+1 := Fk+1 /\ c
c) If Fk =Fk+1: done


k=2:
++++

a) do we have F2 /\ T => P' ?

spec Check19 =
  pred a,b,c,d,a',b',c',d': ()
  axiom (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ (not b \/ c)
        /\ (not b \/ d)
  axiom    (
        (a' <=> (b /\ c /\ d) \/ (a /\ not b /\ not c)
\/ (a /\ not b /\ d))
        /\ (b' <=> (not a /\ b) \/ (not a /\ c /\ not d))
        /\ (c' <=> (not b /\ c /\ not d) \/ (not a /\ b /\ d)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not d))
        /\ (d' <=> (not a /\ b /\ not c) \/ (not a /\ b /\ d)
\/ (a /\ c /\ d))
        )
then %implies
 axiom  (not d' \/ c' \/ not b' \/ not a')
/\ (b' \/ not d' \/ a' \/ not c')
/\ (b' \/ c' \/ a' \/ d')
end
```

```
[disproved with HETS]


Therefore find the state(s) s that cause F2 /\ T => P'
to not hold.
s = (not a /\ not b /\ not c /\ d)

Find a j where Fj /\ neg s /\ T => neg s'


Check F0 => neg s

spec Check20 =
  pred a,b,c,d,a',b',c',d': ()
  axiom (a /\ b /\ c /\ d) \/ (a /\ not b /\ not c /\ not d)
then %implies
 axiom  not (not a /\ not b /\ not c /\ d)
end

[proved with hets]


Check F2 /\ neg s /\ T => neg s'

spec Check21 =
  pred a,b,c,d,a',b',c',d': ()
  axiom (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ (not b \/ c)
        /\ (not b \/ d)
  axiom    (not (not a /\ not b /\ not c /\ d))
  axiom    (
        (a' <=> (b /\ c /\ d) \/ (a /\ not b /\ not c)
\/ (a /\ not b /\ d))
        /\ (b' <=> (not a /\ b) \/ (not a /\ c /\ not d))
        /\ (c' <=> (not b /\ c /\ not d) \/ (not a /\ b /\ d)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not d))
        /\ (d' <=> (not a /\ b /\ not c) \/ (not a /\ b /\ d)
\/ (a /\ c /\ d))
        )
then %implies
```

```
 axiom  not (not a' /\ not b' /\ not c' /\ d')
end

Drop a literal

Clause c: not a /\ not b /\ not c

Check F0 => neg c
[proved]
Check Fk /\ neg c /\ T => neg c'
[proved]


Thus, for all Fj, Fj = Fj /\ neg c

   F1 = (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ a
        /\ (not b \/ c)
        /\ (not d \/ c)
        /\ (not b \/ d)
        /\ (not (not a /\ not b /\ not c))

   F2 = (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ (not b \/ c)
        /\ (not b \/ d)
        /\ (not (not a /\ not b /\ not c))


Check F2 /\ T => P'

spec Check22 =
  pred a,b,c,d,a',b',c',d': ()
  axiom (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ (not b \/ c)
        /\ (not b \/ d)
        /\ (not (not a /\ not b /\ not c))
```

```
  axiom     (
        (a' <=> (b /\ c /\ d) \/ (a /\ not b /\ not c)
\/ (a /\ not b /\ d))
        /\ (b' <=> (not a /\ b) \/ (not a /\ c /\ not d))
        /\ (c' <=> (not b /\ c /\ not d) \/ (not a /\ b /\ d)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not d))
        /\ (d' <=> (not a /\ b /\ not c) \/ (not a /\ b /\ d)
\/ (a /\ c /\ d))
        )
then %implies
 axiom  (not d' \/ c' \/ not b' \/ not a')
/\ (b' \/ not d' \/ a' \/ not c')
/\ (b' \/ c' \/ a' \/ d')
end


[successfully proved with HETS]


b-1) F3 = (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)

b-2) clauses(F2) = {(not d \/ c \/ not b \/ not a),
        (b \/ not d \/ a \/ not c),
        (b \/ c \/ a \/ d),
        (not b \/ c),
        (not b \/ d),
        (not (not a /\ not b /\ not c))}


    let c = (not d \/ c \/ not b \/ not a)
    check F2 /\ T => c'

spec Check23 =
  pred a,b,c,d,a',b',c',d': ()
  axiom (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ (not b \/ c)
        /\ (not b \/ d)
        /\ (not (not a /\ not b /\ not c))
```

```
  axiom       (
        (a' <=> (b /\ c /\ d) \/ (a /\ not b /\ not c)
\/ (a /\ not b /\ d))
        /\ (b' <=> (not a /\ b) \/ (not a /\ c /\ not d))
        /\ (c' <=> (not b /\ c /\ not d) \/ (not a /\ b /\ d)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not d))
        /\ (d' <=> (not a /\ b /\ not c) \/ (not a /\ b /\ d)
\/ (a /\ c /\ d))
        )
then %implies
 axiom (not d' \/ c' \/ not b' \/ not a')
end

[successfully proved with HETS]

thus F3 = (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ (not d \/ c \/ not b \/ not a)


    let c = (b \/ not d \/ a \/ not c)
    check F2 /\ T => c'

spec Check24 =
  pred a,b,c,d,a',b',c',d': ()
  axiom (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ (not b \/ c)
        /\ (not b \/ d)
        /\ (not (not a /\ not b /\ not c))
  axiom       (
        (a' <=> (b /\ c /\ d) \/ (a /\ not b /\ not c)
\/ (a /\ not b /\ d))
        /\ (b' <=> (not a /\ b) \/ (not a /\ c /\ not d))
        /\ (c' <=> (not b /\ c /\ not d) \/ (not a /\ b /\ d)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not d))
        /\ (d' <=> (not a /\ b /\ not c) \/ (not a /\ b /\ d)
\/ (a /\ c /\ d))
        )
then %implies
```

```
 axiom (b' \/ not d' \/ a' \/ not c')
end

[successfully proved with HETS]

thus F3 = (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)


    let c = (b \/ c \/ a \/ d)
    check F2 /\ T => c'

spec Check25 =
  pred a,b,c,d,a',b',c',d': ()
  axiom (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ (not b \/ c)
        /\ (not b \/ d)
        /\ (not (not a /\ not b /\ not c))
  axiom       (
        (a' <=> (b /\ c /\ d) \/ (a /\ not b /\ not c)
\/ (a /\ not b /\ d))
        /\ (b' <=> (not a /\ b) \/ (not a /\ c /\ not d))
        /\ (c' <=> (not b /\ c /\ not d) \/ (not a /\ b /\ d)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not d))
        /\ (d' <=> (not a /\ b /\ not c) \/ (not a /\ b /\ d)
\/ (a /\ c /\ d))
        )
then %implies
 axiom (b' \/ c' \/ a' \/ d')
end

[successfully proved with HETS]

thus F3 = (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ (not d \/ c \/ not b \/ not a)
```

```
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)


    let c = (not b \/ c)
    check F2 /\ T => c'

spec Check26 =
  pred a,b,c,d,a',b',c',d': ()
  axiom (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ (not b \/ c)
        /\ (not b \/ d)
        /\ (not (not a /\ not b /\ not c))
  axiom      (
        (a' <=> (b /\ c /\ d) \/ (a /\ not b /\ not c)
\/ (a /\ not b /\ d))
        /\ (b' <=> (not a /\ b) \/ (not a /\ c /\ not d))
        /\ (c' <=> (not b /\ c /\ not d) \/ (not a /\ b /\ d)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not d))
        /\ (d' <=> (not a /\ b /\ not c) \/ (not a /\ b /\ d)
\/ (a /\ c /\ d))
        )
then %implies
 axiom (not b' \/ c')
end

[successfully proved with HETS]

thus F3 = (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ (not b \/ c)


    let c = (not b \/ d)
    check F2 /\ T => c'
```

```
spec Check27 =
  pred a,b,c,d,a',b',c',d': ()
  axiom (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ (not b \/ c)
        /\ (not b \/ d)
        /\ (not (not a /\ not b /\ not c))
  axiom      (
        (a' <=> (b /\ c /\ d) \/ (a /\ not b /\ not c)
\/ (a /\ not b /\ d))
        /\ (b' <=> (not a /\ b) \/ (not a /\ c /\ not d))
        /\ (c' <=> (not b /\ c /\ not d) \/ (not a /\ b /\ d)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not d))
        /\ (d' <=> (not a /\ b /\ not c) \/ (not a /\ b /\ d)
\/ (a /\ c /\ d))
        )
then %implies
 axiom (not b' \/ d')
end

[disproved with HETS]

Thus, we do not add (not b \/ d) to F3


    let c = (not (not a /\ not c))
    check F2 /\ T => c'

spec Check28 =
  pred a,b,c,d,a',b',c',d': ()
  axiom (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ (not b \/ c)
        /\ (not b \/ d)
        /\ (not (not a /\ not b /\ not c))
  axiom      (
        (a' <=> (b /\ c /\ d) \/ (a /\ not b /\ not c)
\/ (a /\ not b /\ d))
        /\ (b' <=> (not a /\ b) \/ (not a /\ c /\ not d))
        /\ (c' <=> (not b /\ c /\ not d) \/ (not a /\ b /\ d)
```

```
\/ (b /\ c /\ d) \/ (a /\ not b /\ not d))
        /\ (d' <=> (not a /\ b /\ not c) \/ (not a /\ b /\ d)
\/ (a /\ c /\ d))
        )
then %implies
 axiom (not (not a' /\ not b' /\ not c'))
end

[successfully proved with HETS]

thus F3 = (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ (not b \/ c)
        /\ (not (not a /\ not b /\ not c))


c) thus F3 = (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ (not b \/ c)
        /\ (not (not a /\ not b /\ not c))
        != F2

i.e. F3 = F2.

Part IV: Repeat part 3 until Fk =Fk+1:
--------------------------------------


a) Check: Is it the case that Fk /\ T => P'?
b) If it is, then
 b-1) set Fk+1 := P
 b-2) for every clause c $\in$ Fk , check if Fk/\T=>c'?
      If it is, set Fk+1 := Fk+1 /\ c
c) If Fk =Fk+1: done


k=3:
++++
```

```
a) do we have F3 /\ T => P' ?

spec Check29 =
  pred a,b,c,d,a',b',c',d': ()
  axiom (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ (not b \/ c)
        /\ (not (not a /\ not b /\ not c))
  axiom    (
        (a' <=> (b /\ c /\ d) \/ (a /\ not b /\ not c)
\/ (a /\ not b /\ d))
        /\ (b' <=> (not a /\ b) \/ (not a /\ c /\ not d))
        /\ (c' <=> (not b /\ c /\ not d) \/ (not a /\ b /\ d)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not d))
        /\ (d' <=> (not a /\ b /\ not c) \/ (not a /\ b /\ d)
\/ (a /\ c /\ d))
        )
then %implies
 axiom  (not d' \/ c' \/ not b' \/ not a')
/\ (b' \/ not d' \/ a' \/ not c')
/\ (b' \/ c' \/ a' \/ d')
end

[disproved with HETS]


Therefore find the state(s) s that cause F3 /\ T => P' to not hold.
s = (a /\ b /\ c /\ not d)

Find an j where Fj /\ neg s /\ T => neg s'

Check F3 /\ neg s /\ T => neg s'

spec Check30 =
  pred a,b,c,d,a',b',c',d': ()
  axiom (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ (not b \/ c)
        /\ (not (not a /\ not b /\ not c))
```

```
  axiom not (a /\ b /\ c /\ not d)
  axiom     (
        (a' <=> (b /\ c /\ d) \/ (a /\ not b /\ not c)
\/ (a /\ not b /\ d))
        /\ (b' <=> (not a /\ b) \/ (not a /\ c /\ not d))
        /\ (c' <=> (not b /\ c /\ not d) \/ (not a /\ b /\ d)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not d))
        /\ (d' <=> (not a /\ b /\ not c) \/ (not a /\ b /\ d)
\/ (a /\ c /\ d))
        )
then %implies
 axiom  not (a' /\ b' /\ c' /\ not d')
end


Drop a literal

Clause c: a /\ b /\ c

Check F0 => neg c
[proved]
Check Fk /\ neg c /\ T => neg c'
[disproved]


Drop a different literal

Clause c: a /\ b /\ not d

Check F0 => neg c
[proved]
Check Fk /\ neg c /\ T => neg c'
[proved]


[successfully proved with HETS]

Thus, for all Fj, Fj = Fj /\ neg s

   F1 = (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
```

```
        /\ a
        /\ (not b \/ c)
        /\ (not d \/ c)
        /\ (not b \/ d)
        /\ (not (not a /\ not b /\ not c))
        /\ (not (a /\ b /\ not d))

   F2 = (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ (not b \/ c)
        /\ (not b \/ d)
        /\ (not (not a /\ not b /\ not c))
        /\ (not (a /\ b /\ not d))

   F3 = (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ (not b \/ c)
        /\ (not (not a /\ not b /\ not c))
        /\ (not (a /\ b /\ not d))


b-1) F4 = (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)

b-2) clauses(F3) = {(not d \/ c \/ not b \/ not a),
                    (b \/ not d \/ a \/ not c),
                    (b \/ c \/ a \/ d),
                    (not b \/ c),
                    (not (not a /\ not b /\ not c /\ d),)
                    (not (a /\ b /\ c /\ not d)}


    let c = (not d \/ c \/ not b \/ not a)
    check F3 /\ T => c'

spec Check31 =
  pred a,b,c,d,a',b',c',d': ()
  axiom (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
```

```
        /\ (b \/ c \/ a \/ d)
        /\ (not b \/ c)
        /\ (not (not a /\ not b /\ not c))
        /\ (not (a /\ b /\ not d))
  axiom      (
        (a' <=> (b /\ c /\ d) \/ (a /\ not b /\ not c)
\/ (a /\ not b /\ d))
        /\ (b' <=> (not a /\ b) \/ (not a /\ c /\ not d))
        /\ (c' <=> (not b /\ c /\ not d) \/ (not a /\ b /\ d)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not d))
        /\ (d' <=> (not a /\ b /\ not c) \/ (not a /\ b /\ d)
\/ (a /\ c /\ d))
        )
then %implies
 axiom (not d' \/ c' \/ not b' \/ not a')
end

[successfully proved with HETS]

thus F4 = (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ (not d \/ c \/ not b \/ not a)


    let c = (b \/ not d \/ a \/ not c)
    check F3 /\ T => c'

spec Check32 =
  pred a,b,c,d,a',b',c',d': ()
  axiom (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ (not b \/ c)
        /\ (not (not a /\ not b /\ not c))
        /\ (not (a /\ b /\ not d))
  axiom      (
        (a' <=> (b /\ c /\ d) \/ (a /\ not b /\ not c)
\/ (a /\ not b /\ d))
        /\ (b' <=> (not a /\ b) \/ (not a /\ c /\ not d))
        /\ (c' <=> (not b /\ c /\ not d) \/ (not a /\ b /\ d)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not d))
```

```
        /\ (d' <=> (not a /\ b /\ not c) \/ (not a /\ b /\ d)
\/ (a /\ c /\ d))
        )
then %implies
 axiom (b' \/ not d' \/ a' \/ not c')
end

[successfully proved with HETS]

thus F4 = (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)


    let c = (b \/ c \/ a \/ d)
    check F3 /\ T => c'

spec Check33 =
  pred a,b,c,d,a',b',c',d': ()
  axiom (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ (not b \/ c)
        /\ (not (not a /\ not b /\ not c))
        /\ (not (a /\ b /\ not d))
  axiom       (
        (a' <=> (b /\ c /\ d) \/ (a /\ not b /\ not c)
\/ (a /\ not b /\ d))
        /\ (b' <=> (not a /\ b) \/ (not a /\ c /\ not d))
        /\ (c' <=> (not b /\ c /\ not d) \/ (not a /\ b /\ d)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not d))
        /\ (d' <=> (not a /\ b /\ not c) \/ (not a /\ b /\ d)
\/ (a /\ c /\ d))
        )
then %implies
 axiom (b' \/ c' \/ a' \/ d')
end

[successfully proved with HETS]
```

```
thus F4 = (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)


    let c = (not b \/ c)
    check F3 /\ T => c'

spec Check34 =
  pred a,b,c,d,a',b',c',d': ()
  axiom (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ (not b \/ c)
        /\ (not (not a /\ not b /\ not c))
        /\ (not (a /\ b /\ not d))
  axiom       (
        (a' <=> (b /\ c /\ d) \/ (a /\ not b /\ not c)
\/ (a /\ not b /\ d))
        /\ (b' <=> (not a /\ b) \/ (not a /\ c /\ not d))
        /\ (c' <=> (not b /\ c /\ not d) \/ (not a /\ b /\ d)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not d))
        /\ (d' <=> (not a /\ b /\ not c) \/ (not a /\ b /\ d)
\/ (a /\ c /\ d))
        )
then %implies
 axiom (not b' \/ c')
end

[disproved with HETS]

Thus, we do not add (not b \/ c) to F4


    let c = (not (not a /\ not b /\ not c))
    check F3 /\ T => c'

spec Check35 =
  pred a,b,c,d,a',b',c',d': ()
```

```
  axiom (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ (not b \/ c)
        /\ (not (not a /\ not b /\ not c))
        /\ (not (a /\ b /\ not d))
  axiom       (
        (a' <=> (b /\ c /\ d) \/ (a /\ not b /\ not c)
\/ (a /\ not b /\ d))
        /\ (b' <=> (not a /\ b) \/ (not a /\ c /\ not d))
        /\ (c' <=> (not b /\ c /\ not d) \/ (not a /\ b /\ d)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not d))
        /\ (d' <=> (not a /\ b /\ not c) \/ (not a /\ b /\ d)
\/ (a /\ c /\ d))
        )
then %implies
 axiom (not (not a' /\ not b' /\ not c'))
end

[successfully proved with HETS]

thus F4 = (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ (not (not a /\ not b /\ not c))


    let c = (not (a /\ b /\ not d))
    check F3 /\ T => c'

spec Check36 =
  pred a,b,c,d,a',b',c',d': ()
  axiom (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ (not b \/ c)
        /\ (not (not a /\ not b /\ not c))
        /\ (not (a /\ b /\ not d))
  axiom       (
```

```
        (a' <=> (b /\ c /\ d) \/ (a /\ not b /\ not c)
\/ (a /\ not b /\ d))
        /\ (b' <=> (not a /\ b) \/ (not a /\ c /\ not d))
        /\ (c' <=> (not b /\ c /\ not d) \/ (not a /\ b /\ d)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not d))
        /\ (d' <=> (not a /\ b /\ not c) \/ (not a /\ b /\ d)
\/ (a /\ c /\ d))
        )
then %implies
 axiom (not (a' /\ b' /\ not d'))
end

[successfully proved with HETS]

thus F4 = (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ (not (not a /\ not b /\ not c))
        /\ (not (a /\ b /\ not d))


c) thus F4 = (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ (not (not a /\ not b /\ not c))
        /\ (not (a /\ b /\ not d))

   thus F4 = (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ (not (not a /\ not b /\ not c))
        /\ (not (a /\ b /\ not d))
        != F3


i.e. F4 = F3.
```

```
Part IV: Repeat part 3 until Fk =Fk+1:
--------------------------------------

a) Check: Is it the case that Fk /\ T => P'?
b) If it is, then
 b-1) set Fk+1 := P
 b-2) for every clause c $\in$ Fk , check if Fk/\T=>c'?
      If it is, set Fk+1 := Fk+1 /\ c
c) If Fk =Fk+1: done


k=4:
++++

a) do we have F4 /\ T => P' ?

spec Check37 =
  pred a,b,c,d,a',b',c',d': ()
  axiom (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ (not (not a /\ not b /\ not c))
        /\ (not (a /\ b /\ not d))
  axiom    (
        (a' <=> (b /\ c /\ d) \/ (a /\ not b /\ not c)
\/ (a /\ not b /\ d))
        /\ (b' <=> (not a /\ b) \/ (not a /\ c /\ not d))
        /\ (c' <=> (not b /\ c /\ not d) \/ (not a /\ b /\ d)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not d))
        /\ (d' <=> (not a /\ b /\ not c) \/ (not a /\ b /\ d)
\/ (a /\ c /\ d))
        )
then %implies
 axiom  (not d' \/ c' \/ not b' \/ not a')
/\ (b' \/ not d' \/ a' \/ not c')
/\ (b' \/ c' \/ a' \/ d')
end

[successfully proved with HETS]
```

```
b-1) F5 = (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)

b-2) clauses(F4) = {(not d \/ c \/ not b \/ not a),
                    (b \/ not d \/ a \/ not c),
                    (b \/ c \/ a \/ d),
                    (not (not a /\ not b /\ not c /\ d)),
                    (not (a /\ b /\ c /\ not d)))}


    let c = (not d \/ c \/ not b \/ not a)
    check F4 /\ T => c'

spec Check38 =
  pred a,b,c,d,a',b',c',d': ()
  axiom (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ (not (not a /\ not b /\ not c))
        /\ (not (a /\ b /\ not d))
  axiom       (
        (a' <=> (b /\ c /\ d) \/ (a /\ not b /\ not c)
\/ (a /\ not b /\ d))
        /\ (b' <=> (not a /\ b) \/ (not a /\ c /\ not d))
        /\ (c' <=> (not b /\ c /\ not d) \/ (not a /\ b /\ d)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not d))
        /\ (d' <=> (not a /\ b /\ not c) \/ (not a /\ b /\ d)
\/ (a /\ c /\ d))
        )
then %implies
 axiom (not d' \/ c' \/ not b' \/ not a')
end

[successfully proved with HETS]

thus F5 = (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ (not d \/ c \/ not b \/ not a)
```

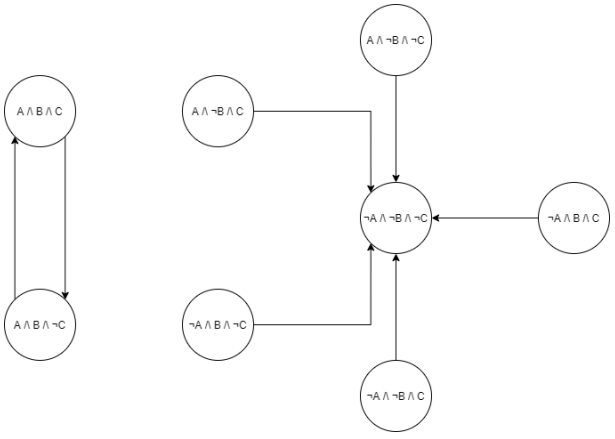
```
    let c = (b \/ not d \/ a \/ not c)
    check F4 /\ T => c'

spec Check39 =
  pred a,b,c,d,a',b',c',d': ()
  axiom (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ (not (not a /\ not b /\ not c))
        /\ (not (a /\ b /\ not d))
  axiom       (
        (a' <=> (b /\ c /\ d) \/ (a /\ not b /\ not c)
\/ (a /\ not b /\ d))
        /\ (b' <=> (not a /\ b) \/ (not a /\ c /\ not d))
        /\ (c' <=> (not b /\ c /\ not d) \/ (not a /\ b /\ d)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not d))
        /\ (d' <=> (not a /\ b /\ not c) \/ (not a /\ b /\ d)
\/ (a /\ c /\ d))
        )
then %implies
 axiom (b' \/ not d' \/ a' \/ not c')
end

[successfully proved with HETS]

thus F5 = (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)


    let c = (b \/ c \/ a \/ d)
    check F4 /\ T => c'

spec Check40 =
  pred a,b,c,d,a',b',c',d': ()
  axiom (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ (not (not a /\ not b /\ not c))
        /\ (not (a /\ b /\ not d))
```

```
  axiom       (
        (a' <=> (b /\ c /\ d) \/ (a /\ not b /\ not c)
\/ (a /\ not b /\ d))
        /\ (b' <=> (not a /\ b) \/ (not a /\ c /\ not d))
        /\ (c' <=> (not b /\ c /\ not d) \/ (not a /\ b /\ d)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not d))
        /\ (d' <=> (not a /\ b /\ not c) \/ (not a /\ b /\ d)
\/ (a /\ c /\ d))
        )
then %implies
 axiom (b' \/ c' \/ a' \/ d')
end

[successfully proved with HETS]

thus F5 = (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)


    let c = (not (not a /\ not b /\ not c))
    check F4 /\ T => c'

spec Check41 =
  pred a,b,c,d,a',b',c',d': ()
  axiom (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ (not (not a /\ not b /\ not c))
        /\ (not (a /\ b /\ not d))
  axiom       (
        (a' <=> (b /\ c /\ d) \/ (a /\ not b /\ not c)
\/ (a /\ not b /\ d))
        /\ (b' <=> (not a /\ b) \/ (not a /\ c /\ not d))
        /\ (c' <=> (not b /\ c /\ not d) \/ (not a /\ b /\ d)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not d))
        /\ (d' <=> (not a /\ b /\ not c) \/ (not a /\ b /\ d)
\/ (a /\ c /\ d))
        )
```

```
then %implies
 axiom (not (not a' /\ not b' /\ not c'))
end

[successfully proved with HETS]

thus F5 = (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ (not (not a /\ not b /\ not c))


    let c = (not (a /\ b /\ c))
    check F4 /\ T => c'

spec Check42 =
  pred a,b,c,d,a',b',c',d': ()
  axiom (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ (not (not a /\ not b /\ not c))
        /\ (not (a /\ b /\ not d))
  axiom      (
        (a' <=> (b /\ c /\ d) \/ (a /\ not b /\ not c)
\/ (a /\ not b /\ d))
        /\ (b' <=> (not a /\ b) \/ (not a /\ c /\ not d))
        /\ (c' <=> (not b /\ c /\ not d) \/ (not a /\ b /\ d)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not d))
        /\ (d' <=> (not a /\ b /\ not c) \/ (not a /\ b /\ d)
\/ (a /\ c /\ d))
        )
then %implies
 axiom (not (a' /\ b' /\ not d'))
end

[successfully proved with HETS]

thus F5 = (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
```

```
        /\ (b \/ c \/ a \/ d)
        /\ (not (not a /\ not b /\ not c))
        /\ (not (a /\ b /\ not d))


c) thus F5 = (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ (not (not a /\ not b /\ not c))
        /\ (not (a /\ b /\ not d))

   thus F5 = (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ (not (not a /\ not b /\ d))
        /\ (not (a /\ b /\ not d))
        = F4

Therefore verification holds
```

B.1.4 Second Three Variable Ladder Logic Example



spec TRANSITION =
preds a : ();
b : ();
c : ()

preds a' : ();
b' : ();
c' : ()
• $a' \Leftrightarrow a \wedge b$
• $b' \Leftrightarrow a \wedge b$
• $c' \Leftrightarrow a \wedge b \wedge \neg\, c$
end

spec TransitionTest =
TRANSITION
then %implies
• $a \wedge b \wedge c \Rightarrow a' \wedge b' \wedge \neg\, c'$ %(t1)%
• $a \wedge b \wedge \neg\, c \Rightarrow a' \wedge b' \wedge c'$ %(t2)%
• $a \wedge \neg\, b \wedge \neg\, c \Rightarrow \neg\, a' \wedge \neg\, b' \wedge \neg\, c'$ %(t3)%
• $a \wedge \neg\, b \wedge c \Rightarrow \neg\, a' \wedge \neg\, b' \wedge \neg\, c'$ %(t4)%
• $\neg\, a \wedge b \wedge \neg\, c \Rightarrow \neg\, a' \wedge \neg\, b' \wedge \neg\, c'$ %(t5)%
• $\neg\, a \wedge b \wedge c \Rightarrow \neg\, a' \wedge \neg\, b' \wedge \neg\, c'$ %(t6)%
• $\neg\, a \wedge \neg\, b \wedge \neg\, c \Rightarrow \neg\, a' \wedge \neg\, b' \wedge \neg\, c'$ %(t7)%
• $\neg\, a \wedge \neg\, b \wedge c \Rightarrow \neg\, a' \wedge \neg\, b' \wedge \neg\, c'$ %(t8)%
end

spec SafeInitial =
TRANSITION
then • $a \wedge b \wedge c$
then %implies
• $(a \wedge b) \vee c$ %(initial_safe)%
end

spec StepSafe =
TRANSITION
then %implies
• $(a \wedge b) \vee c \Rightarrow (a' \wedge b') \vee c'$ %(step_safety)%
end

spec StepSafeWithInvariant =
TRANSITION
then %implies
• $((a \wedge b) \vee c) \wedge ((a \vee c) \wedge (b \vee c) \wedge a \wedge b) \Rightarrow (a' \wedge b') \vee c'$
%(step_safety_with_invariant)%
end

spec InvariantHoldsInitially =

TRANSITION
then • $a \wedge b \wedge c$
then %implies
• $(a \vee c) \wedge (b \vee c) \wedge a \wedge b$ %(invariant_initially)%
end

spec INVARIANTSTEP =
TRANSITION
then %implies
• $(a \vee c) \wedge (b \vee c) \wedge a \wedge b \Rightarrow (a \vee c) \wedge (b \vee c) \wedge a \wedge b$
%(Invariant_step)%
end

IC3 run through:

```
Part I: the formulae how we read them off the automaton
and their CNFs
```

```
Initialisation Formula:

  I =  a /\ b /\ c
in CNF:

  CNF_CASL(I) = a /\ b /\ c

Transition Formula:

  T =   a' <=> a /\ b
        /\ b' <=> a /\ b
        /\ c' <=> a /\ b /\ not c

  CNF_CASL(T) = (not a \/ not b \/ a')
        /\ (not a' \/ a)
        /\ (not a' \/ b)
        /\ (not a \/ not b \/ b')
        /\ (not b' \/ a)
        /\ (not b' \/ b)
        /\ (not a \/ not b \/ c \/ c')
        /\ (not c' \/ a)
        /\ (not c' \/ b)
        /\ (not c' \/ not c)
```

```
Safety property:

  P  = (a /\ b) \/ c
  P' = (a' /\ b') \/ c'

  CNF_CASL(P) = (a \/ c) /\ (b \/ c)
  CNF_CASL(P') = (a' \/ c') /\ (b' \/ c')


Checking with Hets that T and CNF_CASL(T) are equivalent:
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

spec Check1 =
  pred a,b,c,a',b',c': ()
then %implies
 axiom
            (
                (a' <=> a /\ b)
            )
        /\
            (
                (b' <=> a /\ b)
            )
        /\
            (
                (c' <=> a /\ b /\ not c)
            )

  <=>    (
        (not a \/ not b \/ a')
        /\ (not a' \/ a)
        /\ (not a' \/ b)
        /\ (not a \/ not b \/ b')
        /\ (not b' \/ a)
        /\ (not b' \/ b)
        /\ (not a \/ not b \/ c \/ c')
        /\ (not c' \/ a)
        /\ (not c' \/ b)
        /\ (not c' \/ not c)
         )

end
```

```
[successfully proved with HETS interface]

Part II:
------------------------------------

once more the formulae (now all in CNF):

  I =  a /\ b /\ c

  T =     (not a \/ not b \/ a')
        /\ (not a' \/ a)
        /\ (not a' \/ b)
        /\ (not a \/ not b \/ b')
        /\ (not b' \/ a)
        /\ (not b' \/ b)
        /\ (not a \/ not b \/ c \/ c')
        /\ (not c' \/ a)
        /\ (not c' \/ b)
        /\ (not c' \/ not c)

  P = (a \/ c) /\ (b \/ c)

  P' = (a' \/ c') /\ (b' \/ c')

a) Check I=>P?

spec Check2 =
  pred a,b,c: ()
  axiom  a /\ b /\ c
then %implies
 axiom (a \/ c) /\ (b \/ c)
end

[successfully proved with HETS interface]


b) Check I/\T=>P'?

spec Check3 =
  pred a,b,c,a',b',c': ()
```

```
  axiom  a /\ b /\ c
  axiom      (not a \/ not b \/ a')
        /\ (not a' \/ a)
        /\ (not a' \/ b)
        /\ (not a \/ not b \/ b')
        /\ (not b' \/ a)
        /\ (not b' \/ b)
        /\ (not a \/ not b \/ c \/ c')
        /\ (not c' \/ a)
        /\ (not c' \/ b)
        /\ (not c' \/ not c)
then %implies
 axiom (a' \/ c') /\ (b' \/ c')
end

[successfully proved with HETS interface]


c) Set F1 :=P

  F1 = (a \/ c) /\ (b \/ c)

d) For every clause c in clauses(I), if c not-in clauses(F1),
check:
    I /\ T => c'?
    If it does, set F1 := F1 /\ c


clauses(I) = {a, b, c}
clauses(F1) = { (a \/ c) \/ (b \/ c)}

d-1) c = a

check if I /\ T => a':

spec Check4 =
  pred a,b,c,a',b',c': ()
  axiom  a /\ b /\ c
  axiom      (not a \/ not b \/ a')
        /\ (not a' \/ a)
        /\ (not a' \/ b)
        /\ (not a \/ not b \/ b')
```

```
        /\ (not b' \/ a)
        /\ (not b' \/ b)
        /\ (not a \/ not b \/ c \/ c')
        /\ (not c' \/ a)
        /\ (not c' \/ b)
        /\ (not c' \/ not c)
then %implies
 axiom a'
end
```

[proved with Hets]

Thus, we add a to F1, and obtain:

```
   F1 =     (a \/ c) /\ (b \/ c)
        /\ a
```

d−2) c= b

```
spec Check5 =
  pred a,b,c,a',b',c': ()
  axiom  a /\ b /\ c
  axiom      (not a \/ not b \/ a')
        /\ (not a' \/ a)
        /\ (not a' \/ b)
        /\ (not a \/ not b \/ b')
        /\ (not b' \/ a)
        /\ (not b' \/ b)
        /\ (not a \/ not b \/ c \/ c')
        /\ (not c' \/ a)
        /\ (not c' \/ b)
        /\ (not c' \/ not c)
then %implies
 axiom b'
end
```

Thus, we add b to F1, and obtain:

```
   F1 =     (a \/ c) /\ (b \/ c)
        /\ a
        /\ b
```

```
d-3) c= c

spec Check6 =
  pred a,b,c,a',b',c': ()
  axiom  a /\ b /\ c
  axiom       (not a \/ not b \/ a')
        /\ (not a' \/ a)
        /\ (not a' \/ b)
        /\ (not a \/ not b \/ b')
        /\ (not b' \/ a)
        /\ (not b' \/ b)
        /\ (not a \/ not b \/ c \/ c')
        /\ (not c' \/ a)
        /\ (not c' \/ b)
        /\ (not c' \/ not c)
then %implies
 axiom c'
end

[disproved with Hets]

Thus, we do not add c' to F1.

Part III:
-------------------------------------


a) Check: Is it the case that Fk /\ T => P'?
b) If it is, then
 b-1) set Fk+1 := P
 b-2) for every clause c in Fk , check if Fk/\T=>c'?
      If it is, set Fk+1 := Fk+1 /\ c
c) If Fk =Fk+1: done


k=1:
++++

a) do we have F1 /\ T => P' ?

spec Check7 =
  pred a,b,c,a',b',c': ()
  axiom       (a \/ c) /\ (b \/ c)
```

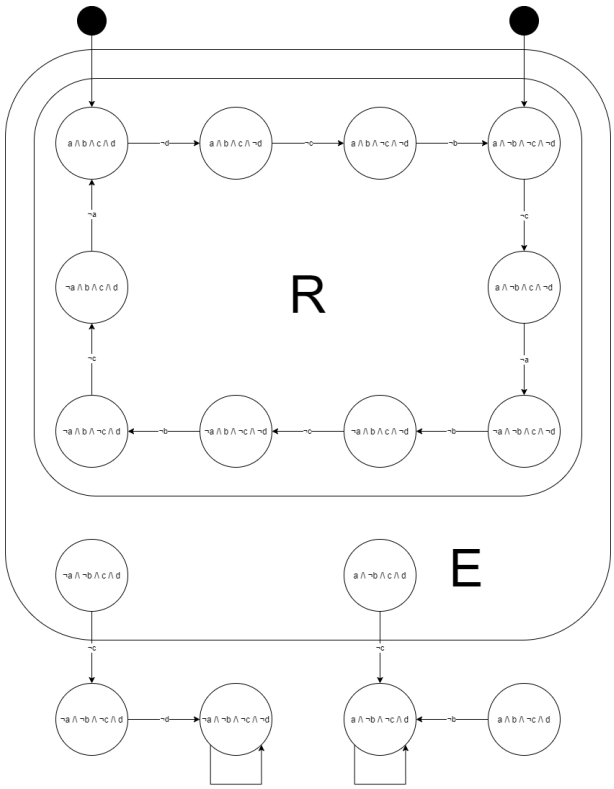
```
        /\ a /\ b
  axiom    (not a \/ not b \/ a')
        /\ (not a' \/ a)
        /\ (not a' \/ b)
        /\ (not a \/ not b \/ b')
        /\ (not b' \/ a)
        /\ (not b' \/ b)
        /\ (not a \/ not b \/ c \/ c')
        /\ (not c' \/ a)
        /\ (not c' \/ b)
        /\ (not c' \/ not c)
then %implies
 axiom  (a' \/ c') /\ (b' \/ c')
end

[proved with Hets]

b-1) F2 = (a \/ c) /\ (b \/ c)

b-2) clauses(F1) = { (a \/ c), (b \/ c), a, b }

    let c = (a \/ c)
    check F1 /\ T => c'

spec Check8 =
  pred a,b,c,a',b',c': ()
  axiom  (a \/ c) /\ (b \/ c)
        /\ a /\ b
  axiom      (not a \/ not b \/ a')
        /\ (not a' \/ a)
        /\ (not a' \/ b)
        /\ (not a \/ not b \/ b')
        /\ (not b' \/ a)
        /\ (not b' \/ b)
        /\ (not a \/ not b \/ c \/ c')
        /\ (not c' \/ a)
        /\ (not c' \/ b)
        /\ (not c' \/ not c)
then %implies
 axiom (a' \/ c')
end
```

```
[successfully proved with HETS]

thus F2 = (a \/ c) /\ (b \/ c)
        /\ (a \/ c)

    let c = (b \/ c)
    check F1 /\ T => c'


spec Check9 =
  pred a,b,c,a',b',c': ()
  axiom  (a \/ c) /\ (b \/ c)
        /\ a /\ b
  axiom       (not a \/ not b \/ a')
        /\ (not a' \/ a)
        /\ (not a' \/ b)
        /\ (not a \/ not b \/ b')
        /\ (not b' \/ a)
        /\ (not b' \/ b)
        /\ (not a \/ not b \/ c \/ c')
        /\ (not c' \/ a)
        /\ (not c' \/ b)
        /\ (not c' \/ not c)
then %implies
 axiom (b' \/ c')
end

[successfully proved with HETS]

thus F2 = (a \/ c) /\ (b \/ c)
        /\ (a \/ c) /\ (b \/ c)


    let c = a
    check F1 /\ T => c'

spec Check9 =
  pred a,b,c,a',b',c': ()
  axiom  (a \/ c) /\ (b \/ c)
        /\ a /\ b
  axiom       (not a \/ not b \/ a')
        /\ (not a' \/ a)
```

```
        /\ (not a' \/ b)
        /\ (not a \/ not b \/ b')
        /\ (not b' \/ a)
        /\ (not b' \/ b)
        /\ (not a \/ not b \/ c \/ c')
        /\ (not c' \/ a)
        /\ (not c' \/ b)
        /\ (not c' \/ not c)
then %implies
 axiom a'
end

[successfully proved with HETS]

thus F2 = (a \/ c) /\ (b \/ c)
        /\ (a \/ c)
        /\ (b \/ c)
        /\ a


    let c = b
    check F1 /\ T => c'

spec Check10 =
  pred a,b,c,a',b',c': ()
  axiom  (a \/ c) /\ (b \/ c)
        /\ a /\ b
  axiom       (not a \/ not b \/ a')
        /\ (not a' \/ a)
        /\ (not a' \/ b)
        /\ (not a \/ not b \/ b')
        /\ (not b' \/ a)
        /\ (not b' \/ b)
        /\ (not a \/ not b \/ c \/ c')
        /\ (not c' \/ a)
        /\ (not c' \/ b)
        /\ (not c' \/ not c)
then %implies
 axiom b'
end

[successfully proved with HETS]
```

```
thus F2 = (a \/ c) /\ (b \/ c)
        /\ (a \/ c)
        /\ (b \/ c)
        /\ a
        /\ b

c) thus F2 = (a \/ c) /\ (b \/ c) /\ (a \/ c) /\ a
        = (a \/ c) /\ (b \/ c) /\ a /\ b
        = F1

i.e. F2 = F1.
```

B.1.5 Second Four Variable Ladder Logic Example



spec TRANSITION =
preds a : ();
b : ();
c : ();

$d : ()$

preds $a' : ()$;
$b' : ()$;
$c' : ()$;
$d' : ()$

- $a' \Leftrightarrow (a \wedge \neg\, c) \vee (a \wedge d) \vee (a \wedge b) \vee (b \wedge c \wedge d)$
- $b' \Leftrightarrow (\neg\, a \wedge b) \vee (b \wedge c) \vee (\neg\, a \wedge c \wedge \neg\, d)$
- c'
  $\Leftrightarrow (\neg\, b \wedge c \wedge \neg\, d) \vee (\neg\, a \wedge b \wedge d) \vee (b \wedge c \wedge d)$
  $\vee\, (a \wedge \neg\, b \wedge \neg\, d)$
- d'
  $\Leftrightarrow (\neg\, a \wedge c \wedge d) \vee (\neg\, a \wedge b \wedge \neg\, c) \vee (b \wedge \neg\, c \wedge d)$
  $\vee\, (a \wedge \neg\, b \wedge d)$

end

spec TransitionTest =
TRANSITION
then %implies

- $a \wedge b \wedge c \wedge d \Rightarrow a' \wedge b' \wedge c' \wedge \neg\, d'$ %(t1)%
- $a \wedge b \wedge c \wedge \neg\, d \Rightarrow a' \wedge b' \wedge \neg\, c' \wedge \neg\, d'$ %(t2)%
- $a \wedge b \wedge \neg\, c \wedge \neg\, d \Rightarrow a' \wedge \neg\, b' \wedge \neg\, c' \wedge \neg\, d'$ %(t3)%
- $a \wedge \neg\, b \wedge \neg\, c \wedge \neg\, d \Rightarrow a' \wedge \neg\, b' \wedge c' \wedge \neg\, d'$ %(t4)%
- $a \wedge \neg\, b \wedge c \wedge \neg\, d \Rightarrow \neg\, a' \wedge \neg\, b' \wedge c' \wedge \neg\, d'$ %(t5)%
- $\neg\, a \wedge \neg\, b \wedge c \wedge \neg\, d \Rightarrow \neg\, a' \wedge b' \wedge c' \wedge \neg\, d'$ %(t6)%
- $\neg\, a \wedge b \wedge c \wedge \neg\, d \Rightarrow \neg\, a' \wedge b' \wedge \neg\, c' \wedge \neg\, d'$ %(t7)%
- $\neg\, a \wedge b \wedge \neg\, c \wedge \neg\, d \Rightarrow \neg\, a' \wedge b' \wedge \neg\, c' \wedge d'$ %(t8)%
- $\neg\, a \wedge b \wedge \neg\, c \wedge d \Rightarrow \neg\, a' \wedge b' \wedge c' \wedge d'$ %(t9)%
- $\neg\, a \wedge b \wedge c \wedge d \Rightarrow a' \wedge b' \wedge c' \wedge d'$ %(t10)%
- $\neg\, a \wedge \neg\, b \wedge c \wedge d \Rightarrow \neg\, a' \wedge \neg\, b' \wedge \neg\, c' \wedge d'$ %(t11)%
- $\neg\, a \wedge \neg\, b \wedge \neg\, c \wedge d \Rightarrow \neg\, a' \wedge \neg\, b' \wedge \neg\, c' \wedge \neg\, d'$ %(t12)%
- $\neg\, a \wedge \neg\, b \wedge \neg\, c \wedge \neg\, d \Rightarrow \neg\, a' \wedge \neg\, b' \wedge \neg\, c' \wedge \neg\, d'$ %(t13)%
- $a \wedge \neg\, b \wedge c \wedge d \Rightarrow a' \wedge \neg\, b' \wedge \neg\, c' \wedge d'$ %(t14)%
- $a \wedge b \wedge \neg\, c \wedge d \Rightarrow a' \wedge \neg\, b' \wedge \neg\, c' \wedge d'$ %(t15)%
- $a \wedge \neg\, b \wedge \neg\, c \wedge d \Rightarrow a' \wedge \neg\, b' \wedge \neg\, c' \wedge d'$ %(t16)%

end

spec SafeInitial =
TRANSITION
then $\bullet\ (a \wedge b \wedge c \wedge d) \vee (a \wedge \neg\, b \wedge \neg\, c \wedge \neg\, d)$
then %implies

- $c \vee (\neg\, a \wedge b) \vee (a \wedge \neg\, d)$ %(initial_safe)%

end

spec StepSafe =
  Transition
then %implies
  • $c \vee (\neg\ a \wedge b) \vee (a \wedge \neg\ d) \Rightarrow c' \vee (\neg\ a' \wedge b') \vee (a' \wedge \neg\ d')$
    %(step_safety)%
end

spec StepSafeWithInvariant =
  Transition
then %implies
  • $(c \vee (\neg\ a \wedge b) \vee (a \wedge \neg\ d))$
    $\wedge\ ((b \vee c \vee a) \wedge (\neg\ a \vee c \vee \neg\ d) \wedge (b \vee c \vee \neg\ d) \wedge (\neg\ d \vee b))$
    $\Rightarrow c' \vee (\neg\ a' \wedge b') \vee (a' \wedge \neg\ d')$
    %(step_safety_with_invariant)%
end

spec InvariantHoldsInitially =
  Transition
then • $(a \wedge b \wedge c \wedge d) \vee (a \wedge \neg\ b \wedge \neg\ c \wedge \neg\ d)$
then %implies
  • $(b \vee c \vee a) \wedge (\neg\ a \vee c \vee \neg\ d) \wedge (b \vee c \vee \neg\ d) \wedge (\neg\ d \vee b)$
    %(invariant_initially)%
end

spec InvariantStep =
  Transition
then %implies
  • $(b \vee c \vee a) \wedge (\neg\ a \vee c \vee \neg\ d) \wedge (b \vee c \vee \neg\ d) \wedge (\neg\ d \vee b)$
    $\Rightarrow (b \vee c \vee a) \wedge (\neg\ a \vee c \vee \neg\ d) \wedge (b \vee c \vee \neg\ d) \wedge (\neg\ d \vee b)$
    %(Invariant_step)%
end

IC3 run through:

Part I: the formulae how we read them off the automaton and their CNFs

---

Initialisation Formula:

I = (a /\ b /\ c /\ d) \/ (a /\ not b /\ not c /\ not d)
in CNF:

```
  CNF_CASL(I) = a /\ (not c \/ b) /\ (not d \/ b) /\ (not b \/ c)
/\ (not d \/ c)
        /\ (not b \/ d) /\ (not c \/ d)


Transition Formula:

  T =   a' <=> (a /\ not c) \/ (a /\ d) \/ (a /\ b)
\/ (b /\ c /\ d);
        b' <=> (not a /\ b) \/ (b /\ c) \/ (not a /\ c /\ not d);
        c' <=> (not b /\ c /\ not d) \/ (not a /\ b /\ d)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not d);
        d' <=> (not a /\ c /\ d) \/ (not a /\ b /\ not c)
\/ (b /\ not c /\ d) \/ (a /\ not b /\ d);

  CNF_CASL(T) = (not a \/ c \/ a')
        /\ (not a \/ not d \/ a')
        /\ (not a \/ not b \/ a')
        /\ (not b \/ not c \/ not d \/ a')
        /\ (a \/ b \/ not a')
        /\ (d \/ a \/ b \/ not a')
        /\ (a \/ not c \/ b \/ not a')
        /\ (d \/ not c \/ a \/ b \/ not a')
        /\ (d \/ not c \/ b \/ not a')
        /\ (a \/ c \/ not a')
        /\ (d \/ a \/ c \/ not a')
        /\ (a \/ b \/ c \/ not a')
        /\ (d \/ a \/ b \/ c \/ not a')
        /\ (a \/ d \/ not a')
        /\ (a \/ not c \/ d \/ not a')
        /\ (d \/ not c \/ a \/ not a')
        /\ (a \/ b \/ d \/ not a')
        /\ (a \/ not c \/ b \/ d \/ not a')
        /\ (a \/ not b \/ b')
        /\ (not b \/ not c \/ b')
        /\ (a \/ not c \/ d \/ b')
        /\ (b \/ not a \/ not b')
        /\ (c \/ not a \/ not b')
        /\ (c \/ b \/ not a \/ not b')
        /\ (b \/ not a \/ c \/ not b')
        /\ (b \/ c \/ not b')
        /\ (b \/ not a \/ not d \/ not b')
```

```
/\ (c \/ not a \/ not d \/ not b')
/\ (b \/ not d \/ not b')
/\ (c \/ b \/ not d \/ not b')
/\ (b \/ not c \/ d \/ c')
/\ (a \/ not b \/ not d \/ c')
/\ (not b \/ not c \/ not d \/ c')
/\ (not a \/ b \/ d \/ c')
/\ (b \/ c \/ a \/ not c')
/\ (d \/ c \/ b \/ a \/ not c')
/\ (b \/ not d \/ a \/ not c')
/\ (d \/ not b \/ c \/ a \/ not c')
/\ (d \/ c \/ a \/ not c')
/\ (b \/ not d \/ c \/ a \/ not c')
/\ (d \/ not b \/ a \/ not c')
/\ (b \/ c \/ d \/ a \/ not c')
/\ (not a \/ not b \/ c \/ not c')
/\ (d \/ not b \/ c \/ not c')
/\ (not a \/ c \/ not b \/ not c')
/\ (d \/ c \/ not b \/ not c')
/\ (not a \/ not d \/ c \/ not b \/ not c')
/\ (not a \/ not b \/ d \/ not c')
/\ (d \/ not b \/ not c')
/\ (not a \/ c \/ d \/ not b \/ not c')
/\ (not a \/ c \/ b \/ not d \/ not c')
/\ (b \/ c \/ not d \/ not c')
/\ (not a \/ not d \/ b \/ not c')
/\ (b \/ not d \/ not c')
/\ (not a \/ not b \/ c \/ not d \/ not c')
/\ (not a \/ c \/ not d \/ not c')
/\ (not a \/ not d \/ c \/ not c')
/\ (b \/ not d \/ c \/ not c')
/\ (a \/ not c \/ not d \/ d')
/\ (a \/ not b \/ c \/ d')
/\ (not b \/ c \/ not d \/ d')
/\ (not a \/ b \/ not d \/ d')
/\ (b \/ c \/ a \/ not d')
/\ (b \/ d \/ a \/ not d')
/\ (not c \/ d \/ b \/ a \/ not d')
/\ (b \/ d \/ not c \/ a \/ not d')
/\ (not c \/ d \/ a \/ not d')
/\ (b \/ c \/ d \/ a \/ not d')
/\ (not a \/ not c \/ not b \/ not d')
```

```
        /\ (not a \/ d \/ not c \/ not b \/ not d')
        /\ (not c \/ d \/ not b \/ not d')
        /\ (not a \/ d \/ not b \/ not d')
        /\ (not c \/ not a \/ d \/ not b \/ not d')
        /\ (not a \/ c \/ d \/ not b \/ not d')
        /\ (not a \/ b \/ d \/ not d')
        /\ (not c \/ not a \/ b \/ d \/ not d')
        /\ (not a \/ c \/ b \/ d \/ not d')
        /\ (b \/ c \/ d \/ not d')
        /\ (not a \/ d \/ b \/ not d')
        /\ (b \/ d \/ not d')
        /\ (not c \/ d \/ b \/ not d')
        /\ (not a \/ not c \/ d \/ not d')
        /\ (b \/ not a \/ not c \/ d \/ not d')
        /\ (not a \/ d \/ not c \/ not d')
        /\ (b \/ d \/ not c \/ not d')
        /\ (not c \/ d \/ not d')
        /\ (not a \/ d \/ not d')
        /\ (not a \/ c \/ d \/ not d')

Safety property:

  P  = c \/ (not a /\ b) \/ (a /\ not d)
  P' = c' \/ (not a' /\ b') \/ (a' /\ not d')

  CNF_CASL(P) = (b \/ c \/ a) /\ (not a \/ c \/ not d)
/\ (b \/ c \/ not d)

  CNF_CASL(P') = (b' \/ c' \/ a') /\ (not a' \/ c' \/ not d')
/\ (b' \/ c' \/ not d')

Checking with Hets that T and CNF_CASL(T) are equivalent:
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

spec Check1 =
  pred a,b,c,d,a',b',c',d': ()
then %implies
 axiom
        (a' <=> (a /\ not c) \/ (a /\ d) \/ (a /\ b)
\/ (b /\ c /\ d))
        /\ (b' <=> (not a /\ b) \/ (b /\ c) \/ (not a /\ c /\ not d))
        /\ (c' <=> (not b /\ c /\ not d) \/ (not a /\ b /\ d)
```

```
\/ (b /\ c /\ d) \/ (a /\ not b /\ not d))
        /\ (d' <=> (not a /\ c /\ d) \/ (not a /\ b /\ not c)
\/ (b /\ not c /\ d) \/ (a /\ not b /\ d))

  <=>     (
         (not a \/ c \/ a')
         /\ (not a \/ not d \/ a')
         /\ (not a \/ not b \/ a')
         /\ (not b \/ not c \/ not d \/ a')
         /\ (a \/ b \/ not a')
         /\ (d \/ a \/ b \/ not a')
         /\ (a \/ not c \/ b \/ not a')
         /\ (d \/ not c \/ a \/ b \/ not a')
         /\ (d \/ not c \/ b \/ not a')
         /\ (a \/ c \/ not a')
         /\ (d \/ a \/ c \/ not a')
         /\ (a \/ b \/ c \/ not a')
         /\ (d \/ a \/ b \/ c \/ not a')
         /\ (a \/ d \/ not a')
         /\ (a \/ not c \/ d \/ not a')
         /\ (d \/ not c \/ a \/ not a')
         /\ (a \/ b \/ d \/ not a')
         /\ (a \/ not c \/ b \/ d \/ not a')
         /\ (a \/ not b \/ b')
         /\ (not b \/ not c \/ b')
         /\ (a \/ not c \/ d \/ b')
         /\ (b \/ not a \/ not b')
         /\ (c \/ not a \/ not b')
         /\ (c \/ b \/ not a \/ not b')
         /\ (b \/ not a \/ c \/ not b')
         /\ (b \/ c \/ not b')
         /\ (b \/ not a \/ not d \/ not b')
         /\ (c \/ not a \/ not d \/ not b')
         /\ (b \/ not d \/ not b')
         /\ (c \/ b \/ not d \/ not b')
         /\ (b \/ not c \/ d \/ c')
         /\ (a \/ not b \/ not d \/ c')
         /\ (not b \/ not c \/ not d \/ c')
         /\ (not a \/ b \/ d \/ c')
         /\ (b \/ c \/ a \/ not c')
         /\ (d \/ c \/ b \/ a \/ not c')
         /\ (b \/ not d \/ a \/ not c')
```

```
/\ (d \/ not b \/ c \/ a \/ not c')
/\ (d \/ c \/ a \/ not c')
/\ (b \/ not d \/ c \/ a \/ not c')
/\ (d \/ not b \/ a \/ not c')
/\ (b \/ c \/ d \/ a \/ not c')
/\ (not a \/ not b \/ c \/ not c')
/\ (d \/ not b \/ c \/ not c')
/\ (not a \/ c \/ not b \/ not c')
/\ (d \/ c \/ not b \/ not c')
/\ (not a \/ not d \/ c \/ not b \/ not c')
/\ (not a \/ not b \/ d \/ not c')
/\ (d \/ not b \/ not c')
/\ (not a \/ c \/ d \/ not b \/ not c')
/\ (not a \/ c \/ b \/ not d \/ not c')
/\ (b \/ c \/ not d \/ not c')
/\ (not a \/ not d \/ b \/ not c')
/\ (b \/ not d \/ not c')
/\ (not a \/ not b \/ c \/ not d \/ not c')
/\ (not a \/ c \/ not d \/ not c')
/\ (not a \/ not d \/ c \/ not c')
/\ (b \/ not d \/ c \/ not c')
/\ (a \/ not c \/ not d \/ d')
/\ (a \/ not b \/ c \/ d')
/\ (not b \/ c \/ not d \/ d')
/\ (not a \/ b \/ not d \/ d')
/\ (b \/ c \/ a \/ not d')
/\ (b \/ d \/ a \/ not d')
/\ (not c \/ d \/ b \/ a \/ not d')
/\ (b \/ d \/ not c \/ a \/ not d')
/\ (not c \/ d \/ a \/ not d')
/\ (b \/ c \/ d \/ a \/ not d')
/\ (not a \/ not c \/ not b \/ not d')
/\ (not a \/ d \/ not c \/ not b \/ not d')
/\ (not c \/ d \/ not b \/ not d')
/\ (not a \/ d \/ not b \/ not d')
/\ (not c \/ not a \/ d \/ not b \/ not d')
/\ (not a \/ c \/ d \/ not b \/ not d')
/\ (not a \/ b \/ d \/ not d')
/\ (not c \/ not a \/ b \/ d \/ not d')
/\ (not a \/ c \/ b \/ d \/ not d')
/\ (b \/ c \/ d \/ not d')
/\ (not a \/ d \/ b \/ not d')
```

```
        /\ (b \/ d \/ not d')
        /\ (not c \/ d \/ b \/ not d')
        /\ (not a \/ not c \/ d \/ not d')
        /\ (b \/ not a \/ not c \/ d \/ not d')
        /\ (not a \/ d \/ not c \/ not d')
        /\ (b \/ d \/ not c \/ not d')
        /\ (not c \/ d \/ not d')
        /\ (not a \/ d \/ not d')
        /\ (not a \/ c \/ d \/ not d')
         )

end

[successfully proved with HETS interface]


Part II:
------------------------------------

once more the formulae (now all in CNF):

  I = a /\ (not c \/ b) /\ (not d \/ b) /\ (not b \/ c) /\ (not d \/ c)
/\ (not b \/ d) /\ (not c \/ d)

  T =     (a' <=> (a /\ not c) \/ (a /\ d) \/ (a /\ b)
\/ (b /\ c /\ d))
        /\ (b' <=> (not a /\ b) \/ (b /\ c)
\/ (not a /\ c /\ not d))
        /\ (c' <=> (not b /\ c /\ not d) \/ (not a /\ b /\ d)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not d))
        /\ (d' <=> (not a /\ c /\ d) \/ (not a /\ b /\ not c)
\/ (b /\ not c /\ d) \/ (a /\ not b /\ d))

  P = (b \/ c \/ a) /\ (not a \/ c \/ not d) /\ (b \/ c \/ not d)

  P' = (b' \/ c' \/ a') /\ (not a' \/ c' \/ not d')
/\ (b' \/ c' \/ not d')

a) Check I=>P?

spec Check2 =
  pred a,b,c,d: ()
```

```
  axiom  a /\ (not c \/ b) /\ (not d \/ b) /\ (not b \/ c)
/\ (not d \/ c) /\ (not b \/ d) /\ (not c \/ d)
then %implies
 axiom (b \/ c \/ a) /\ (not a \/ c \/ not d) /\ (b \/ c \/ not d)
end

[successfully proved with HETS interface]
```

b) Check I/\T=>P‘?

```
spec Check3 =
  pred a,b,c,d,a',b',c',d': ()
  axiom  a /\ (not c \/ b) /\ (not d \/ b) /\ (not b \/ c)
/\ (not d \/ c) /\ (not b \/ d) /\ (not c \/ d)
  axiom      (a' <=> (a /\ not c) \/ (a /\ d) \/ (a /\ b)
\/ (b /\ c /\ d))
        /\ (b' <=> (not a /\ b) \/ (b /\ c)
\/ (not a /\ c /\ not d))
        /\ (c' <=> (not b /\ c /\ not d) \/ (not a /\ b /\ d)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not d))
        /\ (d' <=> (not a /\ c /\ d) \/ (not a /\ b /\ not c)
\/ (b /\ not c /\ d) \/ (a /\ not b /\ d))

then %implies
 axiom (b' \/ c' \/ a') /\ (not a' \/ c' \/ not d')
/\ (b' \/ c' \/ not d')
end

[successfully proved with HETS interface]
```

c) Set F1 :=P

```
  F1 = (b \/ c \/ a) /\ (not a \/ c \/ not d) /\ (b \/ c \/ not d)
```

d) For every clause c in clauses(I), if c not−in clauses(F1), check:
    I /\ T => c‘?
    If it does, set F1 := F1 /\ c

```
clauses(I) = {  a,
                (not c \/ b),
                (not d \/ b),
                (not b \/ c),
                (not d \/ c),
                (not b \/ d),
                (not c \/ d)}

clauses(F1) = { (b \/ c \/ a),
                (not a \/ c \/ not d),
                (b \/ c \/ not d)}

d-1) c = a

check if I /\ T => a':

spec Check4 =
  pred a,b,c,d,a',b',c',d': ()
  axiom  a /\ (not c \/ b) /\ (not d \/ b) /\ (not b \/ c)
/\ (not d \/ c) /\ (not b \/ d) /\ (not c \/ d)
  axiom      (a' <=> (a /\ not c) \/ (a /\ d) \/ (a /\ b)
\/ (b /\ c /\ d))
        /\ (b' <=> (not a /\ b) \/ (b /\ c)
\/ (not a /\ c /\ not d))
        /\ (c' <=> (not b /\ c /\ not d) \/ (not a /\ b /\ d)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not d))
        /\ (d' <=> (not a /\ c /\ d) \/ (not a /\ b /\ not c)
\/ (b /\ not c /\ d) \/ (a /\ not b /\ d))
then %implies
 axiom a'
end

[proved with Hets]

Thus, we add a to F1, and obtain:

   F1 = (b \/ c \/ a)
        /\ (not a \/ c \/ not d)
        /\ (b \/ c \/ not d)
        /\ a

d-2) c= (not c \/ b)
```

```
spec Check5 =
  pred a,b,c,d,a',b',c',d': ()
  axiom  a /\ (not c \/ b) /\ (not d \/ b) /\ (not b \/ c)
/\ (not d \/ c) /\ (not b \/ d) /\ (not c \/ d)
  axiom      (a' <=> (a /\ not c) \/ (a /\ d) \/ (a /\ b)
\/ (b /\ c /\ d))
        /\ (b' <=> (not a /\ b) \/ (b /\ c)
\/ (not a /\ c /\ not d))
        /\ (c' <=> (not b /\ c /\ not d) \/ (not a /\ b /\ d)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not d))
        /\ (d' <=> (not a /\ c /\ d) \/ (not a /\ b /\ not c)
\/ (b /\ not c /\ d) \/ (a /\ not b /\ d))
then %implies
 axiom (not c' \/ b')
end
```

[disproved with Hets]

Thus, we do not add (not c' \/ b') to F1.

d–3) c= (not d \/ b)

```
spec Check6 =
  pred a,b,c,d,a',b',c',d': ()
  axiom  a /\ (not c \/ b) /\ (not d \/ b) /\ (not b \/ c)
/\ (not d \/ c) /\ (not b \/ d) /\ (not c \/ d)
  axiom      (a' <=> (a /\ not c) \/ (a /\ d) \/ (a /\ b)
\/ (b /\ c /\ d))
        /\ (b' <=> (not a /\ b) \/ (b /\ c)
\/ (not a /\ c /\ not d))
        /\ (c' <=> (not b /\ c /\ not d) \/ (not a /\ b /\ d)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not d))
        /\ (d' <=> (not a /\ c /\ d) \/ (not a /\ b /\ not c)
\/ (b /\ not c /\ d) \/ (a /\ not b /\ d))
then %implies
 axiom (not d' \/ b')
end
```

[proved with Hets]

Thus, we add (not d \/ b) to F1, and obtain:

```
   F1 = (b \/ c \/ a)
        /\ (not a \/ c \/ not d)
        /\ (b \/ c \/ not d)
        /\ a
        /\ (not d \/ b)

d-4) c= (not b \/ c)

spec Check7 =
  pred a,b,c,d,a',b',c',d': ()
  axiom  a /\ (not c \/ b) /\ (not d \/ b) /\ (not b \/ c)
/\ (not d \/ c) /\ (not b \/ d) /\ (not c \/ d)
  axiom      (a' <=> (a /\ not c) \/ (a /\ d) \/ (a /\ b)
\/ (b /\ c /\ d))
        /\ (b' <=> (not a /\ b) \/ (b /\ c)
\/ (not a /\ c /\ not d))
        /\ (c' <=> (not b /\ c /\ not d) \/ (not a /\ b /\ d)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not d))
        /\ (d' <=> (not a /\ c /\ d) \/ (not a /\ b /\ not c)
\/ (b /\ not c /\ d) \/ (a /\ not b /\ d))
then %implies
 axiom (not b' \/ c')
end

[proved with Hets]

Thus, we add (not b \/ c) to F1, and obtain:

   F1 = (b \/ c \/ a)
        /\ (not a \/ c \/ not d)
        /\ (b \/ c \/ not d)
        /\ a
        /\ (not d \/ b)
        /\ (not b \/ c)

d-5) c= (not d \/ c)

spec Check8 =
  pred a,b,c,d,a',b',c',d': ()
  axiom  a /\ (not c \/ b) /\ (not d \/ b) /\ (not b \/ c)
/\ (not d \/ c) /\ (not b \/ d) /\ (not c \/ d)
```

```
  axiom       (a' <=> (a /\ not c) \/ (a /\ d) \/ (a /\ b)
\/ (b /\ c /\ d))
        /\ (b' <=> (not a /\ b) \/ (b /\ c)
\/ (not a /\ c /\ not d))
        /\ (c' <=> (not b /\ c /\ not d) \/ (not a /\ b /\ d)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not d))
        /\ (d' <=> (not a /\ c /\ d) \/ (not a /\ b /\ not c)
\/ (b /\ not c /\ d) \/ (a /\ not b /\ d))
then %implies
 axiom (not d' \/ c')
end

[proved with Hets]

Thus, we add (not d \/ c) to F1, and obtain:

   F1 = (b \/ c \/ a)
        /\ (not a \/ c \/ not d)
        /\ (b \/ c \/ not d)
        /\ a
        /\ (not d \/ b)
        /\ (not b \/ c)
        /\ (not d \/ c)

d-6) c= (not b \/ d)

spec Check9 =
  pred a,b,c,d,a',b',c',d': ()
  axiom  a /\ (not c \/ b) /\ (not d \/ b) /\ (not b \/ c)
/\ (not d \/ c) /\ (not b \/ d) /\ (not c \/ d)
  axiom       (a' <=> (a /\ not c) \/ (a /\ d) \/ (a /\ b)
\/ (b /\ c /\ d))
        /\ (b' <=> (not a /\ b) \/ (b /\ c)
\/ (not a /\ c /\ not d))
        /\ (c' <=> (not b /\ c /\ not d) \/ (not a /\ b /\ d)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not d))
        /\ (d' <=> (not a /\ c /\ d) \/ (not a /\ b /\ not c)
\/ (b /\ not c /\ d) \/ (a /\ not b /\ d))
then %implies
 axiom (not b' \/ d')
end
```

```
[disproved with Hets]

Thus, we do not add (not b' \/ d') to F1.

d-7) c= (not c \/ d)

spec Check10 =
  pred a,b,c,d,a',b',c',d': ()
  axiom  a /\ (not c \/ b) /\ (not d \/ b) /\ (not b \/ c)
/\ (not d \/ c) /\ (not b \/ d) /\ (not c \/ d)
  axiom      (a' <=> (a /\ not c) \/ (a /\ d) \/ (a /\ b)
\/ (b /\ c /\ d))
        /\ (b' <=> (not a /\ b) \/ (b /\ c)
\/ (not a /\ c /\ not d))
        /\ (c' <=> (not b /\ c /\ not d) \/ (not a /\ b /\ d)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not d))
        /\ (d' <=> (not a /\ c /\ d) \/ (not a /\ b /\ not c)
\/ (b /\ not c /\ d) \/ (a /\ not b /\ d))
then %implies
 axiom (not c' \/ d')
end

[disproved with Hets]

Thus, we do not add (not c' \/ d') to F1.

Part III:
-------------------------------------

a) Check: Is it the case that Fk /\ T => P'?
b) If it is, then
 b-1) set Fk+1 := P
 b-2) for every clause c in Fk , check if Fk/\T=>c'?
      If it is, set Fk+1 := Fk+1 /\ c
c) If Fk =Fk+1: done


k=1:
++++

a) do we have F1 /\ T => P' ?
```

```
spec Check11 =
  pred a,b,c,d,a',b',c',d': ()
  axiom (b \/ c \/ a)
        /\ (not a \/ c \/ not d)
        /\ (b \/ c \/ not d)
        /\ a
        /\ (not d \/ b)
        /\ (not b \/ c)
        /\ (not d \/ c)
  axiom     (a' <=> (a /\ not c) \/ (a /\ d) \/ (a /\ b)
\/ (b /\ c /\ d))
        /\ (b' <=> (not a /\ b) \/ (b /\ c)
\/ (not a /\ c /\ not d))
        /\ (c' <=> (not b /\ c /\ not d) \/ (not a /\ b /\ d)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not d))
        /\ (d' <=> (not a /\ c /\ d) \/ (not a /\ b /\ not c)
\/ (b /\ not c /\ d) \/ (a /\ not b /\ d))


then %implies
 axiom  (b' \/ c' \/ a') /\ (not a' \/ c' \/ not d')
/\ (b' \/ c' \/ not d')
end

[proved with Hets]

b-1) F2 = (b \/ c \/ a) /\ (not a \/ c \/ not d)
/\ (b \/ c \/ not d)

b-2) clauses(F1) = {(b \/ c \/ a),
        (not a \/ c \/ not d),
        (b \/ c \/ not d),
        a,
        (not d \/ b),
        (not b \/ c),
        (not d \/ c)}

    let c = (b \/ c \/ a)
    check F1 /\ T => c'

spec Check12 =
  pred a,b,c,d,a',b',c',d': ()
```

```
  axiom  (b \/ c \/ a)
        /\ (not a \/ c \/ not d)
        /\ (b \/ c \/ not d)
        /\ a
        /\ (not d \/ b)
        /\ (not b \/ c)
        /\ (not d \/ c)

  axiom      (a' <=> (a /\ not c) \/ (a /\ d) \/ (a /\ b)
\/ (b /\ c /\ d))
        /\ (b' <=> (not a /\ b) \/ (b /\ c)
\/ (not a /\ c /\ not d))
        /\ (c' <=> (not b /\ c /\ not d) \/ (not a /\ b /\ d)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not d))
        /\ (d' <=> (not a /\ c /\ d) \/ (not a /\ b /\ not c)
\/ (b /\ not c /\ d) \/ (a /\ not b /\ d))
then %implies
 axiom (b' \/ c' \/ a')
end

[successfully proved with HETS]

thus F2 =  (b \/ c \/ a)
        /\ (not a \/ c \/ not d)
        /\ (b \/ c \/ not d)
        /\ (b \/ c \/ a)


    let c = (not a \/ c \/ not d)
    check F1 /\ T => c'

spec Check13 =
  pred a,b,c,d,a',b',c',d': ()
  axiom  (b \/ c \/ a)
        /\ (not a \/ c \/ not d)
        /\ (b \/ c \/ not d)
        /\ a
        /\ (not d \/ b)
        /\ (not b \/ c)
        /\ (not d \/ c)
  axiom      (a' <=> (a /\ not c) \/ (a /\ d) \/ (a /\ b)
\/ (b /\ c /\ d))
```

```
        /\ (b' <=> (not a /\ b) \/ (b /\ c)
\/ (not a /\ c /\ not d))
        /\ (c' <=> (not b /\ c /\ not d) \/ (not a /\ b /\ d)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not d))
        /\ (d' <=> (not a /\ c /\ d) \/ (not a /\ b /\ not c)
\/ (b /\ not c /\ d) \/ (a /\ not b /\ d))
then %implies
 axiom (not a' \/ c' \/ not d')
end

[successfully proved with HETS]

thus F2 =  (b \/ c \/ a)
        /\ (not a \/ c \/ not d)
        /\ (b \/ c \/ not d)
        /\ (b \/ c \/ a)
        /\ (not a \/ c \/ not d)


    let c = (b \/ c \/ not d)
    check F1 /\ T => c'

spec Check14 =
  pred a,b,c,d,a',b',c',d': ()
  axiom  (b \/ c \/ a)
        /\ (not a \/ c \/ not d)
        /\ (b \/ c \/ not d)
        /\ a
        /\ (not d \/ b)
        /\ (not b \/ c)
        /\ (not d \/ c)
  axiom      (a' <=> (a /\ not c) \/ (a /\ d) \/ (a /\ b)
\/ (b /\ c /\ d))
        /\ (b' <=> (not a /\ b) \/ (b /\ c)
\/ (not a /\ c /\ not d))
        /\ (c' <=> (not b /\ c /\ not d) \/ (not a /\ b /\ d)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not d))
        /\ (d' <=> (not a /\ c /\ d) \/ (not a /\ b /\ not c)
\/ (b /\ not c /\ d) \/ (a /\ not b /\ d))
then %implies
 axiom (b' \/ c' \/ not d')
end
```

```
[successfully proved with HETS]

thus F2 =  (b \/ c \/ a)
        /\ (not a \/ c \/ not d)
        /\ (b \/ c \/ not d)
        /\ (b \/ c \/ a)
        /\ (not a \/ c \/ not d)
        /\ (b \/ c \/ not d)


    let c = a
    check F1 /\ T => c'

spec Check15 =
  pred a,b,c,d,a',b',c',d': ()
  axiom  (b \/ c \/ a)
        /\ (not a \/ c \/ not d)
        /\ (b \/ c \/ not d)
        /\ a
        /\ (not d \/ b)
        /\ (not b \/ c)
        /\ (not d \/ c)
  axiom      (a' <=> (a /\ not c) \/ (a /\ d) \/ (a /\ b)
\/ (b /\ c /\ d))
        /\ (b' <=> (not a /\ b) \/ (b /\ c)
\/ (not a /\ c /\ not d))
        /\ (c' <=> (not b /\ c /\ not d) \/ (not a /\ b /\ d)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not d))
        /\ (d' <=> (not a /\ c /\ d) \/ (not a /\ b /\ not c)
\/ (b /\ not c /\ d) \/ (a /\ not b /\ d))
then %implies
 axiom a'
end

[disproved with Hets]

Thus, we do not add a' to F1.


    let c = (not d \/ b)
    check F1 /\ T => c'
```

```
spec Check16 =
  pred a,b,c,d,a',b',c',d': ()
  axiom  (b \/ c \/ a)
        /\ (not a \/ c \/ not d)
        /\ (b \/ c \/ not d)
        /\ a
        /\ (not d \/ b)
        /\ (not b \/ c)
        /\ (not d \/ c)
  axiom       (a' <=> (a /\ not c) \/ (a /\ d) \/ (a /\ b)
\/ (b /\ c /\ d))
        /\ (b' <=> (not a /\ b) \/ (b /\ c)
\/ (not a /\ c /\ not d))
        /\ (c' <=> (not b /\ c /\ not d) \/ (not a /\ b /\ d)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not d))
        /\ (d' <=> (not a /\ c /\ d) \/ (not a /\ b /\ not c)
\/ (b /\ not c /\ d) \/ (a /\ not b /\ d))
then %implies
 axiom (not d' \/ b')
end

[successfully proved with HETS]

thus F2 =  (b \/ c \/ a)
        /\ (not a \/ c \/ not d)
        /\ (b \/ c \/ not d)
        /\ (b \/ c \/ a)
        /\ (not a \/ c \/ not d)
        /\ (b \/ c \/ not d)
        /\ (not d \/ b)


    let c = (not b \/ c)
    check F1 /\ T => c'

spec Check17 =
  pred a,b,c,d,a',b',c',d': ()
  axiom  (b \/ c \/ a)
        /\ (not a \/ c \/ not d)
        /\ (b \/ c \/ not d)
        /\ a
```

```
        /\ (not d \/ b)
        /\ (not b \/ c)
        /\ (not d \/ c)
  axiom      (a' <=> (a /\ not c) \/ (a /\ d) \/ (a /\ b)
\/ (b /\ c /\ d))
        /\ (b' <=> (not a /\ b) \/ (b /\ c)
\/ (not a /\ c /\ not d))
        /\ (c' <=> (not b /\ c /\ not d) \/ (not a /\ b /\ d)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not d))
        /\ (d' <=> (not a /\ c /\ d) \/ (not a /\ b /\ not c)
\/ (b /\ not c /\ d) \/ (a /\ not b /\ d))
then %implies
 axiom (not b' \/ c')
end

[disproved with Hets]

Thus, we do not add (not b' \/ c') to F1.


    let c = (not d \/ c)
    check F1 /\ T => c'

spec Check18 =
  pred a,b,c,d,a',b',c',d': ()
  axiom  (b \/ c \/ a)
        /\ (not a \/ c \/ not d)
        /\ (b \/ c \/ not d)
        /\ a
        /\ (not d \/ b)
        /\ (not b \/ c)
        /\ (not d \/ c)
  axiom      (a' <=> (a /\ not c) \/ (a /\ d) \/ (a /\ b) \/ (b /\ c /\ d))
        /\ (b' <=> (not a /\ b) \/ (b /\ c) \/ (not a /\ c /\ not d))
        /\ (c' <=> (not b /\ c /\ not d) \/ (not a /\ b /\ d)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not d))
        /\ (d' <=> (not a /\ c /\ d) \/ (not a /\ b /\ not c)
\/ (b /\ not c /\ d) \/ (a /\ not b /\ d))
then %implies
 axiom (not d' \/ c')
end
```

```
[successfully proved with HETS]

thus F2 =  (b \/ c \/ a)
        /\ (not a \/ c \/ not d)
        /\ (b \/ c \/ not d)
        /\ (b \/ c \/ a)
        /\ (not a \/ c \/ not d)
        /\ (b \/ c \/ not d)
        /\ (not d \/ b)
        /\ (not d \/ c)


c) thus F2 = (b \/ c \/ a)
        /\ (not a \/ c \/ not d)
        /\ (b \/ c \/ not d)
        /\ (b \/ c \/ a)
        /\ (not a \/ c \/ not d)
        /\ (b \/ c \/ not d)
        /\ (not d \/ b)
        /\ (not d \/ c)

        = (b \/ c \/ a)
        /\ (not a \/ c \/ not d)
        /\ (b \/ c \/ not d)
        /\ (not d \/ b)
        /\ (not d \/ c)

        != F1

i.e. F2 = F1.

Part IV: Repeat part 3 until Fk =Fk+1:
--------------------------------------

a) Check: Is it the case that Fk /\ T => P'?
b) If it is, then
 b-1) set Fk+1 := P
 b-2) for every clause c in Fk , check if Fk/\T=>c'?
      If it is, set Fk+1 := Fk+1 /\ c
c) If Fk =Fk+1: done
```

```
k=1:
++++

a) do we have F2 /\ T => P' ?

spec Check19 =
  pred a,b,c,d,a',b',c',d': ()
  axiom       (b \/ c \/ a)
        /\ (not a \/ c \/ not d)
        /\ (b \/ c \/ not d)
        /\ (not d \/ b)
        /\ (not d \/ c)
  axiom      (a' <=> (a /\ not c) \/ (a /\ d) \/ (a /\ b)
\/ (b /\ c /\ d))
        /\ (b' <=> (not a /\ b) \/ (b /\ c)
\/ (not a /\ c /\ not d))
        /\ (c' <=> (not b /\ c /\ not d) \/ (not a /\ b /\ d)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not d))
        /\ (d' <=> (not a /\ c /\ d) \/ (not a /\ b /\ not c)
\/ (b /\ not c /\ d) \/ (a /\ not b /\ d))
then %implies
 axiom  (b' \/ c' \/ a') /\ (not a' \/ c' \/ not d')
/\ (b' \/ c' \/ not d')
end

[proved with Hets] - "YEEEEEEEESSSSSSSSSSS"

b-1) F3 = (b \/ c \/ a) /\ (not a \/ c \/ not d)
/\ (b \/ c \/ not d)

b-2) clauses(F2) = { (b \/ c \/ a),
        (not a \/ c \/ not d),
        (b \/ c \/ not d),
        (not d \/ b),
        (not d \/ c)}

    let c = (b \/ c \/ a)
    check F2 /\ T => c'

spec Check20 =
  pred a,b,c,d,a',b',c',d': ()
  axiom  (b \/ c \/ a)
```

```
        /\ (not a \/ c \/ not d)
        /\ (b \/ c \/ not d)
        /\ (not d \/ b)
        /\ (not d \/ c)
  axiom      (a' <=> (a /\ not c) \/ (a /\ d) \/ (a /\ b) \/ (b /\ c /\ d))
        /\ (b' <=> (not a /\ b) \/ (b /\ c) \/ (not a /\ c /\ not d))
        /\ (c' <=> (not b /\ c /\ not d) \/ (not a /\ b /\ d) \/ (b /\ c /\
        /\ (d' <=> (not a /\ c /\ d) \/ (not a /\ b /\ not c) \/ (b /\ not c
then %implies
 axiom (b' \/ c' \/ a')
end

[successfully proved with HETS]

thus F3 = (b \/ c \/ a)
        /\ (not a \/ c \/ not d)
        /\ (b \/ c \/ not d)
        /\ (b \/ c \/ a)


    let c = (not a \/ c \/ not d)
    check F2 /\ T => c'

spec Check21 =
  pred a,b,c,d,a',b',c',d': ()
  axiom  (b \/ c \/ a)
        /\ (not a \/ c \/ not d)
        /\ (b \/ c \/ not d)
        /\ (not d \/ b)
        /\ (not d \/ c)
  axiom      (a' <=> (a /\ not c) \/ (a /\ d) \/ (a /\ b)
\/ (b /\ c /\ d))
        /\ (b' <=> (not a /\ b) \/ (b /\ c)
\/ (not a /\ c /\ not d))
        /\ (c' <=> (not b /\ c /\ not d) \/ (not a /\ b /\ d)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not d))
        /\ (d' <=> (not a /\ c /\ d) \/ (not a /\ b /\ not c)
\/ (b /\ not c /\ d) \/ (a /\ not b /\ d))
then %implies
 axiom (not a' \/ c' \/ not d')
end
```

```
[successfully proved with HETS]

thus F3 = (b \/ c \/ a)
        /\ (not a \/ c \/ not d)
        /\ (b \/ c \/ not d)
        /\ (b \/ c \/ a)
        /\ (not a \/ c \/ not d)


    let c = (b \/ c \/ not d)
    check F2 /\ T => c'

spec Check21 =
  pred a,b,c,d,a',b',c',d': ()
  axiom  (b \/ c \/ a)
        /\ (not a \/ c \/ not d)
        /\ (b \/ c \/ not d)
        /\ (not d \/ b)
        /\ (not d \/ c)
  axiom       (a' <=> (a /\ not c) \/ (a /\ d) \/ (a /\ b)
\/ (b /\ c /\ d))
        /\ (b' <=> (not a /\ b) \/ (b /\ c)
\/ (not a /\ c /\ not d))
        /\ (c' <=> (not b /\ c /\ not d) \/ (not a /\ b /\ d)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not d))
        /\ (d' <=> (not a /\ c /\ d) \/ (not a /\ b /\ not c)
\/ (b /\ not c /\ d) \/ (a /\ not b /\ d))
then %implies
 axiom (b' \/ c' \/ not d')
end


[successfully proved with HETS]

thus F3 = (b \/ c \/ a)
        /\ (not a \/ c \/ not d)
        /\ (b \/ c \/ not d)
        /\ (b \/ c \/ a)
        /\ (not a \/ c \/ not d)
        /\ (b \/ c \/ not d)
```

```
    let c = (not d \/ b)
    check F2 /\ T => c'

spec Check22 =
  pred a,b,c,d,a',b',c',d': ()
  axiom  (b \/ c \/ a)
        /\ (not a \/ c \/ not d)
        /\ (b \/ c \/ not d)
        /\ (not d \/ b)
        /\ (not d \/ c)
  axiom      (a' <=> (a /\ not c) \/ (a /\ d) \/ (a /\ b)
\/ (b /\ c /\ d))
        /\ (b' <=> (not a /\ b) \/ (b /\ c)
\/ (not a /\ c /\ not d))
        /\ (c' <=> (not b /\ c /\ not d) \/ (not a /\ b /\ d)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not d))
        /\ (d' <=> (not a /\ c /\ d) \/ (not a /\ b /\ not c)
\/ (b /\ not c /\ d) \/ (a /\ not b /\ d))
then %implies
 axiom (not d' \/ b')
end


[successfully proved with HETS]

thus F3 = (b \/ c \/ a)
        /\ (not a \/ c \/ not d)
        /\ (b \/ c \/ not d)
        /\ (b \/ c \/ a)
        /\ (not a \/ c \/ not d)
        /\ (b \/ c \/ not d)
        /\ (not d \/ b)


    let c = (not d \/ c)
    check F2 /\ T => c'

spec Check23 =
  pred a,b,c,d,a',b',c',d': ()
  axiom  (b \/ c \/ a)
        /\ (not a \/ c \/ not d)
```

```
        /\ (b \/ c \/ not d)
        /\ (not d \/ b)
        /\ (not d \/ c)
  axiom       (a' <=> (a /\ not c) \/ (a /\ d) \/ (a /\ b)
\/ (b /\ c /\ d))
        /\ (b' <=> (not a /\ b) \/ (b /\ c)
\/ (not a /\ c /\ not d))
        /\ (c' <=> (not b /\ c /\ not d) \/ (not a /\ b /\ d)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not d))
        /\ (d' <=> (not a /\ c /\ d) \/ (not a /\ b /\ not c)
\/ (b /\ not c /\ d) \/ (a /\ not b /\ d))
then %implies
 axiom (not d' \/ c')
end
```

[disproved with HETS]

```
c) thus F3 = (b \/ c \/ a)
        /\ (not a \/ c \/ not d)
        /\ (b \/ c \/ not d)
        /\ (b \/ c \/ a)
        /\ (not a \/ c \/ not d)
        /\ (b \/ c \/ not d)
        /\ (not d \/ b)

        = (b \/ c \/ a)
        /\ (not a \/ c \/ not d)
        /\ (b \/ c \/ not d)
        /\ (not d \/ b)
        != F2

i.e. F3 = F2.
```

Part IV: Repeat part 3 until Fk =Fk+1:

```
a) Check: Is it the case that Fk /\ T => P'?
b) If it is, then
 b-1) set Fk+1 := P
```

```
 b-2) for every clause c in Fk , check if Fk/\T=>c'?
      If it is, set Fk+1 := Fk+1 /\ c
c) If Fk =Fk+1: done


k=1:
++++

a) do we have F2 /\ T => P' ?

spec Check24 =
  pred a,b,c,d,a',b',c',d': ()
  axiom      (b \/ c \/ a)
        /\ (not a \/ c \/ not d)
        /\ (b \/ c \/ not d)
        /\ (not d \/ b)
  axiom     (a' <=> (a /\ not c) \/ (a /\ d) \/ (a /\ b)
\/ (b /\ c /\ d))
        /\ (b' <=> (not a /\ b) \/ (b /\ c)
\/ (not a /\ c /\ not d))
        /\ (c' <=> (not b /\ c /\ not d) \/ (not a /\ b /\ d)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not d))
        /\ (d' <=> (not a /\ c /\ d) \/ (not a /\ b /\ not c)
\/ (b /\ not c /\ d) \/ (a /\ not b /\ d))
then %implies
 axiom  (b' \/ c' \/ a') /\ (not a' \/ c' \/ not d')
/\ (b' \/ c' \/ not d')
end

[proved with Hets] - "YOU LOVE TO SEE IT!!!"

b-1) F4 = (b \/ c \/ a) /\ (not a \/ c \/ not d)
/\ (b \/ c \/ not d)

b-2) clauses(F3) = { (b \/ c \/ a),
        (not a \/ c \/ not d),
        (b \/ c \/ not d),
        (not d \/ b)}

    let c = (b \/ c \/ a)
    check F3 /\ T => c'
```

```
spec Check25 =
  pred a,b,c,d,a',b',c',d': ()
  axiom  (b \/ c \/ a)
        /\ (not a \/ c \/ not d)
        /\ (b \/ c \/ not d)
        /\ (not d \/ b)
  axiom       (a' <=> (a /\ not c) \/ (a /\ d) \/ (a /\ b)
\/ (b /\ c /\ d))
        /\ (b' <=> (not a /\ b) \/ (b /\ c)
\/ (not a /\ c /\ not d))
        /\ (c' <=> (not b /\ c /\ not d) \/ (not a /\ b /\ d)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not d))
        /\ (d' <=> (not a /\ c /\ d) \/ (not a /\ b /\ not c)
\/ (b /\ not c /\ d) \/ (a /\ not b /\ d))
then %implies
 axiom (b' \/ c' \/ a')
end

[successfully proved with HETS]

thus F4 = (b \/ c \/ a)
        /\ (not a \/ c \/ not d)
        /\ (b \/ c \/ not d)
        /\ (b \/ c \/ a)


    let c = (not a \/ c \/ not d)
    check F3 /\ T => c'

spec Check26 =
  pred a,b,c,d,a',b',c',d': ()
  axiom   (b \/ c \/ a)
        /\ (not a \/ c \/ not d)
        /\ (b \/ c \/ not d)
        /\ (not d \/ b)
  axiom       (a' <=> (a /\ not c) \/ (a /\ d) \/ (a /\ b)
\/ (b /\ c /\ d))
        /\ (b' <=> (not a /\ b) \/ (b /\ c)
\/ (not a /\ c /\ not d))
        /\ (c' <=> (not b /\ c /\ not d) \/ (not a /\ b /\ d)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not d))
        /\ (d' <=> (not a /\ c /\ d) \/ (not a /\ b /\ not c)
```

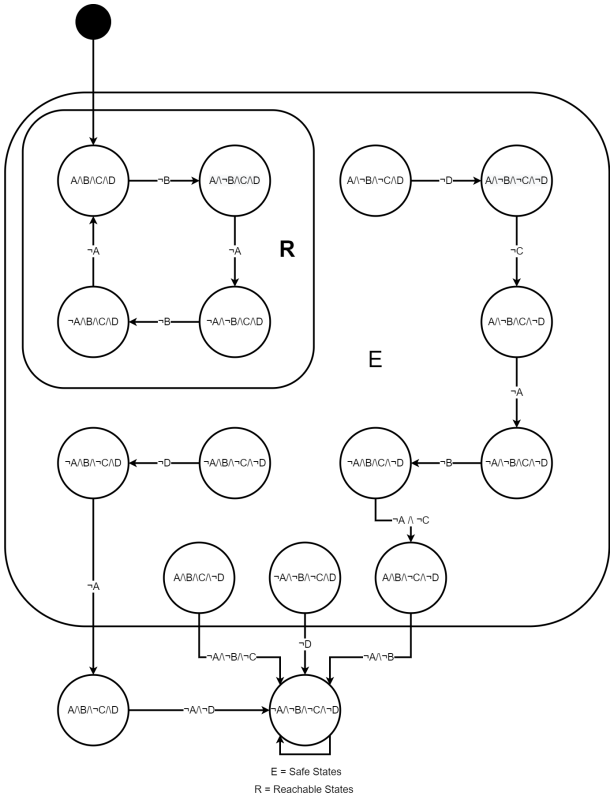
```
\/ (b /\ not c /\ d) \/ (a /\ not b /\ d))
then %implies
 axiom (not a' \/ c' \/ not d')
end


[successfully proved with HETS]

thus F4 = (b \/ c \/ a)
        /\ (not a \/ c \/ not d)
        /\ (b \/ c \/ not d)
        /\ (b \/ c \/ a)
        /\ (not a \/ c \/ not d)


    let c = (b \/ c \/ not d)
    check F3 /\ T => c'

spec Check27 =
  pred a,b,c,d,a',b',c',d': ()
  axiom  (b \/ c \/ a)
        /\ (not a \/ c \/ not d)
        /\ (b \/ c \/ not d)
        /\ (not d \/ b)
  axiom       (a' <=> (a /\ not c) \/ (a /\ d) \/ (a /\ b)
\/ (b /\ c /\ d))
        /\ (b' <=> (not a /\ b) \/ (b /\ c)
\/ (not a /\ c /\ not d))
        /\ (c' <=> (not b /\ c /\ not d) \/ (not a /\ b /\ d)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not d))
        /\ (d' <=> (not a /\ c /\ d) \/ (not a /\ b /\ not c)
\/ (b /\ not c /\ d) \/ (a /\ not b /\ d))
then %implies
 axiom (b' \/ c' \/ not d')
end


[successfully proved with HETS]

thus F4 = (b \/ c \/ a)
        /\ (not a \/ c \/ not d)
        /\ (b \/ c \/ not d)
```

```
        /\ (b \/ c \/ a)
        /\ (not a \/ c \/ not d)
        /\ (b \/ c \/ not d)


    let c = (not d \/ b)
    check F2 /\ T => c'

spec Check28 =
  pred a,b,c,d,a',b',c',d': ()
  axiom  (b \/ c \/ a)
        /\ (not a \/ c \/ not d)
        /\ (b \/ c \/ not d)
        /\ (not d \/ b)
  axiom      (a' <=> (a /\ not c) \/ (a /\ d) \/ (a /\ b)
\/ (b /\ c /\ d))
        /\ (b' <=> (not a /\ b) \/ (b /\ c)
\/ (not a /\ c /\ not d))
        /\ (c' <=> (not b /\ c /\ not d) \/ (not a /\ b /\ d)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not d))
        /\ (d' <=> (not a /\ c /\ d) \/ (not a /\ b /\ not c)
\/ (b /\ not c /\ d) \/ (a /\ not b /\ d))
then %implies
 axiom (not d' \/ b')
end


[successfully proved with HETS]

thus F4 = (b \/ c \/ a)
        /\ (not a \/ c \/ not d)
        /\ (b \/ c \/ not d)
        /\ (b \/ c \/ a)
        /\ (not a \/ c \/ not d)
        /\ (b \/ c \/ not d)
        /\ (not d \/ b)


c) thus F4 = (b \/ c \/ a)
        /\ (not a \/ c \/ not d)
        /\ (b \/ c \/ not d)
        /\ (b \/ c \/ a)
```

```
/\ (not a \/ c \/ not d)
/\ (b \/ c \/ not d)
/\ (not d \/ b)

= (b \/ c \/ a)
/\ (not a \/ c \/ not d)
/\ (b \/ c \/ not d)
/\ (not d \/ b)
= F3
```

The algorithms terminates and we have proven that P holds.

B.1.6 Third Four Variable Ladder Logic Example



library *temp35507101*

spec TRANSITION =
preds a : ();
b : ();

$c : ()$;
$d : ()$

preds $a' : ()$;
$b' : ()$;
$c' : ()$;
$d' : ()$

- a'
  $\Leftrightarrow (\neg\ a \wedge b \wedge d) \vee (\neg\ a \wedge b \wedge c) \vee (b \wedge c \wedge d)$
  $\vee\ (a \wedge \neg\ b \wedge \neg\ c)$
- $b' \Leftrightarrow (\neg\ a \wedge c) \vee (\neg\ a \wedge b)$
- $c' \Leftrightarrow (\neg\ b \wedge c) \vee (c \wedge d) \vee (a \wedge \neg\ b \wedge \neg\ d)$
- $d' \Leftrightarrow (c \wedge d) \vee (\neg\ a \wedge b \wedge \neg\ c)$

end

spec TransitionTest =
TRANSITION
then %implies

- $\neg\ a \wedge \neg\ b \wedge \neg\ c \wedge \neg\ d \Rightarrow \neg\ a' \wedge \neg\ b' \wedge \neg\ c' \wedge \neg\ d'$ %(t1)%
- $\neg\ a \wedge \neg\ b \wedge \neg\ c \wedge d \Rightarrow \neg\ a' \wedge \neg\ b' \wedge \neg\ c' \wedge \neg\ d'$ %(t2)%
- $\neg\ a \wedge \neg\ b \wedge c \wedge \neg\ d \Rightarrow \neg\ a' \wedge b' \wedge c' \wedge \neg\ d'$ %(t3)%
- $\neg\ a \wedge \neg\ b \wedge c \wedge d \Rightarrow \neg\ a' \wedge b' \wedge c' \wedge d'$ %(t4)%
- $\neg\ a \wedge b \wedge \neg\ c \wedge \neg\ d \Rightarrow \neg\ a' \wedge b' \wedge \neg\ c' \wedge d'$ %(t5)%
- $\neg\ a \wedge b \wedge \neg\ c \wedge d \Rightarrow a' \wedge b' \wedge \neg\ c' \wedge d'$ %(t6)%
- $\neg\ a \wedge b \wedge c \wedge \neg\ d \Rightarrow a' \wedge b' \wedge \neg\ c' \wedge \neg\ d'$ %(t7)%
- $\neg\ a \wedge b \wedge c \wedge d \Rightarrow a' \wedge b' \wedge c' \wedge d'$ %(t8)%
- $a \wedge \neg\ b \wedge \neg\ c \wedge \neg\ d \Rightarrow a' \wedge \neg\ b' \wedge c' \wedge \neg\ d'$ %(t9)%
- $a \wedge \neg\ b \wedge \neg\ c \wedge d \Rightarrow a' \wedge \neg\ b' \wedge \neg\ c' \wedge \neg\ d'$ %(t10)%
- $a \wedge \neg\ b \wedge c \wedge \neg\ d \Rightarrow \neg\ a' \wedge \neg\ b' \wedge c' \wedge \neg\ d'$ %(t11)%
- $a \wedge \neg\ b \wedge c \wedge d \Rightarrow \neg\ a' \wedge \neg\ b' \wedge c' \wedge d'$ %(t12)%
- $a \wedge b \wedge \neg\ c \wedge \neg\ d \Rightarrow \neg\ a' \wedge \neg\ b' \wedge \neg\ c' \wedge \neg\ d'$ %(t13)%
- $a \wedge b \wedge \neg\ c \wedge d \Rightarrow \neg\ a' \wedge \neg\ b' \wedge \neg\ c' \wedge \neg\ d'$ %(t14)%
- $a \wedge b \wedge c \wedge \neg\ d \Rightarrow \neg\ a' \wedge \neg\ b' \wedge \neg\ c' \wedge \neg\ d'$ %(t15)%
- $a \wedge b \wedge c \wedge d \Rightarrow a' \wedge \neg\ b' \wedge c' \wedge d'$ %(t16)%

end

spec SafeInitial =
TRANSITION
then $\bullet\ a \wedge b \wedge c \wedge d$
then %implies

- $c \vee (\neg\ a \wedge d) \vee (b \wedge \neg\ d) \vee (a \wedge \neg\ b)$ %(initial_safe)%

end

spec StepSafe =
Transition
then %implies
• $c \vee (\neg\ a \wedge d) \vee (b \wedge \neg\ d) \vee (a \wedge \neg\ b)$
$\Rightarrow c' \vee (\neg\ a' \wedge d') \vee (b' \wedge \neg\ d') \vee (a' \wedge \neg\ b')$
%(step_safety)%
end

IC3 run through:

CNF converter:

https://www.erpelstolz.at/gateway/formular-uk-zentral.html

[note that I changed the CNF converter, when I used the previous one

http://formal.cs.utah.edu:8080/pbl/PBL.php,

I could not establish the result that T and CNF(T) are equivalent.]

Hets:

http://rest.hets.eu

Part I: the formulae how we read them off the automaton and their CNFs

Initialisation Formula:

I = (a /\ b /\ c /\ d)
in CNF:

CNF_CASL(I) = (a /\ b /\ c /\ d)

Transition Formula:

```
  T =   (a' <=> ((not a /\ b /\ d) \/ (not a /\ b /\ c)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not c)))
        /\ (b' <=> ((not a /\ c) \/ (not a /\ b)))
        /\ (c' <=> ((not b /\ c) \/ (c /\ d)
\/ (a /\ not b /\ not d)))
        /\ (d' <=> ((c /\ d) \/ (not a /\ b /\ not c)))

  CNF_CASL(T) = (a \/ not b \/ not d \/ a')
        /\ (a \/ not b \/ not c \/ a')
        /\ (not b \/ not c \/ not d \/ a')
        /\ (not a \/ b \/ c \/ a')
        /\ (b \/ a \/ not a')
        /\ (c \/ b \/ a \/ not a')
        /\ (b \/ d \/ a \/ not a')
        /\ (c \/ d \/ b \/ a \/ not a')
        /\ (b \/ d \/ c \/ a \/ not a')
        /\ (c \/ d \/ a \/ not a')
        /\ (c \/ b \/ d \/ a \/ not a')
        /\ (not a \/ c \/ not b \/ not a')
        /\ (not a \/ d \/ c \/ not b \/ not a')
        /\ (c \/ d \/ not b \/ not a')
        /\ (not a \/ d \/ not b \/ not a')
        /\ (c \/ not a \/ d \/ not b \/ not a')
        /\ (not a \/ b \/ not c \/ not a')
        /\ (b \/ not c \/ not a')
        /\ (not a \/ d \/ b \/ not c \/ not a')
        /\ (b \/ d \/ not c \/ not a')
        /\ (not a \/ d \/ not c \/ not a')
        /\ (b \/ not a \/ d \/ not c \/ not a')
        /\ (a \/ not c \/ b')
        /\ (a \/ not b \/ b')
        /\ (not a \/ not b')
        /\ (b \/ not a \/ not b')
        /\ (not a \/ c \/ not b')
        /\ (b \/ c \/ not b')
        /\ (b \/ not c \/ c')
        /\ (not c \/ not d \/ c')
        /\ (not a \/ b \/ d \/ c')
        /\ (c \/ not b \/ a \/ not c')
        /\ (d \/ not b \/ a \/ not c')
        /\ (c \/ a \/ not c')
```

```
        /\ (d \/ c \/ a \/ not c')
        /\ (c \/ not b \/ not c')
        /\ (d \/ not b \/ not c')
        /\ (d \/ c \/ not b \/ not c')
        /\ (c \/ not b \/ not d \/ not c')
        /\ (c \/ not d \/ not c')
        /\ (not c \/ not d \/ d')
        /\ (a \/ not b \/ c \/ d')
        /\ (not a \/ c \/ not d')
        /\ (b \/ c \/ not d')
        /\ (not a \/ d \/ not d')
        /\ (b \/ d \/ not d')
        /\ (not c \/ d \/ not d')
```

Safety property:

```
  P  = (c \/ (not a /\ d) \/ (b /\ not d) \/ (a /\ not b))
  P' = (c' \/ (not a' /\ d') \/ (b' /\ not d') \/ (a' /\ not b'))

  CNF_CASL(P)  = (d \/ c \/ b \/ a)
/\ (not a \/ c \/ not d \/ not b)
  CNF_CASL(P') = (d' \/ c' \/ b' \/ a')
/\ (not a' \/ c' \/ not d' \/ not b')
```

Checking with Hets that T and CNF_CASL(T) are equivalent:

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
spec Check1 =
  pred a,b,c,d,a',b',c',d': ()
then %implies
 axiom
(
            (a' <=> ((not a /\ b /\ d) \/ (not a /\ b /\ c)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not c)))
        /\ (b' <=> ((not a /\ c) \/ (not a /\ b)))
        /\ (c' <=> ((not b /\ c) \/ (c /\ d)
\/ (a /\ not b /\ not d)))
        /\ (d' <=> ((c /\ d) \/ (not a /\ b /\ not c)))
)
  <=>    (
        (a \/ not b \/ not d \/ a')
```

```
/\ (a \/ not b \/ not c \/ a')
/\ (not b \/ not c \/ not d \/ a')
/\ (not a \/ b \/ c \/ a')
/\ (b \/ a \/ not a')
/\ (c \/ b \/ a \/ not a')
/\ (b \/ d \/ a \/ not a')
/\ (c \/ d \/ b \/ a \/ not a')
/\ (b \/ d \/ c \/ a \/ not a')
/\ (c \/ d \/ a \/ not a')
/\ (c \/ b \/ d \/ a \/ not a')
/\ (not a \/ c \/ not b \/ not a')
/\ (not a \/ d \/ c \/ not b \/ not a')
/\ (c \/ d \/ not b \/ not a')
/\ (not a \/ d \/ not b \/ not a')
/\ (c \/ not a \/ d \/ not b \/ not a')
/\ (not a \/ b \/ not c \/ not a')
/\ (b \/ not c \/ not a')
/\ (not a \/ d \/ b \/ not c \/ not a')
/\ (b \/ d \/ not c \/ not a')
/\ (not a \/ d \/ not c \/ not a')
/\ (b \/ not a \/ d \/ not c \/ not a')
/\ (a \/ not c \/ b')
/\ (a \/ not b \/ b')
/\ (not a \/ not b')
/\ (b \/ not a \/ not b')
/\ (not a \/ c \/ not b')
/\ (b \/ c \/ not b')
/\ (b \/ not c \/ c')
/\ (not c \/ not d \/ c')
/\ (not a \/ b \/ d \/ c')
/\ (c \/ not b \/ a \/ not c')
/\ (d \/ not b \/ a \/ not c')
/\ (c \/ a \/ not c')
/\ (d \/ c \/ a \/ not c')
/\ (c \/ not b \/ not c')
/\ (d \/ not b \/ not c')
/\ (d \/ c \/ not b \/ not c')
/\ (c \/ not b \/ not d \/ not c')
/\ (c \/ not d \/ not c')
/\ (not c \/ not d \/ d')
/\ (a \/ not b \/ c \/ d')
/\ (not a \/ c \/ not d')
```

```
        /\ (b \/ c \/ not d')
        /\ (not a \/ d \/ not d')
        /\ (b \/ d \/ not d')
        /\ (not c \/ d \/ not d')
         )
end

[successfully proved with HETS interface]
```

Part II: Page 11 from the slides:

---

once more the formulae (now all in CNF):

```
  I =  (a /\ b /\ c /\ d)

  T =   (
        (a' <=> ((not a /\ b /\ d) \/ (not a /\ b /\ c)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not c)))
        /\ (b' <=> ((not a /\ c) \/ (not a /\ b)))
        /\ (c' <=> ((not b /\ c) \/ (c /\ d)
\/ (a /\ not b /\ not d)))
        /\ (d' <=> ((c /\ d) \/ (not a /\ b /\ not c)))
        )

  P = (d \/ c \/ b \/ a) /\ (not a \/ c \/ not d \/ not b)

  P' = (d' \/ c' \/ b' \/ a')
/\ (not a' \/ c' \/ not d' \/ not b')
```

a) Check I =>P?

```
spec Check2 =
  pred a,b,c,d: ()
  axiom  (a /\ b /\ c /\ d)
then %implies
 axiom (d \/ c \/ b \/ a) /\ (not a \/ c \/ not d \/ not b)
end

[successfully proved with HETS interface]
```

```
b) Check I /\T =>P‘?

spec Check3 =
  pred a,b,c,d,a',b',c',d': ()
  axiom (a /\ b /\ c /\ d)
  axiom      (
         (a' <=> ((not a /\ b /\ d) \/ (not a /\ b /\ c)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not c)))
         /\ (b' <=> ((not a /\ c) \/ (not a /\ b)))
         /\ (c' <=> ((not b /\ c) \/ (c /\ d)
\/ (a /\ not b /\ not d)))
         /\ (d' <=> ((c /\ d) \/ (not a /\ b /\ not c)))
         )
then %implies
 axiom (d' \/ c' \/ b' \/ a')
/\ (not a' \/ c' \/ not d' \/ not b')
end


c) Set F1 :=P

  F1 = (a /\ b /\ c /\ d)

d) For every clause c in clauses(I), if c not-in clauses(F1),
check:
    I  /\ T  => c‘?
    If it does, set F1 := F1  /\ c


clauses(I) = {a,
         b,
         c,
         d}

clauses(F1) = {(d \/ c \/ b \/ a),
         (not a \/ c \/ not d \/ not b)}

d-1) c = a

check if I /\ T => a':
```

```
spec Check4 =
  pred a,b,c,d,a',b',c',d': ()
  axiom (a /\ b /\ c /\ d)
  axiom      (
        (a' <=> ((not a /\ b /\ d) \/ (not a /\ b /\ c)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not c)))
        /\ (b' <=> ((not a /\ c) \/ (not a /\ b)))
        /\ (c' <=> ((not b /\ c) \/ (c /\ d)
\/ (a /\ not b /\ not d)))
        /\ (d' <=> ((c /\ d) \/ (not a /\ b /\ not c)))
        )
then %implies
 axiom a'
end

[proved with Hets]

Thus, we add a to F1, and obtain:

   F1 = (d \/ c \/ b \/ a) /\ (not a \/ c \/ not d \/ not b)
        /\ a


d-2) c = b

check if I /\ T => b':

spec Check5 =
  pred a,b,c,d,a',b',c',d': ()
  axiom (a /\ b /\ c /\ d)
  axiom      (
        (a' <=> ((not a /\ b /\ d) \/ (not a /\ b /\ c)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not c)))
        /\ (b' <=> ((not a /\ c) \/ (not a /\ b)))
        /\ (c' <=> ((not b /\ c) \/ (c /\ d)
\/ (a /\ not b /\ not d)))
        /\ (d' <=> ((c /\ d) \/ (not a /\ b /\ not c)))
        )
then %implies
 axiom b'
end
```

```
[disproved with Hets]

Thus, we do not add not b to F1


d-3) c = c

check if I /\ T => c':

spec Check6 =
  pred a,b,c,d,a',b',c',d': ()
  axiom (a /\ b /\ c /\ d)
  axiom      (
        (a' <=> ((not a /\ b /\ d) \/ (not a /\ b /\ c)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not c)))
        /\ (b' <=> ((not a /\ c) \/ (not a /\ b)))
        /\ (c' <=> ((not b /\ c) \/ (c /\ d)
\/ (a /\ not b /\ not d)))
        /\ (d' <=> ((c /\ d) \/ (not a /\ b /\ not c)))
        )
then %implies
 axiom c'
end

Thus, we add c to F1, and obtain:

   F1 = (d \/ c \/ b \/ a) /\ (not a \/ c \/ not d \/ not b)
        /\ a
        /\ c


d-d) c = d

check if I /\ T => d':

spec Check7 =
  pred a,b,c,d,a',b',c',d': ()
  axiom (a /\ b /\ c /\ d)
  axiom      (
        (a' <=> ((not a /\ b /\ d) \/ (not a /\ b /\ c)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not c)))
        /\ (b' <=> ((not a /\ c) \/ (not a /\ b)))
```

```
        /\ (c' <=> ((not b /\ c) \/ (c /\ d)
\/ (a /\ not b /\ not d)))
        /\ (d' <=> ((c /\ d) \/ (not a /\ b /\ not c)))
        )
then %implies
 axiom d'
end

[proved with Hets]

Thus, we add d to F1, and obtain:

   F1 = (d \/ c \/ b \/ a) /\ (not a \/ c \/ not d \/ not b)
        /\ a
        /\ c
        /\ d


Part III: Page 12 from the slides:
----------------------------------


a) Check: Is it the case that Fk  /\ T  => P'?
b) If it is, then
 b-1) set Fk+1 := P
 b-2) for every clause c in Fk , check if Fk /\T =>c'?
      If it is, set Fk+1 := Fk+1  /\ c
c) If Fk =Fk+1: done


k=1:
++++

a) do we have F1 /\ T => P' ?

spec Check8 =
  pred a,b,c,d,a',b',c',d': ()
  axiom (d \/ c \/ b \/ a) /\ (not a \/ c \/ not d \/ not b)
        /\ a
        /\ c
        /\ d
  axiom    (
        (a' <=> ((not a /\ b /\ d) \/ (not a /\ b /\ c)
```

```
\/ (b /\ c /\ d) \/ (a /\ not b /\ not c)))
        /\ (b' <=> ((not a /\ c) \/ (not a /\ b)))
        /\ (c' <=> ((not b /\ c) \/ (c /\ d)
\/ (a /\ not b /\ not d)))
        /\ (d' <=> ((c /\ d) \/ (not a /\ b /\ not c)))
        )
then %implies
 axiom  (d' \/ c' \/ b' \/ a')
/\ (not a' \/ c' \/ not d' \/ not b')
end

[proved with Hets]


b-1) F2 = (d \/ c \/ b \/ a) /\ (not a \/ c \/ not d \/ not b)

b-2) clauses(F1) = {(d \/ c \/ b \/ a),
        (not a \/ c \/ not d \/ not b),
        a,
        c,
        d}


    let c = (d \/ c \/ b \/ a)
    check F1 /\ T => c'

spec Check9 =
  pred a,b,c,d,a',b',c',d': ()
  axiom (d \/ c \/ b \/ a) /\ (not a \/ c \/ not d \/ not b)
        /\ a
        /\ c
        /\ d
  axiom      (
        (a' <=> ((not a /\ b /\ d) \/ (not a /\ b /\ c)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not c)))
        /\ (b' <=> ((not a /\ c) \/ (not a /\ b)))
        /\ (c' <=> ((not b /\ c) \/ (c /\ d)
\/ (a /\ not b /\ not d)))
        /\ (d' <=> ((c /\ d) \/ (not a /\ b /\ not c)))
        )
then %implies
 axiom (d' \/ c' \/ b' \/ a')
```

```
end

[successfully proved with HETS]

thus F2 = (d \/ c \/ b \/ a) /\ (not a \/ c \/ not d \/ not b)
        /\ (d \/ c \/ b \/ a)


    let c = (not a \/ c \/ not d \/ not b)
    check F1 /\ T => c'

spec Check10 =
  pred a,b,c,d,a',b',c',d': ()
  axiom (d \/ c \/ b \/ a) /\ (not a \/ c \/ not d \/ not b)
        /\ a
        /\ c
        /\ d
  axiom       (
        (a' <=> ((not a /\ b /\ d) \/ (not a /\ b /\ c)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not c)))
        /\ (b' <=> ((not a /\ c) \/ (not a /\ b)))
        /\ (c' <=> ((not b /\ c) \/ (c /\ d)
\/ (a /\ not b /\ not d)))
        /\ (d' <=> ((c /\ d) \/ (not a /\ b /\ not c)))
        )
then %implies
 axiom (not a' \/ c' \/ not d' \/ not b')
end

[successfully proved with HETS]

thus F2 = (d \/ c \/ b \/ a) /\ (not a \/ c \/ not d \/ not b)
        /\ (d \/ c \/ b \/ a)
        /\ (not a \/ c \/ not d \/ not b)


    let c = a
    check F1 /\ T => c'

spec Check11 =
  pred a,b,c,d,a',b',c',d': ()
  axiom (d \/ c \/ b \/ a) /\ (not a \/ c \/ not d \/ not b)
```

```
        /\ a
        /\ c
        /\ d
  axiom      (
        (a' <=> ((not a /\ b /\ d) \/ (not a /\ b /\ c)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not c)))
        /\ (b' <=> ((not a /\ c) \/ (not a /\ b)))
        /\ (c' <=> ((not b /\ c) \/ (c /\ d)
\/ (a /\ not b /\ not d)))
        /\ (d' <=> ((c /\ d) \/ (not a /\ b /\ not c)))
        )
then %implies
 axiom a'
end

[disproved with HETS]

Thus, we do not add a to F2


    let c = c
    check F1 /\ T => c'

spec Check12 =
  pred a,b,c,d,a',b',c',d': ()
  axiom (d \/ c \/ b \/ a) /\ (not a \/ c \/ not d \/ not b)
        /\ a
        /\ c
        /\ d
  axiom      (
        (a' <=> ((not a /\ b /\ d) \/ (not a /\ b /\ c)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not c)))
        /\ (b' <=> ((not a /\ c) \/ (not a /\ b)))
        /\ (c' <=> ((not b /\ c) \/ (c /\ d)
\/ (a /\ not b /\ not d)))
        /\ (d' <=> ((c /\ d) \/ (not a /\ b /\ not c)))
        )
then %implies
 axiom c'
end

[successfully proved with HETS]
```

```
thus F2 = (d \/ c \/ b \/ a) /\ (not a \/ c \/ not d \/ not b)
        /\ (d \/ c \/ b \/ a)
        /\ (not a \/ c \/ not d \/ not b)
        /\ c


    let c = d
    check F1 /\ T => c'

spec Check13 =
  pred a,b,c,d,a',b',c',d': ()
  axiom (d \/ c \/ b \/ a) /\ (not a \/ c \/ not d \/ not b)
        /\ a
        /\ c
        /\ d
  axiom      (
        (a' <=> ((not a /\ b /\ d) \/ (not a /\ b /\ c)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not c)))
        /\ (b' <=> ((not a /\ c) \/ (not a /\ b)))
        /\ (c' <=> ((not b /\ c) \/ (c /\ d)
\/ (a /\ not b /\ not d)))
        /\ (d' <=> ((c /\ d) \/ (not a /\ b /\ not c)))
        )
then %implies
 axiom d'
end

[successfully proved with HETS]

thus F2 = (d \/ c \/ b \/ a) /\ (not a \/ c \/ not d \/ not b)
        /\ (d \/ c \/ b \/ a)
        /\ (not a \/ c \/ not d \/ not b)
        /\ c
        /\ d


c) thus F2 = (d \/ c \/ b \/ a) /\ (not a \/ c \/ not d \/ not b)
        /\ c
        /\ d
        != F1
```

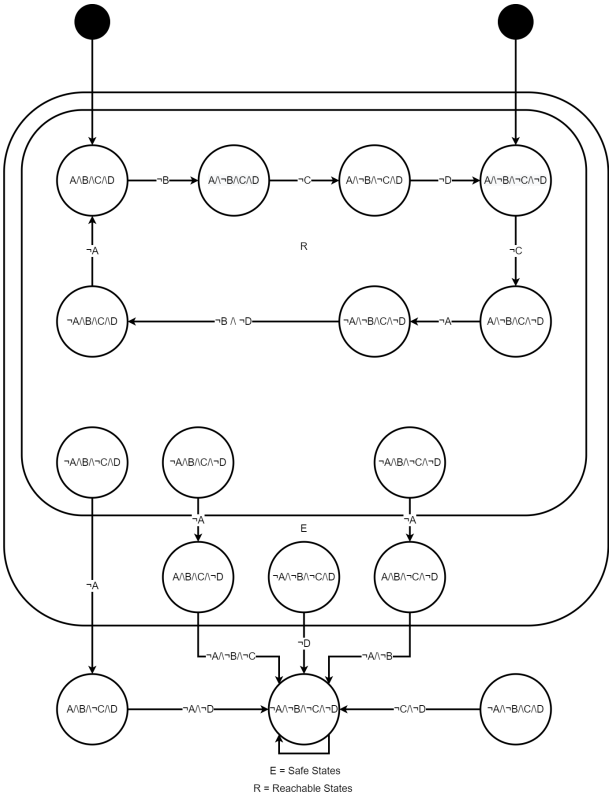
```
i.e. F2 = F1


Part IV: Repeat part 3 until Fk =Fk+1:
--------------------------------------


a) Check: Is it the case that Fk  /\ T  => P'?
b) If it is, then
 b-1) set Fk+1 := P
 b-2) for every clause c in Fk , check if Fk /\T =>c'?
      If it is, set Fk+1 := Fk+1  /\ c
c) If Fk =Fk+1: done


k=2:
++++

a) do we have F2 /\ T => P' ?

spec Check14 =
  pred a,b,c,d,a',b',c',d': ()
  axiom (d \/ c \/ b \/ a) /\ (not a \/ c \/ not d \/ not b)
        /\ c
        /\ d
  axiom    (
        (a' <=> ((not a /\ b /\ d) \/ (not a /\ b /\ c)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not c)))
        /\ (b' <=> ((not a /\ c) \/ (not a /\ b)))
        /\ (c' <=> ((not b /\ c) \/ (c /\ d)
\/ (a /\ not b /\ not d)))
        /\ (d' <=> ((c /\ d) \/ (not a /\ b /\ not c)))
        )
then %implies
 axiom  (d' \/ c' \/ b' \/ a')
/\ (not a' \/ c' \/ not d' \/ not b')
end


[proved with Hets]


b-1) F3 = (d \/ c \/ b \/ a) /\ (not a \/ c \/ not d \/ not b)
```

```
b-2) clauses(F2) = {(d \/ c \/ b \/ a),
        (not a \/ c \/ not d \/ not b),
        c,
        d}


    let c = (d \/ c \/ b \/ a)
    check F2 /\ T => c'

spec Check15 =
  pred a,b,c,d,a',b',c',d': ()
  axiom (d \/ c \/ b \/ a) /\ (not a \/ c \/ not d \/ not b)
        /\ c
        /\ d
  axiom      (
        (a' <=> ((not a /\ b /\ d) \/ (not a /\ b /\ c)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not c)))
        /\ (b' <=> ((not a /\ c) \/ (not a /\ b)))
        /\ (c' <=> ((not b /\ c) \/ (c /\ d)
\/ (a /\ not b /\ not d)))
        /\ (d' <=> ((c /\ d) \/ (not a /\ b /\ not c)))
        )
then %implies
 axiom (d' \/ c' \/ b' \/ a')
end

[successfully proved with HETS]

thus F3 = (d \/ c \/ b \/ a) /\ (not a \/ c \/ not d \/ not b)
        /\ (d \/ c \/ b \/ a)


    let c = (not a \/ c \/ not d \/ not b)
    check F2 /\ T => c'

spec Check16 =
  pred a,b,c,d,a',b',c',d': ()
  axiom (d \/ c \/ b \/ a) /\ (not a \/ c \/ not d \/ not b)
        /\ c
        /\ d
  axiom      (
```

```
        (a' <=> ((not a /\ b /\ d) \/ (not a /\ b /\ c)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not c)))
        /\ (b' <=> ((not a /\ c) \/ (not a /\ b)))
        /\ (c' <=> ((not b /\ c) \/ (c /\ d)
\/ (a /\ not b /\ not d)))
        /\ (d' <=> ((c /\ d) \/ (not a /\ b /\ not c)))
        )
then %implies
 axiom (not a' \/ c' \/ not d' \/ not b')
end

[successfully proved with HETS]

thus F3 = (d \/ c \/ b \/ a) /\ (not a \/ c \/ not d \/ not b)
        /\ (d \/ c \/ b \/ a)
        /\ (not a \/ c \/ not d \/ not b)


    let c = c
    check F2 /\ T => c'

spec Check17 =
  pred a,b,c,d,a',b',c',d': ()
  axiom (d \/ c \/ b \/ a) /\ (not a \/ c \/ not d \/ not b)
        /\ c
        /\ d
  axiom       (
        (a' <=> ((not a /\ b /\ d) \/ (not a /\ b /\ c)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not c)))
        /\ (b' <=> ((not a /\ c) \/ (not a /\ b)))
        /\ (c' <=> ((not b /\ c) \/ (c /\ d)
\/ (a /\ not b /\ not d)))
        /\ (d' <=> ((c /\ d) \/ (not a /\ b /\ not c)))
        )
then %implies
 axiom c'
end

[successfully proved with HETS]

thus F3 = (d \/ c \/ b \/ a) /\ (not a \/ c \/ not d \/ not b)
        /\ (d \/ c \/ b \/ a)
```

```
        /\ (not a \/ c \/ not d \/ not b)
        /\ a


    let c = d
    check F2 /\ T => c'

spec Check18 =
  pred a,b,c,d,a',b',c',d': ()
  axiom (d \/ c \/ b \/ a) /\ (not a \/ c \/ not d \/ not b)
        /\ c
        /\ d
  axiom      (
        (a' <=> ((not a /\ b /\ d) \/ (not a /\ b /\ c)
\/ (b /\ c /\ d) \/ (a /\ not b /\ not c)))
        /\ (b' <=> ((not a /\ c) \/ (not a /\ b)))
        /\ (c' <=> ((not b /\ c) \/ (c /\ d)
\/ (a /\ not b /\ not d)))
        /\ (d' <=> ((c /\ d) \/ (not a /\ b /\ not c)))
        )
then %implies
 axiom d'
end

[successfully proved with HETS]

thus F3 = (d \/ c \/ b \/ a) /\ (not a \/ c \/ not d \/ not b)
        /\ (d \/ c \/ b \/ a)
        /\ (not a \/ c \/ not d \/ not b)
        /\ a
        /\ b
        = F2

Therefore verification holds
```

B.1.7 Fourth Four Variable Ladder Logic Example



spec TRANSITION =
preds a : ();
b : ();
c : ();
d : ()
preds a' : ();
b' : ();
c' : ();
d' : ()
• $a' \Leftrightarrow (\neg\ a \wedge b) \vee (a \wedge c \wedge d) \vee (a \wedge \neg\ b \wedge \neg\ c)$
• $b' \Leftrightarrow (\neg\ a \wedge b) \vee (\neg\ a \wedge c \wedge \neg\ d)$
• $c' \Leftrightarrow (\neg\ a \wedge c \wedge \neg\ d) \vee (b \wedge c \wedge d) \vee (a \wedge \neg\ b \wedge \neg\ d)$
• $d' \Leftrightarrow (\neg\ a \wedge b \wedge d) \vee (a \wedge c \wedge d) \vee (\neg\ a \wedge \neg\ b \wedge c \wedge \neg\ d)$
end

spec TRANSITIONTEST =
TRANSITION
then %implies

• $\neg\ a \wedge \neg\ b \wedge \neg\ c \wedge \neg\ d \Rightarrow \neg\ a' \wedge \neg\ b' \wedge \neg\ c' \wedge \neg\ d'$ %(t1)%
• $\neg\ a \wedge \neg\ b \wedge \neg\ c \wedge d \Rightarrow \neg\ a' \wedge \neg\ b' \wedge \neg\ c' \wedge \neg\ d'$ %(t2)%
• $\neg\ a \wedge \neg\ b \wedge c \wedge \neg\ d \Rightarrow \neg\ a' \wedge b' \wedge c' \wedge d'$ %(t3)%
• $\neg\ a \wedge \neg\ b \wedge c \wedge d \Rightarrow \neg\ a' \wedge \neg\ b' \wedge \neg\ c' \wedge \neg\ d'$ %(t4)%
• $\neg\ a \wedge b \wedge \neg\ c \wedge \neg\ d \Rightarrow a' \wedge b' \wedge \neg\ c' \wedge \neg\ d'$ %(t5)%
• $\neg\ a \wedge b \wedge \neg\ c \wedge d \Rightarrow a' \wedge b' \wedge \neg\ c' \wedge d'$ %(t6)%
• $\neg\ a \wedge b \wedge c \wedge \neg\ d \Rightarrow a' \wedge b' \wedge c' \wedge \neg\ d'$ %(t7)%
• $\neg\ a \wedge b \wedge c \wedge d \Rightarrow a' \wedge b' \wedge c' \wedge d'$ %(t8)%
• $a \wedge \neg\ b \wedge \neg\ c \wedge \neg\ d \Rightarrow a' \wedge \neg\ b' \wedge c' \wedge \neg\ d'$ %(t9)%
• $a \wedge \neg\ b \wedge \neg\ c \wedge d \Rightarrow a' \wedge \neg\ b' \wedge \neg\ c' \wedge \neg\ d'$ %(t10)%
• $a \wedge \neg\ b \wedge c \wedge \neg\ d \Rightarrow \neg\ a' \wedge \neg\ b' \wedge c' \wedge \neg\ d'$ %(t11)%
• $a \wedge \neg\ b \wedge c \wedge d \Rightarrow a' \wedge \neg\ b' \wedge \neg\ c' \wedge d'$ %(t12)%
• $a \wedge b \wedge \neg\ c \wedge \neg\ d \Rightarrow \neg\ a' \wedge \neg\ b' \wedge \neg\ c' \wedge \neg\ d'$ %(t13)%
• $a \wedge b \wedge \neg\ c \wedge d \Rightarrow \neg\ a' \wedge \neg\ b' \wedge \neg\ c' \wedge \neg\ d'$ %(t14)%
• $a \wedge b \wedge c \wedge \neg\ d \Rightarrow \neg\ a' \wedge \neg\ b' \wedge \neg\ c' \wedge \neg\ d'$ %(t15)%
• $a \wedge b \wedge c \wedge d \Rightarrow a' \wedge \neg\ b' \wedge c' \wedge d'$ %(t16)%

end

spec SafeInitial =
Transition
then • $(a \wedge \neg\ b \wedge \neg\ c \wedge \neg\ d) \vee (a \wedge b \wedge c \wedge d)$
then %implies
• $(c \wedge \neg\ d) \vee (b \wedge \neg\ d) \vee (b \wedge c) \vee (a \wedge \neg\ b) \vee (\neg\ a \wedge \neg\ c \wedge d)$
%(initial_safe)%

end

spec StepSafe =
Transition
then %implies
• $(c \wedge \neg\ d) \vee (b \wedge \neg\ d) \vee (b \wedge c) \vee (a \wedge \neg\ b) \vee (\neg\ a \wedge \neg\ c \wedge d)$
$\Rightarrow (c' \wedge \neg\ d') \vee (b' \wedge \neg\ d') \vee (b' \wedge c') \vee (a' \wedge \neg\ b')$
$\vee (\neg\ a' \wedge \neg\ c' \wedge d')$
%(step_safety)%

end

IC3 run through:

CNF converter:

https://www.erpelstolz.at/gateway/formular-uk-zentral.html

[note that I changed the CNF converter, when I used the previous one

```
  http://formal.cs.utah.edu:8080/pbl/PBL.php,

I could not establish the result that T and CNF(T) are equivalent.]


Hets:
-----

   http://rest.hets.eu

==========================


Part I: the formulae how we read them off the automaton and
their CNFs
------------------------------------------------------------


Initialisation Formula:

  I =  ((a /\ not b /\ not c /\ not d) \/ (a /\ b /\ c /\ d))
in CNF:

  CNF_CASL(I) = (a /\ (c \/ not b) /\ (d \/ not b)
/\ (b \/ not c) /\ (d \/ not c) /\ (b \/ not d) /\ (c \/ not d))

Transition Formula:

  T =   (a' <=> ((not a /\ b) \/ (a /\ c /\ d)
\/ (a /\ not b /\ not c)))
        /\ (b' <=> ((not a /\ b) \/ (not a /\ c /\ not d)))
        /\ (c' <=> ((not a /\ c /\ not d) \/ (b /\ c /\ d)
\/ (a /\ not b /\ not d)))
        /\ (d' <=> ((not a /\ b /\ d) \/ (a /\ c /\ d)
\/ (not a /\ not b /\ c /\ not d)))

  CNF_CASL(T) = (a \/ not b \/ a') /\ (not a \/ not c \/ not d \/ a')
/\ (not a \/ b \/ c \/ a') /\ (a \/ b \/ not a')
/\ (c \/ b \/ a \/ not a') /\ (d \/ b \/ a \/ not a')
/\ (c \/ not a \/ not b \/ not a') /\ (d \/ not a \/ not b \/ not a')
/\ (d \/ not a \/ not c \/ not a') /\ (a \/ b \/ not c \/ not a')
```

```
/\ (d \/ b \/ not c \/ not a') /\ (a \/ not b \/ b')
/\ (a \/ not c \/ d \/ b') /\ (not a \/ not b')
/\ (c \/ not a \/ not b') /\ (not d \/ not a \/ not b')
/\ (not a \/ b \/ not b') /\ (c \/ b \/ not b')
/\ (not d \/ b \/ not b') /\ (a \/ not c \/ d \/ c')
/\ (not b \/ not c \/ not d \/ c') /\ (not a \/ b \/ d \/ c')
/\ (b \/ c \/ a \/ not c') /\ (c \/ a \/ not c')
/\ (d \/ c \/ a \/ not c') /\ (b \/ not d \/ a \/ not c')
/\ (c \/ not d \/ a \/ not c') /\ (c \/ not a \/ not b \/ not c')
/\ (d \/ not a \/ not b \/ not c') /\ (c \/ not b \/ not c')
/\ (d \/ c \/ not b \/ not c') /\ (c \/ not d \/ not b \/ not c')
/\ (b \/ not a \/ not d \/ not c') /\ (c \/ not a \/ not d \/ not c')
/\ (b \/ c \/ not d \/ not c') /\ (c \/ not d \/ not c')
/\ (b \/ not d \/ not c') /\ (a \/ not b \/ not d \/ d')
/\ (not a \/ not c \/ not d \/ d') /\ (a \/ b \/ not c \/ d \/ d')
/\ (c \/ not a \/ not d') /\ (d \/ not a \/ not d')
/\ (c \/ b \/ not a \/ not d') /\ (d \/ b \/ not a \/ not d')
/\ (c \/ d \/ not a \/ not d') /\ (c \/ not a \/ not b \/ not d')
/\ (d \/ not a \/ not b \/ not d') /\ (a \/ d \/ not b \/ not d')
/\ (c \/ d \/ not b \/ not d') /\ (d \/ not b \/ not d')
/\ (d \/ not a \/ c \/ not d') /\ (a \/ b \/ c \/ not d' )
/\ (c \/ b \/ not d') /\ (d \/ b \/ c \/ not d')
/\ (a \/ d \/ c \/ not d') /\ (c \/ d \/ not d')
/\ (c \/ not a \/ not d \/ not d') /\ (a \/ b \/ not d \/ not d')
/\ (c \/ b \/ not d \/ not d')
```

Safety property:

```
  P  = ((c /\ not d) \/ (b /\ not d) \/ (b /\ c)
\/ (a /\ not b) \/ (not a /\ not c /\ d))
  P' = ((c' /\ not d') \/ (b' /\ not d') \/ (b' /\ c')
\/ (a' /\ not b') \/ (not a' /\ not c' /\ d'))

  CNF_CASL(P)  = (not d \/ c \/ not b \/ not a)
/\ (b \/ not d \/ a \/ not c) /\ (b \/ c \/ a \/ d)
  CNF_CASL(P') = (not d' \/ c' \/ not b' \/ not a')
/\ (b' \/ not d' \/ a' \/ not c') /\ (b' \/ c' \/ a' \/ d')
```

Checking with Hets that T and CNF_CASL(T) are equivalent:

++

```
spec Check1 =
  pred a,b,c,d,a',b',c',d': ()
then %implies
 axiom
(
            (a' <=> ((not a /\ b) \/ (a /\ c /\ d)
\/ (a /\ not b /\ not c)))
        /\ (b' <=> ((not a /\ b)
\/ (not a /\ c /\ not d)))
        /\ (c' <=> ((not a /\ c /\ not d) \/ (b /\ c /\ d)
\/ (a /\ not b /\ not d)))
        /\ (d' <=> ((not a /\ b /\ d) \/ (a /\ c /\ d)
\/ (not a /\ not b /\ c /\ not d)))
)
  <=>     (
        (a \/ not b \/ a') /\ (not a \/ not c \/ not d \/ a')
/\ (not a \/ b \/ c \/ a') /\ (a \/ b \/ not a')
/\ (c \/ b \/ a \/ not a') /\ (d \/ b \/ a \/ not a')
/\ (c \/ not a \/ not b \/ not a') /\ (d \/ not a \/ not b \/ not a')
/\ (d \/ not a \/ not c \/ not a') /\ (a \/ b \/ not c \/ not a')
/\ (d \/ b \/ not c \/ not a') /\ (a \/ not b \/ b')
/\ (a \/ not c \/ d \/ b') /\ (not a \/ not b')
/\ (c \/ not a \/ not b') /\ (not d \/ not a \/ not b')
/\ (not a \/ b \/ not b') /\ (c \/ b \/ not b')
/\ (not d \/ b \/ not b') /\ (a \/ not c \/ d \/ c')
/\ (not b \/ not c \/ not d \/ c') /\ (not a \/ b \/ d \/ c')
/\ (b \/ c \/ a \/ not c') /\ (c \/ a \/ not c')
/\ (d \/ c \/ a \/ not c') /\ (b \/ not d \/ a \/ not c')
/\ (c \/ not d \/ a \/ not c') /\ (c \/ not a \/ not b \/ not c')
/\ (d \/ not a \/ not b \/ not c') /\ (c \/ not b \/ not c')
/\ (d \/ c \/ not b \/ not c') /\ (c \/ not d \/ not b \/ not c')
/\ (b \/ not a \/ not d \/ not c') /\ (c \/ not a \/ not d \/ not c')
/\ (b \/ c \/ not d \/ not c') /\ (c \/ not d \/ not c')
/\ (b \/ not d \/ not c') /\ (a \/ not b \/ not d \/ d')
/\ (not a \/ not c \/ not d \/ d') /\ (a \/ b \/ not c \/ d \/ d')
/\ (c \/ not a \/ not d') /\ (d \/ not a \/ not d')
/\ (c \/ b \/ not a \/ not d') /\ (d \/ b \/ not a \/ not d')
/\ (c \/ d \/ not a \/ not d') /\ (c \/ not a \/ not b \/ not d')
/\ (d \/ not a \/ not b \/ not d') /\ (a \/ d \/ not b \/ not d')
/\ (c \/ d \/ not b \/ not d') /\ (d \/ not b \/ not d')
/\ (d \/ not a \/ c \/ not d') /\ (a \/ b \/ c \/ not d' )
/\ (c \/ b \/ not d') /\ (d \/ b \/ c \/ not d')
```

```
/\ (a \/ d \/ c \/ not d') /\ (c \/ d \/ not d')
/\ (c \/ not a \/ not d \/ not d') /\ (a \/ b \/ not d \/ not d')
/\ (c \/ b \/ not d \/ not d')
        )
end

[successfully proved with HETS interface]
```

Part II: Page 11 from the slides:

---

once more the formulae (now all in CNF):

```
  I =  (a /\ (c \/ not b) /\ (d \/ not b) /\ (b \/ not c)
/\ (d \/ not c) /\ (b \/ not d) /\ (c \/ not d))

  T =   (a' <=> ((not a /\ b) \/ (a /\ c /\ d)
\/ (a /\ not b /\ not c)))
        /\ (b' <=> ((not a /\ b) \/ (not a /\ c /\ not d)))
        /\ (c' <=> ((not a /\ c /\ not d) \/ (b /\ c /\ d)
\/ (a /\ not b /\ not d)))
        /\ (d' <=> ((not a /\ b /\ d) \/ (a /\ c /\ d)
\/ (not a /\ not b /\ c /\ not d)))

  P = (not d \/ c \/ not b \/ not a) /\ (b \/ not d \/ a \/ not c)
/\ (b \/ c \/ a \/ d)

  P' = (not d' \/ c' \/ not b' \/ not a')
/\ (b' \/ not d' \/ a' \/ not c') /\ (b' \/ c' \/ a' \/ d')
```

a) Check I=>P?

```
spec Check2 =
  pred a,b,c,d: ()
  axiom  ((a /\ not b /\ not c /\ not d) \/ (a /\ b /\ c /\ d))
then %implies
 axiom (not d \/ c \/ not b \/ not a)
/\ (b \/ not d \/ a \/ not c) /\ (b \/ c \/ a \/ d)
end

[successfully proved with HETS interface]
```

```
b) Check I/\T=>P‘?

spec Check3 =
  pred a,b,c,d,a',b',c',d': ()
  axiom ((a /\ not b /\ not c /\ not d) \/ (a /\ b /\ c /\ d))
  axiom (a' <=> ((not a /\ b) \/ (a /\ c /\ d)
\/ (a /\ not b /\ not c)))
        /\ (b' <=> ((not a /\ b) \/ (not a /\ c /\ not d)))
        /\ (c' <=> ((not a /\ c /\ not d) \/ (b /\ c /\ d)
\/ (a /\ not b /\ not d)))
        /\ (d' <=> ((not a /\ b /\ d) \/ (a /\ c /\ d)
\/ (not a /\ not b /\ c /\ not d)))
then %implies
 axiom (not d' \/ c' \/ not b' \/ not a')
/\ (b' \/ not d' \/ a' \/ not c') /\ (b' \/ c' \/ a' \/ d')
end


c) Set F1 :=P

  F1 = (not d \/ c \/ not b \/ not a)
/\ (b \/ not d \/ a \/ not c) /\ (b \/ c \/ a \/ d)


d) For every clause c in clauses(I), if c not-in clauses(F1),
check:
    I /\ T => c‘?
    If it does, set F1 := F1 /\ c


clauses(I) = {a,
        (c \/ not b),
        (d \/ not b),
        (b \/ not c),
        (d \/ not c),
        (b \/ not d),
        (c \/ not d)}

clauses(F1) = {(not d \/ c \/ not b \/ not a),
        (b \/ not d \/ a \/ not c),
```

```
        (b \/ c \/ a \/ d)}

d-1) c = a

check if I /\ T => a':

spec Check4 =
  pred a,b,c,d,a',b',c',d': ()
  axiom (a /\ (c \/ not b) /\ (d \/ not b) /\ (b \/ not c)
/\ (d \/ not c) /\ (b \/ not d) /\ (c \/ not d))
  axiom (a' <=> ((not a /\ b) \/ (a /\ c /\ d)
\/ (a /\ not b /\ not c)))
        /\ (b' <=> ((not a /\ b) \/ (not a /\ c /\ not d)))
        /\ (c' <=> ((not a /\ c /\ not d) \/ (b /\ c /\ d)
\/ (a /\ not b /\ not d)))
        /\ (d' <=> ((not a /\ b /\ d) \/ (a /\ c /\ d)
\/ (not a /\ not b /\ c /\ not d)))
then %implies
 axiom a'
end

[proved with Hets]

Thus, we add a to F1, and obtain:

   F1 = (not d \/ c \/ not b \/ not a) /\ (b \/ not d \/ a \/ not c)
/\ (b \/ c \/ a \/ d)
        /\ a


d-2) c = (c \/ not b)

check if I /\ T => b':

spec Check5 =
  pred a,b,c,d,a',b',c',d': ()
  axiom (a /\ b /\ c /\ d)
  axiom      (
        (a' <=> ((not a /\ b) \/ (a /\ c /\ d)
\/ (a /\ not b /\ not c)))
        /\ (b' <=> ((not a /\ b) \/ (not a /\ c /\ not d)))
        /\ (c' <=> ((not a /\ c /\ not d) \/ (b /\ c /\ d)
```

```
\/ (a /\ not b /\ not d)))
        /\ (d' <=> ((not a /\ b /\ d) \/ (a /\ c /\ d)
\/ (not a /\ not b /\ c /\ not d)))
        )
then %implies
 axiom (c' \/ not b')
end

Thus, we add (c \/ not b) to F1, and obtain:

   F1 = (not d \/ c \/ not b \/ not a)
/\ (b \/ not d \/ a \/ not c)
/\ (b \/ c \/ a \/ d)
        /\ a
        /\ (c \/ not b)


d-3) c = (d \/ not b)

check if I /\ T => b':

spec Check6 =
  pred a,b,c,d,a',b',c',d': ()
  axiom (a /\ b /\ c /\ d)
  axiom      (
        (a' <=> ((not a /\ b) \/ (a /\ c /\ d)
\/ (a /\ not b /\ not c)))
        /\ (b' <=> ((not a /\ b) \/ (not a /\ c /\ not d)))
        /\ (c' <=> ((not a /\ c /\ not d) \/ (b /\ c /\ d)
\/ (a /\ not b /\ not d)))
        /\ (d' <=> ((not a /\ b /\ d) \/ (a /\ c /\ d)
\/ (not a /\ not b /\ c /\ not d)))
        )
then %implies
 axiom (d' \/ not b')
end

Thus, we add (d \/ not b) to F1, and obtain:

   F1 = (not d \/ c \/ not b \/ not a)
/\ (b \/ not d \/ a \/ not c) /\ (b \/ c \/ a \/ d)
        /\ a
```

```
        /\ (c \/ not b)
        /\ (d \/ not b)


d-4) c = (b \/ not c)

check if I /\ T => b':

spec Check7 =
  pred a,b,c,d,a',b',c',d': ()
  axiom (a /\ b /\ c /\ d)
  axiom      (
        (a' <=> ((not a /\ b) \/ (a /\ c /\ d)
\/ (a /\ not b /\ not c)))
        /\ (b' <=> ((not a /\ b) \/ (not a /\ c /\ not d)))
        /\ (c' <=> ((not a /\ c /\ not d) \/ (b /\ c /\ d)
\/ (a /\ not b /\ not d)))
        /\ (d' <=> ((not a /\ b /\ d) \/ (a /\ c /\ d)
\/ (not a /\ not b /\ c /\ not d)))
        )
then %implies
 axiom (b' \/ not c')
end

[disproved with Hets]

Thus, we do not add b /\ not c to F1


d-5) c = (d \/ not c)

check if I /\ T => c':

spec Check8 =
  pred a,b,c,d,a',b',c',d': ()
  axiom (a /\ b /\ c /\ d)
  axiom      (
        (a' <=> ((not a /\ b) \/ (a /\ c /\ d)
\/ (a /\ not b /\ not c)))
        /\ (b' <=> ((not a /\ b) \/ (not a /\ c /\ not d)))
        /\ (c' <=> ((not a /\ c /\ not d) \/ (b /\ c /\ d)
\/ (a /\ not b /\ not d)))
```

```
        /\ (d' <=> ((not a /\ b /\ d) \/ (a /\ c /\ d)
\/ (not a /\ not b /\ c /\ not d)))
        )
then %implies
 axiom (d' \/ not c')
end
```

Thus, we add (d \/ not c) to F1, and obtain:

```
   F1 = (not d \/ c \/ not b \/ not a)
/\ (b \/ not d \/ a \/ not c) /\ (b \/ c \/ a \/ d)
        /\ a
        /\ (c \/ not b)
        /\ (d \/ not b)
        /\ (d \/ not c)
```

d-6) c = (b \/ not d)

check if I /\ T => d':

```
spec Check9 =
  pred a,b,c,d,a',b',c',d': ()
  axiom (a /\ b /\ c /\ d)
  axiom      (
        (a' <=> ((not a /\ b) \/ (a /\ c /\ d)
\/ (a /\ not b /\ not c)))
        /\ (b' <=> ((not a /\ b)
\/ (not a /\ c /\ not d)))
        /\ (c' <=> ((not a /\ c /\ not d) \/ (b /\ c /\ d)
\/ (a /\ not b /\ not d)))
        /\ (d' <=> ((not a /\ b /\ d) \/ (a /\ c /\ d)
\/ (not a /\ not b /\ c /\ not d)))
        )
then %implies
 axiom (b' \/ not d')
end
```

[disproved with Hets]

Thus, we do not add b /\ not d to F1

```
d-7) c = (c \/ not d)

check if I /\ T => d':

spec Check9 =
  pred a,b,c,d,a',b',c',d': ()
  axiom (a /\ b /\ c /\ d)
  axiom      (
        (a' <=> ((not a /\ b) \/ (a /\ c /\ d)
\/ (a /\ not b /\ not c)))
        /\ (b' <=> ((not a /\ b) \/ (not a /\ c /\ not d)))
        /\ (c' <=> ((not a /\ c /\ not d) \/ (b /\ c /\ d)
\/ (a /\ not b /\ not d)))
        /\ (d' <=> ((not a /\ b /\ d) \/ (a /\ c /\ d)
\/ (not a /\ not b /\ c /\ not d)))
        )
then %implies
 axiom (c' \/ not d')
end

Thus, we add (c \/ not d) to F1, and obtain:

   F1 = (not d \/ c \/ not b \/ not a)
/\ (b \/ not d \/ a \/ not c) /\ (b \/ c \/ a \/ d)
        /\ a
        /\ (c \/ not b)
        /\ (d \/ not b)
        /\ (d \/ not c)
        /\ (c \/ not d)


Part III: Page 12 from the slides:
-----------------------------------


a) Check: Is it the case that Fk /\ T => P'?
b) If it is, then
 b-1) set Fk+1 := P
 b-2) for every clause c $\in$ Fk , check if Fk/\T=>c'?
      If it is, set Fk+1 := Fk+1 /\ c
c) If Fk =Fk+1: done
```

```
k=1:
++++

a) do we have F1 /\ T => P' ?

spec Check10 =
  pred a,b,c,d,a',b',c',d': ()
  axiom (not d \/ c \/ not b \/ not a)
/\ (b \/ not d \/ a \/ not c) /\ (b \/ c \/ a \/ d)
        /\ a
        /\ (c \/ not b)
        /\ (d \/ not b)
        /\ (d \/ not c)
        /\ (c \/ not d)
  axiom    (
        (a' <=> ((not a /\ b) \/ (a /\ c /\ d)
\/ (a /\ not b /\ not c)))
        /\ (b' <=> ((not a /\ b) \/ (not a /\ c /\ not d)))
        /\ (c' <=> ((not a /\ c /\ not d) \/ (b /\ c /\ d)
\/ (a /\ not b /\ not d)))
        /\ (d' <=> ((not a /\ b /\ d) \/ (a /\ c /\ d)
\/ (not a /\ not b /\ c /\ not d)))
        )
then %implies
 axiom  (not d' \/ c' \/ not b' \/ not a')
/\ (b' \/ not d' \/ a' \/ not c') /\ (b' \/ c' \/ a' \/ d')
end

[proved with Hets]


b-1) F2 = (not d \/ c \/ not b \/ not a)
/\ (b \/ not d \/ a \/ not c) /\ (b \/ c \/ a \/ d)

b-2) clauses(F1) = {(not d \/ c \/ not b \/ not a),
        (b \/ not d \/ a \/ not c),
        (b \/ c \/ a \/ d),
        a,
        (c \/ not b),
        (d \/ not b),
        (d \/ not c),
```

```
        (c \/ not d)}


    let c = (not d \/ c \/ not b \/ not a)
    check F1 /\ T => c'

spec Check11 =
  pred a,b,c,d,a',b',c',d': ()
  axiom (not d \/ c \/ not b \/ not a)
/\ (b \/ not d \/ a \/ not c) /\ (b \/ c \/ a \/ d)
        /\ a
        /\ (c \/ not b)
        /\ (d \/ not b)
        /\ (d \/ not c)
        /\ (c \/ not d)
  axiom      (
        (a' <=> ((not a /\ b) \/ (a /\ c /\ d)
\/ (a /\ not b /\ not c)))
        /\ (b' <=> ((not a /\ b) \/ (not a /\ c /\ not d)))
        /\ (c' <=> ((not a /\ c /\ not d) \/ (b /\ c /\ d)
\/ (a /\ not b /\ not d)))
        /\ (d' <=> ((not a /\ b /\ d) \/ (a /\ c /\ d)
\/ (not a /\ not b /\ c /\ not d)))
        )
then %implies
 axiom (not d' \/ c' \/ not b' \/ not a')
end

[successfully proved with HETS]

thus F2 = (not d \/ c \/ not b \/ not a)
/\ (b \/ not d \/ a \/ not c) /\ (b \/ c \/ a \/ d)
        /\ (not d \/ c \/ not b \/ not a)


    let c = (b \/ not d \/ a \/ not c)
    check F1 /\ T => c'

spec Check12 =
  pred a,b,c,d,a',b',c',d': ()
  axiom (not d \/ c \/ not b \/ not a)
/\ (b \/ not d \/ a \/ not c) /\ (b \/ c \/ a \/ d)
```

```
        /\ a
        /\ (c \/ not b)
        /\ (d \/ not b)
        /\ (d \/ not c)
        /\ (c \/ not d)
  axiom      (
        (a' <=> ((not a /\ b) \/ (a /\ c /\ d)
\/ (a /\ not b /\ not c)))
        /\ (b' <=> ((not a /\ b) \/ (not a /\ c /\ not d)))
        /\ (c' <=> ((not a /\ c /\ not d) \/ (b /\ c /\ d)
\/ (a /\ not b /\ not d)))
        /\ (d' <=> ((not a /\ b /\ d) \/ (a /\ c /\ d)
\/ (not a /\ not b /\ c /\ not d)))
        )
then %implies
 axiom (b' \/ not d' \/ a' \/ not c')
end

[successfully proved with HETS]

thus F2 = (not d \/ c \/ not b \/ not a)
/\ (b \/ not d \/ a \/ not c) /\ (b \/ c \/ a \/ d)
        /\ (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)


    let c = (b \/ not d \/ a \/ not c)
    check F1 /\ T => c'

spec Check13 =
  pred a,b,c,d,a',b',c',d': ()
  axiom (not d \/ c \/ not b \/ not a)
/\ (b \/ not d \/ a \/ not c) /\ (b \/ c \/ a \/ d)
        /\ a
        /\ (c \/ not b)
        /\ (d \/ not b)
        /\ (d \/ not c)
        /\ (c \/ not d)
  axiom      (
        (a' <=> ((not a /\ b) \/ (a /\ c /\ d)
\/ (a /\ not b /\ not c)))
        /\ (b' <=> ((not a /\ b) \/ (not a /\ c /\ not d)))
```

```
        /\ (c' <=> ((not a /\ c /\ not d) \/ (b /\ c /\ d)
\/ (a /\ not b /\ not d)))
        /\ (d' <=> ((not a /\ b /\ d) \/ (a /\ c /\ d)
\/ (not a /\ not b /\ c /\ not d)))
        )
then %implies
 axiom (b' \/ c' \/ a' \/ d')
end

[successfully proved with HETS]

thus F2 = (not d \/ c \/ not b \/ not a)
/\ (b \/ not d \/ a \/ not c) /\ (b \/ c \/ a \/ d)
        /\ (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)


    let c = a
    check F1 /\ T => c'

spec Check14 =
  pred a,b,c,d,a',b',c',d': ()
  axiom (not d \/ c \/ not b \/ not a)
/\ (b \/ not d \/ a \/ not c) /\ (b \/ c \/ a \/ d)
        /\ a
        /\ (c \/ not b)
        /\ (d \/ not b)
        /\ (d \/ not c)
        /\ (c \/ not d)
  axiom      (
        (a' <=> ((not a /\ b) \/ (a /\ c /\ d)
\/ (a /\ not b /\ not c)))
        /\ (b' <=> ((not a /\ b) \/ (not a /\ c /\ not d)))
        /\ (c' <=> ((not a /\ c /\ not d) \/ (b /\ c /\ d)
\/ (a /\ not b /\ not d)))
        /\ (d' <=> ((not a /\ b /\ d) \/ (a /\ c /\ d)
\/ (not a /\ not b /\ c /\ not d)))
        )
then %implies
 axiom a'
end
```

```
[proved with HETS]

Thus, we add a to F1, and obtain:

   F2 = (not d \/ c \/ not b \/ not a)
/\ (b \/ not d \/ a \/ not c) /\ (b \/ c \/ a \/ d)
        /\ a


    let c = (c \/ not b)
    check F1 /\ T => c'

spec Check15 =
  pred a,b,c,d,a',b',c',d': ()
  axiom (not d \/ c \/ not b \/ not a)
/\ (b \/ not d \/ a \/ not c) /\ (b \/ c \/ a \/ d)
        /\ a
        /\ (c \/ not b)
        /\ (d \/ not b)
        /\ (d \/ not c)
        /\ (c \/ not d)
  axiom      (
        (a' <=> ((not a /\ b) \/ (a /\ c /\ d)
\/ (a /\ not b /\ not c)))
        /\ (b' <=> ((not a /\ b) \/ (not a /\ c /\ not d)))
        /\ (c' <=> ((not a /\ c /\ not d) \/ (b /\ c /\ d)
\/ (a /\ not b /\ not d)))
        /\ (d' <=> ((not a /\ b /\ d) \/ (a /\ c /\ d)
\/ (not a /\ not b /\ c /\ not d)))
        )
then %implies
 axiom (c' \/ not b')
end

[successfully proved with HETS]

thus F2 = (not d \/ c \/ not b \/ not a)
/\ (b \/ not d \/ a \/ not c) /\ (b \/ c \/ a \/ d)
        /\ a
        /\ (c \/ not b)
```

```
    let c = (d \/ not b)
    check F1 /\ T => c'

spec Check16 =
  pred a,b,c,d,a',b',c',d': ()
  axiom (not d \/ c \/ not b \/ not a)
/\ (b \/ not d \/ a \/ not c) /\ (b \/ c \/ a \/ d)
        /\ a
        /\ (c \/ not b)
        /\ (d \/ not b)
        /\ (d \/ not c)
        /\ (c \/ not d)
  axiom      (
        (a' <=> ((not a /\ b) \/ (a /\ c /\ d)
\/ (a /\ not b /\ not c)))
        /\ (b' <=> ((not a /\ b) \/ (not a /\ c /\ not d)))
        /\ (c' <=> ((not a /\ c /\ not d) \/ (b /\ c /\ d)
\/ (a /\ not b /\ not d)))
        /\ (d' <=> ((not a /\ b /\ d) \/ (a /\ c /\ d)
\/ (not a /\ not b /\ c /\ not d)))
        )
then %implies
 axiom (d' \/ not b')
end

[successfully proved with HETS]

thus F2 = (not d \/ c \/ not b \/ not a)
/\ (b \/ not d \/ a \/ not c) /\ (b \/ c \/ a \/ d)
        /\ a
        /\ (c \/ not b)
        /\ (d \/ not d)


    let c = (d \/ not c)
    check F1 /\ T => c'

spec Check17 =
  pred a,b,c,d,a',b',c',d': ()
  axiom (not d \/ c \/ not b \/ not a)
/\ (b \/ not d \/ a \/ not c) /\ (b \/ c \/ a \/ d)
```

```
        /\ a
        /\ (c \/ not b)
        /\ (d \/ not b)
        /\ (d \/ not c)
        /\ (c \/ not d)
  axiom      (
        (a' <=> ((not a /\ b) \/ (a /\ c /\ d)
\/ (a /\ not b /\ not c)))
        /\ (b' <=> ((not a /\ b) \/ (not a /\ c /\ not d)))
        /\ (c' <=> ((not a /\ c /\ not d) \/ (b /\ c /\ d)
\/ (a /\ not b /\ not d)))
        /\ (d' <=> ((not a /\ b /\ d) \/ (a /\ c /\ d)
\/ (not a /\ not b /\ c /\ not d)))
        )
then %implies
 axiom (d' \/ not c')
end

[disproved with Hets]

Thus, we do not add d /\ not c to F1


    let c = (c \/ not d)
    check F1 /\ T => c'

spec Check18 =
  pred a,b,c,d,a',b',c',d': ()
  axiom (not d \/ c \/ not b \/ not a)
/\ (b \/ not d \/ a \/ not c) /\ (b \/ c \/ a \/ d)
        /\ a
        /\ (c \/ not b)
        /\ (d \/ not b)
        /\ (d \/ not c)
        /\ (c \/ not d)
  axiom      (
        (a' <=> ((not a /\ b) \/ (a /\ c /\ d)
\/ (a /\ not b /\ not c)))
        /\ (b' <=> ((not a /\ b) \/ (not a /\ c /\ not d)))
        /\ (c' <=> ((not a /\ c /\ not d) \/ (b /\ c /\ d)
\/ (a /\ not b /\ not d)))
        /\ (d' <=> ((not a /\ b /\ d) \/ (a /\ c /\ d)
```

```
\/ (not a /\ not b /\ c /\ not d)))
        )
then %implies
 axiom (c' \/ not d')
end

[disproved with Hets]

Thus, we do not add c /\ not d to F2


c) thus F2 = (not d \/ c \/ not b \/ not a)
/\ (b \/ not d \/ a \/ not c) /\ (b \/ c \/ a \/ d)
        /\ a
        /\ (c \/ not b)
        /\ (d \/ not b)
        != F1

i.e. F2 = F1


Part IV: Repeat part 3 until Fk =Fk+1:
--------------------------------------


a) Check: Is it the case that Fk /\ T => P'?
b) If it is, then
 b-1) set Fk+1 := P
 b-2) for every clause c $\in$ Fk , check if Fk/\T=>c'?
      If it is, set Fk+1 := Fk+1 /\ c
c) If Fk =Fk+1: done


k=2:
++++

a) do we have F2 /\ T => P' ?

spec Check19 =
  pred a,b,c,d,a',b',c',d': ()
  axiom (not d \/ c \/ not b \/ not a)
/\ (b \/ not d \/ a \/ not c) /\ (b \/ c \/ a \/ d)
        /\ a
```

```
        /\ (c \/ not b)
        /\ (d \/ not b)
  axiom     (
        (a' <=> ((not a /\ b) \/ (a /\ c /\ d)
\/ (a /\ not b /\ not c)))
        /\ (b' <=> ((not a /\ b) \/ (not a /\ c /\ not d)))
        /\ (c' <=> ((not a /\ c /\ not d) \/ (b /\ c /\ d)
\/ (a /\ not b /\ not d)))
        /\ (d' <=> ((not a /\ b /\ d) \/ (a /\ c /\ d)
\/ (not a /\ not b /\ c /\ not d)))
        )
then %implies
 axiom  (not d' \/ c' \/ not b' \/ not a')
/\ (b' \/ not d' \/ a' \/ not c') /\ (b' \/ c' \/ a' \/ d')
end


[successfully proved with Hets]


b-1) F3 = (not d \/ c \/ not b \/ not a)
/\ (b \/ not d \/ a \/ not c) /\ (b \/ c \/ a \/ d)

b-2) clauses(F2) = {(not d \/ c \/ not b \/ not a),
        (b \/ not d \/ a \/ not c),
        (b \/ c \/ a \/ d)
        a,
        (c \/ not b),
        (d \/ not b),}


    let c = (not d \/ c \/ not b \/ not a)
    check F2 /\ T => c'

spec Check20 =
  pred a,b,c,d,a',b',c',d': ()
  axiom (not d \/ c \/ not b \/ not a)
/\ (b \/ not d \/ a \/ not c) /\ (b \/ c \/ a \/ d)
        /\ a
        /\ (c \/ not b)
        /\ (d \/ not b)
  axiom      (
```

```
        (a' <=> ((not a /\ b) \/ (a /\ c /\ d)
\/ (a /\ not b /\ not c)))
        /\ (b' <=> ((not a /\ b) \/ (not a /\ c /\ not d)))
        /\ (c' <=> ((not a /\ c /\ not d) \/ (b /\ c /\ d)
\/ (a /\ not b /\ not d)))
        /\ (d' <=> ((not a /\ b /\ d) \/ (a /\ c /\ d)
\/ (not a /\ not b /\ c /\ not d)))
        )
then %implies
 axiom (not d' \/ c' \/ not b' \/ not a')
end

[successfully proved with HETS]

thus F3 = (not d \/ c \/ not b \/ not a)
/\ (b \/ not d \/ a \/ not c) /\ (b \/ c \/ a \/ d)
        /\ (not d \/ c \/ not b \/ not a)


    let c = (b \/ not d \/ a \/ not c)
    check F2 /\ T => c'

spec Check21 =
  pred a,b,c,d,a',b',c',d': ()
  axiom (not d \/ c \/ not b \/ not a)
/\ (b \/ not d \/ a \/ not c) /\ (b \/ c \/ a \/ d)
        /\ a
        /\ (c \/ not b)
        /\ (d \/ not b)
  axiom      (
        (a' <=> ((not a /\ b) \/ (a /\ c /\ d)
\/ (a /\ not b /\ not c)))
        /\ (b' <=> ((not a /\ b) \/ (not a /\ c /\ not d)))
        /\ (c' <=> ((not a /\ c /\ not d) \/ (b /\ c /\ d)
\/ (a /\ not b /\ not d)))
        /\ (d' <=> ((not a /\ b /\ d) \/ (a /\ c /\ d)
\/ (not a /\ not b /\ c /\ not d)))
        )
then %implies
 axiom (b' \/ not d' \/ a' \/ not c')
end
```

```
[successfully proved with HETS]

thus F3 = (not d \/ c \/ not b \/ not a)
/\ (b \/ not d \/ a \/ not c) /\ (b \/ c \/ a \/ d)
        /\ (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)


    let c = (b \/ c \/ a \/ d)
    check F2 /\ T => c'

spec Check22 =
  pred a,b,c,d,a',b',c',d': ()
  axiom (not d \/ c \/ not b \/ not a)
/\ (b \/ not d \/ a \/ not c) /\ (b \/ c \/ a \/ d)
        /\ a
        /\ (c \/ not b)
        /\ (d \/ not b)
  axiom      (
        (a' <=> ((not a /\ b) \/ (a /\ c /\ d)
\/ (a /\ not b /\ not c)))
        /\ (b' <=> ((not a /\ b) \/ (not a /\ c /\ not d)))
        /\ (c' <=> ((not a /\ c /\ not d) \/ (b /\ c /\ d)
\/ (a /\ not b /\ not d)))
        /\ (d' <=> ((not a /\ b /\ d) \/ (a /\ c /\ d)
\/ (not a /\ not b /\ c /\ not d)))
        )
then %implies
 axiom (b' \/ c' \/ a' \/ d')
end

[successfully proved with HETS]

thus F3 = (not d \/ c \/ not b \/ not a)
/\ (b \/ not d \/ a \/ not c) /\ (b \/ c \/ a \/ d)
        /\ (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)


    let c = a
    check F2 /\ T => c'
```

```
spec Check23 =
  pred a,b,c,d,a',b',c',d': ()
  axiom (not d \/ c \/ not b \/ not a)
/\ (b \/ not d \/ a \/ not c) /\ (b \/ c \/ a \/ d)
        /\ a
        /\ (c \/ not b)
        /\ (d \/ not b)
  axiom      (
        (a' <=> ((not a /\ b) \/ (a /\ c /\ d)
\/ (a /\ not b /\ not c)))
        /\ (b' <=> ((not a /\ b) \/ (not a /\ c /\ not d)))
        /\ (c' <=> ((not a /\ c /\ not d) \/ (b /\ c /\ d)
\/ (a /\ not b /\ not d)))
        /\ (d' <=> ((not a /\ b /\ d) \/ (a /\ c /\ d)
\/ (not a /\ not b /\ c /\ not d)))
        )
then %implies
 axiom a'
end

[disproved with Hets]

Thus, we do not add a to F3


    let c = (c \/ not b)
    check F2 /\ T => c'

spec Check24 =
  pred a,b,c,d,a',b',c',d': ()
  axiom (not d \/ c \/ not b \/ not a)
/\ (b \/ not d \/ a \/ not c) /\ (b \/ c \/ a \/ d)
        /\ a
        /\ (c \/ not b)
        /\ (d \/ not b)
  axiom      (
        (a' <=> ((not a /\ b) \/ (a /\ c /\ d)
\/ (a /\ not b /\ not c)))
        /\ (b' <=> ((not a /\ b) \/ (not a /\ c /\ not d)))
        /\ (c' <=> ((not a /\ c /\ not d) \/ (b /\ c /\ d)
\/ (a /\ not b /\ not d)))
```

```
        /\ (d' <=> ((not a /\ b /\ d) \/ (a /\ c /\ d)
\/ (not a /\ not b /\ c /\ not d)))
        )
then %implies
 axiom (c' \/ not b')
end


[successfully proved with HETS]

thus F3 = (not d \/ c \/ not b \/ not a)
/\ (b \/ not d \/ a \/ not c) /\ (b \/ c \/ a \/ d)
        /\ (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ (c \/ not b)


    let c = (d \/ not b)
    check F2 /\ T => c'

spec Check25 =
  pred a,b,c,d,a',b',c',d': ()
  axiom (not d \/ c \/ not b \/ not a)
/\ (b \/ not d \/ a \/ not c) /\ (b \/ c \/ a \/ d)
        /\ a
        /\ (c \/ not b)
        /\ (d \/ not b)
  axiom       (
        (a' <=> ((not a /\ b) \/ (a /\ c /\ d)
\/ (a /\ not b /\ not c)))
        /\ (b' <=> ((not a /\ b) \/ (not a /\ c /\ not d)))
        /\ (c' <=> ((not a /\ c /\ not d) \/ (b /\ c /\ d)
\/ (a /\ not b /\ not d)))
        /\ (d' <=> ((not a /\ b /\ d) \/ (a /\ c /\ d)
\/ (not a /\ not b /\ c /\ not d)))
        )
then %implies
 axiom (d' \/ not b')
end
```

```
[successfully proved with HETS]

thus F3 = (not d \/ c \/ not b \/ not a)
/\ (b \/ not d \/ a \/ not c) /\ (b \/ c \/ a \/ d)
        /\ (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ (c \/ not b)
        /\ (d \/ not b)

c) thus F3 = (not d \/ c \/ not b \/ not a)
/\ (b \/ not d \/ a \/ not c) /\ (b \/ c \/ a \/ d)
        /\ (c \/ not b)
        /\ (d \/ not b)
        != F2

Part IV: Repeat part 3 until Fk =Fk+1:
--------------------------------------


a) Check: Is it the case that Fk /\ T => P'?
b) If it is, then
 b-1) set Fk+1 := P
 b-2) for every clause c $\in$ Fk , check if Fk/\T=>c'?
      If it is, set Fk+1 := Fk+1 /\ c
c) If Fk =Fk+1: done


k=2:
++++

a) do we have F3 /\ T => P' ?

spec Check26 =
  pred a,b,c,d,a',b',c',d': ()
  axiom (not d \/ c \/ not b \/ not a)
/\ (b \/ not d \/ a \/ not c) /\ (b \/ c \/ a \/ d)
        /\ (c \/ not b)
        /\ (d \/ not b)
  axiom    (
        (a' <=> ((not a /\ b) \/ (a /\ c /\ d)
\/ (a /\ not b /\ not c)))
        /\ (b' <=> ((not a /\ b) \/ (not a /\ c /\ not d)))
```

```
        /\ (c' <=> ((not a /\ c /\ not d) \/ (b /\ c /\ d)
\/ (a /\ not b /\ not d)))
        /\ (d' <=> ((not a /\ b /\ d) \/ (a /\ c /\ d)
\/ (not a /\ not b /\ c /\ not d)))
        )
then %implies
 axiom  (not d' \/ c' \/ not b' \/ not a')
/\ (b' \/ not d' \/ a' \/ not c') /\ (b' \/ c' \/ a' \/ d')
end


[disproved with Hets]


Therefore find the state(s) s that cause F3 /\ T => P'
to not hold.
s = (not a /\ not b /\ not c /\ d)


Check F3 /\ neg s /\ T => neg s'

spec Check27 =
  pred a,b,c,d,a',b',c',d': ()
  axiom (not d \/ c \/ not b \/ not a)
/\ (b \/ not d \/ a \/ not c) /\ (b \/ c \/ a \/ d)
        /\ (c \/ not b)
        /\ (d \/ not b)
  axiom    (not (not a /\ not b /\ not c /\ d))
  axiom    (
        (a' <=> ((not a /\ b) \/ (a /\ c /\ d)
\/ (a /\ not b /\ not c)))
        /\ (b' <=> ((not a /\ b) \/ (not a /\ c /\ not d)))
        /\ (c' <=> ((not a /\ c /\ not d) \/ (b /\ c /\ d)
\/ (a /\ not b /\ not d)))
        /\ (d' <=> ((not a /\ b /\ d) \/ (a /\ c /\ d)
\/ (not a /\ not b /\ c /\ not d)))
        )
then %implies
 axiom  (not (not a' /\ not b' /\ not c' /\ d'))
end
```

```
Drop a literal

Clause c: not a /\ not b /\ not c

Check F0 => neg c
[proved]
Check Fk /\ neg c /\ T => neg c'
[proved]

Therefore, set F3 to F3 /\ neg q

F3 = (not d \/ c \/ not b \/ not a)
/\ (b \/ not d \/ a \/ not c) /\ (b \/ c \/ a \/ d)
        /\ (c \/ not b)
        /\ (d \/ not b)
        /\ not(not a /\ not b /\ not c)


Check F3 /\ T => P'

spec Check28 =
  pred a,b,c,d,a',b',c',d': ()
  axiom (not d \/ c \/ not b \/ not a)
/\ (b \/ not d \/ a \/ not c) /\ (b \/ c \/ a \/ d)
        /\ (c \/ not b)
        /\ (d \/ not b)
        /\ not(not a /\ not b /\ not c)
  axiom    (
        (a' <=> ((not a /\ b) \/ (a /\ c /\ d)
\/ (a /\ not b /\ not c)))
        /\ (b' <=> ((not a /\ b) \/ (not a /\ c /\ not d)))
        /\ (c' <=> ((not a /\ c /\ not d) \/ (b /\ c /\ d)
\/ (a /\ not b /\ not d)))
        /\ (d' <=> ((not a /\ b /\ d) \/ (a /\ c /\ d)
\/ (not a /\ not b /\ c /\ not d)))
        )
then %implies
 axiom  (not d' \/ c' \/ not b' \/ not a')
/\ (b' \/ not d' \/ a' \/ not c') /\ (b' \/ c' \/ a' \/ d')
end
```

```
[successfully proved with HETS]


b-1) F4 = (not d \/ c \/ not b \/ not a)
/\ (b \/ not d \/ a \/ not c) /\ (b \/ c \/ a \/ d)

b-2) clauses(F3) = {(not d \/ c \/ not b \/ not a),
        (b \/ not d \/ a \/ not c),
        (b \/ c \/ a \/ d),
        (c \/ not b),
        (d \/ not b),
        not(not a /\ not b /\ not c)}


    let c = (not d \/ c \/ not b \/ not a)
    check F3 /\ T => c'

spec Check29 =
  pred a,b,c,d,a',b',c',d': ()
  axiom (not d \/ c \/ not b \/ not a)
/\ (b \/ not d \/ a \/ not c) /\ (b \/ c \/ a \/ d)
        /\ (c \/ not b)
        /\ (d \/ not b)
        /\ not(not a /\ not b /\ not c)
  axiom      (
        (a' <=> ((not a /\ b) \/ (a /\ c /\ d)
\/ (a /\ not b /\ not c)))
        /\ (b' <=> ((not a /\ b) \/ (not a /\ c /\ not d)))
        /\ (c' <=> ((not a /\ c /\ not d) \/ (b /\ c /\ d)
\/ (a /\ not b /\ not d)))
        /\ (d' <=> ((not a /\ b /\ d) \/ (a /\ c /\ d)
\/ (not a /\ not b /\ c /\ not d)))
        )
then %implies
 axiom (not d' \/ c' \/ not b' \/ not a')
end

[successfully proved with HETS]

thus F4 = (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
```

```
        /\ (not d \/ c \/ not b \/ not a)


    let c = (b \/ not d \/ a \/ not c)
    check F3 /\ T => c'

spec Check30 =
  pred a,b,c,d,a',b',c',d': ()
  axiom (not d \/ c \/ not b \/ not a)
/\ (b \/ not d \/ a \/ not c) /\ (b \/ c \/ a \/ d)
        /\ (c \/ not b)
        /\ (d \/ not b)
        /\ not(not a /\ not b /\ not c)
  axiom      (
        (a' <=> ((not a /\ b) \/ (a /\ c /\ d)
\/ (a /\ not b /\ not c)))
        /\ (b' <=> ((not a /\ b) \/ (not a /\ c /\ not d)))
        /\ (c' <=> ((not a /\ c /\ not d) \/ (b /\ c /\ d)
\/ (a /\ not b /\ not d)))
        /\ (d' <=> ((not a /\ b /\ d) \/ (a /\ c /\ d)
\/ (not a /\ not b /\ c /\ not d)))
        )
then %implies
 axiom (b' \/ not d' \/ a' \/ not c')
end

[successfully proved with HETS]

thus F4 = (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)


    let c = (c \/ not b)
    check F3 /\ T => c'

spec Check32 =
  pred a,b,c,d,a',b',c',d': ()
  axiom (not d \/ c \/ not b \/ not a)
/\ (b \/ not d \/ a \/ not c) /\ (b \/ c \/ a \/ d)
```

```
        /\ (c \/ not b)
        /\ (d \/ not b)
        /\ not(not a /\ not b /\ not c)
  axiom      (
        (a' <=> ((not a /\ b) \/ (a /\ c /\ d)
\/ (a /\ not b /\ not c)))
        /\ (b' <=> ((not a /\ b) \/ (not a /\ c /\ not d)))
        /\ (c' <=> ((not a /\ c /\ not d) \/ (b /\ c /\ d)
\/ (a /\ not b /\ not d)))
        /\ (d' <=> ((not a /\ b /\ d) \/ (a /\ c /\ d)
\/ (not a /\ not b /\ c /\ not d)))
        )
then %implies
 axiom (c' \/ not b')
end

[successfully proved with HETS]

thus F4 = (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ (c \/ not b)


    let c = (d \/ not b)
    check F4 /\ T => c'

spec Check33 =
  pred a,b,c,d,a',b',c',d': ()
  axiom (not d \/ c \/ not b \/ not a)
/\ (b \/ not d \/ a \/ not c) /\ (b \/ c \/ a \/ d)
        /\ (c \/ not b)
        /\ (d \/ not b)
        /\ not(not a /\ not b /\ not c)
  axiom      (
        (a' <=> ((not a /\ b) \/ (a /\ c /\ d)
\/ (a /\ not b /\ not c)))
        /\ (b' <=> ((not a /\ b) \/ (not a /\ c /\ not d)))
        /\ (c' <=> ((not a /\ c /\ not d) \/ (b /\ c /\ d)
```

```
\/ (a /\ not b /\ not d)))
        /\ (d' <=> ((not a /\ b /\ d) \/ (a /\ c /\ d)
\/ (not a /\ not b /\ c /\ not d)))
        )
then %implies
 axiom (d' \/ not b')
end

[successfully proved with HETS]

thus F4 = (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ (c \/ not b)
        /\ (d \/ not b)


    let c = not(not a /\ not b /\ not c)
    check F3 /\ T => c'

spec Check34 =
  pred a,b,c,d,a',b',c',d': ()
  axiom (not d \/ c \/ not b \/ not a)
/\ (b \/ not d \/ a \/ not c) /\ (b \/ c \/ a \/ d)
        /\ (c \/ not b)
        /\ (d \/ not b)
        /\ not(not a /\ not b /\ not c)
  axiom      (
        (a' <=> ((not a /\ b) \/ (a /\ c /\ d)
\/ (a /\ not b /\ not c)))
        /\ (b' <=> ((not a /\ b) \/ (not a /\ c /\ not d)))
        /\ (c' <=> ((not a /\ c /\ not d) \/ (b /\ c /\ d)
\/ (a /\ not b /\ not d)))
        /\ (d' <=> ((not a /\ b /\ d) \/ (a /\ c /\ d)
\/ (not a /\ not b /\ c /\ not d)))
        )
then %implies
 axiom not(not a' /\ not b' /\ not c')
end
```

```
[successfully proved with HETS]

thus F4 = (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ (c \/ not b)
        /\ (d \/ not b)
        /\ not(not a /\ not b /\ not c)

Therefore, F4 = (not d \/ c \/ not b \/ not a)
        /\ (b \/ not d \/ a \/ not c)
        /\ (b \/ c \/ a \/ d)
        /\ (c \/ not b)
        /\ (d \/ not b)
        /\ not(not a /\ not b /\ not c)

        = F3

Therefore verification holds
```

B.2 Real-World Ladder Logic Examples

B.2.1 Level Crossing CASL Specifications

spec TRANSITION =

preds *cars_crossing*, *train_approach*, *lights_flash*, *barriers_down*, *train_crossing*, *train_passed*, *cars_crossing'*, *train_approach'*, *lights_flash'*, *barriers_down'*, *train_crossing'*, *train_passed'* : ()

- $cars\_crossing'$
  $\Leftrightarrow (\neg\ train\_approach \wedge \neg\ lights\_flash \wedge \neg\ barriers\_down \wedge \neg\ train\_crossing \wedge \neg\ train\_passed)$
  $\vee\ (cars\_crossing \wedge train\_approach \wedge \neg\ barriers\_down \wedge \neg\ train\_crossing \wedge \neg\ train\_passed)$
- $train\_approach'$
  $\Leftrightarrow (train\_approach \wedge lights\_flash \wedge barriers\_down \wedge \neg\ train\_crossing \wedge \neg\ train\_passed)$
  $\vee\ (cars\_crossing \wedge \neg\ lights\_flash \wedge \neg\ barriers\_down$

$\wedge \neg$ *train_crossing* $\wedge \neg$ *train_passed*)
$\vee$ (*cars_crossing* $\wedge$ *train_approach* $\wedge \neg$ *barriers_down*
$\wedge \neg$ *train_crossing* $\wedge \neg$ *train_passed*)

- *lights_flash'*
  $\Leftrightarrow$ ($\neg$ *cars_crossing* $\wedge \neg$ *train_approach* $\wedge$ *lights_flash*
  $\wedge$ *barriers_down* $\wedge$ *train_passed*)
  $\vee$ ($\neg$ *cars_crossing* $\wedge$ *lights_flash* $\wedge$ *barriers_down*
  $\wedge$ *train_crossing* $\wedge \neg$ *train_passed*)
  $\vee$ (*train_approach* $\wedge$ *lights_flash* $\wedge$ *barriers_down*
  $\wedge \neg$ *train_crossing* $\wedge \neg$ *train_passed*)
  $\vee$ (*cars_crossing* $\wedge$ *train_approach* $\wedge \neg$ *barriers_down*
  $\wedge \neg$ *train_crossing* $\wedge \neg$ *train_passed*)
- *barriers_down'*
  $\Leftrightarrow$ ($\neg$ *cars_crossing* $\wedge \neg$ *train_approach* $\wedge$ *lights_flash*
  $\wedge$ *barriers_down*)
  $\vee$ ($\neg$ *cars_crossing* $\wedge$ *lights_flash* $\wedge$ *barriers_down*
  $\wedge \neg$ *train_passed*)
  $\vee$ (*cars_crossing* $\wedge$ *train_approach* $\wedge$ *lights_flash*
  $\wedge \neg$ *train_crossing* $\wedge \neg$ *train_passed*)
- *train_crossing'*
  $\Leftrightarrow$ ($\neg$ *cars_crossing* $\wedge$ *lights_flash* $\wedge$ *barriers_down*
  $\wedge$ *train_crossing* $\wedge \neg$ *train_passed*)
  $\vee$ ($\neg$ *cars_crossing* $\wedge$ *train_approach* $\wedge$ *lights_flash*
  $\wedge$ *barriers_down* $\wedge \neg$ *train_passed*)
- *train_passed'*
  $\Leftrightarrow \neg$ *cars_crossing* $\wedge \neg$ *train_approach* $\wedge$ *lights_flash*
  $\wedge$ *barriers_down* $\wedge$ *train_crossing*

end

spec SafeInitial =
TRANSITION
then • *cars_crossing* $\wedge \neg$ *train_approach* $\wedge \neg$ *lights_flash*
$\wedge \neg$ *barriers_down* $\wedge \neg$ *train_crossing* $\wedge \neg$ *train_passed*
then %implies
• $\neg$ (*cars_crossing* $\wedge \neg$ *barriers_down*) $\vee \neg$ *train_crossing*
%(initialsafe)%
end

spec StepSafe =
TRANSITION
then %implies
• $\neg$ (*cars_crossing* $\wedge \neg$ *barriers_down*) $\vee \neg$ *train_crossing*

$$\Rightarrow \neg\ (cars\_crossing' \wedge \neg\ barriers\_down') \\ \vee \neg\ train\_crossing'$$

%(stepsafety)%

end

APPENDIX C

Classifying the Siemens Railway Interlocking Testbed

C.1 Classification of the Siemens Interlocking Tested

Interlocking	Ladder Logic/Safety Property Pair	LL-Verifier			IC3ref		ABC	
		IV Result	BMC Result	Type	IC3ref Result	Type	ABC Result	Type
A	A__Ref12_Train_detection_indicates_clear	yes	yes	A	yes	A	yes	A
A	A__Ref1_Route_set_and_locked	yes	yes	A	yes	A	yes	A
A	A__Ref22_Indicators_Proved	no	no	B	no	B	no	B
A	A__Ref32_All_signals_on	yes	yes	A	yes	A	yes	A
A	A__Ref35_Approach_Locking	no	no	B	yes	D	yes	D
B	B__Ref1_Route_set_and_locked	no	no	B	no	B	no	B
B	B__Ref22_Indicators_Proved	no	no	B	no	B	no	B
B	B__Ref32_All_signals_on	no	no	B	no	B	no	B
B	B__Ref35_Approach_Locking	no	no	B	no	B	no	B
DesignX	DesignX_GSP_2_1_11_RoutesInStaffLockoutLODP_group0	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_1_12_RoutesInStaffLockoutLODK_group0	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_12_RoutesInStaffLockoutLODK_group1	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_12_RoutesInStaffLockoutLODK_group2	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_12_RoutesInStaffLockoutLODK_group3	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_12_RoutesInStaffLockoutLODK_group4	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_12_RoutesInStaffLockoutLODK_group5	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_12_RoutesInStaffLockoutLODK_group6	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_13_RoutesInStaffLockoutLODKOverlap_group0	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_13_RoutesInStaffLockoutLODKOverlap_group1	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_13_RoutesInStaffLockoutLODKOverlap_group2	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_13_RoutesInStaffLockoutLODKOverlap_group3	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_13_RoutesInStaffLockoutLODKOverlap_group4	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_13_RoutesInStaffLockoutLODKOverlap_group5	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_13_RoutesInStaffLockoutLODKOverlap_group6	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_16_SignalReminder_group0	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_16_SignalReminder_group1	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_16_SignalReminder_group10	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_1_16_SignalReminder_group11	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_1_16_SignalReminder_group12	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_16_SignalReminder_group13	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_16_SignalReminder_group14	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_16_SignalReminder_group15	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_16_SignalReminder_group16	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_16_SignalReminder_group17	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_1_16_SignalReminder_group18	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_16_SignalReminder_group19	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_16_SignalReminder_group2	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_16_SignalReminder_group20	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_16_SignalReminder_group21	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_1_16_SignalReminder_group22	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_16_SignalReminder_group23	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_1_16_SignalReminder_group24	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_16_SignalReminder_group25	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_16_SignalReminder_group26	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_16_SignalReminder_group27	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_1_16_SignalReminder_group3	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_1_16_SignalReminder_group4	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_16_SignalReminder_group5	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_16_SignalReminder_group6	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_16_SignalReminder_group7	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_1_16_SignalReminder_group8	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_1_16_SignalReminder_group9	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_1_17_TechniciansRouteBar_group0	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_17_TechniciansRouteBar_group1	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_17_TechniciansRouteBar_group10	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_17_TechniciansRouteBar_group11	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_17_TechniciansRouteBar_group12	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_17_TechniciansRouteBar_group13	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_17_TechniciansRouteBar_group14	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_17_TechniciansRouteBar_group15	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_17_TechniciansRouteBar_group16	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_17_TechniciansRouteBar_group17	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_17_TechniciansRouteBar_group18	yes	yes	A	yes	A	yes	A

DesignX	DesignX_GSP_2_1_17_TechniciansRouteBar_group19	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_17_TechniciansRouteBar_group2	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_17_TechniciansRouteBar_group20	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_17_TechniciansRouteBar_group21	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_17_TechniciansRouteBar_group22	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_17_TechniciansRouteBar_group23	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_17_TechniciansRouteBar_group24	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_17_TechniciansRouteBar_group25	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_17_TechniciansRouteBar_group26	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_17_TechniciansRouteBar_group27	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_17_TechniciansRouteBar_group3	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_17_TechniciansRouteBar_group4	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_17_TechniciansRouteBar_group5	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_17_TechniciansRouteBar_group6	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_17_TechniciansRouteBar_group7	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_17_TechniciansRouteBar_group8	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_17_TechniciansRouteBar_group9	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_18_SignalGroupReplacement_group0	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_18_SignalGroupReplacement_group1	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_18_SignalGroupReplacement_group10	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_18_SignalGroupReplacement_group11	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_18_SignalGroupReplacement_group12	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_18_SignalGroupReplacement_group13	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_18_SignalGroupReplacement_group14	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_18_SignalGroupReplacement_group15	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_18_SignalGroupReplacement_group16	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_18_SignalGroupReplacement_group17	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_18_SignalGroupReplacement_group18	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_18_SignalGroupReplacement_group19	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_18_SignalGroupReplacement_group2	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_18_SignalGroupReplacement_group20	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_18_SignalGroupReplacement_group21	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_18_SignalGroupReplacement_group22	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_18_SignalGroupReplacement_group23	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_18_SignalGroupReplacement_group24	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_18_SignalGroupReplacement_group25	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_18_SignalGroupReplacement_group26	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_18_SignalGroupReplacement_group27	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_18_SignalGroupReplacement_group3	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_18_SignalGroupReplacement_group4	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_18_SignalGroupReplacement_group5	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_18_SignalGroupReplacement_group6	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_18_SignalGroupReplacement_group7	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_18_SignalGroupReplacement_group8	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_18_SignalGroupReplacement_group9	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_1_RoutesFromEntranceSignal_group0	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_1_RoutesFromEntranceSignal_group1	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_1_RoutesFromEntranceSignal_group10	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_1_RoutesFromEntranceSignal_group11	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_1_RoutesFromEntranceSignal_group12	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_1_RoutesFromEntranceSignal_group13	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_1_RoutesFromEntranceSignal_group14	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_1_RoutesFromEntranceSignal_group15	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_1_RoutesFromEntranceSignal_group16	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_1_RoutesFromEntranceSignal_group17	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_1_RoutesFromEntranceSignal_group18	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_1_RoutesFromEntranceSignal_group19	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_1_RoutesFromEntranceSignal_group2	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_1_RoutesFromEntranceSignal_group20	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_1_RoutesFromEntranceSignal_group3	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_1_RoutesFromEntranceSignal_group4	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_1_RoutesFromEntranceSignal_group5	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_1_RoutesFromEntranceSignal_group6	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_1_RoutesFromEntranceSignal_group7	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_1_RoutesFromEntranceSignal_group8	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_1_RoutesFromEntranceSignal_group9	no	yes	C	yes	CT	yes	CT

DesignX	DesignX_GSP_2_1_20_PermissiveRouteSettingExitSignal_group0	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_20_PermissiveRouteSettingExitSignal_group1	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_20_PermissiveRouteSettingExitSignal_group2	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_21_PermissiveRouteSettingOppMovements_group0	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_21_PermissiveRouteSettingOppMovements_group1	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_1_21_PermissiveRouteSettingOppMovements_group2	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_1_22_RoutesFromPermissiveExit_group0	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_22_RoutesFromPermissiveExit_group1	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_22_RoutesFromPermissiveExit_group2	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_22_RoutesFromPermissiveExit_group3	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_22_RoutesFromPermissiveExit_group4	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_23_TrackSectionsInLineOfRoute_group0	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_1_23_TrackSectionsInLineOfRoute_group1	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_1_23_TrackSectionsInLineOfRoute_group10	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_1_23_TrackSectionsInLineOfRoute_group11	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_1_23_TrackSectionsInLineOfRoute_group12	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_23_TrackSectionsInLineOfRoute_group13	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_1_23_TrackSectionsInLineOfRoute_group14	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_23_TrackSectionsInLineOfRoute_group15	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_23_TrackSectionsInLineOfRoute_group16	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_1_23_TrackSectionsInLineOfRoute_group17	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_1_23_TrackSectionsInLineOfRoute_group18	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_1_23_TrackSectionsInLineOfRoute_group19	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_23_TrackSectionsInLineOfRoute_group2	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_23_TrackSectionsInLineOfRoute_group20	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_1_23_TrackSectionsInLineOfRoute_group21	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_1_23_TrackSectionsInLineOfRoute_group22	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_23_TrackSectionsInLineOfRoute_group23	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_1_23_TrackSectionsInLineOfRoute_group24	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_23_TrackSectionsInLineOfRoute_group25	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_23_TrackSectionsInLineOfRoute_group26	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_23_TrackSectionsInLineOfRoute_group27	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_23_TrackSectionsInLineOfRoute_group3	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_23_TrackSectionsInLineOfRoute_group4	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_1_23_TrackSectionsInLineOfRoute_group5	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_1_23_TrackSectionsInLineOfRoute_group6	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_1_23_TrackSectionsInLineOfRoute_group7	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_1_23_TrackSectionsInLineOfRoute_group8	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_1_23_TrackSectionsInLineOfRoute_group9	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_1_24_OverlapSectionsInLineOfRoute_group0	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_24_OverlapSectionsInLineOfRoute_group1	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_24_OverlapSectionsInLineOfRoute_group10	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_24_OverlapSectionsInLineOfRoute_group11	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_24_OverlapSectionsInLineOfRoute_group12	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_24_OverlapSectionsInLineOfRoute_group13	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_24_OverlapSectionsInLineOfRoute_group2	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_24_OverlapSectionsInLineOfRoute_group3	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_24_OverlapSectionsInLineOfRoute_group4	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_24_OverlapSectionsInLineOfRoute_group5	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_24_OverlapSectionsInLineOfRoute_group6	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_24_OverlapSectionsInLineOfRoute_group7	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_24_OverlapSectionsInLineOfRoute_group8	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_1_24_OverlapSectionsInLineOfRoute_group9	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_1_2_OpposingRoutes_group0	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_2_OpposingRoutes_group1	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_2_OpposingRoutes_group10	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_2_OpposingRoutes_group11	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_2_OpposingRoutes_group12	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_2_OpposingRoutes_group13	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_2_OpposingRoutes_group14	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_2_OpposingRoutes_group15	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_2_OpposingRoutes_group16	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_2_OpposingRoutes_group17	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_2_OpposingRoutes_group18	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_2_OpposingRoutes_group19	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_2_OpposingRoutes_group2	no	yes	C	yes	CT	yes	CT

DesignX	DesignX_GSP_2_1_2_OpposingRoutes_group20	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_2_OpposingRoutes_group21	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_2_OpposingRoutes_group22	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_2_OpposingRoutes_group23	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_2_OpposingRoutes_group24	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_2_OpposingRoutes_group25	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_2_OpposingRoutes_group26	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_2_OpposingRoutes_group3	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_2_OpposingRoutes_group4	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_2_OpposingRoutes_group5	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_2_OpposingRoutes_group6	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_2_OpposingRoutes_group7	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_2_OpposingRoutes_group8	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_2_OpposingRoutes_group9	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_3_ConflictingRoutes_group0	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_3_ConflictingRoutes_group1	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_3_ConflictingRoutes_group10	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_3_ConflictingRoutes_group11	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_3_ConflictingRoutes_group12	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_3_ConflictingRoutes_group13	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_3_ConflictingRoutes_group14	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_3_ConflictingRoutes_group15	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_3_ConflictingRoutes_group16	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_3_ConflictingRoutes_group17	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_3_ConflictingRoutes_group18	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_3_ConflictingRoutes_group19	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_3_ConflictingRoutes_group2	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_3_ConflictingRoutes_group20	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_3_ConflictingRoutes_group21	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_3_ConflictingRoutes_group3	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_3_ConflictingRoutes_group4	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_3_ConflictingRoutes_group5	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_3_ConflictingRoutes_group6	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_3_ConflictingRoutes_group7	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_3_ConflictingRoutes_group8	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_3_ConflictingRoutes_group9	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_4_LastOpposingRouteSection_group0	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_1_4_LastOpposingRouteSection_group1	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_1_4_LastOpposingRouteSection_group10	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_4_LastOpposingRouteSection_group11	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_1_4_LastOpposingRouteSection_group12	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_4_LastOpposingRouteSection_group13	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_4_LastOpposingRouteSection_group14	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_1_4_LastOpposingRouteSection_group15	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_1_4_LastOpposingRouteSection_group16	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_4_LastOpposingRouteSection_group17	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_1_4_LastOpposingRouteSection_group18	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_1_4_LastOpposingRouteSection_group19	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_4_LastOpposingRouteSection_group2	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_4_LastOpposingRouteSection_group20	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_4_LastOpposingRouteSection_group21	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_4_LastOpposingRouteSection_group22	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_4_LastOpposingRouteSection_group23	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_4_LastOpposingRouteSection_group3	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_1_4_LastOpposingRouteSection_group4	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_1_4_LastOpposingRouteSection_group5	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_1_4_LastOpposingRouteSection_group6	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_1_4_LastOpposingRouteSection_group7	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_4_LastOpposingRouteSection_group8	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_1_4_LastOpposingRouteSection_group9	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_1_5_LastOpposingOverlapRouteSection_group0	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_1_5_LastOpposingOverlapRouteSection_group1	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_1_5_LastOpposingOverlapRouteSection_group2	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_1_5_LastOpposingOverlapRouteSection_group3	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_5_LastOpposingOverlapRouteSection_group4	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_5_LastOpposingOverlapRouteSection_group5	yes	yes	A	yes	A	yes	A

DesignX	DesignX_GSP_2_1_5_LastOpposingOverlapRouteSection_group6	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_1_5_LastOpposingOverlapRouteSection_group7	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_5_LastOpposingOverlapRouteSection_group8	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_6_PointsInTheRoute_group0	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_6_PointsInTheRoute_group1	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_6_PointsInTheRoute_group10	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_6_PointsInTheRoute_group11	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_6_PointsInTheRoute_group12	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_6_PointsInTheRoute_group13	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_6_PointsInTheRoute_group14	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_6_PointsInTheRoute_group15	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_6_PointsInTheRoute_group16	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_6_PointsInTheRoute_group17	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_6_PointsInTheRoute_group18	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_6_PointsInTheRoute_group19	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_6_PointsInTheRoute_group2	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_6_PointsInTheRoute_group20	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_6_PointsInTheRoute_group21	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_6_PointsInTheRoute_group3	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_6_PointsInTheRoute_group4	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_6_PointsInTheRoute_group5	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_6_PointsInTheRoute_group6	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_6_PointsInTheRoute_group7	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_6_PointsInTheRoute_group8	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_6_PointsInTheRoute_group9	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_8_PointsInTheOverlap_group0	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_8_PointsInTheOverlap_group1	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_8_PointsInTheOverlap_group2	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_1_8_PointsInTheOverlap_group3	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_12_ProceedMCBCCTVCrossing_group0	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_12_ProceedMCBCCTVCrossing_group1	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_12_ProceedMCBCCTVCrossing_group2	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_12_ProceedMCBCCTVCrossing_group3	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_16_ProceedWithExitSignal_group0	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_16_ProceedWithExitSignal_group1	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_16_ProceedWithExitSignal_group10	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_16_ProceedWithExitSignal_group11	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_16_ProceedWithExitSignal_group12	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_16_ProceedWithExitSignal_group13	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_16_ProceedWithExitSignal_group14	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_16_ProceedWithExitSignal_group15	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_16_ProceedWithExitSignal_group16	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_16_ProceedWithExitSignal_group2	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_16_ProceedWithExitSignal_group3	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_16_ProceedWithExitSignal_group4	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_16_ProceedWithExitSignal_group5	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_16_ProceedWithExitSignal_group6	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_16_ProceedWithExitSignal_group7	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_16_ProceedWithExitSignal_group8	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_16_ProceedWithExitSignal_group9	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_1_ProceedAspectWithRouteSet_group0	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_1_ProceedAspectWithRouteSet_group1	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_1_ProceedAspectWithRouteSet_group10	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_1_ProceedAspectWithRouteSet_group11	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_1_ProceedAspectWithRouteSet_group12	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_1_ProceedAspectWithRouteSet_group13	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_1_ProceedAspectWithRouteSet_group14	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_1_ProceedAspectWithRouteSet_group15	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_1_ProceedAspectWithRouteSet_group16	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_1_ProceedAspectWithRouteSet_group17	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_1_ProceedAspectWithRouteSet_group2	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_1_ProceedAspectWithRouteSet_group3	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_1_ProceedAspectWithRouteSet_group4	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_1_ProceedAspectWithRouteSet_group5	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_1_ProceedAspectWithRouteSet_group6	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_1_ProceedAspectWithRouteSet_group7	yes	yes	A	yes	A	yes	A

DesignX	DesignX_GSP_2_2_1_ProceedAspectWithRouteSet_group8	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_1_ProceedAspectWithRouteSet_group9	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_21_ProceedWithDisengagement_group0	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_21_ProceedWithDisengagement_group1	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_21_ProceedWithDisengagement_group10	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_21_ProceedWithDisengagement_group11	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_21_ProceedWithDisengagement_group12	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_21_ProceedWithDisengagement_group13	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_21_ProceedWithDisengagement_group14	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_21_ProceedWithDisengagement_group15	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_21_ProceedWithDisengagement_group16	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_21_ProceedWithDisengagement_group17	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_21_ProceedWithDisengagement_group2	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_21_ProceedWithDisengagement_group3	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_21_ProceedWithDisengagement_group4	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_21_ProceedWithDisengagement_group5	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_21_ProceedWithDisengagement_group6	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_21_ProceedWithDisengagement_group7	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_21_ProceedWithDisengagement_group8	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_21_ProceedWithDisengagement_group9	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_22_ProceedWithSGRC_group0	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_22_ProceedWithSGRC_group1	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_22_ProceedWithSGRC_group10	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_22_ProceedWithSGRC_group11	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_22_ProceedWithSGRC_group12	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_22_ProceedWithSGRC_group13	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_22_ProceedWithSGRC_group2	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_22_ProceedWithSGRC_group3	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_22_ProceedWithSGRC_group4	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_22_ProceedWithSGRC_group5	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_22_ProceedWithSGRC_group6	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_22_ProceedWithSGRC_group7	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_22_ProceedWithSGRC_group8	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_22_ProceedWithSGRC_group9	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_23_ProceedWithCSFC_group0	no	no	B	no	B	no	B
DesignX	DesignX_GSP_2_2_23_ProceedWithCSFC_group1	no	no	B	no	B	no	B
DesignX	DesignX_GSP_2_2_23_ProceedWithCSFC_group10	no	no	B	no	B	no	B
DesignX	DesignX_GSP_2_2_23_ProceedWithCSFC_group11	no	no	B	no	B	no	B
DesignX	DesignX_GSP_2_2_23_ProceedWithCSFC_group12	no	no	B	no	B	no	B
DesignX	DesignX_GSP_2_2_23_ProceedWithCSFC_group13	no	no	B	no	B	no	B
DesignX	DesignX_GSP_2_2_23_ProceedWithCSFC_group2	no	no	B	no	B	no	B
DesignX	DesignX_GSP_2_2_23_ProceedWithCSFC_group3	no	no	B	no	B	no	B
DesignX	DesignX_GSP_2_2_23_ProceedWithCSFC_group4	no	no	B	no	B	no	B
DesignX	DesignX_GSP_2_2_23_ProceedWithCSFC_group5	no	no	B	no	B	no	B
DesignX	DesignX_GSP_2_2_23_ProceedWithCSFC_group6	no	no	B	no	B	no	B
DesignX	DesignX_GSP_2_2_23_ProceedWithCSFC_group7	no	no	B	no	B	no	B
DesignX	DesignX_GSP_2_2_23_ProceedWithCSFC_group8	no	no	B	no	B	no	B
DesignX	DesignX_GSP_2_2_23_ProceedWithCSFC_group9	no	no	B	no	B	no	B
DesignX	DesignX_GSP_2_2_24_ProceedDuringEPR_group0	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_24_ProceedDuringEPR_group1	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_24_ProceedDuringEPR_group10	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_24_ProceedDuringEPR_group11	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_2_24_ProceedDuringEPR_group12	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_24_ProceedDuringEPR_group13	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_24_ProceedDuringEPR_group14	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_24_ProceedDuringEPR_group15	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_24_ProceedDuringEPR_group16	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_24_ProceedDuringEPR_group17	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_24_ProceedDuringEPR_group18	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_24_ProceedDuringEPR_group19	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_24_ProceedDuringEPR_group2	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_24_ProceedDuringEPR_group20	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_24_ProceedDuringEPR_group21	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_2_24_ProceedDuringEPR_group22	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_24_ProceedDuringEPR_group23	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_2_24_ProceedDuringEPR_group24	yes	yes	A	yes	A	yes	A

DesignX	DesignX_GSP_2_2_24_ProceedDuringEPR_group25	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_24_ProceedDuringEPR_group26	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_24_ProceedDuringEPR_group27	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_24_ProceedDuringEPR_group3	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_24_ProceedDuringEPR_group4	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_24_ProceedDuringEPR_group5	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_24_ProceedDuringEPR_group6	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_24_ProceedDuringEPR_group7	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_24_ProceedDuringEPR_group8	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_24_ProceedDuringEPR_group9	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_2_25_ProceedDuringSTR_group0	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_25_ProceedDuringSTR_group1	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_25_ProceedDuringSTR_group10	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_25_ProceedDuringSTR_group11	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_25_ProceedDuringSTR_group12	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_2_25_ProceedDuringSTR_group13	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_25_ProceedDuringSTR_group14	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_25_ProceedDuringSTR_group15	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_25_ProceedDuringSTR_group16	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_25_ProceedDuringSTR_group17	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_25_ProceedDuringSTR_group18	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_2_25_ProceedDuringSTR_group19	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_25_ProceedDuringSTR_group2	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_25_ProceedDuringSTR_group20	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_2_25_ProceedDuringSTR_group21	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_25_ProceedDuringSTR_group22	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_25_ProceedDuringSTR_group23	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_25_ProceedDuringSTR_group24	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_25_ProceedDuringSTR_group3	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_25_ProceedDuringSTR_group4	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_25_ProceedDuringSTR_group5	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_25_ProceedDuringSTR_group6	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_25_ProceedDuringSTR_group7	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_2_25_ProceedDuringSTR_group8	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_25_ProceedDuringSTR_group9	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_2_26_ProceedDuringSTRPermissive_group0	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_2_26_ProceedDuringSTRPermissive_group1	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_2_26_ProceedDuringSTRPermissive_group2	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_2_27_ProceedWithAspRestriction_group0	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_2_27_ProceedWithAspRestriction_group1	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_2_27_ProceedWithAspRestriction_group10	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_2_27_ProceedWithAspRestriction_group11	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_2_27_ProceedWithAspRestriction_group12	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_2_27_ProceedWithAspRestriction_group13	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_2_27_ProceedWithAspRestriction_group14	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_2_27_ProceedWithAspRestriction_group15	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_2_27_ProceedWithAspRestriction_group16	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_2_27_ProceedWithAspRestriction_group17	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_2_27_ProceedWithAspRestriction_group18	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_2_27_ProceedWithAspRestriction_group19	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_2_27_ProceedWithAspRestriction_group2	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_2_27_ProceedWithAspRestriction_group20	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_2_27_ProceedWithAspRestriction_group21	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_2_27_ProceedWithAspRestriction_group22	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_27_ProceedWithAspRestriction_group23	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_2_27_ProceedWithAspRestriction_group24	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_27_ProceedWithAspRestriction_group25	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_27_ProceedWithAspRestriction_group26	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_27_ProceedWithAspRestriction_group27	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_27_ProceedWithAspRestriction_group3	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_2_27_ProceedWithAspRestriction_group4	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_2_27_ProceedWithAspRestriction_group5	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_2_27_ProceedWithAspRestriction_group6	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_2_27_ProceedWithAspRestriction_group7	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_2_27_ProceedWithAspRestriction_group8	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_2_27_ProceedWithAspRestriction_group9	no	no	B	yes	D	yes	D

DesignX	DesignX_GSP_2_2_28_AspRestrictionReminderDevice_group0	yes	yes	A	no	E	no	E
DesignX	DesignX_GSP_2_2_28_AspRestrictionReminderDevice_group1	yes	yes	A	no	E	no	E
DesignX	DesignX_GSP_2_2_28_AspRestrictionReminderDevice_group10	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_28_AspRestrictionReminderDevice_group11	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_28_AspRestrictionReminderDevice_group12	yes	yes	A	no	E	no	E
DesignX	DesignX_GSP_2_2_28_AspRestrictionReminderDevice_group13	yes	yes	A	no	E	no	E
DesignX	DesignX_GSP_2_2_28_AspRestrictionReminderDevice_group2	yes	yes	A	no	E	no	E
DesignX	DesignX_GSP_2_2_28_AspRestrictionReminderDevice_group3	yes	yes	A	no	E	no	E
DesignX	DesignX_GSP_2_2_28_AspRestrictionReminderDevice_group4	yes	yes	A	no	E	no	E
DesignX	DesignX_GSP_2_2_28_AspRestrictionReminderDevice_group5	yes	yes	A	no	E	no	E
DesignX	DesignX_GSP_2_2_28_AspRestrictionReminderDevice_group6	yes	yes	A	no	E	no	E
DesignX	DesignX_GSP_2_2_28_AspRestrictionReminderDevice_group7	yes	yes	A	no	E	no	E
DesignX	DesignX_GSP_2_2_28_AspRestrictionReminderDevice_group8	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_28_AspRestrictionReminderDevice_group9	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_2_ProceedWithRouteSectionsLocked_group0	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_2_2_ProceedWithRouteSectionsLocked_group1	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_2_2_ProceedWithRouteSectionsLocked_group10	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_2_ProceedWithRouteSectionsLocked_group11	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_2_ProceedWithRouteSectionsLocked_group12	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_2_ProceedWithRouteSectionsLocked_group13	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_2_2_ProceedWithRouteSectionsLocked_group14	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_2_ProceedWithRouteSectionsLocked_group15	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_2_ProceedWithRouteSectionsLocked_group16	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_2_ProceedWithRouteSectionsLocked_group17	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_2_ProceedWithRouteSectionsLocked_group18	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_2_2_ProceedWithRouteSectionsLocked_group19	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_2_ProceedWithRouteSectionsLocked_group2	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_2_ProceedWithRouteSectionsLocked_group20	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_2_2_ProceedWithRouteSectionsLocked_group21	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_2_ProceedWithRouteSectionsLocked_group22	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_2_ProceedWithRouteSectionsLocked_group23	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_2_ProceedWithRouteSectionsLocked_group24	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_2_ProceedWithRouteSectionsLocked_group25	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_2_ProceedWithRouteSectionsLocked_group26	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_2_ProceedWithRouteSectionsLocked_group27	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_2_ProceedWithRouteSectionsLocked_group3	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_2_ProceedWithRouteSectionsLocked_group4	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_2_2_ProceedWithRouteSectionsLocked_group5	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_2_2_ProceedWithRouteSectionsLocked_group6	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_2_2_ProceedWithRouteSectionsLocked_group7	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_2_2_ProceedWithRouteSectionsLocked_group8	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_2_ProceedWithRouteSectionsLocked_group9	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_30_ProceedDisengagementAspRes_group0	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_30_ProceedDisengagementAspRes_group1	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_30_ProceedDisengagementAspRes_group10	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_30_ProceedDisengagementAspRes_group11	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_30_ProceedDisengagementAspRes_group12	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_30_ProceedDisengagementAspRes_group13	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_30_ProceedDisengagementAspRes_group14	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_30_ProceedDisengagementAspRes_group15	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_30_ProceedDisengagementAspRes_group16	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_30_ProceedDisengagementAspRes_group17	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_30_ProceedDisengagementAspRes_group18	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_30_ProceedDisengagementAspRes_group19	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_30_ProceedDisengagementAspRes_group2	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_30_ProceedDisengagementAspRes_group20	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_30_ProceedDisengagementAspRes_group21	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_30_ProceedDisengagementAspRes_group22	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_30_ProceedDisengagementAspRes_group23	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_30_ProceedDisengagementAspRes_group24	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_30_ProceedDisengagementAspRes_group25	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_30_ProceedDisengagementAspRes_group26	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_30_ProceedDisengagementAspRes_group27	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_30_ProceedDisengagementAspRes_group3	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_30_ProceedDisengagementAspRes_group4	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_30_ProceedDisengagementAspRes_group5	yes	yes	A	yes	A	yes	A

DesignX	DesignX_GSP_2_2_30_ProceedDisengagementAspRes_group6	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_30_ProceedDisengagementAspRes_group7	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_30_ProceedDisengagementAspRes_group8	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_30_ProceedDisengagementAspRes_group9	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_31_ProceedWithApproachLocking_group0	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_31_ProceedWithApproachLocking_group1	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_31_ProceedWithApproachLocking_group10	yes	yes	A	no	E	no	E
DesignX	DesignX_GSP_2_2_31_ProceedWithApproachLocking_group11	yes	yes	A	no	E	no	E
DesignX	DesignX_GSP_2_2_31_ProceedWithApproachLocking_group12	yes	yes	A	no	E	no	E
DesignX	DesignX_GSP_2_2_31_ProceedWithApproachLocking_group13	yes	yes	A	no	E	no	E
DesignX	DesignX_GSP_2_2_31_ProceedWithApproachLocking_group2	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_31_ProceedWithApproachLocking_group3	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_31_ProceedWithApproachLocking_group4	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_31_ProceedWithApproachLocking_group5	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_31_ProceedWithApproachLocking_group6	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_31_ProceedWithApproachLocking_group7	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_31_ProceedWithApproachLocking_group8	yes	yes	A	no	E	no	E
DesignX	DesignX_GSP_2_2_31_ProceedWithApproachLocking_group9	yes	yes	A	no	E	no	E
DesignX	DesignX_GSP_2_2_32_ProceedWithTPWSSuppressed_group0	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_32_ProceedWithTPWSSuppressed_group1	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_32_ProceedWithTPWSSuppressed_group10	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_32_ProceedWithTPWSSuppressed_group11	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_32_ProceedWithTPWSSuppressed_group12	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_32_ProceedWithTPWSSuppressed_group13	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_32_ProceedWithTPWSSuppressed_group2	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_32_ProceedWithTPWSSuppressed_group3	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_32_ProceedWithTPWSSuppressed_group4	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_32_ProceedWithTPWSSuppressed_group5	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_2_32_ProceedWithTPWSSuppressed_group6	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_32_ProceedWithTPWSSuppressed_group7	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_32_ProceedWithTPWSSuppressed_group8	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_32_ProceedWithTPWSSuppressed_group9	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_34_RYGAspectWithSOMOutput_group0	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_2_34_RYGAspectWithSOMOutput_group1	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_2_34_RYGAspectWithSOMOutput_group2	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_2_34_RYGAspectWithSOMOutput_group3	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_2_3_ProceedWithPointSetLockedDetected_group0	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_3_ProceedWithPointSetLockedDetected_group1	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_3_ProceedWithPointSetLockedDetected_group10	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_3_ProceedWithPointSetLockedDetected_group11	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_3_ProceedWithPointSetLockedDetected_group12	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_3_ProceedWithPointSetLockedDetected_group13	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_3_ProceedWithPointSetLockedDetected_group14	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_3_ProceedWithPointSetLockedDetected_group15	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_3_ProceedWithPointSetLockedDetected_group16	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_3_ProceedWithPointSetLockedDetected_group17	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_3_ProceedWithPointSetLockedDetected_group18	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_3_ProceedWithPointSetLockedDetected_group19	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_3_ProceedWithPointSetLockedDetected_group2	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_3_ProceedWithPointSetLockedDetected_group20	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_3_ProceedWithPointSetLockedDetected_group21	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_3_ProceedWithPointSetLockedDetected_group3	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_3_ProceedWithPointSetLockedDetected_group4	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_3_ProceedWithPointSetLockedDetected_group5	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_3_ProceedWithPointSetLockedDetected_group6	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_3_ProceedWithPointSetLockedDetected_group7	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_3_ProceedWithPointSetLockedDetected_group8	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_3_ProceedWithPointSetLockedDetected_group9	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_43_ProceedWithTechAspectDisable_group0	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_43_ProceedWithTechAspectDisable_group1	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_43_ProceedWithTechAspectDisable_group10	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_43_ProceedWithTechAspectDisable_group11	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_43_ProceedWithTechAspectDisable_group12	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_43_ProceedWithTechAspectDisable_group13	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_43_ProceedWithTechAspectDisable_group14	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_43_ProceedWithTechAspectDisable_group15	yes	yes	A	yes	A	yes	A

DesignX	DesignX_GSP_2_2_43_ProceedWithTechAspectDisable_group16	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_43_ProceedWithTechAspectDisable_group17	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_43_ProceedWithTechAspectDisable_group2	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_43_ProceedWithTechAspectDisable_group3	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_43_ProceedWithTechAspectDisable_group4	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_43_ProceedWithTechAspectDisable_group5	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_43_ProceedWithTechAspectDisable_group6	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_43_ProceedWithTechAspectDisable_group7	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_43_ProceedWithTechAspectDisable_group8	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_43_ProceedWithTechAspectDisable_group9	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_48_ProceedWithLODKRoute_group0	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_48_ProceedWithLODKRoute_group1	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_48_ProceedWithLODKRoute_group2	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_48_ProceedWithLODKRoute_group3	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_2_48_ProceedWithLODKRoute_group4	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_48_ProceedWithLODKRoute_group5	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_48_ProceedWithLODKRoute_group6	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_50_ProceedWithLODP_group0	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_2_51_ProceedWithSuppAWS_group0	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_2_51_ProceedWithSuppAWS_group1	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_2_51_ProceedWithSuppAWS_group10	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_2_51_ProceedWithSuppAWS_group11	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_2_51_ProceedWithSuppAWS_group12	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_2_51_ProceedWithSuppAWS_group13	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_2_51_ProceedWithSuppAWS_group14	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_2_51_ProceedWithSuppAWS_group15	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_2_51_ProceedWithSuppAWS_group16	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_2_51_ProceedWithSuppAWS_group2	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_2_51_ProceedWithSuppAWS_group3	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_2_51_ProceedWithSuppAWS_group4	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_2_51_ProceedWithSuppAWS_group5	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_2_51_ProceedWithSuppAWS_group6	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_2_51_ProceedWithSuppAWS_group7	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_2_51_ProceedWithSuppAWS_group8	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_2_51_ProceedWithSuppAWS_group9	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_2_52_ProceedWithPermissiveTrackProved_group0	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_2_52_ProceedWithPermissiveTrackProved_group1	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_2_52_ProceedWithPermissiveTrackProved_group2	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_2_53_ProceedWithExitRestrictive_group0	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_53_ProceedWithExitRestrictive_group1	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_53_ProceedWithExitRestrictive_group2	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_54_ProceedWithTPWSExit_group0	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_54_ProceedWithTPWSExit_group1	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_54_ProceedWithTPWSExit_group10	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_54_ProceedWithTPWSExit_group11	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_54_ProceedWithTPWSExit_group12	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_54_ProceedWithTPWSExit_group13	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_54_ProceedWithTPWSExit_group14	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_54_ProceedWithTPWSExit_group15	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_54_ProceedWithTPWSExit_group16	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_54_ProceedWithTPWSExit_group2	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_54_ProceedWithTPWSExit_group3	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_54_ProceedWithTPWSExit_group4	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_54_ProceedWithTPWSExit_group5	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_54_ProceedWithTPWSExit_group6	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_54_ProceedWithTPWSExit_group7	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_54_ProceedWithTPWSExit_group8	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_54_ProceedWithTPWSExit_group9	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_55_SignalDisengagement_group0	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_55_SignalDisengagement_group1	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_55_SignalDisengagement_group10	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_55_SignalDisengagement_group11	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_55_SignalDisengagement_group12	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_55_SignalDisengagement_group13	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_55_SignalDisengagement_group14	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_55_SignalDisengagement_group15	yes	yes	A	yes	A	yes	A

DesignX	DesignX_GSP_2_2_55_SignalDisengagement_group16	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_55_SignalDisengagement_group17	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_55_SignalDisengagement_group18	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_55_SignalDisengagement_group19	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_55_SignalDisengagement_group2	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_55_SignalDisengagement_group20	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_55_SignalDisengagement_group21	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_55_SignalDisengagement_group22	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_55_SignalDisengagement_group23	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_55_SignalDisengagement_group24	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_55_SignalDisengagement_group25	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_55_SignalDisengagement_group26	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_55_SignalDisengagement_group27	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_55_SignalDisengagement_group3	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_55_SignalDisengagement_group4	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_55_SignalDisengagement_group5	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_55_SignalDisengagement_group6	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_55_SignalDisengagement_group7	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_55_SignalDisengagement_group8	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_55_SignalDisengagement_group9	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_5_ProceedWithPointSLDOverlap_group0	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_5_ProceedWithPointSLDOverlap_group1	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_5_ProceedWithPointSLDOverlap_group2	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_5_ProceedWithPointSLDOverlap_group3	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_6_ProceedWithTracksClearPermissive_group0	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_6_ProceedWithTracksClearPermissive_group1	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_6_ProceedWithTracksClearPermissive_group2	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_7_ProceedWithTracksClear_group0	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_7_ProceedWithTracksClear_group1	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_7_ProceedWithTracksClear_group10	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_7_ProceedWithTracksClear_group11	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_7_ProceedWithTracksClear_group12	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_2_7_ProceedWithTracksClear_group13	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_7_ProceedWithTracksClear_group14	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_7_ProceedWithTracksClear_group15	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_7_ProceedWithTracksClear_group16	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_7_ProceedWithTracksClear_group17	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_7_ProceedWithTracksClear_group18	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_2_7_ProceedWithTracksClear_group19	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_7_ProceedWithTracksClear_group2	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_7_ProceedWithTracksClear_group20	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_2_7_ProceedWithTracksClear_group21	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_7_ProceedWithTracksClear_group22	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_7_ProceedWithTracksClear_group23	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_7_ProceedWithTracksClear_group24	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_7_ProceedWithTracksClear_group3	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_7_ProceedWithTracksClear_group4	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_7_ProceedWithTracksClear_group5	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_7_ProceedWithTracksClear_group6	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_7_ProceedWithTracksClear_group7	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_2_7_ProceedWithTracksClear_group8	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_7_ProceedWithTracksClear_group9	no	no	B	yes	D	yes	D
DesignX	DesignX_GSP_2_2_8_ProceedWithOverlapTracksClear_group0	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_8_ProceedWithOverlapTracksClear_group1	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_8_ProceedWithOverlapTracksClear_group10	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_8_ProceedWithOverlapTracksClear_group11	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_8_ProceedWithOverlapTracksClear_group12	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_8_ProceedWithOverlapTracksClear_group13	yes	yes	A	yes	A	yes	A
DesignX	DesignX_GSP_2_2_8_ProceedWithOverlapTracksClear_group2	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_8_ProceedWithOverlapTracksClear_group3	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_8_ProceedWithOverlapTracksClear_group4	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_8_ProceedWithOverlapTracksClear_group5	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_8_ProceedWithOverlapTracksClear_group6	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_8_ProceedWithOverlapTracksClear_group7	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_8_ProceedWithOverlapTracksClear_group8	no	yes	C	yes	CT	yes	CT
DesignX	DesignX_GSP_2_2_8_ProceedWithOverlapTracksClear_group9	no	yes	C	yes	CT	yes	CT

A	A_additional_A_A_GSP_ConflictingRoutes_group0	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_ConflictingRoutes_group1	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_ConflictingRoutes_group10	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group100	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group101	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group102	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group103	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group104	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group105	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group106	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group107	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group108	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group109	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group11	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group110	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group111	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group112	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group113	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group114	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group115	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_ConflictingRoutes_group116	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_ConflictingRoutes_group117	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_ConflictingRoutes_group118	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_ConflictingRoutes_group119	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_ConflictingRoutes_group12	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group120	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group121	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group122	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group123	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group124	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group125	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group126	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group127	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group128	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group129	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group13	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group130	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group131	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group132	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group133	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group134	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group135	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group136	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group137	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group138	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group139	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group14	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group140	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group141	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group142	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group143	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group144	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group145	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group146	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group147	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group148	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group149	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group15	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group150	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group151	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group152	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group153	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group154	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group155	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group156	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group157	no	yes	C	yes	CT	yes	CT

A	A_additional_A_A_GSP_ConflictingRoutes_group158	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group159	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group16	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group160	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group161	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group162	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group163	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group164	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group165	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group166	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group167	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group168	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group169	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group17	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group170	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group171	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group172	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group173	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group174	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group175	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group176	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group177	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group178	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group179	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group18	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group180	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group181	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group182	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group183	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group184	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group185	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group186	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group187	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group188	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group189	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group19	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group190	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group191	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group192	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group193	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group194	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group195	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group196	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group197	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group198	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group199	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group2	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_ConflictingRoutes_group20	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group200	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group201	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group202	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group203	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group204	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group205	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group206	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group207	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group208	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group209	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group21	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group210	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group211	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group212	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group213	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group214	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group215	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group216	no	yes	C	yes	CT	yes	CT

A	A_additional_A_A_GSP_ConflictingRoutes_group217	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group218	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group219	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group22	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group220	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group221	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group222	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group223	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group224	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group225	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group226	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group227	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group228	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group229	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group23	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_ConflictingRoutes_group230	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group231	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group232	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group233	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group234	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group235	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group236	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group237	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group238	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group239	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group24	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_ConflictingRoutes_group240	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group241	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group242	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group243	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group244	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group245	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group246	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group247	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group248	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group249	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group25	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_ConflictingRoutes_group250	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group251	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group252	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group253	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group254	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group255	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group256	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group257	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group258	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group259	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group26	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_ConflictingRoutes_group260	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group261	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group262	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group263	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group264	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group265	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group266	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group267	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group268	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group269	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group27	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_ConflictingRoutes_group270	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group271	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group272	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group273	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group274	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group275	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group276	no	yes	C	yes	CT	yes	CT

A	A_additional_A_A_GSP_ConflictingRoutes_group277	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group278	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group279	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group28	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group280	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group281	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group282	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group283	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group284	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group285	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group286	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group287	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group288	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group289	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group29	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group290	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group291	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group292	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group293	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group294	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group295	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group296	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group297	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group298	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group299	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group3	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_ConflictingRoutes_group30	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group300	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group301	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group302	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group303	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group304	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group305	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group306	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group307	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group308	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group309	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group31	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group310	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group311	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group312	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group313	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group314	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group315	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group316	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group317	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group318	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group319	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group32	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group320	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group321	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group322	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group323	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group324	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group325	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group326	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group327	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group328	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group329	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group33	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group330	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group331	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group332	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group333	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group334	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group335	no	yes	C	yes	CT	yes	CT

A	A_additional_A_A_GSP_ConflictingRoutes_group336	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group337	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group338	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group339	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group34	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group340	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group341	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group342	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group343	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group344	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group345	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group346	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group347	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group348	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group349	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group35	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group350	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group351	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group352	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group353	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group354	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group355	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group356	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group357	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group358	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group359	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group36	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group360	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group361	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group362	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group363	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group364	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group365	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group366	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group367	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group368	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group369	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group37	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group370	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group371	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group372	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group373	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group374	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group375	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group376	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group377	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group378	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group379	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group38	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group380	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group381	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group382	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group383	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group384	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group385	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group386	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group387	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group388	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group389	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group39	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group390	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group391	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group392	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group393	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group394	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group395	no	yes	C	yes	CT	yes	CT

A	A_additional_A_A_GSP_ConflictingRoutes_group396	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group397	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group398	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group399	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group4	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_ConflictingRoutes_group40	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group400	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group401	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group402	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group403	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group404	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group405	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group406	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group407	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group408	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group409	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group41	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group410	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group411	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group412	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group413	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group414	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group415	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group416	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group417	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group418	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group419	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group42	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group420	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group421	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group422	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group423	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group424	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group425	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group426	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group427	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group428	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group429	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group43	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group430	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group431	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group432	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_ConflictingRoutes_group433	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_ConflictingRoutes_group434	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_ConflictingRoutes_group435	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group436	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group437	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group438	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group439	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group44	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group440	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group441	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group442	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group443	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group444	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group445	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group446	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group447	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group448	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group449	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group45	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group450	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group451	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group452	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group453	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group454	no	yes	C	yes	CT	yes	CT

A	A_additional_A_A_GSP_ConflictingRoutes_group455	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_ConflictingRoutes_group456	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group457	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_ConflictingRoutes_group458	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group459	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group46	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_ConflictingRoutes_group460	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group461	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group462	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group463	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group464	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group465	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group466	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group467	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group468	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group469	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group47	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_ConflictingRoutes_group470	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group471	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group472	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group473	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group474	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group475	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group476	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group477	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group478	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_ConflictingRoutes_group479	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group48	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_ConflictingRoutes_group480	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_ConflictingRoutes_group481	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group482	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group483	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group484	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group485	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group486	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group487	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group488	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group489	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group49	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_ConflictingRoutes_group490	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group491	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group492	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group493	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group494	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group495	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group496	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group497	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group498	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group499	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group5	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group50	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_ConflictingRoutes_group500	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group501	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_ConflictingRoutes_group502	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_ConflictingRoutes_group503	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_ConflictingRoutes_group504	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group505	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group506	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_ConflictingRoutes_group507	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_ConflictingRoutes_group508	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group509	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group51	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group510	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group511	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group512	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group513	no	yes	C	yes	CT	yes	CT

A	A_additional_A_A_GSP_ConflictingRoutes_group514	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group515	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group516	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group517	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group518	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group519	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group52	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group520	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group521	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group522	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group523	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group524	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group525	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group526	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group527	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group528	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group529	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group53	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group530	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_ConflictingRoutes_group531	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group532	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group533	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group534	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group535	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group536	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group537	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group538	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group539	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group54	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group540	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group541	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group542	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group543	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group544	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group545	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group546	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group547	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group548	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group549	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group55	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group550	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group551	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group552	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group553	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_ConflictingRoutes_group554	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_ConflictingRoutes_group555	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group556	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group557	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group558	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group559	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group56	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group560	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_ConflictingRoutes_group561	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group562	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group563	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group564	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_ConflictingRoutes_group565	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group566	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group567	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group568	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group569	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group57	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group570	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_ConflictingRoutes_group571	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_ConflictingRoutes_group572	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group573	no	yes	C	yes	CT	yes	CT

A	A_additional_A_A_GSP_ConflictingRoutes_group574	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group575	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_ConflictingRoutes_group58	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group59	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group6	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group60	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group61	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group62	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group63	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group64	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group65	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group66	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group67	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group68	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group69	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_ConflictingRoutes_group7	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group70	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_ConflictingRoutes_group71	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_ConflictingRoutes_group72	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_ConflictingRoutes_group73	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_ConflictingRoutes_group74	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group75	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group76	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group77	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group78	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group79	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group8	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group80	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group81	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group82	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group83	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group84	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group85	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group86	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group87	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group88	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group89	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group9	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group90	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group91	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group92	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_ConflictingRoutes_group93	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_ConflictingRoutes_group94	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_ConflictingRoutes_group95	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_ConflictingRoutes_group96	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_ConflictingRoutes_group97	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group98	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group99	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_Fast_group0	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_ConflictingRoutes_Fast_group1	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_ConflictingRoutes_Fast_group10	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_Fast_group11	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_Fast_group12	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_Fast_group13	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_Fast_group14	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_Fast_group15	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_Fast_group16	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_Fast_group17	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_Fast_group18	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_ConflictingRoutes_Fast_group19	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_ConflictingRoutes_Fast_group2	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_ConflictingRoutes_Fast_group20	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_ConflictingRoutes_Fast_group21	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_ConflictingRoutes_Fast_group22	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_ConflictingRoutes_Fast_group23	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_ConflictingRoutes_Fast_group24	no	no	B	yes	D	yes	D

A	A_additional_A_A_GSP_ConflictingRoutes_Fast_group25	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_ConflictingRoutes_Fast_group26	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_Fast_group27	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_ConflictingRoutes_Fast_group28	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_ConflictingRoutes_Fast_group29	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_ConflictingRoutes_Fast_group3	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_ConflictingRoutes_Fast_group30	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_ConflictingRoutes_Fast_group4	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_ConflictingRoutes_Fast_group5	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_ConflictingRoutes_Fast_group6	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_Fast_group7	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_Fast_group8	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_ConflictingRoutes_Fast_group9	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_FirstOverlapWithPreviousRouteSectionRel_Fast_group0	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_FirstOverlapWithPreviousRouteSectionRel_Fast_group1	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group0	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group1	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group10	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group11	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group12	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group13	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group14	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group15	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group16	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group17	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group18	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group19	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group2	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group20	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group21	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group22	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group23	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group24	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group25	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group26	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group27	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group28	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group29	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group3	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group30	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group31	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group32	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group33	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group34	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group35	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group36	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group37	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group38	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group39	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group4	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group40	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group41	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group42	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group43	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group44	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group5	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group6	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group7	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group8	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group9	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_OpposingRoutes_Fast_group0	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_OpposingRoutes_Fast_group1	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_OpposingRoutes_Fast_group10	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_OpposingRoutes_Fast_group11	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_OpposingRoutes_Fast_group12	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_OpposingRoutes_Fast_group13	no	yes	C	yes	CT	yes	CT

A	A_additional_A_A_GSP_OpposingRoutes_Fast_group14	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_OpposingRoutes_Fast_group15	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_OpposingRoutes_Fast_group16	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_OpposingRoutes_Fast_group17	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_OpposingRoutes_Fast_group18	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_OpposingRoutes_Fast_group19	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_OpposingRoutes_Fast_group2	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_OpposingRoutes_Fast_group20	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_OpposingRoutes_Fast_group21	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_OpposingRoutes_Fast_group22	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_OpposingRoutes_Fast_group23	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_OpposingRoutes_Fast_group24	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_OpposingRoutes_Fast_group25	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_OpposingRoutes_Fast_group26	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_OpposingRoutes_Fast_group27	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_OpposingRoutes_Fast_group28	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_OpposingRoutes_Fast_group29	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_OpposingRoutes_Fast_group3	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_OpposingRoutes_Fast_group30	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_OpposingRoutes_Fast_group4	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_OpposingRoutes_Fast_group5	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_OpposingRoutes_Fast_group6	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_OpposingRoutes_Fast_group7	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_OpposingRoutes_Fast_group8	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_OpposingRoutes_Fast_group9	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_OverlapSectionsInLineOfRoute_group0	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_OverlapSectionsInLineOfRoute_group1	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_PointControlWhenFreeToMove_group0	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_PointControlWhenFreeToMove_group1	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_PointControlWhenFreeToMove_group2	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_PointControlWhenFreeToMove_group3	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_PointControlWhenFreeToMove_group4	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_PointControlWhenFreeToMove_group5	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_PointControlWhenFreeToMove_group6	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_PointControlWhenFreeToMove_group7	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_PointControlWhenKeyed_group0	yes	yes	A	no	E	no	E
A	A_additional_A_A_GSP_PointControlWhenKeyed_group1	yes	yes	A	no	E	no	E
A	A_additional_A_A_GSP_PointControlWhenKeyed_group2	yes	yes	A	no	E	no	E
A	A_additional_A_A_GSP_PointControlWhenKeyed_group3	yes	yes	A	no	E	no	E
A	A_additional_A_A_GSP_PointControlWhenKeyed_group4	yes	yes	A	no	E	no	E
A	A_additional_A_A_GSP_PointControlWhenKeyed_group5	yes	yes	A	no	E	no	E
A	A_additional_A_A_GSP_PointControlWhenKeyed_group6	yes	yes	A	no	E	no	E
A	A_additional_A_A_GSP_PointControlWhenKeyed_group7	yes	yes	A	no	E	no	E
A	A_additional_A_A_GSP_PointControlWithSectionsOverPoints_group0	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_PointControlWithSectionsOverPoints_group1	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_PointControlWithSectionsOverPoints_group10	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_PointControlWithSectionsOverPoints_group11	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_PointControlWithSectionsOverPoints_group12	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_PointControlWithSectionsOverPoints_group13	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_PointControlWithSectionsOverPoints_group14	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_PointControlWithSectionsOverPoints_group15	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_PointControlWithSectionsOverPoints_group16	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_PointControlWithSectionsOverPoints_group17	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_PointControlWithSectionsOverPoints_group18	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_PointControlWithSectionsOverPoints_group19	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_PointControlWithSectionsOverPoints_group2	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_PointControlWithSectionsOverPoints_group20	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_PointControlWithSectionsOverPoints_group21	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_PointControlWithSectionsOverPoints_group22	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_PointControlWithSectionsOverPoints_group23	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_PointControlWithSectionsOverPoints_group24	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_PointControlWithSectionsOverPoints_group25	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_PointControlWithSectionsOverPoints_group26	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_PointControlWithSectionsOverPoints_group27	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_PointControlWithSectionsOverPoints_group28	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_PointControlWithSectionsOverPoints_group29	yes	yes	A	yes	A	yes	A

A	A_additional_A_A_GSP_PointControlWithSectionsOverPoints_group3	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_PointControlWithSectionsOverPoints_group4	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_PointControlWithSectionsOverPoints_group5	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_PointControlWithSectionsOverPoints_group6	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_PointControlWithSectionsOverPoints_group7	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_PointControlWithSectionsOverPoints_group8	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_PointControlWithSectionsOverPoints_group9	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_PointControlWithTechniciansControls_group0	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_PointControlWithTechniciansControls_group1	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_PointControlWithTechniciansControls_group2	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_PointControlWithTechniciansControls_group3	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_PointControlWithTechniciansControls_group4	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_PointControlWithTechniciansControls_group5	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_PointControlWithTechniciansControls_group6	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_PointControlWithTechniciansControls_group7	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_PointsCalledNormal_group0	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_PointsCalledNormal_group1	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_PointsCalledNormal_group2	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_PointsCalledNormal_group3	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_PointsCalledNormal_group4	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_PointsCalledNormal_group5	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_PointsCalledNormal_group6	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_PointsCalledNormal_group7	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_PointsCalledNormal_group8	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_PointsCalledReverse_group0	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_PointsCalledReverse_group1	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_PointsCalledReverse_group2	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_PointsCalledReverse_group3	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_PointsCalledReverse_group4	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_PointsCalledReverse_group5	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_PointsCalledReverse_group6	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_PointsCalledReverse_group7	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_PointsCalledReverse_group8	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_ProceedWithExitSignal_group0	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_ProceedWithExitSignal_group1	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_ProceedWithExitSignal_group2	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_ProceedWithExitSignal_group3	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_ProceedWithExitSignal_group4	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_ProceedWithExitSignal_group5	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_ProceedWithExitSignal_group6	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_RouteSectionLockedAheadOfTrain_group0	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_RouteSectionLockedAheadOfTrain_group1	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_RouteSectionLockedAheadOfTrain_group10	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_RouteSectionLockedAheadOfTrain_group11	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_RouteSectionLockedAheadOfTrain_group12	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_RouteSectionLockedAheadOfTrain_group13	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_RouteSectionLockedAheadOfTrain_group14	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_RouteSectionLockedAheadOfTrain_group15	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_RouteSectionLockedAheadOfTrain_group16	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_RouteSectionLockedAheadOfTrain_group17	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_RouteSectionLockedAheadOfTrain_group18	yes	yes	A	no	E	no	E
A	A_additional_A_A_GSP_RouteSectionLockedAheadOfTrain_group19	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_RouteSectionLockedAheadOfTrain_group2	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_RouteSectionLockedAheadOfTrain_group20	yes	yes	A	no	E	no	E
A	A_additional_A_A_GSP_RouteSectionLockedAheadOfTrain_group21	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_RouteSectionLockedAheadOfTrain_group22	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_RouteSectionLockedAheadOfTrain_group23	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_RouteSectionLockedAheadOfTrain_group24	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_RouteSectionLockedAheadOfTrain_group3	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_RouteSectionLockedAheadOfTrain_group4	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_RouteSectionLockedAheadOfTrain_group5	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_RouteSectionLockedAheadOfTrain_group6	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_RouteSectionLockedAheadOfTrain_group7	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_RouteSectionLockedAheadOfTrain_group8	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_RouteSectionLockedAheadOfTrain_group9	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_RouteSectionLockedOverPoints_group0	no	yes	C	yes	CT	yes	CT

A	A_additional_A_A_GSP_RouteSectionLockedOverPoints_group1	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_RouteSectionLockedOverPoints_group10	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_RouteSectionLockedOverPoints_group11	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_RouteSectionLockedOverPoints_group12	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_RouteSectionLockedOverPoints_group13	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RouteSectionLockedOverPoints_group14	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RouteSectionLockedOverPoints_group15	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RouteSectionLockedOverPoints_group16	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_RouteSectionLockedOverPoints_group17	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_RouteSectionLockedOverPoints_group18	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RouteSectionLockedOverPoints_group19	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_RouteSectionLockedOverPoints_group2	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RouteSectionLockedOverPoints_group20	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_RouteSectionLockedOverPoints_group21	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RouteSectionLockedOverPoints_group22	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RouteSectionLockedOverPoints_group23	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RouteSectionLockedOverPoints_group24	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RouteSectionLockedOverPoints_group25	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_RouteSectionLockedOverPoints_group26	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RouteSectionLockedOverPoints_group27	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RouteSectionLockedOverPoints_group28	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_RouteSectionLockedOverPoints_group29	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_RouteSectionLockedOverPoints_group3	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_RouteSectionLockedOverPoints_group4	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_RouteSectionLockedOverPoints_group5	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_RouteSectionLockedOverPoints_group6	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_RouteSectionLockedOverPoints_group7	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_RouteSectionLockedOverPoints_group8	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_RouteSectionLockedOverPoints_group9	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group0	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group1	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group10	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group100	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group101	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group102	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group103	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group104	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group105	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group106	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group107	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group108	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group109	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group11	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group110	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group111	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group112	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group113	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group114	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group115	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group116	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group117	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group118	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group119	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group12	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group120	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group121	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group122	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group123	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group124	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group125	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group126	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group127	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group128	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group129	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group13	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group130	no	yes	C	yes	CT	yes	CT

A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group131	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group132	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group133	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group134	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group135	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group136	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group137	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group138	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group139	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group14	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group140	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group141	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group142	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group143	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group144	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group145	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group146	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group147	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group148	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group149	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group15	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group150	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group151	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group152	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group153	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group154	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group155	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group156	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group157	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group158	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group159	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group16	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group160	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group161	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group162	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group163	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group164	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group165	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group166	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group167	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group168	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group169	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group17	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group170	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group171	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group172	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group173	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group174	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group175	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group176	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group177	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group178	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group179	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group18	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group180	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group181	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group182	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group183	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group19	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group2	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group20	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group21	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group22	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group23	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group24	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group25	no	no	B	yes	D	yes	D

A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group26	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group27	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group28	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group29	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group3	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group30	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group31	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group32	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group33	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group34	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group35	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group36	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group37	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group38	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group39	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group4	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group40	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group41	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group42	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group43	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group44	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group45	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group46	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group47	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group48	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group49	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group5	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group50	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group51	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group52	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group53	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group54	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group55	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group56	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group57	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group58	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group59	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group6	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group60	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group61	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group62	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group63	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group64	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group65	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group66	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group67	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group68	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group69	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group7	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group70	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group71	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group72	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group73	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group74	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group75	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group76	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group77	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group78	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group79	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group8	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group80	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group81	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group82	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group83	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group84	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group85	no	yes	C	yes	CT	yes	CT

A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group86	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group87	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group88	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group89	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group9	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group90	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group91	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group92	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group93	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group94	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group95	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group96	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group97	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group98	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group99	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_SignalGroupReplacement_group0	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SignalGroupReplacement_group1	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SignalGroupReplacement_group10	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SignalGroupReplacement_group11	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SignalGroupReplacement_group12	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SignalGroupReplacement_group13	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SignalGroupReplacement_group14	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SignalGroupReplacement_group15	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SignalGroupReplacement_group16	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SignalGroupReplacement_group17	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SignalGroupReplacement_group18	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SignalGroupReplacement_group19	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SignalGroupReplacement_group2	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SignalGroupReplacement_group20	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SignalGroupReplacement_group21	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SignalGroupReplacement_group22	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SignalGroupReplacement_group23	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SignalGroupReplacement_group24	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SignalGroupReplacement_group25	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SignalGroupReplacement_group26	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SignalGroupReplacement_group27	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SignalGroupReplacement_group28	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SignalGroupReplacement_group29	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SignalGroupReplacement_group3	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SignalGroupReplacement_group30	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SignalGroupReplacement_group31	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SignalGroupReplacement_group32	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SignalGroupReplacement_group33	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SignalGroupReplacement_group34	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SignalGroupReplacement_group35	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SignalGroupReplacement_group4	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SignalGroupReplacement_group5	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SignalGroupReplacement_group6	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SignalGroupReplacement_group7	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SignalGroupReplacement_group8	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SignalGroupReplacement_group9	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SignalReminder_group0	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SignalReminder_group1	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SignalReminder_group10	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SignalReminder_group11	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SignalReminder_group12	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SignalReminder_group13	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SignalReminder_group14	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SignalReminder_group15	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SignalReminder_group16	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SignalReminder_group17	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SignalReminder_group18	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_SignalReminder_group19	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SignalReminder_group2	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SignalReminder_group20	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SignalReminder_group21	yes	yes	A	yes	A	yes	A

A	A_additional_A_A_GSP_SignalReminder_group22	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SignalReminder_group23	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SignalReminder_group24	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SignalReminder_group25	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SignalReminder_group26	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SignalReminder_group27	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SignalReminder_group28	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SignalReminder_group29	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SignalReminder_group3	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SignalReminder_group30	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SignalReminder_group31	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SignalReminder_group32	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SignalReminder_group33	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SignalReminder_group34	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SignalReminder_group35	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SignalReminder_group36	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SignalReminder_group4	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SignalReminder_group5	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SignalReminder_group6	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SignalReminder_group7	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SignalReminder_group8	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SignalReminder_group9	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_group0	no	no	B	no	B	no	B
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_group1	yes	yes	A	no	E	no	E
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_group10	no	no	B	no	B	no	B
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_group11	no	no	B	no	B	no	B
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_group12	no	no	B	no	B	no	B
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_group13	no	no	B	no	B	no	B
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_group14	no	no	B	no	B	no	B
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_group15	no	no	B	no	B	no	B
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_group16	no	no	B	no	B	no	B
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_group17	no	no	B	no	B	no	B
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_group18	no	no	B	no	B	no	B
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_group19	no	no	B	no	B	no	B
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_group2	no	no	B	no	B	no	B
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_group20	no	no	B	no	B	no	B
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_group21	no	no	B	no	B	no	B
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_group22	no	no	B	no	B	no	B
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_group23	no	no	B	no	B	no	B
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_group24	no	no	B	no	B	no	B
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_group3	no	no	B	no	B	no	B
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_group4	no	no	B	no	B	no	B
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_group5	no	no	B	no	B	no	B
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_group6	no	no	B	no	B	no	B
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_group7	no	no	B	no	B	no	B
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_group8	no	no	B	no	B	no	B
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_group9	no	no	B	no	B	no	B
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_Fast_group0	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_Fast_group1	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_Fast_group10	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_Fast_group11	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_Fast_group12	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_Fast_group13	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_Fast_group14	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_Fast_group15	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_Fast_group16	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_Fast_group17	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_Fast_group18	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_Fast_group19	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_Fast_group2	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_Fast_group20	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_Fast_group21	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_Fast_group22	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_Fast_group23	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_Fast_group24	yes	yes	A	no	E	no	E
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_Fast_group25	yes	yes	A	yes	A	yes	A

A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_Fast_group26	yes	yes	A	no	E	no	E
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_Fast_group27	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_Fast_group28	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_Fast_group29	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_Fast_group3	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_Fast_group30	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_Fast_group31	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_Fast_group32	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_Fast_group33	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_Fast_group34	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_Fast_group35	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_Fast_group36	yes	yes	A	no	E	no	E
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_Fast_group4	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_Fast_group5	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_Fast_group6	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_Fast_group7	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_Fast_group8	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_Fast_group9	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group0	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group1	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group10	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group100	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group101	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group102	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group103	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group104	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group105	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group106	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group107	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group108	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group109	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group11	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group110	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group111	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group112	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group113	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group114	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group115	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group116	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group117	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group118	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group119	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group12	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group120	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group121	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group122	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group123	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group124	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group125	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group126	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group127	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group128	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group129	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group13	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group130	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group131	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group132	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group14	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group15	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group16	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group17	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group18	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group19	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group2	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group20	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group21	no	no	B	yes	D	yes	D

A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group22	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group23	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group24	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group25	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group26	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group27	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group28	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group29	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group3	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group30	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group31	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group32	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group33	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group34	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group35	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group36	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group37	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group38	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group39	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group4	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group40	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group41	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group42	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group43	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group44	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group45	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group46	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group47	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group48	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group49	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group5	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group50	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group51	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group52	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group53	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group54	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group55	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group56	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group57	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group58	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group59	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group6	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group60	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group61	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group62	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group63	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group64	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group65	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group66	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group67	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group68	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group69	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group7	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group70	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group71	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group72	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group73	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group74	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group75	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group76	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group77	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group78	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group79	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group8	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group80	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group81	yes	yes	A	yes	A	yes	A

A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group82	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group83	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group84	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group85	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group86	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group87	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group88	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group89	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group9	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group90	no	yes	C	yes	CT	yes	CT
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group91	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group92	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group93	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group94	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group95	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group96	yes	yes	A	yes	A	yes	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group97	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group98	no	no	B	yes	D	yes	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group99	yes	yes	A	yes	A	yes	A
B	B_additional_B_B_GSP_ConflictingRoutes_group0	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group10	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group100	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group101	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group102	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group103	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group104	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group105	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group106	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group107	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group108	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group109	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group11	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group110	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group111	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group112	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group113	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group114	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group115	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group116	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group117	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group118	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group119	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group12	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group120	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group121	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group122	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group123	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group124	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group125	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group126	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group127	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group128	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group129	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group13	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group130	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group131	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group14	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group15	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group16	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group17	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group18	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group19	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group2	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group20	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group21	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group22	no	no	B	no	B	no	B

B	B_additional_B_B_GSP_ConflictingRoutes_group23	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group24	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group25	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group26	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group27	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group28	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group29	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group3	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group30	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group31	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group32	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group33	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group34	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group35	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group36	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group37	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group38	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group39	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group4	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group40	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group41	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group42	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group43	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group44	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group45	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group46	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group47	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group48	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group49	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group5	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group50	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group51	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group52	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group53	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group54	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group55	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group56	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group57	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group58	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group59	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group6	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group60	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group61	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group62	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group63	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group64	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group65	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group66	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group67	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group68	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group69	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group7	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group70	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group71	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group72	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group73	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group74	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group75	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group76	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group77	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group78	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group79	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group8	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group80	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group81	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group82	no	no	B	no	B	no	B

B	B_additional_B_B_GSP_ConflictingRoutes_group83	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group84	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group85	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group86	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group87	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group88	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group89	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group9	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group90	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group91	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group92	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group93	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group94	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group95	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group96	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group97	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group98	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ConflictingRoutes_group99	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_FirstOverlapWithPreviousRouteSectionRel_group0	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_FirstOverlapWithPreviousRouteSectionRel_Fast_group0	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_FirstRouteSectionReleaseWithLocking_group0	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_FirstRouteSectionReleaseWithLocking_group1	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_FirstRouteSectionReleaseWithLocking_group10	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_FirstRouteSectionReleaseWithLocking_group11	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_FirstRouteSectionReleaseWithLocking_group12	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_FirstRouteSectionReleaseWithLocking_group13	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_FirstRouteSectionReleaseWithLocking_group2	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_FirstRouteSectionReleaseWithLocking_group3	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_FirstRouteSectionReleaseWithLocking_group4	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_FirstRouteSectionReleaseWithLocking_group5	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_FirstRouteSectionReleaseWithLocking_group6	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_FirstRouteSectionReleaseWithLocking_group7	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_FirstRouteSectionReleaseWithLocking_group8	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_FirstRouteSectionReleaseWithLocking_group9	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_OverlapSectionsInLineOfRoute_group0	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointControlWhenFreeToMove_group0	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointControlWhenFreeToMove_group1	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointControlWhenFreeToMove_group2	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointControlWhenFreeToMove_group3	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointControlWhenFreeToMove_group4	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointControlWhenFreeToMove_group5	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointControlWhenFreeToMove_group6	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointControlWhenKeyed_group0	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointControlWhenKeyed_group1	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointControlWhenKeyed_group2	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointControlWhenKeyed_group3	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointControlWhenKeyed_group4	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointControlWhenKeyed_group5	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointControlWhenKeyed_group6	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointControlWithSectionsOverPoints_group0	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointControlWithSectionsOverPoints_group1	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointControlWithSectionsOverPoints_group10	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointControlWithSectionsOverPoints_group11	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointControlWithSectionsOverPoints_group12	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointControlWithSectionsOverPoints_group13	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointControlWithSectionsOverPoints_group2	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointControlWithSectionsOverPoints_group3	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointControlWithSectionsOverPoints_group4	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointControlWithSectionsOverPoints_group5	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointControlWithSectionsOverPoints_group6	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointControlWithSectionsOverPoints_group7	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointControlWithSectionsOverPoints_group8	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointControlWithSectionsOverPoints_group9	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointControlWithTechniciansControls_group0	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointControlWithTechniciansControls_group1	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointControlWithTechniciansControls_group2	no	no	B	no	B	no	B

B	B_additional_B_B_GSP_PointControlWithTechniciansControls_group3	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointControlWithTechniciansControls_group4	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointControlWithTechniciansControls_group5	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointControlWithTechniciansControls_group6	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointReminderDevice_group0	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointReminderDevice_group1	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointReminderDevice_group2	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointReminderDevice_group3	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointReminderDevice_group4	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointReminderDevice_group5	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointReminderDevice_group6	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointsCalledNormal_group0	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointsCalledNormal_group1	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointsCalledNormal_group2	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointsCalledNormal_group3	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointsCalledNormal_group4	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointsCalledNormal_group5	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointsCalledNormal_group6	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointsCalledReverse_group0	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointsCalledReverse_group1	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointsCalledReverse_group2	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointsCalledReverse_group3	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointsCalledReverse_group4	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointsCalledReverse_group5	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointsCalledReverse_group6	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointsInTheRoute_group0	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointsInTheRoute_group1	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointsInTheRoute_group10	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointsInTheRoute_group11	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointsInTheRoute_group12	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointsInTheRoute_group13	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointsInTheRoute_group14	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointsInTheRoute_group15	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointsInTheRoute_group16	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointsInTheRoute_group17	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointsInTheRoute_group18	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointsInTheRoute_group19	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointsInTheRoute_group2	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointsInTheRoute_group20	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointsInTheRoute_group21	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointsInTheRoute_group22	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointsInTheRoute_group23	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointsInTheRoute_group24	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointsInTheRoute_group25	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointsInTheRoute_group26	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointsInTheRoute_group27	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointsInTheRoute_group28	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointsInTheRoute_group29	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointsInTheRoute_group3	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointsInTheRoute_group30	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointsInTheRoute_group31	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointsInTheRoute_group32	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointsInTheRoute_group33	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointsInTheRoute_group34	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointsInTheRoute_group35	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointsInTheRoute_group36	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointsInTheRoute_group37	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointsInTheRoute_group38	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointsInTheRoute_group39	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointsInTheRoute_group4	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointsInTheRoute_group40	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointsInTheRoute_group41	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointsInTheRoute_group5	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointsInTheRoute_group6	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointsInTheRoute_group7	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_PointsInTheRoute_group8	no	no	B	no	B	no	B

B	B_additional_B_B_GSP_PointsInTheRoute_group9	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ProceedWithExitSignal_group0	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ProceedWithOverlapTracksClear_group0	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ProceedWithTracksClear_group0	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ProceedWithTracksClear_group1	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ProceedWithTracksClear_group10	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ProceedWithTracksClear_group11	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ProceedWithTracksClear_group12	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ProceedWithTracksClear_group13	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ProceedWithTracksClear_group14	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ProceedWithTracksClear_group15	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ProceedWithTracksClear_group16	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ProceedWithTracksClear_group17	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ProceedWithTracksClear_group18	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ProceedWithTracksClear_group19	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ProceedWithTracksClear_group2	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ProceedWithTracksClear_group3	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ProceedWithTracksClear_group4	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ProceedWithTracksClear_group5	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ProceedWithTracksClear_group6	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ProceedWithTracksClear_group7	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ProceedWithTracksClear_group8	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_ProceedWithTracksClear_group9	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RouteCancellation_group0	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RouteCancellation_group1	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RouteCancellation_group10	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RouteCancellation_group11	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RouteCancellation_group12	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RouteCancellation_group13	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RouteCancellation_group2	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RouteCancellation_group3	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RouteCancellation_group4	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RouteCancellation_group5	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RouteCancellation_group6	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RouteCancellation_group7	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RouteCancellation_group8	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RouteCancellation_group9	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RouteSectionLockedAheadOfTrain_group0	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RouteSectionLockedAheadOfTrain_group1	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RouteSectionLockedAheadOfTrain_group10	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RouteSectionLockedAheadOfTrain_group11	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RouteSectionLockedAheadOfTrain_group12	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RouteSectionLockedAheadOfTrain_group2	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RouteSectionLockedAheadOfTrain_group3	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RouteSectionLockedAheadOfTrain_group4	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RouteSectionLockedAheadOfTrain_group5	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RouteSectionLockedAheadOfTrain_group6	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RouteSectionLockedAheadOfTrain_group7	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RouteSectionLockedAheadOfTrain_group8	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RouteSectionLockedAheadOfTrain_group9	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RouteSectionLockedAheadOfTrainAlt_group0	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RouteSectionLockedAheadOfTrainAlt_group1	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RouteSectionLockedAheadOfTrainAlt_group10	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RouteSectionLockedAheadOfTrainAlt_group11	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RouteSectionLockedAheadOfTrainAlt_group12	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RouteSectionLockedAheadOfTrainAlt_group2	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RouteSectionLockedAheadOfTrainAlt_group3	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RouteSectionLockedAheadOfTrainAlt_group4	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RouteSectionLockedAheadOfTrainAlt_group5	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RouteSectionLockedAheadOfTrainAlt_group6	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RouteSectionLockedAheadOfTrainAlt_group7	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RouteSectionLockedAheadOfTrainAlt_group8	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RouteSectionLockedAheadOfTrainAlt_group9	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RouteSectionLockedOverPoints_group0	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RouteSectionLockedOverPoints_group1	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RouteSectionLockedOverPoints_group10	no	no	B	no	B	no	B

B	B_additional_B_B_GSP_RouteSectionLockedOverPoints_group11	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RouteSectionLockedOverPoints_group12	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RouteSectionLockedOverPoints_group13	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RouteSectionLockedOverPoints_group2	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RouteSectionLockedOverPoints_group3	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RouteSectionLockedOverPoints_group4	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RouteSectionLockedOverPoints_group5	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RouteSectionLockedOverPoints_group6	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RouteSectionLockedOverPoints_group7	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RouteSectionLockedOverPoints_group8	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RouteSectionLockedOverPoints_group9	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group0	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group1	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group10	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group100	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group101	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group102	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group103	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group104	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group105	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group106	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group107	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group108	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group109	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group11	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group110	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group111	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group112	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group113	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group114	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group115	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group116	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group117	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group118	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group119	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group12	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group120	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group121	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group122	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group123	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group124	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group125	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group126	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group127	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group128	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group129	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group13	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group130	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group131	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group14	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group15	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group16	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group17	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group18	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group19	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group2	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group20	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group21	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group22	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group23	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group24	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group25	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group26	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group27	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group28	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group29	no	no	B	no	B	no	B

B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group3	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group30	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group31	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group32	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group33	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group34	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group35	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group36	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group37	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group38	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group39	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group4	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group40	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group41	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group42	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group43	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group44	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group45	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group46	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group47	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group48	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group49	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group5	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group50	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group51	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group52	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group53	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group54	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group55	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group56	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group57	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group58	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group59	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group6	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group60	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group61	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group62	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group63	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group64	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group65	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group66	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group67	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group68	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group69	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group7	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group70	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group71	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group72	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group73	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group74	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group75	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group76	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group77	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group78	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group79	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group8	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group80	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group81	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group82	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group83	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group84	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group85	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group86	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group87	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group88	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group89	no	no	B	no	B	no	B

B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group9	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group90	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group91	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group92	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group93	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group94	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group95	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group96	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group97	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group98	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group99	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_SignalGroupReplacement_group0	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_SignalGroupReplacement_group1	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_SignalGroupReplacement_group10	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_SignalGroupReplacement_group11	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_SignalGroupReplacement_group12	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_SignalGroupReplacement_group13	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_SignalGroupReplacement_group2	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_SignalGroupReplacement_group3	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_SignalGroupReplacement_group4	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_SignalGroupReplacement_group5	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_SignalGroupReplacement_group6	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_SignalGroupReplacement_group7	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_SignalGroupReplacement_group8	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_SignalGroupReplacement_group9	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_SignalReminder_group0	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_SignalReminder_group1	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_SignalReminder_group10	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_SignalReminder_group11	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_SignalReminder_group12	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_SignalReminder_group13	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_SignalReminder_group2	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_SignalReminder_group3	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_SignalReminder_group4	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_SignalReminder_group5	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_SignalReminder_group6	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_SignalReminder_group7	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_SignalReminder_group8	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_SignalReminder_group9	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_SubsequentRouteSectionRelease_group0	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_SubsequentRouteSectionRelease_group1	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_SubsequentRouteSectionRelease_group10	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_SubsequentRouteSectionRelease_group11	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_SubsequentRouteSectionRelease_group12	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_SubsequentRouteSectionRelease_group13	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_SubsequentRouteSectionRelease_group14	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_SubsequentRouteSectionRelease_group15	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_SubsequentRouteSectionRelease_group2	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_SubsequentRouteSectionRelease_group3	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_SubsequentRouteSectionRelease_group4	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_SubsequentRouteSectionRelease_group5	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_SubsequentRouteSectionRelease_group6	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_SubsequentRouteSectionRelease_group7	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_SubsequentRouteSectionRelease_group8	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_SubsequentRouteSectionRelease_group9	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_SubsequentRouteSectionRelease_Fast_group0	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_SubsequentRouteSectionRelease_Fast_group1	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_SubsequentRouteSectionRelease_Fast_group10	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_SubsequentRouteSectionRelease_Fast_group11	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_SubsequentRouteSectionRelease_Fast_group12	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_SubsequentRouteSectionRelease_Fast_group13	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_SubsequentRouteSectionRelease_Fast_group14	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_SubsequentRouteSectionRelease_Fast_group15	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_SubsequentRouteSectionRelease_Fast_group16	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_SubsequentRouteSectionRelease_Fast_group2	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_SubsequentRouteSectionRelease_Fast_group3	no	no	B	no	B	no	B

B	B_additional_B_B_GSP_SubsequentRouteSectionRelease_Fast_group4	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_SubsequentRouteSectionRelease_Fast_group5	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_SubsequentRouteSectionRelease_Fast_group6	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_SubsequentRouteSectionRelease_Fast_group7	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_SubsequentRouteSectionRelease_Fast_group8	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_SubsequentRouteSectionRelease_Fast_group9	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TechniciansRouteBar_group0	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TechniciansRouteBar_group1	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TechniciansRouteBar_group10	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TechniciansRouteBar_group11	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TechniciansRouteBar_group12	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TechniciansRouteBar_group13	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TechniciansRouteBar_group2	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TechniciansRouteBar_group3	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TechniciansRouteBar_group4	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TechniciansRouteBar_group5	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TechniciansRouteBar_group6	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TechniciansRouteBar_group7	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TechniciansRouteBar_group8	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TechniciansRouteBar_group9	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group0	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group1	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group10	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group11	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group12	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group13	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group14	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group15	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group16	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group17	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group18	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group19	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group2	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group20	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group21	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group22	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group23	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group24	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group25	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group26	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group27	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group28	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group29	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group3	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group30	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group31	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group32	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group33	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group34	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group35	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group36	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group37	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group38	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group39	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group4	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group40	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group41	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group42	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group43	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group44	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group45	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group46	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group47	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group48	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group49	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group5	no	no	B	no	B	no	B

B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group50	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group51	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group52	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group53	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group54	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group55	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group56	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group57	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group58	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group59	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group6	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group60	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group61	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group62	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group63	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group64	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group65	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group66	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group67	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group7	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group8	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group9	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_Fast_group0	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_Fast_group1	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_Fast_group10	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_Fast_group11	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_Fast_group12	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_Fast_group13	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_Fast_group14	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_Fast_group15	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_Fast_group16	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_Fast_group17	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_Fast_group18	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_Fast_group19	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_Fast_group2	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_Fast_group20	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_Fast_group21	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_Fast_group22	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_Fast_group23	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_Fast_group24	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_Fast_group25	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_Fast_group26	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_Fast_group27	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_Fast_group28	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_Fast_group29	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_Fast_group3	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_Fast_group30	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_Fast_group31	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_Fast_group32	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_Fast_group33	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_Fast_group34	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_Fast_group4	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_Fast_group5	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_Fast_group6	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_Fast_group7	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_Fast_group8	no	no	B	no	B	no	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_Fast_group9	no	no	B	no	B	no	B

C.2 Runtimes of classifying the Siemens Interlocking Testbed

Interlocking	Ladder Logic/Safety Property Pair	LL-Verifier			IC3ref		ABC	
		IV Time (Sec)	BMC Time (Sec)	Type	IC3ref Time (Sec)	Type	ABC Time (Sec)	Type
A	A__Ref12_Train_detection_indicates_clear	1.091	41.84	A	1.8	A	0.03	A
A	A__Ref1_Route_set_and_locked	1.086	45.46	A	1.183	A	0.03	A
A	A__Ref22_Indicators_Proved	1.097	58.82	B	0.428	B	0.03	B
A	A__Ref32_All_signals_on	1.086	45.74	A	1.496	A	0.03	A
A	A__Ref35_Approach_Locking	1.098	56.24	B	1.203	D	0.03	D
B	B__Ref1_Route_set_and_locked	1.119	55.3	B	0.329	B	0.03	B
B	B__Ref22_Indicators_Proved	1.118	55.12	B	0.327	B	0.03	B
B	B__Ref32_All_signals_on	1.115	87.6	B	0.328	B	0.03	B
B	B__Ref35_Approach_Locking	1.117	54.46	B	0.327	B	0.03	B
DesignX	DesignX_GSP_2_1_11_RoutesInStaffLockoutLODP_group0	1.564	73.29	B	0.74	D	0.03	D
DesignX	DesignX_GSP_2_1_12_RoutesInStaffLockoutLODK_group0	1.428	75.73	A	0.609	A	0.03	A
DesignX	DesignX_GSP_2_1_12_RoutesInStaffLockoutLODK_group1	1.421	62.78	A	0.606	A	0.03	A
DesignX	DesignX_GSP_2_1_12_RoutesInStaffLockoutLODK_group2	1.423	66.3	A	0.608	A	0.03	A
DesignX	DesignX_GSP_2_1_12_RoutesInStaffLockoutLODK_group3	1.424	63.14	A	0.608	A	0.03	A
DesignX	DesignX_GSP_2_1_12_RoutesInStaffLockoutLODK_group4	1.422	62.8	A	0.609	A	0.03	A
DesignX	DesignX_GSP_2_1_12_RoutesInStaffLockoutLODK_group5	1.425	67.77	A	0.608	A	0.03	A
DesignX	DesignX_GSP_2_1_12_RoutesInStaffLockoutLODK_group6	1.423	61.45	A	0.609	A	0.03	A
DesignX	DesignX_GSP_2_1_13_RoutesInStaffLockoutLODKOverlap_group0	1.42	75.58	A	0.608	A	0.03	A
DesignX	DesignX_GSP_2_1_13_RoutesInStaffLockoutLODKOverlap_group1	1.423	62.81	A	0.61	A	0.03	A
DesignX	DesignX_GSP_2_1_13_RoutesInStaffLockoutLODKOverlap_group2	1.42	66.32	A	0.609	A	0.03	A
DesignX	DesignX_GSP_2_1_13_RoutesInStaffLockoutLODKOverlap_group3	1.421	63.04	A	0.609	A	0.03	A
DesignX	DesignX_GSP_2_1_13_RoutesInStaffLockoutLODKOverlap_group4	1.425	62.73	A	0.608	A	0.03	A
DesignX	DesignX_GSP_2_1_13_RoutesInStaffLockoutLODKOverlap_group5	1.422	67.8	A	0.608	A	0.03	A
DesignX	DesignX_GSP_2_1_13_RoutesInStaffLockoutLODKOverlap_group6	1.423	61.66	A	0.608	A	0.03	A
DesignX	DesignX_GSP_2_1_16_SignalReminder_group0	1.526	187.8	A	0.887	A	0.03	A
DesignX	DesignX_GSP_2_1_16_SignalReminder_group1	1.529	192.2	A	0.896	A	0.03	A
DesignX	DesignX_GSP_2_1_16_SignalReminder_group10	1.549	69.94	B	0.638	D	0.03	D
DesignX	DesignX_GSP_2_1_16_SignalReminder_group11	1.546	74.37	B	0.731	D	0.03	D
DesignX	DesignX_GSP_2_1_16_SignalReminder_group12	1.53	197.3	A	0.819	A	0.03	A
DesignX	DesignX_GSP_2_1_16_SignalReminder_group13	1.531	182.5	A	0.834	A	0.03	A
DesignX	DesignX_GSP_2_1_16_SignalReminder_group14	1.527	42.18	A	0.801	A	0.03	A
DesignX	DesignX_GSP_2_1_16_SignalReminder_group15	1.529	169.6	A	0.745	A	0.03	A
DesignX	DesignX_GSP_2_1_16_SignalReminder_group16	1.527	175.1	A	0.666	A	0.03	A
DesignX	DesignX_GSP_2_1_16_SignalReminder_group17	1.553	78.69	B	0.645	D	0.03	D
DesignX	DesignX_GSP_2_1_16_SignalReminder_group18	1.527	138.7	A	0.643	A	0.03	A
DesignX	DesignX_GSP_2_1_16_SignalReminder_group19	1.535	274.9	A	0.791	A	0.03	A
DesignX	DesignX_GSP_2_1_16_SignalReminder_group2	1.532	226	A	0.839	A	0.03	A
DesignX	DesignX_GSP_2_1_16_SignalReminder_group20	1.549	241	A	0.792	A	0.03	A
DesignX	DesignX_GSP_2_1_16_SignalReminder_group21	1.562	71.63	B	0.66	D	0.03	D
DesignX	DesignX_GSP_2_1_16_SignalReminder_group22	1.526	136.6	A	0.691	A	0.03	A
DesignX	DesignX_GSP_2_1_16_SignalReminder_group23	1.574	70.53	B	0.692	D	0.03	D
DesignX	DesignX_GSP_2_1_16_SignalReminder_group24	1.531	188.1	A	0.694	A	0.03	A
DesignX	DesignX_GSP_2_1_16_SignalReminder_group25	1.526	134.3	A	0.698	A	0.03	A
DesignX	DesignX_GSP_2_1_16_SignalReminder_group26	1.528	183.1	A	0.694	A	0.03	A
DesignX	DesignX_GSP_2_1_16_SignalReminder_group27	1.54	132.4	B	0.568	D	0.031	D
DesignX	DesignX_GSP_2_1_16_SignalReminder_group3	1.556	77.76	B	0.847	D	0.03	D
DesignX	DesignX_GSP_2_1_16_SignalReminder_group4	1.526	227.5	A	0.845	A	0.03	A
DesignX	DesignX_GSP_2_1_16_SignalReminder_group5	1.529	238.6	A	0.812	A	0.03	A
DesignX	DesignX_GSP_2_1_16_SignalReminder_group6	1.529	176	A	0.655	A	0.03	A
DesignX	DesignX_GSP_2_1_16_SignalReminder_group7	1.545	76.86	B	0.657	D	0.029	D
DesignX	DesignX_GSP_2_1_16_SignalReminder_group8	1.557	76.98	B	0.67	D	0.03	D
DesignX	DesignX_GSP_2_1_16_SignalReminder_group9	1.557	73.06	B	0.706	D	0.03	D
DesignX	DesignX_GSP_2_1_17_TechniciansRouteBar_group0	1.421	62.03	A	0.721	A	0.03	A
DesignX	DesignX_GSP_2_1_17_TechniciansRouteBar_group1	1.422	73.1	A	0.755	A	0.031	A
DesignX	DesignX_GSP_2_1_17_TechniciansRouteBar_group10	1.421	67.33	A	0.669	A	0.03	A
DesignX	DesignX_GSP_2_1_17_TechniciansRouteBar_group11	1.421	70.39	A	0.736	A	0.03	A
DesignX	DesignX_GSP_2_1_17_TechniciansRouteBar_group12	1.418	62.12	A	0.882	A	0.03	A
DesignX	DesignX_GSP_2_1_17_TechniciansRouteBar_group13	1.423	67.1	A	0.843	A	0.03	A
DesignX	DesignX_GSP_2_1_17_TechniciansRouteBar_group14	1.42	68.4	A	0.748	A	0.03	A
DesignX	DesignX_GSP_2_1_17_TechniciansRouteBar_group15	1.42	79.14	A	0.732	A	0.03	A
DesignX	DesignX_GSP_2_1_17_TechniciansRouteBar_group16	1.42	63.43	A	0.661	A	0.03	A
DesignX	DesignX_GSP_2_1_17_TechniciansRouteBar_group17	1.422	61.74	A	0.66	A	0.03	A

DesignX	DesignX_GSP_2_1_17_TechniciansRouteBar_group18	1.42	61.92	A	0.666	A	0.03	A
DesignX	DesignX_GSP_2_1_17_TechniciansRouteBar_group19	1.42	65.65	A	0.717	A	0.03	A
DesignX	DesignX_GSP_2_1_17_TechniciansRouteBar_group2	1.423	63.57	A	0.829	A	0.03	A
DesignX	DesignX_GSP_2_1_17_TechniciansRouteBar_group20	1.42	73.91	A	0.718	A	0.031	A
DesignX	DesignX_GSP_2_1_17_TechniciansRouteBar_group21	1.424	67	A	0.67	A	0.03	A
DesignX	DesignX_GSP_2_1_17_TechniciansRouteBar_group22	1.421	62.44	A	0.696	A	0.03	A
DesignX	DesignX_GSP_2_1_17_TechniciansRouteBar_group23	1.423	68.84	A	0.696	A	0.03	A
DesignX	DesignX_GSP_2_1_17_TechniciansRouteBar_group24	1.426	70.68	A	0.694	A	0.03	A
DesignX	DesignX_GSP_2_1_17_TechniciansRouteBar_group25	1.425	62.53	A	0.693	A	0.03	A
DesignX	DesignX_GSP_2_1_17_TechniciansRouteBar_group26	1.426	66.91	A	0.698	A	0.03	A
DesignX	DesignX_GSP_2_1_17_TechniciansRouteBar_group27	1.422	61.98	A	0.645	A	0.03	A
DesignX	DesignX_GSP_2_1_17_TechniciansRouteBar_group3	1.425	64.29	A	0.569	A	0.031	A
DesignX	DesignX_GSP_2_1_17_TechniciansRouteBar_group4	1.421	65.67	A	0.833	A	0.03	A
DesignX	DesignX_GSP_2_1_17_TechniciansRouteBar_group5	1.42	65.9	A	0.804	A	0.031	A
DesignX	DesignX_GSP_2_1_17_TechniciansRouteBar_group6	1.424	62.11	A	0.661	A	0.031	A
DesignX	DesignX_GSP_2_1_17_TechniciansRouteBar_group7	1.423	62.83	A	0.642	A	0.03	A
DesignX	DesignX_GSP_2_1_17_TechniciansRouteBar_group8	1.425	64.55	A	0.679	A	0.03	A
DesignX	DesignX_GSP_2_1_17_TechniciansRouteBar_group9	1.425	68.36	A	0.753	A	0.03	A
DesignX	DesignX_GSP_2_1_18_SignalGroupReplacement_group0	1.521	42.39	A	0.395	A	0.03	A
DesignX	DesignX_GSP_2_1_18_SignalGroupReplacement_group1	1.521	42.47	A	0.394	A	0.03	A
DesignX	DesignX_GSP_2_1_18_SignalGroupReplacement_group10	1.524	42.58	A	0.398	A	0.03	A
DesignX	DesignX_GSP_2_1_18_SignalGroupReplacement_group11	1.523	42.56	A	0.392	A	0.03	A
DesignX	DesignX_GSP_2_1_18_SignalGroupReplacement_group12	1.518	64.79	A	0.394	A	0.03	A
DesignX	DesignX_GSP_2_1_18_SignalGroupReplacement_group13	1.52	63.21	A	0.39	A	0.03	A
DesignX	DesignX_GSP_2_1_18_SignalGroupReplacement_group14	1.523	62.01	A	0.394	A	0.031	A
DesignX	DesignX_GSP_2_1_18_SignalGroupReplacement_group15	1.523	62.34	A	0.392	A	0.03	A
DesignX	DesignX_GSP_2_1_18_SignalGroupReplacement_group16	1.522	62.18	A	0.391	A	0.03	A
DesignX	DesignX_GSP_2_1_18_SignalGroupReplacement_group17	1.521	61.66	A	0.39	A	0.03	A
DesignX	DesignX_GSP_2_1_18_SignalGroupReplacement_group18	1.516	61.93	A	0.396	A	0.031	A
DesignX	DesignX_GSP_2_1_18_SignalGroupReplacement_group19	1.521	62.2	A	0.394	A	0.03	A
DesignX	DesignX_GSP_2_1_18_SignalGroupReplacement_group2	1.52	61.61	A	0.397	A	0.031	A
DesignX	DesignX_GSP_2_1_18_SignalGroupReplacement_group20	1.522	63.94	A	0.39	A	0.03	A
DesignX	DesignX_GSP_2_1_18_SignalGroupReplacement_group21	1.526	42.48	A	0.394	A	0.031	A
DesignX	DesignX_GSP_2_1_18_SignalGroupReplacement_group22	1.522	61.73	A	0.39	A	0.03	A
DesignX	DesignX_GSP_2_1_18_SignalGroupReplacement_group23	1.52	61.29	A	0.393	A	0.03	A
DesignX	DesignX_GSP_2_1_18_SignalGroupReplacement_group24	1.515	42.5	A	0.392	A	0.031	A
DesignX	DesignX_GSP_2_1_18_SignalGroupReplacement_group25	1.518	62.37	A	0.394	A	0.031	A
DesignX	DesignX_GSP_2_1_18_SignalGroupReplacement_group26	1.52	70.4	A	0.396	A	0.03	A
DesignX	DesignX_GSP_2_1_18_SignalGroupReplacement_group27	1.517	61.39	A	0.395	A	0.031	A
DesignX	DesignX_GSP_2_1_18_SignalGroupReplacement_group3	1.522	63.79	A	0.396	A	0.03	A
DesignX	DesignX_GSP_2_1_18_SignalGroupReplacement_group4	1.521	62.49	A	0.396	A	0.031	A
DesignX	DesignX_GSP_2_1_18_SignalGroupReplacement_group5	1.52	61.9	A	0.393	A	0.03	A
DesignX	DesignX_GSP_2_1_18_SignalGroupReplacement_group6	1.52	61.32	A	0.395	A	0.03	A
DesignX	DesignX_GSP_2_1_18_SignalGroupReplacement_group7	1.518	42.3	A	0.396	A	0.03	A
DesignX	DesignX_GSP_2_1_18_SignalGroupReplacement_group8	1.523	62.87	A	0.394	A	0.031	A
DesignX	DesignX_GSP_2_1_18_SignalGroupReplacement_group9	1.519	62.63	A	0.394	A	0.031	A
DesignX	DesignX_GSP_2_1_1_RoutesFromEntranceSignal_group0	1.542	68.65	C	0.397	CT	0.03	CT
DesignX	DesignX_GSP_2_1_1_RoutesFromEntranceSignal_group1	1.543	66.49	C	0.397	CT	0.03	CT
DesignX	DesignX_GSP_2_1_1_RoutesFromEntranceSignal_group10	1.569	66.48	C	0.392	CT	0.03	CT
DesignX	DesignX_GSP_2_1_1_RoutesFromEntranceSignal_group11	1.555	73.18	C	0.4	CT	0.03	CT
DesignX	DesignX_GSP_2_1_1_RoutesFromEntranceSignal_group12	1.551	70.94	C	0.396	CT	0.03	CT
DesignX	DesignX_GSP_2_1_1_RoutesFromEntranceSignal_group13	1.54	68.43	C	0.398	CT	0.03	CT
DesignX	DesignX_GSP_2_1_1_RoutesFromEntranceSignal_group14	1.547	63.07	C	0.396	CT	0.03	CT
DesignX	DesignX_GSP_2_1_1_RoutesFromEntranceSignal_group15	1.556	65.81	C	0.394	CT	0.031	CT
DesignX	DesignX_GSP_2_1_1_RoutesFromEntranceSignal_group16	1.544	63.17	C	0.397	CT	0.031	CT
DesignX	DesignX_GSP_2_1_1_RoutesFromEntranceSignal_group17	1.538	62.91	C	0.398	CT	0.03	CT
DesignX	DesignX_GSP_2_1_1_RoutesFromEntranceSignal_group18	1.541	68.61	C	0.397	CT	0.03	CT
DesignX	DesignX_GSP_2_1_1_RoutesFromEntranceSignal_group19	1.54	64.02	C	0.392	CT	0.03	CT
DesignX	DesignX_GSP_2_1_1_RoutesFromEntranceSignal_group2	1.539	63.3	C	0.395	CT	0.03	CT
DesignX	DesignX_GSP_2_1_1_RoutesFromEntranceSignal_group20	1.548	72.56	C	0.393	CT	0.03	CT
DesignX	DesignX_GSP_2_1_1_RoutesFromEntranceSignal_group3	1.537	77.82	C	0.393	CT	0.031	CT
DesignX	DesignX_GSP_2_1_1_RoutesFromEntranceSignal_group4	1.535	68.7	C	0.398	CT	0.03	CT
DesignX	DesignX_GSP_2_1_1_RoutesFromEntranceSignal_group5	1.538	66.68	C	0.396	CT	0.03	CT
DesignX	DesignX_GSP_2_1_1_RoutesFromEntranceSignal_group6	1.537	63.01	C	0.398	CT	0.03	CT
DesignX	DesignX_GSP_2_1_1_RoutesFromEntranceSignal_group7	1.541	65.54	C	0.394	CT	0.03	CT
DesignX	DesignX_GSP_2_1_1_RoutesFromEntranceSignal_group8	1.541	66.09	C	0.395	CT	0.03	CT

DesignX	DesignX_GSP_2_1_1_RoutesFromEntranceSignal_group9	1.539	71.34	C	0.392	CT	0.03	CT
DesignX	DesignX_GSP_2_1_20_PermissiveRouteSettingExitSignal_group0	1.552	69.13	A	0.911	A	0.03	A
DesignX	DesignX_GSP_2_1_20_PermissiveRouteSettingExitSignal_group1	1.535	70.2	A	0.687	A	0.03	A
DesignX	DesignX_GSP_2_1_20_PermissiveRouteSettingExitSignal_group2	1.54	71.65	A	0.666	A	0.03	A
DesignX	DesignX_GSP_2_1_21_PermissiveRouteSettingOppMovements_group0	1.559	77.74	C	0.749	CT	0.03	CT
DesignX	DesignX_GSP_2_1_21_PermissiveRouteSettingOppMovements_group1	1.556	73.54	B	0.694	D	0.03	D
DesignX	DesignX_GSP_2_1_21_PermissiveRouteSettingOppMovements_group2	1.549	79.53	B	0.671	D	0.03	D
DesignX	DesignX_GSP_2_1_22_RoutesFromPermissiveExit_group0	1.556	74.41	C	0.73	CT	0.031	CT
DesignX	DesignX_GSP_2_1_22_RoutesFromPermissiveExit_group1	1.558	127.5	C	0.825	CT	0.03	CT
DesignX	DesignX_GSP_2_1_22_RoutesFromPermissiveExit_group2	1.56	172.2	C	0.921	CT	0.03	CT
DesignX	DesignX_GSP_2_1_22_RoutesFromPermissiveExit_group3	1.555	238.9	C	0.89	CT	0.03	CT
DesignX	DesignX_GSP_2_1_22_RoutesFromPermissiveExit_group4	1.551	230.8	C	0.776	CT	0.031	CT
DesignX	DesignX_GSP_2_1_23_TrackSectionsInLineOfRoute_group0	1.549	119.9	B	0.688	D	0.031	D
DesignX	DesignX_GSP_2_1_23_TrackSectionsInLineOfRoute_group1	1.552	74.9	B	0.718	D	0.031	D
DesignX	DesignX_GSP_2_1_23_TrackSectionsInLineOfRoute_group10	1.562	76.05	B	0.829	D	0.031	D
DesignX	DesignX_GSP_2_1_23_TrackSectionsInLineOfRoute_group11	1.553	70.58	B	0.401	D	0.031	D
DesignX	DesignX_GSP_2_1_23_TrackSectionsInLineOfRoute_group12	1.54	69.25	A	0.732	A	0.031	A
DesignX	DesignX_GSP_2_1_23_TrackSectionsInLineOfRoute_group13	1.573	77.62	B	0.732	D	0.031	D
DesignX	DesignX_GSP_2_1_23_TrackSectionsInLineOfRoute_group14	1.536	66.18	A	0.773	A	0.031	A
DesignX	DesignX_GSP_2_1_23_TrackSectionsInLineOfRoute_group15	1.425	69.53	A	0.85	A	0.031	A
DesignX	DesignX_GSP_2_1_23_TrackSectionsInLineOfRoute_group16	1.574	74.73	B	0.677	D	0.031	D
DesignX	DesignX_GSP_2_1_23_TrackSectionsInLineOfRoute_group17	1.563	70.26	B	0.677	D	0.031	D
DesignX	DesignX_GSP_2_1_23_TrackSectionsInLineOfRoute_group18	1.55	81.23	B	0.687	D	0.031	D
DesignX	DesignX_GSP_2_1_23_TrackSectionsInLineOfRoute_group19	1.535	66.74	A	0.59	A	0.03	A
DesignX	DesignX_GSP_2_1_23_TrackSectionsInLineOfRoute_group2	1.537	67.31	A	0.705	A	0.031	A
DesignX	DesignX_GSP_2_1_23_TrackSectionsInLineOfRoute_group20	1.561	78.5	B	0.72	D	0.03	D
DesignX	DesignX_GSP_2_1_23_TrackSectionsInLineOfRoute_group21	1.569	80.8	B	0.397	D	0.03	D
DesignX	DesignX_GSP_2_1_23_TrackSectionsInLineOfRoute_group22	1.43	67.76	A	0.83	A	0.03	A
DesignX	DesignX_GSP_2_1_23_TrackSectionsInLineOfRoute_group23	1.567	106.4	B	0.918	D	0.03	D
DesignX	DesignX_GSP_2_1_23_TrackSectionsInLineOfRoute_group24	1.423	65.38	A	0.864	A	0.031	A
DesignX	DesignX_GSP_2_1_23_TrackSectionsInLineOfRoute_group25	1.431	61.35	A	0.88	A	0.031	A
DesignX	DesignX_GSP_2_1_23_TrackSectionsInLineOfRoute_group26	1.424	64.84	A	0.771	A	0.03	A
DesignX	DesignX_GSP_2_1_23_TrackSectionsInLineOfRoute_group27	1.428	62.4	A	0.791	A	0.031	A
DesignX	DesignX_GSP_2_1_23_TrackSectionsInLineOfRoute_group3	1.537	76.47	A	0.706	A	0.031	A
DesignX	DesignX_GSP_2_1_23_TrackSectionsInLineOfRoute_group4	1.56	75.36	B	0.717	D	0.031	D
DesignX	DesignX_GSP_2_1_23_TrackSectionsInLineOfRoute_group5	1.579	104.2	B	0.72	D	0.031	D
DesignX	DesignX_GSP_2_1_23_TrackSectionsInLineOfRoute_group6	1.572	92.81	B	0.675	D	0.031	D
DesignX	DesignX_GSP_2_1_23_TrackSectionsInLineOfRoute_group7	1.558	74.23	B	0.694	D	0.031	D
DesignX	DesignX_GSP_2_1_23_TrackSectionsInLineOfRoute_group8	1.565	67.42	B	0.832	D	0.03	D
DesignX	DesignX_GSP_2_1_23_TrackSectionsInLineOfRoute_group9	1.563	78.5	B	0.398	D	0.03	D
DesignX	DesignX_GSP_2_1_24_OverlapSectionsInLineOfRoute_group0	1.543	70.26	A	1.314	A	0.031	A
DesignX	DesignX_GSP_2_1_24_OverlapSectionsInLineOfRoute_group1	1.537	68.46	A	1.041	A	0.03	A
DesignX	DesignX_GSP_2_1_24_OverlapSectionsInLineOfRoute_group10	1.425	63.29	A	0.958	A	0.03	A
DesignX	DesignX_GSP_2_1_24_OverlapSectionsInLineOfRoute_group11	1.422	61.35	A	0.979	A	0.031	A
DesignX	DesignX_GSP_2_1_24_OverlapSectionsInLineOfRoute_group12	1.422	63.09	A	0.978	A	0.031	A
DesignX	DesignX_GSP_2_1_24_OverlapSectionsInLineOfRoute_group13	1.422	67.14	A	0.666	A	0.031	A
DesignX	DesignX_GSP_2_1_24_OverlapSectionsInLineOfRoute_group2	1.535	71.93	A	0.684	A	0.03	A
DesignX	DesignX_GSP_2_1_24_OverlapSectionsInLineOfRoute_group3	1.545	66.29	A	1.345	A	0.031	A
DesignX	DesignX_GSP_2_1_24_OverlapSectionsInLineOfRoute_group4	1.53	68.21	A	1.065	A	0.03	A
DesignX	DesignX_GSP_2_1_24_OverlapSectionsInLineOfRoute_group5	1.538	73.35	A	0.689	A	0.031	A
DesignX	DesignX_GSP_2_1_24_OverlapSectionsInLineOfRoute_group6	1.532	119.2	A	1.556	A	0.03	A
DesignX	DesignX_GSP_2_1_24_OverlapSectionsInLineOfRoute_group7	1.529	73.56	A	1.18	A	0.03	A
DesignX	DesignX_GSP_2_1_24_OverlapSectionsInLineOfRoute_group8	1.55	69.83	B	0.738	D	0.031	D
DesignX	DesignX_GSP_2_1_24_OverlapSectionsInLineOfRoute_group9	1.555	71.41	B	0.74	D	0.031	D
DesignX	DesignX_GSP_2_1_2_OpposingRoutes_group0	1.534	71.05	C	0.394	CT	0.031	CT
DesignX	DesignX_GSP_2_1_2_OpposingRoutes_group1	1.539	175.8	C	0.396	CT	0.031	CT
DesignX	DesignX_GSP_2_1_2_OpposingRoutes_group10	1.541	66.03	C	0.398	CT	0.031	CT
DesignX	DesignX_GSP_2_1_2_OpposingRoutes_group11	1.538	73.46	C	0.395	CT	0.031	CT
DesignX	DesignX_GSP_2_1_2_OpposingRoutes_group12	1.541	72.18	C	0.393	CT	0.031	CT
DesignX	DesignX_GSP_2_1_2_OpposingRoutes_group13	1.538	68.72	C	0.396	CT	0.031	CT
DesignX	DesignX_GSP_2_1_2_OpposingRoutes_group14	1.538	66.3	C	0.398	CT	0.031	CT
DesignX	DesignX_GSP_2_1_2_OpposingRoutes_group15	1.535	64.2	C	0.394	CT	0.031	CT
DesignX	DesignX_GSP_2_1_2_OpposingRoutes_group16	1.539	272	C	0.396	CT	0.031	CT
DesignX	DesignX_GSP_2_1_2_OpposingRoutes_group17	1.54	226	C	0.397	CT	0.031	CT
DesignX	DesignX_GSP_2_1_2_OpposingRoutes_group18	1.534	170.1	C	0.394	CT	0.031	CT
DesignX	DesignX_GSP_2_1_2_OpposingRoutes_group19	1.539	263.7	C	0.401	CT	0.031	CT

DesignX	DesignX_GSP_2_1_2_OpposingRoutes_group2	1.533	217.7	C	0.401	CT	0.031	CT
DesignX	DesignX_GSP_2_1_2_OpposingRoutes_group20	1.533	121.8	C	0.399	CT	0.031	CT
DesignX	DesignX_GSP_2_1_2_OpposingRoutes_group21	1.536	69.95	C	0.393	CT	0.031	CT
DesignX	DesignX_GSP_2_1_2_OpposingRoutes_group22	1.555	62.74	C	0.395	CT	0.031	CT
DesignX	DesignX_GSP_2_1_2_OpposingRoutes_group23	1.545	64.5	C	0.395	CT	0.031	CT
DesignX	DesignX_GSP_2_1_2_OpposingRoutes_group24	1.539	407.7	C	0.395	CT	0.03	CT
DesignX	DesignX_GSP_2_1_2_OpposingRoutes_group25	1.539	265.2	C	0.399	CT	0.031	CT
DesignX	DesignX_GSP_2_1_2_OpposingRoutes_group26	1.536	66.76	C	0.394	CT	0.03	CT
DesignX	DesignX_GSP_2_1_2_OpposingRoutes_group3	1.538	271.3	C	0.394	CT	0.031	CT
DesignX	DesignX_GSP_2_1_2_OpposingRoutes_group4	1.541	74.38	C	0.4	CT	0.031	CT
DesignX	DesignX_GSP_2_1_2_OpposingRoutes_group5	1.536	70.28	C	0.395	CT	0.031	CT
DesignX	DesignX_GSP_2_1_2_OpposingRoutes_group6	1.539	220.2	C	0.401	CT	0.031	CT
DesignX	DesignX_GSP_2_1_2_OpposingRoutes_group7	1.534	168.8	C	0.398	CT	0.03	CT
DesignX	DesignX_GSP_2_1_2_OpposingRoutes_group8	1.54	65.58	C	0.396	CT	0.031	CT
DesignX	DesignX_GSP_2_1_2_OpposingRoutes_group9	1.544	68.92	C	0.395	CT	0.031	CT
DesignX	DesignX_GSP_2_1_3_ConflictingRoutes_group0	1.538	219	C	0.398	CT	0.03	CT
DesignX	DesignX_GSP_2_1_3_ConflictingRoutes_group1	1.545	367.B	C	0.405	CT	0.031	CT
DesignX	DesignX_GSP_2_1_3_ConflictingRoutes_group10	1.53	405.3	C	0.402	CT	0.031	CT
DesignX	DesignX_GSP_2_1_3_ConflictingRoutes_group11	1.544	536.8	C	0.399	CT	0.031	CT
DesignX	DesignX_GSP_2_1_3_ConflictingRoutes_group12	1.54	740.8	C	0.402	CT	0.031	CT
DesignX	DesignX_GSP_2_1_3_ConflictingRoutes_group13	1.535	660	C	0.4	CT	0.03	CT
DesignX	DesignX_GSP_2_1_3_ConflictingRoutes_group14	1.538	240.1	C	0.398	CT	0.031	CT
DesignX	DesignX_GSP_2_1_3_ConflictingRoutes_group15	1.538	834	C	0.401	CT	0.031	CT
DesignX	DesignX_GSP_2_1_3_ConflictingRoutes_group16	1.54	407.6	C	0.403	CT	0.031	CT
DesignX	DesignX_GSP_2_1_3_ConflictingRoutes_group17	1.541	392.1	C	0.403	CT	0.031	CT
DesignX	DesignX_GSP_2_1_3_ConflictingRoutes_group18	1.542	269.5	C	0.397	CT	0.03	CT
DesignX	DesignX_GSP_2_1_3_ConflictingRoutes_group19	1.537	437.6	C	0.401	CT	0.031	CT
DesignX	DesignX_GSP_2_1_3_ConflictingRoutes_group2	1.542	468.5	C	0.401	CT	0.031	CT
DesignX	DesignX_GSP_2_1_3_ConflictingRoutes_group20	1.542	419.3	C	0.402	CT	0.031	CT
DesignX	DesignX_GSP_2_1_3_ConflictingRoutes_group21	1.541	366.4	C	0.401	CT	0.03	CT
DesignX	DesignX_GSP_2_1_3_ConflictingRoutes_group3	1.537	348.4	C	0.402	CT	0.031	CT
DesignX	DesignX_GSP_2_1_3_ConflictingRoutes_group4	1.539	829.6	C	0.404	CT	0.031	CT
DesignX	DesignX_GSP_2_1_3_ConflictingRoutes_group5	1.535	733.3	C	0.404	CT	0.031	CT
DesignX	DesignX_GSP_2_1_3_ConflictingRoutes_group6	1.527	263.4	C	0.401	CT	0.03	CT
DesignX	DesignX_GSP_2_1_3_ConflictingRoutes_group7	1.536	276	C	0.403	CT	0.031	CT
DesignX	DesignX_GSP_2_1_3_ConflictingRoutes_group8	1.534	346.4	C	0.402	CT	0.03	CT
DesignX	DesignX_GSP_2_1_3_ConflictingRoutes_group9	1.533	838.2	C	0.402	CT	0.03	CT
DesignX	DesignX_GSP_2_1_4_LastOpposingRouteSection_group0	1.552	71.74	B	0.905	D	0.031	D
DesignX	DesignX_GSP_2_1_4_LastOpposingRouteSection_group1	1.557	127.7	B	0.761	D	0.03	D
DesignX	DesignX_GSP_2_1_4_LastOpposingRouteSection_group10	1.538	73.84	A	0.829	A	0.031	A
DesignX	DesignX_GSP_2_1_4_LastOpposingRouteSection_group11	1.56	75.03	B	0.793	D	0.031	D
DesignX	DesignX_GSP_2_1_4_LastOpposingRouteSection_group12	1.531	65.26	A	0.761	A	0.031	A
DesignX	DesignX_GSP_2_1_4_LastOpposingRouteSection_group13	1.541	73.72	A	0.754	A	0.03	A
DesignX	DesignX_GSP_2_1_4_LastOpposingRouteSection_group14	1.549	119.9	B	0.65	D	0.031	D
DesignX	DesignX_GSP_2_1_4_LastOpposingRouteSection_group15	1.569	76.02	B	0.664	D	0.031	D
DesignX	DesignX_GSP_2_1_4_LastOpposingRouteSection_group16	1.573	123.7	C	0.802	CT	0.031	CT
DesignX	DesignX_GSP_2_1_4_LastOpposingRouteSection_group17	1.556	73.65	B	0.722	D	0.031	D
DesignX	DesignX_GSP_2_1_4_LastOpposingRouteSection_group18	1.557	121.7	B	0.674	D	0.031	D
DesignX	DesignX_GSP_2_1_4_LastOpposingRouteSection_group19	1.538	69.92	A	0.737	A	0.031	A
DesignX	DesignX_GSP_2_1_4_LastOpposingRouteSection_group2	1.566	165.1	C	0.579	CT	0.031	CT
DesignX	DesignX_GSP_2_1_4_LastOpposingRouteSection_group20	1.539	65.31	A	0.571	A	0.031	A
DesignX	DesignX_GSP_2_1_4_LastOpposingRouteSection_group21	1.586	219.1	C	0.702	CT	0.031	CT
DesignX	DesignX_GSP_2_1_4_LastOpposingRouteSection_group22	1.549	223.3	C	0.699	CT	0.03	CT
DesignX	DesignX_GSP_2_1_4_LastOpposingRouteSection_group23	1.533	70.55	A	0.703	A	0.031	A
DesignX	DesignX_GSP_2_1_4_LastOpposingRouteSection_group3	1.556	78.94	B	0.844	D	0.031	D
DesignX	DesignX_GSP_2_1_4_LastOpposingRouteSection_group4	1.56	68.92	B	0.709	D	0.031	D
DesignX	DesignX_GSP_2_1_4_LastOpposingRouteSection_group5	1.562	74.65	B	0.669	D	0.031	D
DesignX	DesignX_GSP_2_1_4_LastOpposingRouteSection_group6	1.562	71.36	B	0.702	D	0.031	D
DesignX	DesignX_GSP_2_1_4_LastOpposingRouteSection_group7	1.55	71.37	C	0.721	CT	0.031	CT
DesignX	DesignX_GSP_2_1_4_LastOpposingRouteSection_group8	1.562	76.37	B	0.671	D	0.031	D
DesignX	DesignX_GSP_2_1_4_LastOpposingRouteSection_group9	1.563	125.9	B	0.745	D	0.03	D
DesignX	DesignX_GSP_2_1_5_LastOpposingOverlapRouteSection_group0	1.573	73.36	B	0.793	D	0.031	D
DesignX	DesignX_GSP_2_1_5_LastOpposingOverlapRouteSection_group1	1.551	75.68	B	0.72	D	0.031	D
DesignX	DesignX_GSP_2_1_5_LastOpposingOverlapRouteSection_group2	1.556	68.77	B	0.701	D	0.031	D
DesignX	DesignX_GSP_2_1_5_LastOpposingOverlapRouteSection_group3	1.534	67.93	A	0.666	A	0.031	A
DesignX	DesignX_GSP_2_1_5_LastOpposingOverlapRouteSection_group4	1.564	121.4	C	0.667	CT	0.031	CT

DesignX	DesignX_GSP_2_1_5_LastOpposingOverlapRouteSection_group5	1.548	70.78	A	0.716	A	0.031	A
DesignX	DesignX_GSP_2_1_5_LastOpposingOverlapRouteSection_group6	1.528	67.38	A	0.7	A	0.03	A
DesignX	DesignX_GSP_2_1_5_LastOpposingOverlapRouteSection_group7	1.548	164.3	C	0.699	CT	0.031	CT
DesignX	DesignX_GSP_2_1_5_LastOpposingOverlapRouteSection_group8	1.552	165.3	C	0.699	CT	0.031	CT
DesignX	DesignX_GSP_2_1_6_PointsInTheRoute_group0	1.541	119.5	C	0.394	CT	0.031	CT
DesignX	DesignX_GSP_2_1_6_PointsInTheRoute_group1	1.537	215.2	C	0.401	CT	0.031	CT
DesignX	DesignX_GSP_2_1_6_PointsInTheRoute_group10	1.535	264.2	C	0.396	CT	0.031	CT
DesignX	DesignX_GSP_2_1_6_PointsInTheRoute_group11	1.53	179	C	0.396	CT	0.031	CT
DesignX	DesignX_GSP_2_1_6_PointsInTheRoute_group12	1.534	168.5	C	0.398	CT	0.031	CT
DesignX	DesignX_GSP_2_1_6_PointsInTheRoute_group13	1.536	164.1	C	0.397	CT	0.031	CT
DesignX	DesignX_GSP_2_1_6_PointsInTheRoute_group14	1.535	71.86	C	0.395	CT	0.031	CT
DesignX	DesignX_GSP_2_1_6_PointsInTheRoute_group15	1.533	67.67	C	0.398	CT	0.031	CT
DesignX	DesignX_GSP_2_1_6_PointsInTheRoute_group16	1.532	163.8	C	0.397	CT	0.03	CT
DesignX	DesignX_GSP_2_1_6_PointsInTheRoute_group17	1.541	161.3	C	0.4	CT	0.031	CT
DesignX	DesignX_GSP_2_1_6_PointsInTheRoute_group18	1.538	117.2	C	0.398	CT	0.03	CT
DesignX	DesignX_GSP_2_1_6_PointsInTheRoute_group19	1.536	160.7	C	0.397	CT	0.031	CT
DesignX	DesignX_GSP_2_1_6_PointsInTheRoute_group2	1.533	174	C	0.397	CT	0.031	CT
DesignX	DesignX_GSP_2_1_6_PointsInTheRoute_group20	1.53	173.5	C	0.4	CT	0.03	CT
DesignX	DesignX_GSP_2_1_6_PointsInTheRoute_group21	1.541	285.4	C	0.399	CT	0.031	CT
DesignX	DesignX_GSP_2_1_6_PointsInTheRoute_group3	1.543	75.09	C	0.394	CT	0.031	CT
DesignX	DesignX_GSP_2_1_6_PointsInTheRoute_group4	1.55	163.6	C	0.402	CT	0.031	CT
DesignX	DesignX_GSP_2_1_6_PointsInTheRoute_group5	1.532	220.6	C	0.398	CT	0.031	CT
DesignX	DesignX_GSP_2_1_6_PointsInTheRoute_group6	1.532	121.1	C	0.394	CT	0.031	CT
DesignX	DesignX_GSP_2_1_6_PointsInTheRoute_group7	1.538	119.3	C	0.395	CT	0.031	CT
DesignX	DesignX_GSP_2_1_6_PointsInTheRoute_group8	1.54	267.7	C	0.395	CT	0.031	CT
DesignX	DesignX_GSP_2_1_6_PointsInTheRoute_group9	1.534	217.1	C	0.395	CT	0.031	CT
DesignX	DesignX_GSP_2_1_8_PointsInTheOverlap_group0	1.53	446.2	C	0.397	CT	0.031	CT
DesignX	DesignX_GSP_2_1_8_PointsInTheOverlap_group1	1.54	361.9	C	0.397	CT	0.031	CT
DesignX	DesignX_GSP_2_1_8_PointsInTheOverlap_group2	1.536	174.2	C	0.397	CT	0.031	CT
DesignX	DesignX_GSP_2_1_8_PointsInTheOverlap_group3	1.531	224	C	0.4	CT	0.031	CT
DesignX	DesignX_GSP_2_2_12_ProceedMCBCCTVCrossing_group0	1.503	61.64	A	1.463	A	0.03	A
DesignX	DesignX_GSP_2_2_12_ProceedMCBCCTVCrossing_group1	1.51	68.16	A	0.82	A	0.03	A
DesignX	DesignX_GSP_2_2_12_ProceedMCBCCTVCrossing_group2	1.505	68.44	A	2.943	A	0.03	A
DesignX	DesignX_GSP_2_2_12_ProceedMCBCCTVCrossing_group3	1.507	64.25	A	2.92	A	0.03	A
DesignX	DesignX_GSP_2_2_16_ProceedWithExitSignal_group0	1.554	69.78	C	5.685	CT	0.03	CT
DesignX	DesignX_GSP_2_2_16_ProceedWithExitSignal_group1	1.548	165.9	C	6.704	CT	0.03	CT
DesignX	DesignX_GSP_2_2_16_ProceedWithExitSignal_group10	1.54	69.97	C	2.613	CT	0.03	CT
DesignX	DesignX_GSP_2_2_16_ProceedWithExitSignal_group11	1.546	75.33	C	1.965	CT	0.031	CT
DesignX	DesignX_GSP_2_2_16_ProceedWithExitSignal_group12	1.541	117.5	C	2.072	CT	0.03	CT
DesignX	DesignX_GSP_2_2_16_ProceedWithExitSignal_group13	1.42	42.29	A	3.204	A	0.03	A
DesignX	DesignX_GSP_2_2_16_ProceedWithExitSignal_group14	1.424	62.54	A	3.324	A	0.031	A
DesignX	DesignX_GSP_2_2_16_ProceedWithExitSignal_group15	1.425	63.32	A	3.279	A	0.031	A
DesignX	DesignX_GSP_2_2_16_ProceedWithExitSignal_group16	1.43	61.87	A	1.885	A	0.03	A
DesignX	DesignX_GSP_2_2_16_ProceedWithExitSignal_group2	1.546	127.9	C	6.199	CT	0.03	CT
DesignX	DesignX_GSP_2_2_16_ProceedWithExitSignal_group3	1.543	124.3	C	2.997	CT	0.031	CT
DesignX	DesignX_GSP_2_2_16_ProceedWithExitSignal_group4	1.546	123.1	C	4.991	CT	0.03	CT
DesignX	DesignX_GSP_2_2_16_ProceedWithExitSignal_group5	1.539	74.19	C	3.157	CT	0.03	CT
DesignX	DesignX_GSP_2_2_16_ProceedWithExitSignal_group6	1.547	120.5	C	3.879	CT	0.031	CT
DesignX	DesignX_GSP_2_2_16_ProceedWithExitSignal_group7	1.539	70.87	C	3.752	CT	0.03	CT
DesignX	DesignX_GSP_2_2_16_ProceedWithExitSignal_group8	1.424	62.07	A	1.975	A	0.03	A
DesignX	DesignX_GSP_2_2_16_ProceedWithExitSignal_group9	1.545	67.47	C	2.405	CT	0.03	CT
DesignX	DesignX_GSP_2_2_1_ProceedAspectWithRouteSet_group0	1.432	62.81	A	6.109	A	0.03	A
DesignX	DesignX_GSP_2_2_1_ProceedAspectWithRouteSet_group1	1.423	61.3	A	2.602	A	0.03	A
DesignX	DesignX_GSP_2_2_1_ProceedAspectWithRouteSet_group10	1.429	63.92	A	2.734	A	0.03	A
DesignX	DesignX_GSP_2_2_1_ProceedAspectWithRouteSet_group11	1.424	72.53	A	3.196	A	0.03	A
DesignX	DesignX_GSP_2_2_1_ProceedAspectWithRouteSet_group12	1.431	62.24	A	3.18	A	0.03	A
DesignX	DesignX_GSP_2_2_1_ProceedAspectWithRouteSet_group13	1.428	64.24	A	1.338	A	0.031	A
DesignX	DesignX_GSP_2_2_1_ProceedAspectWithRouteSet_group14	1.426	42.39	A	3.257	A	0.03	A
DesignX	DesignX_GSP_2_2_1_ProceedAspectWithRouteSet_group15	1.427	42.19	A	3.305	A	0.03	A
DesignX	DesignX_GSP_2_2_1_ProceedAspectWithRouteSet_group16	1.428	64.85	A	1.885	A	0.03	A
DesignX	DesignX_GSP_2_2_1_ProceedAspectWithRouteSet_group17	1.424	61.93	A	1.199	A	0.03	A
DesignX	DesignX_GSP_2_2_1_ProceedAspectWithRouteSet_group2	1.425	62.48	A	2.475	A	0.031	A
DesignX	DesignX_GSP_2_2_1_ProceedAspectWithRouteSet_group3	1.427	61.97	A	3.938	A	0.03	A
DesignX	DesignX_GSP_2_2_1_ProceedAspectWithRouteSet_group4	1.418	61.85	A	3.453	A	0.03	A
DesignX	DesignX_GSP_2_2_1_ProceedAspectWithRouteSet_group5	1.425	61.99	A	2.797	A	0.031	A
DesignX	DesignX_GSP_2_2_1_ProceedAspectWithRouteSet_group6	1.425	61.54	A	4.102	A	0.03	A

DesignX	DesignX_GSP_2_2_1_ProceedAspectWithRouteSet_group7	1.424	61.34	A	2.329	A	0.03	A
DesignX	DesignX_GSP_2_2_1_ProceedAspectWithRouteSet_group8	1.424	61.52	A	2.001	A	0.03	A
DesignX	DesignX_GSP_2_2_1_ProceedAspectWithRouteSet_group9	1.425	62.58	A	2.394	A	0.031	A
DesignX	DesignX_GSP_2_2_21_ProceedWithDisengagement_group0	1.43	63.42	A	5.313	A	0.03	A
DesignX	DesignX_GSP_2_2_21_ProceedWithDisengagement_group1	1.424	62.7	A	2.523	A	0.03	A
DesignX	DesignX_GSP_2_2_21_ProceedWithDisengagement_group10	1.428	62.62	A	2.644	A	0.03	A
DesignX	DesignX_GSP_2_2_21_ProceedWithDisengagement_group11	1.425	63.9	A	2.984	A	0.03	A
DesignX	DesignX_GSP_2_2_21_ProceedWithDisengagement_group12	1.429	64.69	A	3.203	A	0.03	A
DesignX	DesignX_GSP_2_2_21_ProceedWithDisengagement_group13	1.427	64.43	A	1.314	A	0.03	A
DesignX	DesignX_GSP_2_2_21_ProceedWithDisengagement_group14	1.423	61.8	A	3.284	A	0.031	A
DesignX	DesignX_GSP_2_2_21_ProceedWithDisengagement_group15	1.428	64.31	A	3.276	A	0.031	A
DesignX	DesignX_GSP_2_2_21_ProceedWithDisengagement_group16	1.429	62.35	A	1.878	A	0.031	A
DesignX	DesignX_GSP_2_2_21_ProceedWithDisengagement_group17	1.42	62.54	A	1.098	A	0.031	A
DesignX	DesignX_GSP_2_2_21_ProceedWithDisengagement_group2	1.426	61.43	A	3.36	A	0.03	A
DesignX	DesignX_GSP_2_2_21_ProceedWithDisengagement_group3	1.425	61.74	A	3.903	A	0.03	A
DesignX	DesignX_GSP_2_2_21_ProceedWithDisengagement_group4	1.424	61.8	A	3.508	A	0.03	A
DesignX	DesignX_GSP_2_2_21_ProceedWithDisengagement_group5	1.423	63.34	A	2.832	A	0.03	A
DesignX	DesignX_GSP_2_2_21_ProceedWithDisengagement_group6	1.531	62.42	A	4.935	A	0.03	A
DesignX	DesignX_GSP_2_2_21_ProceedWithDisengagement_group7	1.538	63.09	A	2.976	A	0.03	A
DesignX	DesignX_GSP_2_2_21_ProceedWithDisengagement_group8	1.531	61.53	A	2.135	A	0.03	A
DesignX	DesignX_GSP_2_2_21_ProceedWithDisengagement_group9	1.424	63.42	A	2.513	A	0.03	A
DesignX	DesignX_GSP_2_2_22_ProceedWithSGRC_group0	1.506	70.63	A	2.264	A	0.03	A
DesignX	DesignX_GSP_2_2_22_ProceedWithSGRC_group1	1.511	63.24	A	4.049	A	0.03	A
DesignX	DesignX_GSP_2_2_22_ProceedWithSGRC_group10	1.522	61.68	A	3.265	A	0.03	A
DesignX	DesignX_GSP_2_2_22_ProceedWithSGRC_group11	1.521	61.22	A	3.274	A	0.031	A
DesignX	DesignX_GSP_2_2_22_ProceedWithSGRC_group12	1.525	61.41	A	1.896	A	0.03	A
DesignX	DesignX_GSP_2_2_22_ProceedWithSGRC_group13	1.527	65.02	A	1.208	A	0.03	A
DesignX	DesignX_GSP_2_2_22_ProceedWithSGRC_group2	1.523	63.9	A	3.958	A	0.03	A
DesignX	DesignX_GSP_2_2_22_ProceedWithSGRC_group3	1.528	64.71	A	2.801	A	0.03	A
DesignX	DesignX_GSP_2_2_22_ProceedWithSGRC_group4	1.507	63.59	A	4.058	A	0.03	A
DesignX	DesignX_GSP_2_2_22_ProceedWithSGRC_group5	1.527	61.18	A	2.037	A	0.03	A
DesignX	DesignX_GSP_2_2_22_ProceedWithSGRC_group6	1.501	63.02	A	2.63	A	0.031	A
DesignX	DesignX_GSP_2_2_22_ProceedWithSGRC_group7	1.521	65.02	A	1.94	A	0.031	A
DesignX	DesignX_GSP_2_2_22_ProceedWithSGRC_group8	1.528	63.82	A	3.196	A	0.03	A
DesignX	DesignX_GSP_2_2_22_ProceedWithSGRC_group9	1.522	61.41	A	1.317	A	0.03	A
DesignX	DesignX_GSP_2_2_23_ProceedWithCSFC_group0	1.53	71.2	B	0.567	B	0.03	B
DesignX	DesignX_GSP_2_2_23_ProceedWithCSFC_group1	1.524	71.19	B	0.573	B	0.03	B
DesignX	DesignX_GSP_2_2_23_ProceedWithCSFC_group10	1.545	74.99	B	0.569	B	0.03	B
DesignX	DesignX_GSP_2_2_23_ProceedWithCSFC_group11	1.557	124.1	B	0.572	B	0.03	B
DesignX	DesignX_GSP_2_2_23_ProceedWithCSFC_group12	1.553	123.7	B	0.568	B	0.031	B
DesignX	DesignX_GSP_2_2_23_ProceedWithCSFC_group13	1.547	73.96	B	0.571	B	0.03	B
DesignX	DesignX_GSP_2_2_23_ProceedWithCSFC_group2	1.541	73.66	B	0.569	B	0.03	B
DesignX	DesignX_GSP_2_2_23_ProceedWithCSFC_group3	1.545	75.42	B	0.568	B	0.03	B
DesignX	DesignX_GSP_2_2_23_ProceedWithCSFC_group4	1.521	77.48	B	0.567	B	0.03	B
DesignX	DesignX_GSP_2_2_23_ProceedWithCSFC_group5	1.545	71.22	B	0.571	B	0.031	B
DesignX	DesignX_GSP_2_2_23_ProceedWithCSFC_group6	1.534	73.34	B	0.573	B	0.031	B
DesignX	DesignX_GSP_2_2_23_ProceedWithCSFC_group7	1.536	78.31	B	0.568	B	0.031	B
DesignX	DesignX_GSP_2_2_23_ProceedWithCSFC_group8	1.55	73.47	B	0.566	B	0.03	B
DesignX	DesignX_GSP_2_2_23_ProceedWithCSFC_group9	1.542	69.95	B	0.568	B	0.03	B
DesignX	DesignX_GSP_2_2_24_ProceedDuringEPR_group0	1.546	72.98	C	6.201	CT	0.03	CT
DesignX	DesignX_GSP_2_2_24_ProceedDuringEPR_group1	1.541	252.4	C	5.724	CT	0.03	CT
DesignX	DesignX_GSP_2_2_24_ProceedDuringEPR_group10	1.537	123.5	C	2.758	CT	0.031	CT
DesignX	DesignX_GSP_2_2_24_ProceedDuringEPR_group11	1.54	72.29	B	2.582	D	0.03	D
DesignX	DesignX_GSP_2_2_24_ProceedDuringEPR_group12	1.533	74.2	C	5.153	CT	0.03	CT
DesignX	DesignX_GSP_2_2_24_ProceedDuringEPR_group13	1.543	119.9	C	3.687	CT	0.031	CT
DesignX	DesignX_GSP_2_2_24_ProceedDuringEPR_group14	1.528	64.29	A	4.341	A	0.03	A
DesignX	DesignX_GSP_2_2_24_ProceedDuringEPR_group15	1.528	65.33	A	2.137	A	0.03	A
DesignX	DesignX_GSP_2_2_24_ProceedDuringEPR_group16	1.539	69.51	C	2.381	CT	0.031	CT
DesignX	DesignX_GSP_2_2_24_ProceedDuringEPR_group17	1.525	62.76	A	2.733	A	0.031	A
DesignX	DesignX_GSP_2_2_24_ProceedDuringEPR_group18	1.537	73.66	C	2.576	CT	0.031	CT
DesignX	DesignX_GSP_2_2_24_ProceedDuringEPR_group19	1.546	74.47	C	2.066	CT	0.031	CT
DesignX	DesignX_GSP_2_2_24_ProceedDuringEPR_group2	1.545	124.7	C	8.098	CT	0.03	CT
DesignX	DesignX_GSP_2_2_24_ProceedDuringEPR_group20	1.542	119.7	C	1.925	CT	0.03	CT
DesignX	DesignX_GSP_2_2_24_ProceedDuringEPR_group21	1.546	72.54	B	2.074	D	0.03	D
DesignX	DesignX_GSP_2_2_24_ProceedDuringEPR_group22	1.524	62.51	A	3.189	A	0.031	A
DesignX	DesignX_GSP_2_2_24_ProceedDuringEPR_group23	1.541	78.39	B	1.322	D	0.031	D

DesignX	DesignX_GSP_2_2_24_ProceedDuringEPR_group24	1.528	62.86	A	3.268	A	0.03	A
DesignX	DesignX_GSP_2_2_24_ProceedDuringEPR_group25	1.523	65	A	3.23	A	0.031	A
DesignX	DesignX_GSP_2_2_24_ProceedDuringEPR_group26	1.523	62.44	A	1.871	A	0.03	A
DesignX	DesignX_GSP_2_2_24_ProceedDuringEPR_group27	1.529	62.41	A	1.194	A	0.031	A
DesignX	DesignX_GSP_2_2_24_ProceedDuringEPR_group3	1.526	62.3	A	2.675	A	0.031	A
DesignX	DesignX_GSP_2_2_24_ProceedDuringEPR_group4	1.552	71.4	C	2.453	CT	0.031	CT
DesignX	DesignX_GSP_2_2_24_ProceedDuringEPR_group5	1.542	166.2	C	2.532	CT	0.031	CT
DesignX	DesignX_GSP_2_2_24_ProceedDuringEPR_group6	1.547	70.57	C	3.411	CT	0.031	CT
DesignX	DesignX_GSP_2_2_24_ProceedDuringEPR_group7	1.522	61.56	A	4.126	A	0.03	A
DesignX	DesignX_GSP_2_2_24_ProceedDuringEPR_group8	1.543	72.86	C	4.339	CT	0.03	CT
DesignX	DesignX_GSP_2_2_24_ProceedDuringEPR_group9	1.545	72.21	B	4.644	D	0.031	D
DesignX	DesignX_GSP_2_2_25_ProceedDuringSTR_group0	1.544	70.52	C	7.137	CT	0.03	CT
DesignX	DesignX_GSP_2_2_25_ProceedDuringSTR_group1	1.54	172.3	C	5.053	CT	0.031	CT
DesignX	DesignX_GSP_2_2_25_ProceedDuringSTR_group10	1.542	70.63	C	3.638	CT	0.031	CT
DesignX	DesignX_GSP_2_2_25_ProceedDuringSTR_group11	1.55	69.35	C	3.682	CT	0.031	CT
DesignX	DesignX_GSP_2_2_25_ProceedDuringSTR_group12	1.549	71.86	B	4.801	D	0.031	D
DesignX	DesignX_GSP_2_2_25_ProceedDuringSTR_group13	1.541	63.31	C	1.929	CT	0.031	CT
DesignX	DesignX_GSP_2_2_25_ProceedDuringSTR_group14	1.539	67.17	C	2.375	CT	0.031	CT
DesignX	DesignX_GSP_2_2_25_ProceedDuringSTR_group15	1.543	79.5	C	3.686	CT	0.03	CT
DesignX	DesignX_GSP_2_2_25_ProceedDuringSTR_group16	1.542	74.29	C	1.913	CT	0.031	CT
DesignX	DesignX_GSP_2_2_25_ProceedDuringSTR_group17	1.54	125.7	C	2.066	CT	0.031	CT
DesignX	DesignX_GSP_2_2_25_ProceedDuringSTR_group18	1.549	73.64	B	2.063	D	0.03	D
DesignX	DesignX_GSP_2_2_25_ProceedDuringSTR_group19	1.527	65.57	A	3.17	A	0.031	A
DesignX	DesignX_GSP_2_2_25_ProceedDuringSTR_group2	1.538	125	C	5.573	CT	0.031	CT
DesignX	DesignX_GSP_2_2_25_ProceedDuringSTR_group20	1.533	73.46	B	1.333	D	0.03	D
DesignX	DesignX_GSP_2_2_25_ProceedDuringSTR_group21	1.518	66.38	A	3.251	A	0.031	A
DesignX	DesignX_GSP_2_2_25_ProceedDuringSTR_group22	1.524	62.31	A	3.259	A	0.031	A
DesignX	DesignX_GSP_2_2_25_ProceedDuringSTR_group23	1.526	42.32	A	1.872	A	0.031	A
DesignX	DesignX_GSP_2_2_25_ProceedDuringSTR_group24	1.53	65.29	A	1.204	A	0.031	A
DesignX	DesignX_GSP_2_2_25_ProceedDuringSTR_group3	1.545	160.3	C	3.548	CT	0.031	CT
DesignX	DesignX_GSP_2_2_25_ProceedDuringSTR_group4	1.541	78.3	C	3.423	CT	0.031	CT
DesignX	DesignX_GSP_2_2_25_ProceedDuringSTR_group5	1.539	69.51	C	3.546	CT	0.031	CT
DesignX	DesignX_GSP_2_2_25_ProceedDuringSTR_group6	1.537	75.71	C	3.471	CT	0.03	CT
DesignX	DesignX_GSP_2_2_25_ProceedDuringSTR_group7	1.544	70.84	B	4.393	D	0.03	D
DesignX	DesignX_GSP_2_2_25_ProceedDuringSTR_group8	1.545	121.7	C	3.187	CT	0.03	CT
DesignX	DesignX_GSP_2_2_25_ProceedDuringSTR_group9	1.536	71.54	B	2.804	D	0.031	D
DesignX	DesignX_GSP_2_2_26_ProceedDuringSTRPermissive_group0	1.549	70.65	B	3.012	D	0.031	D
DesignX	DesignX_GSP_2_2_26_ProceedDuringSTRPermissive_group1	1.552	73.37	B	4.085	D	0.03	D
DesignX	DesignX_GSP_2_2_26_ProceedDuringSTRPermissive_group2	1.546	74.96	B	2.846	D	0.03	D
DesignX	DesignX_GSP_2_2_27_ProceedWithAspRestriction_group0	1.551	74.87	B	6.362	D	0.031	D
DesignX	DesignX_GSP_2_2_27_ProceedWithAspRestriction_group1	1.553	124.5	B	6.119	D	0.03	D
DesignX	DesignX_GSP_2_2_27_ProceedWithAspRestriction_group10	1.543	120.9	B	2.791	D	0.031	D
DesignX	DesignX_GSP_2_2_27_ProceedWithAspRestriction_group11	1.546	68.03	B	2.99	D	0.03	D
DesignX	DesignX_GSP_2_2_27_ProceedWithAspRestriction_group12	1.541	74.7	B	3.7	D	0.031	D
DesignX	DesignX_GSP_2_2_27_ProceedWithAspRestriction_group13	1.543	72.6	B	4.955	D	0.031	D
DesignX	DesignX_GSP_2_2_27_ProceedWithAspRestriction_group14	1.546	70.74	B	3.76	D	0.031	D
DesignX	DesignX_GSP_2_2_27_ProceedWithAspRestriction_group15	1.549	71.37	B	1.976	D	0.031	D
DesignX	DesignX_GSP_2_2_27_ProceedWithAspRestriction_group16	1.545	80.21	B	2.609	D	0.031	D
DesignX	DesignX_GSP_2_2_27_ProceedWithAspRestriction_group17	1.547	70.16	B	2.863	D	0.031	D
DesignX	DesignX_GSP_2_2_27_ProceedWithAspRestriction_group18	1.545	70.7	B	2.578	D	0.031	D
DesignX	DesignX_GSP_2_2_27_ProceedWithAspRestriction_group19	1.541	69.86	B	1.897	D	0.03	D
DesignX	DesignX_GSP_2_2_27_ProceedWithAspRestriction_group2	1.547	75.35	B	6.581	D	0.031	D
DesignX	DesignX_GSP_2_2_27_ProceedWithAspRestriction_group20	1.544	72.1	B	2.048	D	0.03	D
DesignX	DesignX_GSP_2_2_27_ProceedWithAspRestriction_group21	1.552	120	B	2.729	D	0.03	D
DesignX	DesignX_GSP_2_2_27_ProceedWithAspRestriction_group22	1.527	64.12	A	3.183	A	0.031	A
DesignX	DesignX_GSP_2_2_27_ProceedWithAspRestriction_group23	1.552	70.38	B	1.344	D	0.031	D
DesignX	DesignX_GSP_2_2_27_ProceedWithAspRestriction_group24	1.533	42.46	A	3.257	A	0.03	A
DesignX	DesignX_GSP_2_2_27_ProceedWithAspRestriction_group25	1.533	61.5	A	3.245	A	0.031	A
DesignX	DesignX_GSP_2_2_27_ProceedWithAspRestriction_group26	1.53	62.32	A	1.867	A	0.03	A
DesignX	DesignX_GSP_2_2_27_ProceedWithAspRestriction_group27	1.529	42.1	A	1.204	A	0.031	A
DesignX	DesignX_GSP_2_2_27_ProceedWithAspRestriction_group3	1.556	125.1	B	2.562	D	0.031	D
DesignX	DesignX_GSP_2_2_27_ProceedWithAspRestriction_group4	1.546	72.14	B	2.474	D	0.031	D
DesignX	DesignX_GSP_2_2_27_ProceedWithAspRestriction_group5	1.551	69.03	B	3.292	D	0.031	D
DesignX	DesignX_GSP_2_2_27_ProceedWithAspRestriction_group6	1.554	75.12	B	3.533	D	0.031	D
DesignX	DesignX_GSP_2_2_27_ProceedWithAspRestriction_group7	1.563	78.18	B	3.864	D	0.031	D
DesignX	DesignX_GSP_2_2_27_ProceedWithAspRestriction_group8	1.566	69.96	B	4.643	D	0.03	D

DesignX	DesignX_GSP_2_2_27_ProceedWithAspRestriction_group9	1.546	73.71	B	4.311	D	0.031	D
DesignX	DesignX_GSP_2_2_28_AspRestrictionReminderDevice_group0	1.42	65.48	A	0.52	E	0.031	E
DesignX	DesignX_GSP_2_2_28_AspRestrictionReminderDevice_group1	1.42	61.9	A	0.518	E	0.031	E
DesignX	DesignX_GSP_2_2_28_AspRestrictionReminderDevice_group10	1.415	62.38	A	0.389	A	0.031	A
DesignX	DesignX_GSP_2_2_28_AspRestrictionReminderDevice_group11	1.415	61.54	A	0.391	A	0.03	A
DesignX	DesignX_GSP_2_2_28_AspRestrictionReminderDevice_group12	1.42	62.37	A	0.517	E	0.03	E
DesignX	DesignX_GSP_2_2_28_AspRestrictionReminderDevice_group13	1.417	61.84	A	0.521	E	0.031	E
DesignX	DesignX_GSP_2_2_28_AspRestrictionReminderDevice_group2	1.421	61.71	A	0.522	E	0.03	E
DesignX	DesignX_GSP_2_2_28_AspRestrictionReminderDevice_group3	1.416	62.34	A	0.515	E	0.031	E
DesignX	DesignX_GSP_2_2_28_AspRestrictionReminderDevice_group4	1.419	62.18	A	0.52	E	0.031	E
DesignX	DesignX_GSP_2_2_28_AspRestrictionReminderDevice_group5	1.425	61.3	A	0.521	E	0.031	E
DesignX	DesignX_GSP_2_2_28_AspRestrictionReminderDevice_group6	1.419	62.04	A	0.521	E	0.031	E
DesignX	DesignX_GSP_2_2_28_AspRestrictionReminderDevice_group7	1.416	63.37	A	0.519	E	0.031	E
DesignX	DesignX_GSP_2_2_28_AspRestrictionReminderDevice_group8	1.415	63.29	A	0.391	A	0.031	A
DesignX	DesignX_GSP_2_2_28_AspRestrictionReminderDevice_group9	1.418	62.42	A	0.391	A	0.031	A
DesignX	DesignX_GSP_2_2_2_ProceedWithRouteSectionsLocked_group0	1.552	71.38	B	6.46	D	0.03	D
DesignX	DesignX_GSP_2_2_2_ProceedWithRouteSectionsLocked_group1	1.553	72	B	6.195	D	0.03	D
DesignX	DesignX_GSP_2_2_2_ProceedWithRouteSectionsLocked_group10	1.546	124.1	C	2.796	CT	0.031	CT
DesignX	DesignX_GSP_2_2_2_ProceedWithRouteSectionsLocked_group11	1.547	160.5	C	2.797	CT	0.03	CT
DesignX	DesignX_GSP_2_2_2_ProceedWithRouteSectionsLocked_group12	1.545	73.21	C	7.049	CT	0.03	CT
DesignX	DesignX_GSP_2_2_2_ProceedWithRouteSectionsLocked_group13	1.55	71.58	B	6.601	D	0.031	D
DesignX	DesignX_GSP_2_2_2_ProceedWithRouteSectionsLocked_group14	1.56	68.37	C	4.564	CT	0.031	CT
DesignX	DesignX_GSP_2_2_2_ProceedWithRouteSectionsLocked_group15	1.428	42.23	A	1.966	A	0.031	A
DesignX	DesignX_GSP_2_2_2_ProceedWithRouteSectionsLocked_group16	1.548	67.61	C	2.602	CT	0.031	CT
DesignX	DesignX_GSP_2_2_2_ProceedWithRouteSectionsLocked_group17	1.526	62.99	A	3.045	A	0.031	A
DesignX	DesignX_GSP_2_2_2_ProceedWithRouteSectionsLocked_group18	1.541	75.67	B	2.354	D	0.031	D
DesignX	DesignX_GSP_2_2_2_ProceedWithRouteSectionsLocked_group19	1.551	73.19	C	2.043	CT	0.031	CT
DesignX	DesignX_GSP_2_2_2_ProceedWithRouteSectionsLocked_group2	1.545	127.6	C	6.897	CT	0.031	CT
DesignX	DesignX_GSP_2_2_2_ProceedWithRouteSectionsLocked_group20	1.543	74.12	B	2.077	D	0.03	D
DesignX	DesignX_GSP_2_2_2_ProceedWithRouteSectionsLocked_group21	1.535	267.5	C	1.941	CT	0.031	CT
DesignX	DesignX_GSP_2_2_2_ProceedWithRouteSectionsLocked_group22	1.425	61.23	A	3.158	A	0.031	A
DesignX	DesignX_GSP_2_2_2_ProceedWithRouteSectionsLocked_group23	1.425	66.9	A	1.325	A	0.031	A
DesignX	DesignX_GSP_2_2_2_ProceedWithRouteSectionsLocked_group24	1.425	61.26	A	3.242	A	0.031	A
DesignX	DesignX_GSP_2_2_2_ProceedWithRouteSectionsLocked_group25	1.427	75.14	A	3.249	A	0.031	A
DesignX	DesignX_GSP_2_2_2_ProceedWithRouteSectionsLocked_group26	1.426	66.15	A	1.869	A	0.031	A
DesignX	DesignX_GSP_2_2_2_ProceedWithRouteSectionsLocked_group27	1.422	61.57	A	1.187	A	0.03	A
DesignX	DesignX_GSP_2_2_2_ProceedWithRouteSectionsLocked_group3	1.531	42.48	A	3.153	A	0.031	A
DesignX	DesignX_GSP_2_2_2_ProceedWithRouteSectionsLocked_group4	1.548	76.12	B	4.76	D	0.031	D
DesignX	DesignX_GSP_2_2_2_ProceedWithRouteSectionsLocked_group5	1.548	71.66	B	4.967	D	0.031	D
DesignX	DesignX_GSP_2_2_2_ProceedWithRouteSectionsLocked_group6	1.54	80.58	B	5.125	D	0.031	D
DesignX	DesignX_GSP_2_2_2_ProceedWithRouteSectionsLocked_group7	1.548	128.2	B	3.838	D	0.031	D
DesignX	DesignX_GSP_2_2_2_ProceedWithRouteSectionsLocked_group8	1.546	119.6	C	3.685	CT	0.031	CT
DesignX	DesignX_GSP_2_2_2_ProceedWithRouteSectionsLocked_group9	1.554	118.4	C	3.491	CT	0.031	CT
DesignX	DesignX_GSP_2_2_30_ProceedDisengagementAspRes_group0	1.421	63	A	0.397	A	0.03	A
DesignX	DesignX_GSP_2_2_30_ProceedDisengagementAspRes_group1	1.419	62.49	A	0.396	A	0.031	A
DesignX	DesignX_GSP_2_2_30_ProceedDisengagementAspRes_group10	1.423	61.25	A	0.393	A	0.03	A
DesignX	DesignX_GSP_2_2_30_ProceedDisengagementAspRes_group11	1.421	63.2	A	0.393	A	0.03	A
DesignX	DesignX_GSP_2_2_30_ProceedDisengagementAspRes_group12	1.426	61.71	A	0.393	A	0.031	A
DesignX	DesignX_GSP_2_2_30_ProceedDisengagementAspRes_group13	1.422	62.28	A	0.397	A	0.03	A
DesignX	DesignX_GSP_2_2_30_ProceedDisengagementAspRes_group14	1.427	42.47	A	0.398	A	0.031	A
DesignX	DesignX_GSP_2_2_30_ProceedDisengagementAspRes_group15	1.427	61.37	A	0.397	A	0.031	A
DesignX	DesignX_GSP_2_2_30_ProceedDisengagementAspRes_group16	1.422	61.85	A	0.4	A	0.03	A
DesignX	DesignX_GSP_2_2_30_ProceedDisengagementAspRes_group17	1.425	62.55	A	0.397	A	0.031	A
DesignX	DesignX_GSP_2_2_30_ProceedDisengagementAspRes_group18	1.418	61.48	A	0.398	A	0.03	A
DesignX	DesignX_GSP_2_2_30_ProceedDisengagementAspRes_group19	1.421	61.55	A	0.398	A	0.03	A
DesignX	DesignX_GSP_2_2_30_ProceedDisengagementAspRes_group2	1.423	64.22	A	0.398	A	0.031	A
DesignX	DesignX_GSP_2_2_30_ProceedDisengagementAspRes_group20	1.426	42.16	A	0.396	A	0.031	A
DesignX	DesignX_GSP_2_2_30_ProceedDisengagementAspRes_group21	1.42	62.81	A	0.398	A	0.031	A
DesignX	DesignX_GSP_2_2_30_ProceedDisengagementAspRes_group22	1.429	62.67	A	0.398	A	0.031	A
DesignX	DesignX_GSP_2_2_30_ProceedDisengagementAspRes_group23	1.432	62.28	A	0.401	A	0.03	A
DesignX	DesignX_GSP_2_2_30_ProceedDisengagementAspRes_group24	1.427	61.47	A	0.4	A	0.031	A
DesignX	DesignX_GSP_2_2_30_ProceedDisengagementAspRes_group25	1.433	62.35	A	0.395	A	0.03	A
DesignX	DesignX_GSP_2_2_30_ProceedDisengagementAspRes_group26	1.429	64.64	A	0.402	A	0.03	A
DesignX	DesignX_GSP_2_2_30_ProceedDisengagementAspRes_group27	1.429	65.38	A	0.4	A	0.031	A
DesignX	DesignX_GSP_2_2_30_ProceedDisengagementAspRes_group3	1.429	62.9	A	0.398	A	0.03	A
DesignX	DesignX_GSP_2_2_30_ProceedDisengagementAspRes_group4	1.422	62.57	A	0.396	A	0.03	A

DesignX	DesignX_GSP_2_2_30_ProceedDisengagementAspRes_group5	1.426	62.19	A	0.395	A	0.03	A
DesignX	DesignX_GSP_2_2_30_ProceedDisengagementAspRes_group6	1.421	61.59	A	0.397	A	0.03	A
DesignX	DesignX_GSP_2_2_30_ProceedDisengagementAspRes_group7	1.432	62.32	A	0.397	A	0.031	A
DesignX	DesignX_GSP_2_2_30_ProceedDisengagementAspRes_group8	1.425	61.36	A	0.397	A	0.031	A
DesignX	DesignX_GSP_2_2_30_ProceedDisengagementAspRes_group9	1.429	61.22	A	0.4	A	0.03	A
DesignX	DesignX_GSP_2_2_31_ProceedWithApproachLocking_group0	1.515	64.78	A	3.069	A	0.03	A
DesignX	DesignX_GSP_2_2_31_ProceedWithApproachLocking_group1	1.516	64.12	A	3.934	A	0.031	A
DesignX	DesignX_GSP_2_2_31_ProceedWithApproachLocking_group10	1.441	61.59	A	0.633	E	0.031	E
DesignX	DesignX_GSP_2_2_31_ProceedWithApproachLocking_group11	1.437	42.48	A	0.635	E	0.031	E
DesignX	DesignX_GSP_2_2_31_ProceedWithApproachLocking_group12	1.437	42.39	A	0.634	E	0.031	E
DesignX	DesignX_GSP_2_2_31_ProceedWithApproachLocking_group13	1.44	42.18	A	0.636	E	0.03	E
DesignX	DesignX_GSP_2_2_31_ProceedWithApproachLocking_group2	1.535	64.15	A	3.547	A	0.031	A
DesignX	DesignX_GSP_2_2_31_ProceedWithApproachLocking_group3	1.535	63.63	A	3.74	A	0.03	A
DesignX	DesignX_GSP_2_2_31_ProceedWithApproachLocking_group4	1.509	64.03	A	3.244	A	0.031	A
DesignX	DesignX_GSP_2_2_31_ProceedWithApproachLocking_group5	1.446	61.41	A	1.97	A	0.03	A
DesignX	DesignX_GSP_2_2_31_ProceedWithApproachLocking_group6	1.51	63.35	A	1.737	A	0.03	A
DesignX	DesignX_GSP_2_2_31_ProceedWithApproachLocking_group7	1.537	75.76	A	3.412	A	0.031	A
DesignX	DesignX_GSP_2_2_31_ProceedWithApproachLocking_group8	1.438	42.44	A	0.64	E	0.03	E
DesignX	DesignX_GSP_2_2_31_ProceedWithApproachLocking_group9	1.439	62.13	A	0.635	E	0.03	E
DesignX	DesignX_GSP_2_2_32_ProceedWithTPWSSuppressed_group0	1.509	62.23	A	2.28	A	0.031	A
DesignX	DesignX_GSP_2_2_32_ProceedWithTPWSSuppressed_group1	1.511	62.88	A	3.895	A	0.031	A
DesignX	DesignX_GSP_2_2_32_ProceedWithTPWSSuppressed_group10	1.416	42.38	A	3.295	A	0.03	A
DesignX	DesignX_GSP_2_2_32_ProceedWithTPWSSuppressed_group11	1.419	42.36	A	3.285	A	0.031	A
DesignX	DesignX_GSP_2_2_32_ProceedWithTPWSSuppressed_group12	1.416	62.61	A	1.898	A	0.031	A
DesignX	DesignX_GSP_2_2_32_ProceedWithTPWSSuppressed_group13	1.419	63.7	A	1.134	A	0.031	A
DesignX	DesignX_GSP_2_2_32_ProceedWithTPWSSuppressed_group2	1.414	62.85	A	3.538	A	0.031	A
DesignX	DesignX_GSP_2_2_32_ProceedWithTPWSSuppressed_group3	1.414	61.89	A	3.364	A	0.031	A
DesignX	DesignX_GSP_2_2_32_ProceedWithTPWSSuppressed_group4	1.511	62.09	A	4.111	A	0.03	A
DesignX	DesignX_GSP_2_2_32_ProceedWithTPWSSuppressed_group5	1.56	73.22	B	1.971	D	0.031	D
DesignX	DesignX_GSP_2_2_32_ProceedWithTPWSSuppressed_group6	1.508	62.61	A	2.434	A	0.031	A
DesignX	DesignX_GSP_2_2_32_ProceedWithTPWSSuppressed_group7	1.428	63.17	A	2.127	A	0.031	A
DesignX	DesignX_GSP_2_2_32_ProceedWithTPWSSuppressed_group8	1.417	67.33	A	3.233	A	0.031	A
DesignX	DesignX_GSP_2_2_32_ProceedWithTPWSSuppressed_group9	1.414	68	A	1.249	A	0.031	A
DesignX	DesignX_GSP_2_2_34_RYGAspectWithSOMOutput_group0	1.52	72.07	B	0.4	D	0.03	D
DesignX	DesignX_GSP_2_2_34_RYGAspectWithSOMOutput_group1	1.527	78.5	B	0.398	D	0.031	D
DesignX	DesignX_GSP_2_2_34_RYGAspectWithSOMOutput_group2	1.53	74.34	B	0.401	D	0.031	D
DesignX	DesignX_GSP_2_2_34_RYGAspectWithSOMOutput_group3	1.524	74.9	B	0.399	D	0.031	D
DesignX	DesignX_GSP_2_2_3_ProceedWithPointSetLockedDetected_group0	1.551	68.39	C	5.849	CT	0.031	CT
DesignX	DesignX_GSP_2_2_3_ProceedWithPointSetLockedDetected_group1	1.545	163.8	C	5.575	CT	0.031	CT
DesignX	DesignX_GSP_2_2_3_ProceedWithPointSetLockedDetected_group10	1.543	179.6	C	3.411	CT	0.03	CT
DesignX	DesignX_GSP_2_2_3_ProceedWithPointSetLockedDetected_group11	1.556	116.5	C	2.823	CT	0.031	CT
DesignX	DesignX_GSP_2_2_3_ProceedWithPointSetLockedDetected_group12	1.533	76.55	C	3.759	CT	0.031	CT
DesignX	DesignX_GSP_2_2_3_ProceedWithPointSetLockedDetected_group13	1.547	76.51	C	3.796	CT	0.031	CT
DesignX	DesignX_GSP_2_2_3_ProceedWithPointSetLockedDetected_group14	1.553	65.85	C	3.948	CT	0.031	CT
DesignX	DesignX_GSP_2_2_3_ProceedWithPointSetLockedDetected_group15	1.427	61.19	A	1.959	A	0.031	A
DesignX	DesignX_GSP_2_2_3_ProceedWithPointSetLockedDetected_group16	1.541	66.81	C	2.707	CT	0.03	CT
DesignX	DesignX_GSP_2_2_3_ProceedWithPointSetLockedDetected_group17	1.529	61.52	A	2.615	A	0.03	A
DesignX	DesignX_GSP_2_2_3_ProceedWithPointSetLockedDetected_group18	1.543	77.32	C	2.409	CT	0.031	CT
DesignX	DesignX_GSP_2_2_3_ProceedWithPointSetLockedDetected_group19	1.548	120.4	C	1.9	CT	0.031	CT
DesignX	DesignX_GSP_2_2_3_ProceedWithPointSetLockedDetected_group2	1.544	131.2	C	6.572	CT	0.031	CT
DesignX	DesignX_GSP_2_2_3_ProceedWithPointSetLockedDetected_group20	1.538	166.2	C	1.887	CT	0.03	CT
DesignX	DesignX_GSP_2_2_3_ProceedWithPointSetLockedDetected_group21	1.543	283.3	C	2.078	CT	0.03	CT
DesignX	DesignX_GSP_2_2_3_ProceedWithPointSetLockedDetected_group3	1.531	61.55	A	2.575	A	0.031	A
DesignX	DesignX_GSP_2_2_3_ProceedWithPointSetLockedDetected_group4	1.558	128.5	C	2.511	CT	0.031	CT
DesignX	DesignX_GSP_2_2_3_ProceedWithPointSetLockedDetected_group5	1.544	76.46	C	4.547	CT	0.031	CT
DesignX	DesignX_GSP_2_2_3_ProceedWithPointSetLockedDetected_group6	1.541	71.91	C	3.612	CT	0.03	CT
DesignX	DesignX_GSP_2_2_3_ProceedWithPointSetLockedDetected_group7	1.529	63.35	A	3.906	A	0.031	A
DesignX	DesignX_GSP_2_2_3_ProceedWithPointSetLockedDetected_group8	1.544	118.7	C	3.486	CT	0.031	CT
DesignX	DesignX_GSP_2_2_3_ProceedWithPointSetLockedDetected_group9	1.544	70.88	C	6.085	CT	0.031	CT
DesignX	DesignX_GSP_2_2_43_ProceedWithTechAspectDisable_group0	1.421	62.63	A	5.859	A	0.031	A
DesignX	DesignX_GSP_2_2_43_ProceedWithTechAspectDisable_group1	1.424	61.93	A	2.525	A	0.031	A
DesignX	DesignX_GSP_2_2_43_ProceedWithTechAspectDisable_group10	1.419	62.67	A	2.633	A	0.03	A
DesignX	DesignX_GSP_2_2_43_ProceedWithTechAspectDisable_group11	1.421	63.63	A	2.103	A	0.03	A
DesignX	DesignX_GSP_2_2_43_ProceedWithTechAspectDisable_group12	1.42	73.68	A	3.217	A	0.031	A
DesignX	DesignX_GSP_2_2_43_ProceedWithTechAspectDisable_group13	1.424	64.26	A	1.337	A	0.031	A
DesignX	DesignX_GSP_2_2_43_ProceedWithTechAspectDisable_group14	1.421	61.66	A	3.283	A	0.03	A

DesignX	DesignX_GSP_2_2_43_ProceedWithTechAspectDisable_group15	1.42	61.64	A	3.301	A	0.031	A
DesignX	DesignX_GSP_2_2_43_ProceedWithTechAspectDisable_group16	1.421	61.39	A	1.901	A	0.031	A
DesignX	DesignX_GSP_2_2_43_ProceedWithTechAspectDisable_group17	1.423	62.16	A	1.203	A	0.031	A
DesignX	DesignX_GSP_2_2_43_ProceedWithTechAspectDisable_group2	1.423	62.65	A	3.534	A	0.031	A
DesignX	DesignX_GSP_2_2_43_ProceedWithTechAspectDisable_group3	1.42	61.46	A	4.501	A	0.03	A
DesignX	DesignX_GSP_2_2_43_ProceedWithTechAspectDisable_group4	1.419	62.26	A	3.567	A	0.03	A
DesignX	DesignX_GSP_2_2_43_ProceedWithTechAspectDisable_group5	1.418	62.36	A	2.825	A	0.03	A
DesignX	DesignX_GSP_2_2_43_ProceedWithTechAspectDisable_group6	1.418	62.73	A	3.774	A	0.03	A
DesignX	DesignX_GSP_2_2_43_ProceedWithTechAspectDisable_group7	1.421	62.08	A	3.848	A	0.031	A
DesignX	DesignX_GSP_2_2_43_ProceedWithTechAspectDisable_group8	1.419	61.56	A	1.987	A	0.031	A
DesignX	DesignX_GSP_2_2_43_ProceedWithTechAspectDisable_group9	1.421	62.62	A	2.695	A	0.031	A
DesignX	DesignX_GSP_2_2_48_ProceedWithLODKRoute_group0	1.537	66.01	C	6.744	CT	0.031	CT
DesignX	DesignX_GSP_2_2_48_ProceedWithLODKRoute_group1	1.54	63.9	C	6.181	CT	0.031	CT
DesignX	DesignX_GSP_2_2_48_ProceedWithLODKRoute_group2	1.547	61.76	C	2.579	CT	0.031	CT
DesignX	DesignX_GSP_2_2_48_ProceedWithLODKRoute_group3	1.537	71.14	B	1.97	D	0.031	D
DesignX	DesignX_GSP_2_2_48_ProceedWithLODKRoute_group4	1.545	63.75	C	1.957	CT	0.031	CT
DesignX	DesignX_GSP_2_2_48_ProceedWithLODKRoute_group5	1.543	70.16	C	2.07	CT	0.031	CT
DesignX	DesignX_GSP_2_2_48_ProceedWithLODKRoute_group6	1.422	61.55	A	3.294	A	0.03	A
DesignX	DesignX_GSP_2_2_50_ProceedWithLODP_group0	1.532	122.2	B	6.179	D	0.03	D
DesignX	DesignX_GSP_2_2_51_ProceedWithSuppAWS_group0	1.523	70.6	B	5.492	D	0.031	D
DesignX	DesignX_GSP_2_2_51_ProceedWithSuppAWS_group1	1.54	73.8	B	2.522	D	0.031	D
DesignX	DesignX_GSP_2_2_51_ProceedWithSuppAWS_group10	1.535	73.37	B	2.062	D	0.031	D
DesignX	DesignX_GSP_2_2_51_ProceedWithSuppAWS_group11	1.535	122	B	3.184	D	0.03	D
DesignX	DesignX_GSP_2_2_51_ProceedWithSuppAWS_group12	1.541	72.63	B	1.309	D	0.03	D
DesignX	DesignX_GSP_2_2_51_ProceedWithSuppAWS_group13	1.536	74.65	B	3.266	D	0.03	D
DesignX	DesignX_GSP_2_2_51_ProceedWithSuppAWS_group14	1.544	71.98	B	3.241	D	0.031	D
DesignX	DesignX_GSP_2_2_51_ProceedWithSuppAWS_group15	1.549	121.6	B	1.875	D	0.031	D
DesignX	DesignX_GSP_2_2_51_ProceedWithSuppAWS_group16	1.535	75.78	B	1.08	D	0.03	D
DesignX	DesignX_GSP_2_2_51_ProceedWithSuppAWS_group2	1.526	73.32	B	3.548	D	0.03	D
DesignX	DesignX_GSP_2_2_51_ProceedWithSuppAWS_group3	1.544	72.8	B	3.907	D	0.03	D
DesignX	DesignX_GSP_2_2_51_ProceedWithSuppAWS_group4	1.541	71.14	B	4.395	D	0.031	D
DesignX	DesignX_GSP_2_2_51_ProceedWithSuppAWS_group5	1.525	126.4	B	3.462	D	0.031	D
DesignX	DesignX_GSP_2_2_51_ProceedWithSuppAWS_group6	1.536	81.91	B	3.635	D	0.031	D
DesignX	DesignX_GSP_2_2_51_ProceedWithSuppAWS_group7	1.549	72.8	B	3.803	D	0.031	D
DesignX	DesignX_GSP_2_2_51_ProceedWithSuppAWS_group8	1.527	80.06	B	2.217	D	0.031	D
DesignX	DesignX_GSP_2_2_51_ProceedWithSuppAWS_group9	1.545	72.72	B	2.77	D	0.03	D
DesignX	DesignX_GSP_2_2_52_ProceedWithPermissiveTrackProved_group0	1.536	70.56	B	2.724	D	0.03	D
DesignX	DesignX_GSP_2_2_52_ProceedWithPermissiveTrackProved_group1	1.535	69.29	B	3.925	D	0.031	D
DesignX	DesignX_GSP_2_2_52_ProceedWithPermissiveTrackProved_group2	1.539	71.45	B	2.716	D	0.031	D
DesignX	DesignX_GSP_2_2_53_ProceedWithExitRestrictive_group0	1.526	62.32	A	2.568	A	0.031	A
DesignX	DesignX_GSP_2_2_53_ProceedWithExitRestrictive_group1	1.525	61.33	A	4.014	A	0.031	A
DesignX	DesignX_GSP_2_2_53_ProceedWithExitRestrictive_group2	1.521	61.59	A	2.725	A	0.03	A
DesignX	DesignX_GSP_2_2_54_ProceedWithTPWSExit_group0	1.531	68.83	C	5.741	CT	0.031	CT
DesignX	DesignX_GSP_2_2_54_ProceedWithTPWSExit_group1	1.535	224.4	C	5.436	CT	0.031	CT
DesignX	DesignX_GSP_2_2_54_ProceedWithTPWSExit_group10	1.532	73.65	C	3.551	CT	0.031	CT
DesignX	DesignX_GSP_2_2_54_ProceedWithTPWSExit_group11	1.551	75.28	C	1.9	CT	0.031	CT
DesignX	DesignX_GSP_2_2_54_ProceedWithTPWSExit_group12	1.534	78.98	C	1.95	CT	0.031	CT
DesignX	DesignX_GSP_2_2_54_ProceedWithTPWSExit_group13	1.438	63.18	A	3.164	A	0.031	A
DesignX	DesignX_GSP_2_2_54_ProceedWithTPWSExit_group14	1.416	42.3	A	3.245	A	0.031	A
DesignX	DesignX_GSP_2_2_54_ProceedWithTPWSExit_group15	1.42	65.89	A	3.293	A	0.031	A
DesignX	DesignX_GSP_2_2_54_ProceedWithTPWSExit_group16	1.415	65.08	A	1.882	A	0.03	A
DesignX	DesignX_GSP_2_2_54_ProceedWithTPWSExit_group2	1.538	120.5	C	6.068	CT	0.031	CT
DesignX	DesignX_GSP_2_2_54_ProceedWithTPWSExit_group3	1.53	74.15	C	3.354	CT	0.031	CT
DesignX	DesignX_GSP_2_2_54_ProceedWithTPWSExit_group4	1.542	174	C	2.457	CT	0.03	CT
DesignX	DesignX_GSP_2_2_54_ProceedWithTPWSExit_group5	1.54	69.56	C	3.536	CT	0.03	CT
DesignX	DesignX_GSP_2_2_54_ProceedWithTPWSExit_group6	1.527	122	C	3.752	CT	0.031	CT
DesignX	DesignX_GSP_2_2_54_ProceedWithTPWSExit_group7	1.538	68.06	C	3.659	CT	0.031	CT
DesignX	DesignX_GSP_2_2_54_ProceedWithTPWSExit_group8	1.414	61.57	A	1.956	A	0.031	A
DesignX	DesignX_GSP_2_2_54_ProceedWithTPWSExit_group9	1.536	69.19	C	2.734	CT	0.03	CT
DesignX	DesignX_GSP_2_2_55_SignalDisengagement_group0	1.405	64.53	A	0.722	A	0.031	A
DesignX	DesignX_GSP_2_2_55_SignalDisengagement_group1	1.407	64.46	A	0.748	A	0.031	A
DesignX	DesignX_GSP_2_2_55_SignalDisengagement_group10	1.405	62.21	A	0.648	A	0.03	A
DesignX	DesignX_GSP_2_2_55_SignalDisengagement_group11	1.408	61.4	A	0.735	A	0.031	A
DesignX	DesignX_GSP_2_2_55_SignalDisengagement_group12	1.407	61.61	A	0.878	A	0.031	A
DesignX	DesignX_GSP_2_2_55_SignalDisengagement_group13	1.4	62.51	A	0.843	A	0.031	A
DesignX	DesignX_GSP_2_2_55_SignalDisengagement_group14	1.405	62.23	A	0.747	A	0.03	A

DesignX	DesignX_GSP_2_2_55_SignalDisengagement_group15	1.408	42.36	A	0.736	A	0.031	A
DesignX	DesignX_GSP_2_2_55_SignalDisengagement_group16	1.403	62.63	A	0.647	A	0.031	A
DesignX	DesignX_GSP_2_2_55_SignalDisengagement_group17	1.406	63.09	A	0.664	A	0.03	A
DesignX	DesignX_GSP_2_2_55_SignalDisengagement_group18	1.404	64.16	A	0.667	A	0.031	A
DesignX	DesignX_GSP_2_2_55_SignalDisengagement_group19	1.405	62.76	A	0.716	A	0.031	A
DesignX	DesignX_GSP_2_2_55_SignalDisengagement_group2	1.403	62.58	A	0.716	A	0.031	A
DesignX	DesignX_GSP_2_2_55_SignalDisengagement_group20	1.407	62.52	A	0.721	A	0.031	A
DesignX	DesignX_GSP_2_2_55_SignalDisengagement_group21	1.407	63.92	A	0.669	A	0.031	A
DesignX	DesignX_GSP_2_2_55_SignalDisengagement_group22	1.408	42.37	A	0.697	A	0.031	A
DesignX	DesignX_GSP_2_2_55_SignalDisengagement_group23	1.403	42.34	A	0.693	A	0.03	A
DesignX	DesignX_GSP_2_2_55_SignalDisengagement_group24	1.409	42.37	A	0.696	A	0.03	A
DesignX	DesignX_GSP_2_2_55_SignalDisengagement_group25	1.407	42.27	A	0.697	A	0.031	A
DesignX	DesignX_GSP_2_2_55_SignalDisengagement_group26	1.411	68.9	A	0.699	A	0.031	A
DesignX	DesignX_GSP_2_2_55_SignalDisengagement_group27	1.41	61.83	A	0.648	A	0.031	A
DesignX	DesignX_GSP_2_2_55_SignalDisengagement_group3	1.405	63.57	A	0.718	A	0.031	A
DesignX	DesignX_GSP_2_2_55_SignalDisengagement_group4	1.405	62.72	A	0.833	A	0.03	A
DesignX	DesignX_GSP_2_2_55_SignalDisengagement_group5	1.4	63.22	A	0.801	A	0.031	A
DesignX	DesignX_GSP_2_2_55_SignalDisengagement_group6	1.401	62.28	A	0.66	A	0.031	A
DesignX	DesignX_GSP_2_2_55_SignalDisengagement_group7	1.404	62.73	A	0.647	A	0.03	A
DesignX	DesignX_GSP_2_2_55_SignalDisengagement_group8	1.402	61.61	A	0.697	A	0.031	A
DesignX	DesignX_GSP_2_2_55_SignalDisengagement_group9	1.404	61.64	A	0.749	A	0.03	A
DesignX	DesignX_GSP_2_2_5_ProceedWithPointSLDOverlap_group0	1.554	124.6	C	6.165	CT	0.031	CT
DesignX	DesignX_GSP_2_2_5_ProceedWithPointSLDOverlap_group1	1.528	67.12	C	3.122	CT	0.031	CT
DesignX	DesignX_GSP_2_2_5_ProceedWithPointSLDOverlap_group2	1.536	128.3	C	2.405	CT	0.03	CT
DesignX	DesignX_GSP_2_2_5_ProceedWithPointSLDOverlap_group3	1.54	127.1	C	2.059	CT	0.031	CT
DesignX	DesignX_GSP_2_2_6_ProceedWithTracksClearPermissive_group0	1.519	62.72	A	1.983	A	0.031	A
DesignX	DesignX_GSP_2_2_6_ProceedWithTracksClearPermissive_group1	1.515	62.36	A	3.875	A	0.031	A
DesignX	DesignX_GSP_2_2_6_ProceedWithTracksClearPermissive_group2	1.515	61.6	A	2.772	A	0.031	A
DesignX	DesignX_GSP_2_2_7_ProceedWithTracksClear_group0	1.551	65.51	C	6.537	CT	0.031	CT
DesignX	DesignX_GSP_2_2_7_ProceedWithTracksClear_group1	1.542	180.2	C	6.843	CT	0.03	CT
DesignX	DesignX_GSP_2_2_7_ProceedWithTracksClear_group10	1.527	68.9	C	4.725	CT	0.03	CT
DesignX	DesignX_GSP_2_2_7_ProceedWithTracksClear_group11	1.535	73.62	C	3.734	CT	0.031	CT
DesignX	DesignX_GSP_2_2_7_ProceedWithTracksClear_group12	1.55	120.7	B	3.932	D	0.03	D
DesignX	DesignX_GSP_2_2_7_ProceedWithTracksClear_group13	1.543	63.87	C	1.962	CT	0.03	CT
DesignX	DesignX_GSP_2_2_7_ProceedWithTracksClear_group14	1.531	66.61	C	2.361	CT	0.03	CT
DesignX	DesignX_GSP_2_2_7_ProceedWithTracksClear_group15	1.529	73.4	C	2.406	CT	0.03	CT
DesignX	DesignX_GSP_2_2_7_ProceedWithTracksClear_group16	1.535	123.8	C	1.959	CT	0.03	CT
DesignX	DesignX_GSP_2_2_7_ProceedWithTracksClear_group17	1.532	119.7	C	2.071	CT	0.03	CT
DesignX	DesignX_GSP_2_2_7_ProceedWithTracksClear_group18	1.539	70.2	B	1.879	D	0.03	D
DesignX	DesignX_GSP_2_2_7_ProceedWithTracksClear_group19	1.519	42.59	A	3.179	A	0.03	A
DesignX	DesignX_GSP_2_2_7_ProceedWithTracksClear_group2	1.532	124.5	C	6.62	CT	0.03	CT
DesignX	DesignX_GSP_2_2_7_ProceedWithTracksClear_group20	1.536	71.65	B	1.36	D	0.03	D
DesignX	DesignX_GSP_2_2_7_ProceedWithTracksClear_group21	1.522	62.08	A	3.277	A	0.031	A
DesignX	DesignX_GSP_2_2_7_ProceedWithTracksClear_group22	1.522	67.32	A	3.273	A	0.03	A
DesignX	DesignX_GSP_2_2_7_ProceedWithTracksClear_group23	1.513	61.68	A	1.869	A	0.03	A
DesignX	DesignX_GSP_2_2_7_ProceedWithTracksClear_group24	1.52	62.91	A	1.248	A	0.03	A
DesignX	DesignX_GSP_2_2_7_ProceedWithTracksClear_group3	1.546	77.95	C	2.552	CT	0.03	CT
DesignX	DesignX_GSP_2_2_7_ProceedWithTracksClear_group4	1.537	128.9	C	3.44	CT	0.03	CT
DesignX	DesignX_GSP_2_2_7_ProceedWithTracksClear_group5	1.536	72.95	C	2.441	CT	0.03	CT
DesignX	DesignX_GSP_2_2_7_ProceedWithTracksClear_group6	1.536	160.7	C	3.697	CT	0.03	CT
DesignX	DesignX_GSP_2_2_7_ProceedWithTracksClear_group7	1.536	124.5	B	4.61	D	0.03	D
DesignX	DesignX_GSP_2_2_7_ProceedWithTracksClear_group8	1.538	117.6	C	3.131	CT	0.03	CT
DesignX	DesignX_GSP_2_2_7_ProceedWithTracksClear_group9	1.531	72.04	B	3.247	D	0.03	D
DesignX	DesignX_GSP_2_2_8_ProceedWithOverlapTracksClear_group0	1.538	66.36	C	5.666	CT	0.03	CT
DesignX	DesignX_GSP_2_2_8_ProceedWithOverlapTracksClear_group1	1.54	181.9	C	6.674	CT	0.031	CT
DesignX	DesignX_GSP_2_2_8_ProceedWithOverlapTracksClear_group10	1.421	64.54	A	3.176	A	0.03	A
DesignX	DesignX_GSP_2_2_8_ProceedWithOverlapTracksClear_group11	1.422	63.1	A	3.265	A	0.03	A
DesignX	DesignX_GSP_2_2_8_ProceedWithOverlapTracksClear_group12	1.419	61.64	A	3.249	A	0.03	A
DesignX	DesignX_GSP_2_2_8_ProceedWithOverlapTracksClear_group13	1.416	61.98	A	1.861	A	0.03	A
DesignX	DesignX_GSP_2_2_8_ProceedWithOverlapTracksClear_group2	1.539	120.7	C	6.132	CT	0.03	CT
DesignX	DesignX_GSP_2_2_8_ProceedWithOverlapTracksClear_group3	1.542	77.89	C	2.942	CT	0.03	CT
DesignX	DesignX_GSP_2_2_8_ProceedWithOverlapTracksClear_group4	1.538	73.15	C	4.922	CT	0.03	CT
DesignX	DesignX_GSP_2_2_8_ProceedWithOverlapTracksClear_group5	1.529	67.2	C	3.114	CT	0.03	CT
DesignX	DesignX_GSP_2_2_8_ProceedWithOverlapTracksClear_group6	1.532	70.02	C	3.835	CT	0.03	CT
DesignX	DesignX_GSP_2_2_8_ProceedWithOverlapTracksClear_group7	1.529	68.71	C	3.703	CT	0.03	CT
DesignX	DesignX_GSP_2_2_8_ProceedWithOverlapTracksClear_group8	1.536	70.84	C	2.365	CT	0.03	CT

DesignX	DesignX_GSP_2_2_8_ProceedWithOverlapTracksClear_group9	1.537	73.86	C	2.911	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group0	1.093	67.4	B	0.265	D	0.03	D
A	A_additional_A_A_GSP_ConflictingRoutes_group1	1.101	86.3	B	0.265	D	0.031	D
A	A_additional_A_A_GSP_ConflictingRoutes_group10	1.102	56.1	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group100	1.102	52.34	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group101	1.106	47.94	C	0.263	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group102	1.1	50.19	C	0.263	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group103	1.104	48.71	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group104	1.096	52.86	C	0.262	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group105	1.095	46.24	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group106	1.137	50.43	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group107	1.131	49.83	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group108	1.11	48.02	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group109	1.102	46.3	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group11	1.104	54.63	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group110	1.104	47.65	C	0.263	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group111	1.113	46.18	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group112	1.102	49.15	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group113	1.103	76.46	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group114	1.103	53.96	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group115	1.101	59.05	B	0.265	D	0.031	D
A	A_additional_A_A_GSP_ConflictingRoutes_group116	1.102	69.25	B	0.264	D	0.031	D
A	A_additional_A_A_GSP_ConflictingRoutes_group117	1.103	55.96	B	0.267	D	0.03	D
A	A_additional_A_A_GSP_ConflictingRoutes_group118	1.1	64.96	B	0.266	D	0.031	D
A	A_additional_A_A_GSP_ConflictingRoutes_group119	1.1	56.88	B	0.265	D	0.03	D
A	A_additional_A_A_GSP_ConflictingRoutes_group12	1.102	49.48	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group120	1.101	46.83	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group121	1.105	49.93	C	0.266	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group122	1.099	48.6	C	0.266	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group123	1.102	46.95	C	0.267	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group124	1.137	47.6	C	0.266	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group125	1.125	48.2	C	0.263	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group126	1.111	49.24	C	0.266	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group127	1.101	48.21	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group128	1.104	49.12	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group129	1.103	52.04	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group13	1.101	51.7	C	0.266	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group130	1.101	46.88	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group131	1.096	51.74	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group132	1.097	46.71	C	0.266	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group133	1.097	49.49	C	0.266	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group134	1.105	53.34	C	0.266	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group135	1.102	51.67	C	0.266	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group136	1.096	49.14	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group137	1.099	45.32	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group138	1.099	51.91	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group139	1.1	52.36	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group14	1.103	52.56	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group140	1.097	47.81	C	0.262	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group141	1.112	47.9	C	0.263	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group142	1.139	48.72	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group143	1.12	46.25	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group144	1.105	45.52	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group145	1.094	255.1	C	0.263	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group146	1.1	213.3	C	0.263	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group147	1.095	254	C	0.263	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group148	1.1	196.6	C	0.263	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group149	1.091	155.6	C	0.263	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group15	1.096	50.64	C	0.263	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group150	1.094	250.3	C	0.262	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group151	1.096	243.8	C	0.263	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group152	1.098	215.3	C	0.263	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group153	1.097	209.5	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group154	1.098	157.9	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group155	1.108	45.95	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group156	1.105	49.93	C	0.262	CT	0.031	CT

A	A_additional_A_A_GSP_ConflictingRoutes_group157	1.104	45.93	C	0.263	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group158	1.102	46.74	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group159	1.103	49.01	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group16	1.12	51.35	C	0.262	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group160	1.148	52.14	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group161	1.121	72.63	C	0.261	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group162	1.116	48.92	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group163	1.101	46.59	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group164	1.103	49.16	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group165	1.101	46.69	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group166	1.098	51.87	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group167	1.1	43.55	C	0.263	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group168	1.094	273.8	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group169	1.094	207.2	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group17	1.107	49.6	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group170	1.096	267.6	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group171	1.098	195.4	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group172	1.097	264.4	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group173	1.1	197.4	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group174	1.098	194.8	C	0.263	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group175	1.1	207.2	C	0.263	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group176	1.097	210.1	C	0.263	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group177	1.091	200.3	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group178	1.117	51	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group179	1.141	47.81	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group18	1.115	46.59	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group180	1.105	53.45	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group181	1.099	50.91	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group182	1.101	55.85	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group183	1.1	77.85	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group184	1.098	47.37	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group185	1.096	49.38	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group186	1.095	51.54	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group187	1.1	47.17	C	0.263	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group188	1.098	79.27	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group189	1.098	46.57	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group19	1.104	47.51	C	0.261	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group190	1.101	160.9	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group191	1.096	160.2	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group192	1.1	48.39	C	0.263	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group193	1.1	214.6	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group194	1.096	289.8	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group195	1.095	195.5	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group196	1.113	324.7	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group197	1.132	273.6	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group198	1.115	271.1	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group199	1.105	210.8	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group2	1.108	62.17	B	0.264	D	0.031	D
A	A_additional_A_A_GSP_ConflictingRoutes_group20	1.104	49.64	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group200	1.096	326.3	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group201	1.101	50.16	C	0.263	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group202	1.1	49.21	C	0.263	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group203	1.105	57.37	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group204	1.101	48.04	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group205	1.098	54.99	C	0.263	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group206	1.099	48.77	C	0.263	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group207	1.099	49.34	C	0.266	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group208	1.101	49.74	C	0.261	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group209	1.102	49.15	C	0.262	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group21	1.101	51.27	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group210	1.105	51.22	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group211	1.098	51.34	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group212	1.101	47.52	C	0.263	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group213	1.118	204.8	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group214	1.132	271	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group215	1.115	44.07	C	0.262	CT	0.031	CT

A	A_additional_A_A_GSP_ConflictingRoutes_group216	1.104	198.2	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group217	1.097	211.4	C	0.262	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group218	1.099	319.4	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group219	1.097	267.1	C	0.263	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group22	1.098	51.03	C	0.262	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group220	1.094	203	C	0.262	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group221	1.097	283.7	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group222	1.093	197.1	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group223	1.094	343.9	C	0.263	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group224	1.103	48.63	C	0.263	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group225	1.101	46.94	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group226	1.1	51.91	C	0.263	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group227	1.102	47.34	C	0.263	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group228	1.105	52.06	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group229	1.104	49.9	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group23	1.102	60.43	B	0.268	D	0.03	D
A	A_additional_A_A_GSP_ConflictingRoutes_group230	1.099	50.8	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group231	1.121	47.86	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group232	1.139	49.57	C	0.263	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group233	1.115	48.93	C	0.263	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group234	1.112	48.65	C	0.263	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group235	1.097	52.49	C	0.263	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group236	1.095	223.4	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group237	1.099	256.4	C	0.263	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group238	1.1	241.4	C	0.262	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group239	1.097	271.3	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group24	1.101	55.8	B	0.264	D	0.031	D
A	A_additional_A_A_GSP_ConflictingRoutes_group240	1.097	44.26	C	0.263	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group241	1.094	346.9	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group242	1.094	281.9	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group243	1.098	270.4	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group244	1.097	290.8	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group245	1.099	220.2	C	0.266	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group246	1.096	303	C	0.262	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group247	1.103	49.65	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group248	1.099	53.34	C	0.266	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group249	1.099	52.45	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group25	1.125	56.52	B	0.264	D	0.031	D
A	A_additional_A_A_GSP_ConflictingRoutes_group250	1.129	49.94	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group251	1.116	50.74	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group252	1.106	50.85	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group253	1.101	46.54	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group254	1.1	52.46	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group255	1.1	50.55	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group256	1.107	49.86	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group257	1.102	48.41	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group258	1.097	50.16	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group259	1.096	223	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group26	1.099	57.9	B	0.263	D	0.031	D
A	A_additional_A_A_GSP_ConflictingRoutes_group260	1.096	250.7	C	0.263	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group261	1.1	258.8	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group262	1.101	257.8	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group263	1.102	45.09	C	0.263	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group264	1.102	298.4	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group265	1.099	287.8	C	0.266	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group266	1.093	269.6	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group267	1.096	328.6	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group268	1.13	224.7	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group269	1.121	223.2	C	0.263	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group27	1.115	59	B	0.264	D	0.031	D
A	A_additional_A_A_GSP_ConflictingRoutes_group270	1.106	47.67	C	0.263	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group271	1.1	47.16	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group272	1.102	48.55	C	0.263	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group273	1.103	46.98	C	0.263	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group274	1.1	53.7	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group275	1.1	52.1	C	0.263	CT	0.031	CT

A	A_additional_A_A_GSP_ConflictingRoutes_group276	1.097	49.19	C	0.263	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group277	1.098	51.37	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group278	1.097	75.85	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group279	1.099	46.17	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group28	1.101	47.21	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group280	1.1	50.73	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group281	1.1	51.66	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group282	1.099	218.6	C	0.263	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group283	1.098	213.4	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group284	1.095	207.1	C	0.263	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group285	1.095	206.2	C	0.263	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group286	1.134	323.1	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group287	1.124	359	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group288	1.111	45.24	C	0.263	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group289	1.103	298.9	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group29	1.099	49.39	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group290	1.097	203	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group291	1.098	271.8	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group292	1.098	259.2	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group293	1.101	46.31	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group294	1.096	52.09	C	0.263	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group295	1.098	49.51	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group296	1.101	47.31	C	0.263	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group297	1.1	49.6	C	0.262	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group298	1.103	46.93	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group299	1.102	47.38	C	0.263	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group3	1.1	58.55	B	0.264	D	0.031	D
A	A_additional_A_A_GSP_ConflictingRoutes_group30	1.103	49.6	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group300	1.101	46.65	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group301	1.096	47.15	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group302	1.093	54.47	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group303	1.137	57.82	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group304	1.123	48.78	C	0.262	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group305	1.111	207.2	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group306	1.099	199.1	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group307	1.095	260	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group308	1.1	269.8	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group309	1.098	351.3	C	0.263	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group31	1.105	75.15	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group310	1.094	314.9	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group311	1.097	44.02	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group312	1.094	213.3	C	0.263	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group313	1.1	292.9	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group314	1.098	208.9	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group315	1.095	284.6	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group316	1.103	47.26	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group317	1.105	44.93	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group318	1.103	47.02	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group319	1.104	50.94	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group32	1.1	47.16	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group320	1.098	46.38	C	0.263	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group321	1.142	49.12	C	0.263	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group322	1.124	48.66	C	0.263	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group323	1.11	52.78	C	0.263	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group324	1.098	47.47	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group325	1.097	50.79	C	0.263	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group326	1.1	50.29	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group327	1.102	49.88	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group328	1.099	261	C	0.266	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group329	1.094	222.3	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group33	1.098	53.25	C	0.263	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group330	1.096	214.8	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group331	1.09	265.6	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group332	1.098	214.2	C	0.266	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group333	1.096	284.5	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group334	1.092	267.2	C	0.264	CT	0.031	CT

A	A_additional_A_A_GSP_ConflictingRoutes_group335	1.1	327.8	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group336	1.099	45.83	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group337	1.098	356.8	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group338	1.096	342.6	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group339	1.093	49	C	0.263	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group34	1.141	55.97	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group340	1.129	47.04	C	0.262	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group341	1.111	47.59	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group342	1.104	46.56	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group343	1.1	50.79	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group344	1.1	75.55	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group345	1.099	49.37	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group346	1.099	48.05	C	0.266	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group347	1.1	51.98	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group348	1.1	54.65	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group349	1.099	49.38	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group35	1.103	55.42	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group350	1.099	49.48	C	0.263	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group351	1.098	204.9	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group352	1.104	257.9	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group353	1.099	266.2	C	0.262	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group354	1.101	194.4	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group355	1.099	262.2	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group356	1.095	294.3	C	0.266	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group357	1.094	338.9	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group358	1.144	215.2	C	0.266	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group359	1.132	43.85	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group36	1.111	78.21	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group360	1.099	218.4	C	0.267	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group361	1.096	291.5	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group362	1.1	48.79	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group363	1.101	49.73	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group364	1.103	46.64	C	0.263	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group365	1.097	50.84	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group366	1.102	49.7	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group367	1.099	50.74	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group368	1.097	46.06	C	0.266	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group369	1.1	44.93	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group37	1.106	49.29	C	0.263	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group370	1.108	72.43	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group371	1.105	48.9	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group372	1.109	49.34	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group373	1.108	47.78	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group374	1.092	194.9	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group375	1.101	216.7	C	0.266	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group376	1.133	285.7	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group377	1.124	327.4	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group378	1.108	260.3	C	0.267	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group379	1.097	207.1	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group38	1.105	47.82	C	0.266	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group380	1.095	263.3	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group381	1.1	266.3	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group382	1.097	206.9	C	0.263	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group383	1.095	270.9	C	0.266	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group384	1.095	42.84	C	0.263	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group385	1.099	47.12	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group386	1.098	48.45	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group387	1.099	49.45	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group388	1.102	47.31	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group389	1.108	49.61	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group39	1.102	46.19	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group390	1.102	50.24	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group391	1.108	46.96	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group392	1.098	50.14	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group393	1.108	48.87	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group394	1.138	51.1	C	0.265	CT	0.03	CT

A	A_additional_A_A_GSP_ConflictingRoutes_group395	1.122	48.63	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group396	1.111	46.86	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group397	1.093	163.3	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group398	1.095	155.8	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group399	1.093	210.2	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group4	1.1	59.89	B	0.265	D	0.03	D
A	A_additional_A_A_GSP_ConflictingRoutes_group40	1.098	47.54	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group400	1.095	273.6	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group401	1.093	289.4	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group402	1.095	366.4	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group403	1.095	273.5	C	0.266	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group404	1.096	269.6	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group405	1.1	259	C	0.266	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group406	1.098	304.1	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group407	1.099	54.44	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group408	1.103	54.37	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group409	1.098	51.99	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group41	1.101	46.81	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group410	1.117	50.05	C	0.262	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group411	1.139	46.89	C	0.263	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group412	1.119	46.97	C	0.263	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group413	1.108	50.6	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group414	1.101	54.92	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group415	1.102	47.66	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group416	1.1	47.03	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group417	1.103	51.21	C	0.267	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group418	1.104	49.06	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group419	1.098	49.69	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group42	1.102	47.69	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group420	1.097	48.93	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group421	1.102	53.95	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group422	1.1	48.15	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group423	1.101	48.76	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group424	1.103	105.5	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group425	1.102	47.67	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group426	1.098	52.83	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group427	1.099	53.31	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group428	1.097	52.06	C	0.267	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group429	1.12	46.58	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group43	1.14	49.78	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group430	1.114	46.89	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group431	1.107	46.52	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group432	1.101	56.6	B	0.265	D	0.031	D
A	A_additional_A_A_GSP_ConflictingRoutes_group433	1.099	66.9	B	0.267	D	0.031	D
A	A_additional_A_A_GSP_ConflictingRoutes_group434	1.097	55.4	B	0.264	D	0.031	D
A	A_additional_A_A_GSP_ConflictingRoutes_group435	1.104	46.48	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group436	1.101	51.91	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group437	1.099	45.83	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group438	1.095	47.51	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group439	1.096	46.09	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group44	1.101	46.92	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group440	1.098	46.15	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group441	1.098	48.87	C	0.266	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group442	1.105	48.28	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group443	1.1	46.21	C	0.266	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group444	1.1	47.01	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group445	1.091	50.87	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group446	1.097	47.3	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group447	1.138	48.11	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group448	1.133	47.38	C	0.266	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group449	1.109	73.62	C	0.266	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group45	1.103	50.5	C	0.266	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group450	1.098	49.56	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group451	1.099	50.67	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group452	1.105	47.83	C	0.263	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group453	1.098	48.74	C	0.267	CT	0.031	CT

A	A_additional_A_A_GSP_ConflictingRoutes_group454	1.095	48.87	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group455	1.097	57.88	B	0.263	D	0.031	D
A	A_additional_A_A_GSP_ConflictingRoutes_group456	1.098	44.99	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group457	1.099	57.77	B	0.266	D	0.031	D
A	A_additional_A_A_GSP_ConflictingRoutes_group458	1.103	54.02	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group459	1.103	49.32	C	0.266	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group46	1.103	57.27	B	0.264	D	0.031	D
A	A_additional_A_A_GSP_ConflictingRoutes_group460	1.102	52.02	C	0.267	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group461	1.103	48.11	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group462	1.107	47.87	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group463	1.104	49.64	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group464	1.096	48.32	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group465	1.148	47.66	C	0.267	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group466	1.127	50.27	C	0.266	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group467	1.112	50.85	C	0.266	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group468	1.104	49.71	C	0.263	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group469	1.1	50.32	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group47	1.103	56.41	B	0.263	D	0.031	D
A	A_additional_A_A_GSP_ConflictingRoutes_group470	1.096	48.38	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group471	1.101	51.4	C	0.266	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group472	1.094	52.15	C	0.263	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group473	1.102	47.69	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group474	1.1	49.78	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group475	1.105	50.01	C	0.263	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group476	1.097	47.73	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group477	1.102	50.21	C	0.267	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group478	1.098	58.08	B	0.265	D	0.031	D
A	A_additional_A_A_GSP_ConflictingRoutes_group479	1.099	44.7	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group48	1.104	57	B	0.265	D	0.03	D
A	A_additional_A_A_GSP_ConflictingRoutes_group480	1.104	56.69	B	0.265	D	0.031	D
A	A_additional_A_A_GSP_ConflictingRoutes_group481	1.098	48.01	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group482	1.099	51.96	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group483	1.143	49.54	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group484	1.135	48.5	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group485	1.111	47.55	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group486	1.103	46.82	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group487	1.106	53.15	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group488	1.1	47.5	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group489	1.099	50.4	C	0.266	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group49	1.104	56.97	B	0.263	D	0.031	D
A	A_additional_A_A_GSP_ConflictingRoutes_group490	1.099	52.69	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group491	1.096	53.66	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group492	1.095	46.7	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group493	1.095	54.13	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group494	1.099	55.61	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group495	1.098	49.68	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group496	1.1	50.12	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group497	1.096	49.81	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group498	1.099	77.51	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group499	1.098	52.05	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group5	1.094	53.39	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group50	1.107	58.9	B	0.265	D	0.03	D
A	A_additional_A_A_GSP_ConflictingRoutes_group500	1.138	51.43	C	0.263	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group501	1.125	60.06	B	0.267	D	0.031	D
A	A_additional_A_A_GSP_ConflictingRoutes_group502	1.108	67.18	B	0.263	D	0.031	D
A	A_additional_A_A_GSP_ConflictingRoutes_group503	1.098	58.22	B	0.265	D	0.03	D
A	A_additional_A_A_GSP_ConflictingRoutes_group504	1.104	51.3	C	0.266	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group505	1.103	49.65	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group506	1.106	60.99	B	0.265	D	0.03	D
A	A_additional_A_A_GSP_ConflictingRoutes_group507	1.108	57.01	B	0.265	D	0.031	D
A	A_additional_A_A_GSP_ConflictingRoutes_group508	1.099	51.73	C	0.266	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group509	1.1	50.02	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group51	1.096	54.28	C	0.266	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group510	1.101	47.56	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group511	1.104	48.8	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group512	1.105	56.97	C	0.265	CT	0.03	CT

A	A_additional_A_A_GSP_ConflictingRoutes_group513	1.102	51.02	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group514	1.103	49.88	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group515	1.104	53.2	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group516	1.1	50.77	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group517	1.097	51.46	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group518	1.109	49.33	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group519	1.139	51.63	C	0.266	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group52	1.122	49.92	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group520	1.105	54.51	C	0.263	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group521	1.098	53.91	C	0.266	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group522	1.098	50.87	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group523	1.099	50.03	C	0.266	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group524	1.098	53.46	C	0.266	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group525	1.094	53.58	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group526	1.096	71.47	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group527	1.098	50.81	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group528	1.096	52.38	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group529	1.101	47.79	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group53	1.102	47.1	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group530	1.101	56.35	B	0.263	D	0.031	D
A	A_additional_A_A_GSP_ConflictingRoutes_group531	1.101	49.56	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group532	1.103	51.57	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group533	1.105	48.69	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group534	1.105	48.41	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group535	1.105	47.62	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group536	1.115	50.06	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group537	1.139	46.69	C	0.267	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group538	1.121	54.3	C	0.266	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group539	1.107	50.73	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group54	1.1	52.2	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group540	1.101	47.24	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group541	1.102	48.05	C	0.266	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group542	1.097	48.18	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group543	1.102	53.53	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group544	1.099	49.85	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group545	1.099	54.46	C	0.266	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group546	1.099	49.28	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group547	1.099	50.1	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group548	1.101	51.72	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group549	1.101	46.96	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group55	1.103	51.79	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group550	1.106	51.16	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group551	1.106	54.95	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group552	1.105	49.83	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group553	1.1	66.21	B	0.264	D	0.031	D
A	A_additional_A_A_GSP_ConflictingRoutes_group554	1.116	56.95	B	0.265	D	0.03	D
A	A_additional_A_A_GSP_ConflictingRoutes_group555	1.143	51.18	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group556	1.118	48.04	C	0.263	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group557	1.111	50.51	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group558	1.1	47.47	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group559	1.097	50.03	C	0.266	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group56	1.099	49.43	C	0.263	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group560	1.103	56.38	B	0.265	D	0.03	D
A	A_additional_A_A_GSP_ConflictingRoutes_group561	1.099	52.64	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group562	1.096	52.44	C	0.266	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group563	1.092	51.08	C	0.266	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group564	1.094	89.65	B	0.264	D	0.03	D
A	A_additional_A_A_GSP_ConflictingRoutes_group565	1.098	47.71	C	0.266	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group566	1.093	47.92	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group567	1.097	47.7	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group568	1.098	47.58	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group569	1.099	49.65	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group57	1.103	50.66	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group570	1.097	68.44	B	0.266	D	0.03	D
A	A_additional_A_A_GSP_ConflictingRoutes_group571	1.094	57.69	B	0.264	D	0.031	D
A	A_additional_A_A_GSP_ConflictingRoutes_group572	1.115	49.91	C	0.265	CT	0.03	CT

A	A_additional_A_A_GSP_ConflictingRoutes_group573	1.137	51.58	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group574	1.119	52.08	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group575	1.106	58.56	B	0.265	D	0.03	D
A	A_additional_A_A_GSP_ConflictingRoutes_group58	1.098	53	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group59	1.104	47.77	C	0.267	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group6	1.1	48.5	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group60	1.102	53.88	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group61	1.102	50.85	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group62	1.097	50.71	C	0.263	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group63	1.101	49.05	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group64	1.103	47.22	C	0.266	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group65	1.105	48.61	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group66	1.104	51.32	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group67	1.105	54.02	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group68	1.1	50.52	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group69	1.1	58.93	B	0.266	D	0.031	D
A	A_additional_A_A_GSP_ConflictingRoutes_group7	1.107	49.04	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group70	1.103	58.72	B	0.264	D	0.03	D
A	A_additional_A_A_GSP_ConflictingRoutes_group71	1.102	59.32	B	0.263	D	0.03	D
A	A_additional_A_A_GSP_ConflictingRoutes_group72	1.125	63.28	B	0.265	D	0.031	D
A	A_additional_A_A_GSP_ConflictingRoutes_group73	1.145	58.26	B	0.263	D	0.03	D
A	A_additional_A_A_GSP_ConflictingRoutes_group74	1.125	49.03	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group75	1.115	50.16	C	0.266	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group76	1.1	47.13	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group77	1.101	47.54	C	0.263	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group78	1.105	48.45	C	0.262	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group79	1.101	49.73	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group8	1.096	55.26	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group80	1.099	46.88	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group81	1.099	52.74	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group82	1.101	49.91	C	0.263	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group83	1.1	47.4	C	0.266	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group84	1.099	50.57	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group85	1.101	48.46	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group86	1.102	46.38	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group87	1.102	46.56	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group88	1.108	48.36	C	0.263	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group89	1.101	47.32	C	0.263	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group9	1.103	55.27	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group90	1.122	47.87	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group91	1.133	49.77	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group92	1.117	58.79	B	0.266	D	0.03	D
A	A_additional_A_A_GSP_ConflictingRoutes_group93	1.105	57.7	B	0.264	D	0.031	D
A	A_additional_A_A_GSP_ConflictingRoutes_group94	1.103	55.88	B	0.264	D	0.031	D
A	A_additional_A_A_GSP_ConflictingRoutes_group95	1.103	56.5	B	0.264	D	0.03	D
A	A_additional_A_A_GSP_ConflictingRoutes_group96	1.101	75.18	B	0.264	D	0.031	D
A	A_additional_A_A_GSP_ConflictingRoutes_group97	1.101	49.63	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group98	1.097	46.5	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_group99	1.101	48.41	C	0.266	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_Fast_group0	1.1	56.67	B	0.272	D	0.03	D
A	A_additional_A_A_GSP_ConflictingRoutes_Fast_group1	1.1	62.05	B	0.273	D	0.03	D
A	A_additional_A_A_GSP_ConflictingRoutes_Fast_group10	1.103	542.5	C	0.273	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_Fast_group11	1.1	1018	C	0.271	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_Fast_group12	1.099	957.6	C	0.271	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_Fast_group13	1.102	831.6	C	0.27	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_Fast_group14	1.102	692.1	C	0.273	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_Fast_group15	1.101	1724	C	0.271	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_Fast_group16	1.101	940.5	C	0.271	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_Fast_group17	1.1	1003	C	0.27	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_Fast_group18	1.121	59.18	B	0.271	D	0.03	D
A	A_additional_A_A_GSP_ConflictingRoutes_Fast_group19	1.137	60.09	B	0.272	D	0.03	D
A	A_additional_A_A_GSP_ConflictingRoutes_Fast_group2	1.116	57.58	B	0.276	D	0.03	D
A	A_additional_A_A_GSP_ConflictingRoutes_Fast_group20	1.105	55.18	B	0.27	D	0.03	D
A	A_additional_A_A_GSP_ConflictingRoutes_Fast_group21	1.103	57.68	B	0.272	D	0.03	D
A	A_additional_A_A_GSP_ConflictingRoutes_Fast_group22	1.097	56.08	B	0.266	D	0.03	D
A	A_additional_A_A_GSP_ConflictingRoutes_Fast_group23	1.101	59.17	B	0.274	D	0.03	D

A	A_additional_A_A_GSP_ConflictingRoutes_Fast_group24	1.102	58.68	B	0.272	D	0.03	D
A	A_additional_A_A_GSP_ConflictingRoutes_Fast_group25	1.098	56.4	B	0.262	D	0.03	D
A	A_additional_A_A_GSP_ConflictingRoutes_Fast_group26	1.101	110.6	C	0.268	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_Fast_group27	1.095	56.18	B	0.267	D	0.03	D
A	A_additional_A_A_GSP_ConflictingRoutes_Fast_group28	1.098	56.73	B	0.267	D	0.03	D
A	A_additional_A_A_GSP_ConflictingRoutes_Fast_group29	1.105	57.08	B	0.269	D	0.03	D
A	A_additional_A_A_GSP_ConflictingRoutes_Fast_group3	1.101	92.39	B	0.273	D	0.031	D
A	A_additional_A_A_GSP_ConflictingRoutes_Fast_group30	1.105	55.83	B	0.266	D	0.03	D
A	A_additional_A_A_GSP_ConflictingRoutes_Fast_group4	1.1	56.48	B	0.275	D	0.03	D
A	A_additional_A_A_GSP_ConflictingRoutes_Fast_group5	1.103	61.49	B	0.273	D	0.03	D
A	A_additional_A_A_GSP_ConflictingRoutes_Fast_group6	1.105	701.6	C	0.274	CT	0.031	CT
A	A_additional_A_A_GSP_ConflictingRoutes_Fast_group7	1.097	585.2	C	0.269	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_Fast_group8	1.1	582.2	C	0.272	CT	0.03	CT
A	A_additional_A_A_GSP_ConflictingRoutes_Fast_group9	1.127	518.8	C	0.272	CT	0.031	CT
A	A_additional_A_A_GSP_FirstOverlapWithPreviousRouteSectionRel_Fast_group0	1.129	57.76	B	0.505	D	0.03	D
A	A_additional_A_A_GSP_FirstOverlapWithPreviousRouteSectionRel_Fast_group1	1.037	42.63	A	0.45	A	0.03	A
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group0	1.107	46.06	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group1	1.098	42.81	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group10	1.093	46.88	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group11	1.096	44.98	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group12	1.097	45.3	C	0.266	CT	0.031	CT
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group13	1.091	48.11	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group14	1.094	44.47	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group15	1.092	44.65	C	0.267	CT	0.031	CT
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group16	1.094	45.82	C	0.267	CT	0.03	CT
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group17	1.098	44.58	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group18	1.098	44.35	C	0.267	CT	0.031	CT
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group19	1.099	45.22	C	0.266	CT	0.031	CT
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group2	1.092	44.25	C	0.266	CT	0.031	CT
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group20	1.095	44.82	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group21	1.099	44.47	C	0.267	CT	0.03	CT
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group22	1.097	44.97	C	0.267	CT	0.03	CT
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group23	1.094	43.98	C	0.267	CT	0.031	CT
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group24	1.123	46.33	C	0.266	CT	0.031	CT
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group25	1.119	45.2	C	0.266	CT	0.03	CT
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group26	1.107	44.63	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group27	1.104	58.11	B	0.263	D	0.03	D
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group28	1.103	46.93	C	0.266	CT	0.031	CT
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group29	1.092	46.49	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group3	1.099	43.41	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group30	1.099	42.88	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group31	1.087	56.21	B	0.265	D	0.03	D
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group32	1.095	76.63	C	0.266	CT	0.031	CT
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group33	1.094	44.55	C	0.267	CT	0.03	CT
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group34	1.097	46.71	C	0.266	CT	0.031	CT
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group35	1.097	60.13	B	0.266	D	0.03	D
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group36	1.097	44.79	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group37	1.098	44.84	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group38	1.101	56.12	B	0.265	D	0.03	D
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group39	1.093	43.33	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group4	1.108	46.26	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group40	1.1	43.95	C	0.266	CT	0.03	CT
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group41	1.096	81.5	C	0.267	CT	0.031	CT
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group42	1.124	46.38	C	0.269	CT	0.03	CT
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group43	1.13	57.25	B	0.266	D	0.03	D
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group44	1.116	44.52	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group5	1.109	44.28	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group6	1.101	44.71	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group7	1.104	45.97	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group8	1.103	43.47	C	0.267	CT	0.031	CT
A	A_additional_A_A_GSP_FirstRouteSectionReleaseWithLocking_group9	1.1	45.24	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_OpposingRoutes_Fast_group0	1.096	46.78	C	0.267	CT	0.03	CT
A	A_additional_A_A_GSP_OpposingRoutes_Fast_group1	1.1	45.89	C	0.268	CT	0.031	CT
A	A_additional_A_A_GSP_OpposingRoutes_Fast_group10	1.097	44.61	C	0.266	CT	0.03	CT
A	A_additional_A_A_GSP_OpposingRoutes_Fast_group11	1.101	47.78	C	0.267	CT	0.031	CT
A	A_additional_A_A_GSP_OpposingRoutes_Fast_group12	1.1	47.66	C	0.268	CT	0.031	CT

A	A_additional_A_A_GSP_OpposingRoutes_Fast_group13	1.103	49.15	C	0.267	CT	0.03	CT
A	A_additional_A_A_GSP_OpposingRoutes_Fast_group14	1.103	47.3	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_OpposingRoutes_Fast_group15	1.102	45.95	C	0.267	CT	0.031	CT
A	A_additional_A_A_GSP_OpposingRoutes_Fast_group16	1.1	46.57	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_OpposingRoutes_Fast_group17	1.099	48.77	C	0.267	CT	0.031	CT
A	A_additional_A_A_GSP_OpposingRoutes_Fast_group18	1.101	46.6	C	0.267	CT	0.031	CT
A	A_additional_A_A_GSP_OpposingRoutes_Fast_group19	1.104	45.47	C	0.268	CT	0.03	CT
A	A_additional_A_A_GSP_OpposingRoutes_Fast_group2	1.128	56.14	C	0.268	CT	0.03	CT
A	A_additional_A_A_GSP_OpposingRoutes_Fast_group20	1.133	47.59	C	0.269	CT	0.031	CT
A	A_additional_A_A_GSP_OpposingRoutes_Fast_group21	1.117	46.18	C	0.268	CT	0.031	CT
A	A_additional_A_A_GSP_OpposingRoutes_Fast_group22	1.105	66.29	B	0.268	D	0.03	D
A	A_additional_A_A_GSP_OpposingRoutes_Fast_group23	1.098	48.19	C	0.269	CT	0.031	CT
A	A_additional_A_A_GSP_OpposingRoutes_Fast_group24	1.107	51.04	C	0.268	CT	0.03	CT
A	A_additional_A_A_GSP_OpposingRoutes_Fast_group25	1.096	45.54	C	0.27	CT	0.031	CT
A	A_additional_A_A_GSP_OpposingRoutes_Fast_group26	1.099	57.3	C	0.276	CT	0.03	CT
A	A_additional_A_A_GSP_OpposingRoutes_Fast_group27	1.096	111.6	C	0.273	CT	0.031	CT
A	A_additional_A_A_GSP_OpposingRoutes_Fast_group28	1.097	112.2	C	0.274	CT	0.031	CT
A	A_additional_A_A_GSP_OpposingRoutes_Fast_group29	1.102	108	C	0.274	CT	0.03	CT
A	A_additional_A_A_GSP_OpposingRoutes_Fast_group3	1.099	49.3	C	0.267	CT	0.031	CT
A	A_additional_A_A_GSP_OpposingRoutes_Fast_group30	1.105	55.87	B	0.275	D	0.03	D
A	A_additional_A_A_GSP_OpposingRoutes_Fast_group4	1.103	45.46	C	0.269	CT	0.031	CT
A	A_additional_A_A_GSP_OpposingRoutes_Fast_group5	1.102	46.11	C	0.268	CT	0.03	CT
A	A_additional_A_A_GSP_OpposingRoutes_Fast_group6	1.101	48.08	C	0.267	CT	0.031	CT
A	A_additional_A_A_GSP_OpposingRoutes_Fast_group7	1.101	46.77	C	0.267	CT	0.031	CT
A	A_additional_A_A_GSP_OpposingRoutes_Fast_group8	1.103	49.41	C	0.266	CT	0.031	CT
A	A_additional_A_A_GSP_OpposingRoutes_Fast_group9	1.098	49.37	C	0.268	CT	0.031	CT
A	A_additional_A_A_GSP_OverlapSectionsInLineOfRoute_group0	1.019	46.97	A	0.555	A	0.031	A
A	A_additional_A_A_GSP_OverlapSectionsInLineOfRoute_group1	1.049	44.73	A	0.77	A	0.03	A
A	A_additional_A_A_GSP_PointControlWhenFreeToMove_group0	1.111	45.59	A	0.386	A	0.031	A
A	A_additional_A_A_GSP_PointControlWhenFreeToMove_group1	1.093	47.51	A	0.389	A	0.031	A
A	A_additional_A_A_GSP_PointControlWhenFreeToMove_group2	1.082	43.39	A	0.388	A	0.031	A
A	A_additional_A_A_GSP_PointControlWhenFreeToMove_group3	1.081	56.19	A	0.387	A	0.031	A
A	A_additional_A_A_GSP_PointControlWhenFreeToMove_group4	1.081	50.55	A	0.394	A	0.031	A
A	A_additional_A_A_GSP_PointControlWhenFreeToMove_group5	1.079	47.11	A	0.391	A	0.031	A
A	A_additional_A_A_GSP_PointControlWhenFreeToMove_group6	1.078	45.47	A	0.392	A	0.03	A
A	A_additional_A_A_GSP_PointControlWhenFreeToMove_group7	1.076	46.75	A	0.39	A	0.031	A
A	A_additional_A_A_GSP_PointControlWhenKeyed_group0	1.081	29.51	A	0.356	E	0.031	E
A	A_additional_A_A_GSP_PointControlWhenKeyed_group1	1.08	29.7	A	0.358	E	0.031	E
A	A_additional_A_A_GSP_PointControlWhenKeyed_group2	1.075	29.55	A	0.355	E	0.031	E
A	A_additional_A_A_GSP_PointControlWhenKeyed_group3	1.082	29.57	A	0.359	E	0.03	E
A	A_additional_A_A_GSP_PointControlWhenKeyed_group4	1.081	42.48	A	0.357	E	0.03	E
A	A_additional_A_A_GSP_PointControlWhenKeyed_group5	1.084	29.69	A	0.359	E	0.031	E
A	A_additional_A_A_GSP_PointControlWhenKeyed_group6	1.08	29.59	A	0.356	E	0.031	E
A	A_additional_A_A_GSP_PointControlWhenKeyed_group7	1.083	29.51	A	0.36	E	0.031	E
A	A_additional_A_A_GSP_PointControlWithSectionsOverPoints_group0	1.109	46.62	C	0.385	CT	0.031	CT
A	A_additional_A_A_GSP_PointControlWithSectionsOverPoints_group1	1.023	43.51	A	0.377	A	0.031	A
A	A_additional_A_A_GSP_PointControlWithSectionsOverPoints_group10	1.025	45.38	A	0.376	A	0.031	A
A	A_additional_A_A_GSP_PointControlWithSectionsOverPoints_group11	1.03	45.49	A	0.377	A	0.031	A
A	A_additional_A_A_GSP_PointControlWithSectionsOverPoints_group12	1.057	43	A	0.387	A	0.031	A
A	A_additional_A_A_GSP_PointControlWithSectionsOverPoints_group13	1.131	50.47	C	0.372	CT	0.031	CT
A	A_additional_A_A_GSP_PointControlWithSectionsOverPoints_group14	1.113	44.65	C	0.375	CT	0.03	CT
A	A_additional_A_A_GSP_PointControlWithSectionsOverPoints_group15	1.104	45.32	C	0.374	CT	0.031	CT
A	A_additional_A_A_GSP_PointControlWithSectionsOverPoints_group16	1.023	29.7	A	0.374	A	0.031	A
A	A_additional_A_A_GSP_PointControlWithSectionsOverPoints_group17	1.096	29.62	A	0.374	A	0.031	A
A	A_additional_A_A_GSP_PointControlWithSectionsOverPoints_group18	1.103	42.82	C	0.378	CT	0.031	CT
A	A_additional_A_A_GSP_PointControlWithSectionsOverPoints_group19	1.026	43.93	A	0.378	A	0.031	A
A	A_additional_A_A_GSP_PointControlWithSectionsOverPoints_group2	1.101	48.44	C	0.378	CT	0.031	CT
A	A_additional_A_A_GSP_PointControlWithSectionsOverPoints_group20	1.021	44.42	A	0.377	A	0.031	A
A	A_additional_A_A_GSP_PointControlWithSectionsOverPoints_group21	1.11	78.22	C	0.375	CT	0.031	CT
A	A_additional_A_A_GSP_PointControlWithSectionsOverPoints_group22	1.099	56.21	C	0.388	CT	0.031	CT
A	A_additional_A_A_GSP_PointControlWithSectionsOverPoints_group23	1.101	43.3	C	0.387	CT	0.031	CT
A	A_additional_A_A_GSP_PointControlWithSectionsOverPoints_group24	1.105	42.7	C	0.38	CT	0.031	CT
A	A_additional_A_A_GSP_PointControlWithSectionsOverPoints_group25	1.022	45.05	A	0.387	A	0.031	A
A	A_additional_A_A_GSP_PointControlWithSectionsOverPoints_group26	1.098	49.51	C	0.378	CT	0.03	CT
A	A_additional_A_A_GSP_PointControlWithSectionsOverPoints_group27	1.103	44.98	C	0.376	CT	0.031	CT
A	A_additional_A_A_GSP_PointControlWithSectionsOverPoints_group28	1.097	41.47	A	0.376	A	0.031	A

A	A_additional_A_A_GSP_PointControlWithSectionsOverPoints_group29	1.023	29.57	A	0.377	A	0.031	A
A	A_additional_A_A_GSP_PointControlWithSectionsOverPoints_group3	1.023	44.31	A	0.377	A	0.03	A
A	A_additional_A_A_GSP_PointControlWithSectionsOverPoints_group4	1.073	46.15	A	0.387	A	0.031	A
A	A_additional_A_A_GSP_PointControlWithSectionsOverPoints_group5	1.052	44.58	A	0.378	A	0.03	A
A	A_additional_A_A_GSP_PointControlWithSectionsOverPoints_group6	1.038	44.93	A	0.386	A	0.031	A
A	A_additional_A_A_GSP_PointControlWithSectionsOverPoints_group7	1.027	46.17	A	0.376	A	0.031	A
A	A_additional_A_A_GSP_PointControlWithSectionsOverPoints_group8	1.021	44.11	A	0.376	A	0.031	A
A	A_additional_A_A_GSP_PointControlWithSectionsOverPoints_group9	1.026	44.66	A	0.375	A	0.031	A
A	A_additional_A_A_GSP_PointControlWithTechniciansControls_group0	1.082	50.93	A	0.376	A	0.03	A
A	A_additional_A_A_GSP_PointControlWithTechniciansControls_group1	1.085	47.25	A	0.377	A	0.031	A
A	A_additional_A_A_GSP_PointControlWithTechniciansControls_group2	1.081	47.99	A	0.376	A	0.031	A
A	A_additional_A_A_GSP_PointControlWithTechniciansControls_group3	1.082	48.69	A	0.377	A	0.031	A
A	A_additional_A_A_GSP_PointControlWithTechniciansControls_group4	1.08	47.59	A	0.377	A	0.03	A
A	A_additional_A_A_GSP_PointControlWithTechniciansControls_group5	1.084	56.04	A	0.378	A	0.031	A
A	A_additional_A_A_GSP_PointControlWithTechniciansControls_group6	1.08	46.04	A	0.377	A	0.031	A
A	A_additional_A_A_GSP_PointControlWithTechniciansControls_group7	1.081	50.31	A	0.377	A	0.031	A
A	A_additional_A_A_GSP_PointsCalledNormal_group0	1.1	57.63	B	0.521	D	0.031	D
A	A_additional_A_A_GSP_PointsCalledNormal_group1	1.101	55.2	B	0.533	D	0.031	D
A	A_additional_A_A_GSP_PointsCalledNormal_group2	1.113	58.44	B	0.477	D	0.031	D
A	A_additional_A_A_GSP_PointsCalledNormal_group3	1.118	57.71	B	0.511	D	0.031	D
A	A_additional_A_A_GSP_PointsCalledNormal_group4	1.108	60.58	B	0.533	D	0.031	D
A	A_additional_A_A_GSP_PointsCalledNormal_group5	1.103	57.64	B	0.544	D	0.03	D
A	A_additional_A_A_GSP_PointsCalledNormal_group6	1.129	65	B	0.554	D	0.031	D
A	A_additional_A_A_GSP_PointsCalledNormal_group7	1.156	57.64	B	0.62	D	0.031	D
A	A_additional_A_A_GSP_PointsCalledNormal_group8	1.13	57.57	B	0.653	D	0.031	D
A	A_additional_A_A_GSP_PointsCalledReverse_group0	1.107	56.27	B	0.52	D	0.031	D
A	A_additional_A_A_GSP_PointsCalledReverse_group1	1.103	51.42	C	0.456	CT	0.031	CT
A	A_additional_A_A_GSP_PointsCalledReverse_group2	1.112	76.05	C	0.577	CT	0.031	CT
A	A_additional_A_A_GSP_PointsCalledReverse_group3	1.115	92.92	C	0.568	CT	0.031	CT
A	A_additional_A_A_GSP_PointsCalledReverse_group4	1.115	61.03	B	0.554	D	0.03	D
A	A_additional_A_A_GSP_PointsCalledReverse_group5	1.102	60.23	B	0.602	D	0.031	D
A	A_additional_A_A_GSP_PointsCalledReverse_group6	1.101	57.09	B	0.606	D	0.031	D
A	A_additional_A_A_GSP_PointsCalledReverse_group7	1.099	61.12	B	0.638	D	0.031	D
A	A_additional_A_A_GSP_PointsCalledReverse_group8	1.119	56.9	B	0.647	D	0.031	D
A	A_additional_A_A_GSP_ProceedWithExitSignal_group0	1.022	43.23	A	0.774	A	0.031	A
A	A_additional_A_A_GSP_ProceedWithExitSignal_group1	1.031	45.54	A	1.204	A	0.031	A
A	A_additional_A_A_GSP_ProceedWithExitSignal_group2	1.027	43.55	A	0.806	A	0.031	A
A	A_additional_A_A_GSP_ProceedWithExitSignal_group3	1.105	57.97	B	1.269	D	0.031	D
A	A_additional_A_A_GSP_ProceedWithExitSignal_group4	1.102	55.42	B	1.692	D	0.031	D
A	A_additional_A_A_GSP_ProceedWithExitSignal_group5	1.107	59.92	B	1.507	D	0.031	D
A	A_additional_A_A_GSP_ProceedWithExitSignal_group6	1.103	59.34	B	1.027	D	0.031	D
A	A_additional_A_A_GSP_RouteSectionLockedAheadOfTrain_group0	1.018	43.31	A	0.266	A	0.03	A
A	A_additional_A_A_GSP_RouteSectionLockedAheadOfTrain_group1	1.038	45.57	A	0.268	A	0.031	A
A	A_additional_A_A_GSP_RouteSectionLockedAheadOfTrain_group10	1.056	53.37	A	0.269	A	0.03	A
A	A_additional_A_A_GSP_RouteSectionLockedAheadOfTrain_group11	1.047	43.91	A	0.27	A	0.03	A
A	A_additional_A_A_GSP_RouteSectionLockedAheadOfTrain_group12	1.032	44.31	A	0.266	A	0.031	A
A	A_additional_A_A_GSP_RouteSectionLockedAheadOfTrain_group13	1.026	44.22	A	0.27	A	0.031	A
A	A_additional_A_A_GSP_RouteSectionLockedAheadOfTrain_group14	1.025	45.1	A	0.273	A	0.031	A
A	A_additional_A_A_GSP_RouteSectionLockedAheadOfTrain_group15	1.021	46.07	A	0.27	A	0.03	A
A	A_additional_A_A_GSP_RouteSectionLockedAheadOfTrain_group16	1.024	44.39	A	0.271	A	0.031	A
A	A_additional_A_A_GSP_RouteSectionLockedAheadOfTrain_group17	1.024	44.53	A	0.271	A	0.031	A
A	A_additional_A_A_GSP_RouteSectionLockedAheadOfTrain_group18	0.969	27.27	A	0.332	E	0.03	E
A	A_additional_A_A_GSP_RouteSectionLockedAheadOfTrain_group19	1.021	45.64	A	0.269	A	0.031	A
A	A_additional_A_A_GSP_RouteSectionLockedAheadOfTrain_group2	1.021	43.71	A	0.27	A	0.031	A
A	A_additional_A_A_GSP_RouteSectionLockedAheadOfTrain_group20	0.969	27.11	A	0.332	E	0.031	E
A	A_additional_A_A_GSP_RouteSectionLockedAheadOfTrain_group21	1.025	46.12	A	0.268	A	0.03	A
A	A_additional_A_A_GSP_RouteSectionLockedAheadOfTrain_group22	1.024	45.97	A	0.265	A	0.03	A
A	A_additional_A_A_GSP_RouteSectionLockedAheadOfTrain_group23	1.02	44.99	A	0.265	A	0.031	A
A	A_additional_A_A_GSP_RouteSectionLockedAheadOfTrain_group24	1.019	45.14	A	0.267	A	0.031	A
A	A_additional_A_A_GSP_RouteSectionLockedAheadOfTrain_group3	1.027	43.92	A	0.271	A	0.031	A
A	A_additional_A_A_GSP_RouteSectionLockedAheadOfTrain_group4	1.024	44.12	A	0.266	A	0.031	A
A	A_additional_A_A_GSP_RouteSectionLockedAheadOfTrain_group5	1.027	44.4	A	0.27	A	0.031	A
A	A_additional_A_A_GSP_RouteSectionLockedAheadOfTrain_group6	1.023	29.91	A	0.269	A	0.031	A
A	A_additional_A_A_GSP_RouteSectionLockedAheadOfTrain_group7	1.023	46.47	A	0.271	A	0.031	A
A	A_additional_A_A_GSP_RouteSectionLockedAheadOfTrain_group8	1.059	44.72	A	0.27	A	0.031	A
A	A_additional_A_A_GSP_RouteSectionLockedAheadOfTrain_group9	1.051	44.09	A	0.272	A	0.03	A

A	A_additional_A_A_GSP_RouteSectionLockedOverPoints_group0	1.108	47.81	C	0.267	CT	0.031	CT
A	A_additional_A_A_GSP_RouteSectionLockedOverPoints_group1	1.021	44.17	A	0.266	A	0.031	A
A	A_additional_A_A_GSP_RouteSectionLockedOverPoints_group10	1.018	45.99	A	0.266	A	0.03	A
A	A_additional_A_A_GSP_RouteSectionLockedOverPoints_group11	1.02	44.2	A	0.265	A	0.031	A
A	A_additional_A_A_GSP_RouteSectionLockedOverPoints_group12	1.018	44.9	A	0.266	A	0.03	A
A	A_additional_A_A_GSP_RouteSectionLockedOverPoints_group13	1.099	46.98	C	0.266	CT	0.031	CT
A	A_additional_A_A_GSP_RouteSectionLockedOverPoints_group14	1.105	48.95	C	0.266	CT	0.031	CT
A	A_additional_A_A_GSP_RouteSectionLockedOverPoints_group15	1.098	44.98	C	0.266	CT	0.03	CT
A	A_additional_A_A_GSP_RouteSectionLockedOverPoints_group16	1.019	49.9	A	0.268	A	0.03	A
A	A_additional_A_A_GSP_RouteSectionLockedOverPoints_group17	1.093	29.63	A	0.267	A	0.031	A
A	A_additional_A_A_GSP_RouteSectionLockedOverPoints_group18	1.093	46.58	C	0.267	CT	0.031	CT
A	A_additional_A_A_GSP_RouteSectionLockedOverPoints_group19	1.031	44.03	A	0.266	A	0.031	A
A	A_additional_A_A_GSP_RouteSectionLockedOverPoints_group2	1.132	50.14	C	0.266	CT	0.031	CT
A	A_additional_A_A_GSP_RouteSectionLockedOverPoints_group20	1.042	45.67	A	0.268	A	0.031	A
A	A_additional_A_A_GSP_RouteSectionLockedOverPoints_group21	1.111	107	C	0.267	CT	0.031	CT
A	A_additional_A_A_GSP_RouteSectionLockedOverPoints_group22	1.106	45.13	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_RouteSectionLockedOverPoints_group23	1.102	44.61	C	0.266	CT	0.031	CT
A	A_additional_A_A_GSP_RouteSectionLockedOverPoints_group24	1.1	45.16	C	0.266	CT	0.03	CT
A	A_additional_A_A_GSP_RouteSectionLockedOverPoints_group25	1.022	44.59	A	0.265	A	0.031	A
A	A_additional_A_A_GSP_RouteSectionLockedOverPoints_group26	1.103	44.95	C	0.267	CT	0.03	CT
A	A_additional_A_A_GSP_RouteSectionLockedOverPoints_group27	1.094	46.8	C	0.266	CT	0.031	CT
A	A_additional_A_A_GSP_RouteSectionLockedOverPoints_group28	1.096	29.9	A	0.266	A	0.03	A
A	A_additional_A_A_GSP_RouteSectionLockedOverPoints_group29	1.022	43.62	A	0.266	A	0.031	A
A	A_additional_A_A_GSP_RouteSectionLockedOverPoints_group3	1.014	45.04	A	0.266	A	0.031	A
A	A_additional_A_A_GSP_RouteSectionLockedOverPoints_group4	1.015	45.42	A	0.265	A	0.031	A
A	A_additional_A_A_GSP_RouteSectionLockedOverPoints_group5	1.067	45.3	A	0.266	A	0.03	A
A	A_additional_A_A_GSP_RouteSectionLockedOverPoints_group6	1.04	46.93	A	0.266	A	0.031	A
A	A_additional_A_A_GSP_RouteSectionLockedOverPoints_group7	1.031	48.31	A	0.265	A	0.031	A
A	A_additional_A_A_GSP_RouteSectionLockedOverPoints_group8	1.019	46.71	A	0.265	A	0.031	A
A	A_additional_A_A_GSP_RouteSectionLockedOverPoints_group9	1.016	44.92	A	0.265	A	0.031	A
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group0	1.106	67.96	B	0.264	D	0.03	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group1	1.104	86.95	B	0.264	D	0.031	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group10	1.104	56.51	B	0.264	D	0.031	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group100	1.098	323	C	0.262	CT	0.03	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group101	1.097	359.2	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group102	1.097	44.92	C	0.263	CT	0.03	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group103	1.093	299.7	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group104	1.089	204.4	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group105	1.115	272	C	0.262	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group106	1.13	260.9	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group107	1.112	208.8	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group108	1.103	201.8	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group109	1.098	262.3	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group11	1.101	56.06	B	0.266	D	0.03	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group110	1.095	272.8	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group111	1.096	355.5	C	0.262	CT	0.03	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group112	1.093	317.5	C	0.263	CT	0.03	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group113	1.096	44.54	C	0.263	CT	0.03	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group114	1.094	215.4	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group115	1.092	295.6	C	0.262	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group116	1.092	209.8	C	0.263	CT	0.03	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group117	1.113	287.8	C	0.263	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group118	1.132	258.4	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group119	1.111	218.8	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group12	1.107	56.42	B	0.265	D	0.031	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group120	1.093	211.2	C	0.263	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group121	1.091	261	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group122	1.093	209.2	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group123	1.094	279.8	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group124	1.089	261.5	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group125	1.093	322.2	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group126	1.093	44.65	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group127	1.091	348.7	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group128	1.094	336.3	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group129	1.116	200.5	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group13	1.14	55.88	B	0.263	D	0.031	D

A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group130	1.113	254.5	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group131	1.104	263.3	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group132	1.094	192.1	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group133	1.099	256.3	C	0.263	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group134	1.099	290.8	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group135	1.099	334.5	C	0.263	CT	0.03	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group136	1.1	213.5	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group137	1.095	43.5	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group138	1.094	219.3	C	0.263	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group139	1.093	290.7	C	0.263	CT	0.03	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group14	1.099	57.45	B	0.263	D	0.031	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group140	1.115	192.9	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group141	1.135	216.5	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group142	1.115	285	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group143	1.106	327.4	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group144	1.096	260.7	C	0.262	CT	0.03	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group145	1.097	205.5	C	0.263	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group146	1.098	261.8	C	0.263	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group147	1.095	265.4	C	0.262	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group148	1.097	207.9	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group149	1.092	271.6	C	0.263	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group15	1.098	57.55	B	0.262	D	0.031	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group150	1.092	42.64	C	0.262	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group151	1.088	162.6	C	0.263	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group152	1.113	156.2	C	0.263	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group153	1.129	210.4	C	0.263	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group154	1.115	272.7	C	0.263	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group155	1.105	289.9	C	0.263	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group156	1.093	364.7	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group157	1.091	273.9	C	0.262	CT	0.03	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group158	1.096	269.1	C	0.263	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group159	1.093	258.6	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group16	1.098	57.14	B	0.263	D	0.031	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group160	1.089	302.3	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group161	1.094	53.92	C	0.263	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group162	1.099	56.45	B	0.263	D	0.03	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group163	1.091	66.33	B	0.264	D	0.031	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group164	1.11	54.9	B	0.262	D	0.031	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group165	1.128	56.43	B	0.265	D	0.03	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group166	1.118	43.47	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group167	1.106	56.16	B	0.263	D	0.03	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group168	1.092	56.64	B	0.263	D	0.031	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group169	1.095	43.25	C	0.263	CT	0.03	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group17	1.1	58.18	B	0.264	D	0.03	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group170	1.103	55.26	B	0.264	D	0.03	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group171	1.098	57.8	B	0.263	D	0.031	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group172	1.095	65.49	B	0.261	D	0.031	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group173	1.097	56.27	B	0.263	D	0.03	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group174	1.102	59.31	B	0.263	D	0.031	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group175	1.105	55.33	B	0.263	D	0.031	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group176	1.103	57.01	B	0.262	D	0.031	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group177	1.13	54.68	B	0.264	D	0.031	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group178	1.118	58	B	0.264	D	0.031	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group179	1.106	64.87	B	0.263	D	0.031	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group18	1.096	62.14	B	0.262	D	0.031	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group180	1.102	54.73	B	0.266	D	0.03	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group181	1.098	88.1	B	0.264	D	0.03	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group182	1.095	66.75	B	0.263	D	0.031	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group183	1.095	57.03	B	0.263	D	0.031	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group19	1.097	56.44	B	0.263	D	0.031	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group2	1.104	61.05	B	0.264	D	0.031	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group20	1.097	58.25	B	0.265	D	0.031	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group21	1.092	56.94	B	0.264	D	0.031	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group22	1.108	55.19	B	0.264	D	0.031	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group23	1.13	56.02	B	0.263	D	0.031	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group24	1.124	74.69	B	0.263	D	0.03	D

A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group25	1.11	58.81	B	0.263	D	0.03	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group26	1.098	68.78	B	0.261	D	0.031	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group27	1.099	55.68	B	0.264	D	0.03	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group28	1.099	64.69	B	0.265	D	0.031	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group29	1.102	56.62	B	0.264	D	0.031	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group3	1.099	57.76	B	0.263	D	0.031	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group30	1.093	45.28	C	0.263	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group31	1.091	253.7	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group32	1.093	212.2	C	0.263	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group33	1.092	252.5	C	0.263	CT	0.03	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group34	1.102	194.1	C	0.263	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group35	1.132	154.6	C	0.263	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group36	1.12	249.2	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group37	1.108	244.2	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group38	1.094	215	C	0.263	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group39	1.096	208.3	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group4	1.1	59.67	B	0.263	D	0.031	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group40	1.092	157.4	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group41	1.096	43.35	C	0.262	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group42	1.091	272.1	C	0.262	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group43	1.094	205.9	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group44	1.096	264.8	C	0.263	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group45	1.09	195.6	C	0.263	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group46	1.097	268.7	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group47	1.125	200.7	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group48	1.12	198	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group49	1.105	210.8	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group5	1.101	61.28	B	0.264	D	0.031	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group50	1.095	212.3	C	0.263	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group51	1.094	202.9	C	0.263	CT	0.03	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group52	1.097	163	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group53	1.094	162.9	C	0.263	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group54	1.098	49.51	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group55	1.095	217.2	C	0.267	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group56	1.099	293.5	C	0.263	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group57	1.091	198.4	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group58	1.102	328.9	C	0.263	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group59	1.132	276.6	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group6	1.131	57.32	B	0.265	D	0.031	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group60	1.108	275.1	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group61	1.094	214.4	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group62	1.095	331.2	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group63	1.097	208.1	C	0.263	CT	0.03	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group64	1.096	272.6	C	0.263	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group65	1.098	44.52	C	0.263	CT	0.03	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group66	1.091	199	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group67	1.096	212.7	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group68	1.095	318.7	C	0.262	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group69	1.095	267.9	C	0.262	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group7	1.105	56.12	B	0.264	D	0.031	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group70	1.124	201.6	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group71	1.121	282.2	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group72	1.107	195.2	C	0.263	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group73	1.095	341.5	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group74	1.098	221.9	C	0.262	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group75	1.094	255.8	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group76	1.1	238	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group77	1.091	269	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group78	1.095	44.21	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group79	1.095	344.1	C	0.267	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group8	1.096	57.1	B	0.263	D	0.031	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group80	1.09	282.4	C	0.267	CT	0.03	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group81	1.098	270.2	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group82	1.125	292.2	C	0.266	CT	0.03	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group83	1.122	219	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group84	1.107	297.9	C	0.263	CT	0.031	CT

A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group85	1.096	220.5	C	0.266	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group86	1.099	248.2	C	0.263	CT	0.03	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group87	1.097	256.4	C	0.264	CT	0.03	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group88	1.096	255.9	C	0.263	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group89	1.095	43.73	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group9	1.097	58.63	B	0.263	D	0.031	D
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group90	1.1	296.4	C	0.265	CT	0.03	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group91	1.096	283.7	C	0.266	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group92	1.093	267.2	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group93	1.103	326.5	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group94	1.127	223.2	C	0.267	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group95	1.121	221.4	C	0.264	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group96	1.108	216.2	C	0.266	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group97	1.095	210.5	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group98	1.098	204.5	C	0.262	CT	0.03	CT
A	A_additional_A_A_GSP_RoutesFromEntranceSignal_group99	1.096	202.7	C	0.265	CT	0.031	CT
A	A_additional_A_A_GSP_SignalGroupReplacement_group0	1.089	43.15	A	0.264	A	0.031	A
A	A_additional_A_A_GSP_SignalGroupReplacement_group1	1.094	44.44	A	0.263	A	0.031	A
A	A_additional_A_A_GSP_SignalGroupReplacement_group10	1.088	29.44	A	0.265	A	0.031	A
A	A_additional_A_A_GSP_SignalGroupReplacement_group11	1.091	42.59	A	0.265	A	0.031	A
A	A_additional_A_A_GSP_SignalGroupReplacement_group12	1.09	42.8	A	0.265	A	0.03	A
A	A_additional_A_A_GSP_SignalGroupReplacement_group13	1.085	43.29	A	0.265	A	0.03	A
A	A_additional_A_A_GSP_SignalGroupReplacement_group14	1.086	42.79	A	0.266	A	0.031	A
A	A_additional_A_A_GSP_SignalGroupReplacement_group15	1.127	42.97	A	0.266	A	0.03	A
A	A_additional_A_A_GSP_SignalGroupReplacement_group16	1.122	42.78	A	0.265	A	0.031	A
A	A_additional_A_A_GSP_SignalGroupReplacement_group17	1.1	43.32	A	0.264	A	0.031	A
A	A_additional_A_A_GSP_SignalGroupReplacement_group18	1.091	43.27	A	0.264	A	0.031	A
A	A_additional_A_A_GSP_SignalGroupReplacement_group19	1.088	42.94	A	0.267	A	0.031	A
A	A_additional_A_A_GSP_SignalGroupReplacement_group2	1.094	29.77	A	0.264	A	0.03	A
A	A_additional_A_A_GSP_SignalGroupReplacement_group20	1.091	43.15	A	0.264	A	0.03	A
A	A_additional_A_A_GSP_SignalGroupReplacement_group21	1.093	29.59	A	0.265	A	0.03	A
A	A_additional_A_A_GSP_SignalGroupReplacement_group22	1.091	43.34	A	0.265	A	0.03	A
A	A_additional_A_A_GSP_SignalGroupReplacement_group23	1.086	43.62	A	0.263	A	0.03	A
A	A_additional_A_A_GSP_SignalGroupReplacement_group24	1.089	43.11	A	0.265	A	0.03	A
A	A_additional_A_A_GSP_SignalGroupReplacement_group25	1.081	46.89	A	0.266	A	0.03	A
A	A_additional_A_A_GSP_SignalGroupReplacement_group26	1.088	42.82	A	0.265	A	0.03	A
A	A_additional_A_A_GSP_SignalGroupReplacement_group27	1.126	42.89	A	0.265	A	0.03	A
A	A_additional_A_A_GSP_SignalGroupReplacement_group28	1.12	42.83	A	0.265	A	0.03	A
A	A_additional_A_A_GSP_SignalGroupReplacement_group29	1.105	43.32	A	0.263	A	0.03	A
A	A_additional_A_A_GSP_SignalGroupReplacement_group3	1.093	45.7	A	0.265	A	0.03	A
A	A_additional_A_A_GSP_SignalGroupReplacement_group30	1.09	43.61	A	0.262	A	0.03	A
A	A_additional_A_A_GSP_SignalGroupReplacement_group31	1.091	43.29	A	0.266	A	0.031	A
A	A_additional_A_A_GSP_SignalGroupReplacement_group32	1.088	43.17	A	0.265	A	0.03	A
A	A_additional_A_A_GSP_SignalGroupReplacement_group33	1.092	43.91	A	0.265	A	0.031	A
A	A_additional_A_A_GSP_SignalGroupReplacement_group34	1.087	42.96	A	0.265	A	0.03	A
A	A_additional_A_A_GSP_SignalGroupReplacement_group35	1.087	42.73	A	0.264	A	0.03	A
A	A_additional_A_A_GSP_SignalGroupReplacement_group4	1.087	42.59	A	0.265	A	0.03	A
A	A_additional_A_A_GSP_SignalGroupReplacement_group5	1.085	42.74	A	0.266	A	0.031	A
A	A_additional_A_A_GSP_SignalGroupReplacement_group6	1.085	42.83	A	0.263	A	0.03	A
A	A_additional_A_A_GSP_SignalGroupReplacement_group7	1.126	42.78	A	0.261	A	0.03	A
A	A_additional_A_A_GSP_SignalGroupReplacement_group8	1.115	42.83	A	0.265	A	0.03	A
A	A_additional_A_A_GSP_SignalGroupReplacement_group9	1.102	42.77	A	0.265	A	0.03	A
A	A_additional_A_A_GSP_SignalReminder_group0	1.123	80.01	A	0.46	A	0.031	A
A	A_additional_A_A_GSP_SignalReminder_group1	1.121	56.8	A	0.424	A	0.03	A
A	A_additional_A_A_GSP_SignalReminder_group10	1.12	80.39	A	0.423	A	0.03	A
A	A_additional_A_A_GSP_SignalReminder_group11	1.096	112.9	A	0.377	A	0.031	A
A	A_additional_A_A_GSP_SignalReminder_group12	1.096	120	A	0.375	A	0.03	A
A	A_additional_A_A_GSP_SignalReminder_group13	1.09	86.03	A	0.56	A	0.031	A
A	A_additional_A_A_GSP_SignalReminder_group14	1.092	86.85	A	0.379	A	0.031	A
A	A_additional_A_A_GSP_SignalReminder_group15	1.097	82.11	A	0.376	A	0.03	A
A	A_additional_A_A_GSP_SignalReminder_group16	1.091	112.9	A	0.376	A	0.03	A
A	A_additional_A_A_GSP_SignalReminder_group17	1.091	118.5	A	0.377	A	0.03	A
A	A_additional_A_A_GSP_SignalReminder_group18	1.143	55.2	B	0.562	D	0.03	D
A	A_additional_A_A_GSP_SignalReminder_group19	1.123	131.6	A	0.561	A	0.03	A
A	A_additional_A_A_GSP_SignalReminder_group2	1.135	111.8	A	0.458	A	0.031	A
A	A_additional_A_A_GSP_SignalReminder_group20	1.098	111.4	A	0.375	A	0.03	A

A	A_additional_A_A_GSP_SignalReminder_group21	1.092	113.2	A	0.376	A	0.03	A
A	A_additional_A_A_GSP_SignalReminder_group22	1.095	129.7	A	0.558	A	0.03	A
A	A_additional_A_A_GSP_SignalReminder_group23	1.094	83.38	A	0.562	A	0.03	A
A	A_additional_A_A_GSP_SignalReminder_group24	1.097	81.11	A	0.377	A	0.03	A
A	A_additional_A_A_GSP_SignalReminder_group25	1.092	61.68	A	0.458	A	0.03	A
A	A_additional_A_A_GSP_SignalReminder_group26	1.092	83.95	A	0.377	A	0.03	A
A	A_additional_A_A_GSP_SignalReminder_group27	1.096	86	A	0.375	A	0.03	A
A	A_additional_A_A_GSP_SignalReminder_group28	1.089	61.4	A	0.552	A	0.03	A
A	A_additional_A_A_GSP_SignalReminder_group29	1.095	59.49	A	0.549	A	0.03	A
A	A_additional_A_A_GSP_SignalReminder_group3	1.161	80.41	A	0.423	A	0.031	A
A	A_additional_A_A_GSP_SignalReminder_group30	1.127	78.24	A	0.524	A	0.03	A
A	A_additional_A_A_GSP_SignalReminder_group31	1.107	83.53	A	0.369	A	0.031	A
A	A_additional_A_A_GSP_SignalReminder_group32	1.096	57.11	A	0.439	A	0.031	A
A	A_additional_A_A_GSP_SignalReminder_group33	1.091	114.9	A	0.46	A	0.03	A
A	A_additional_A_A_GSP_SignalReminder_group34	1.096	83.3	A	0.46	A	0.031	A
A	A_additional_A_A_GSP_SignalReminder_group35	1.098	82.55	A	0.483	A	0.031	A
A	A_additional_A_A_GSP_SignalReminder_group36	1.095	114.2	A	0.487	A	0.031	A
A	A_additional_A_A_GSP_SignalReminder_group4	1.09	83.9	A	0.515	A	0.03	A
A	A_additional_A_A_GSP_SignalReminder_group5	1.095	82.09	A	0.374	A	0.03	A
A	A_additional_A_A_GSP_SignalReminder_group6	1.096	84.05	A	0.378	A	0.03	A
A	A_additional_A_A_GSP_SignalReminder_group7	1.091	84.57	A	0.376	A	0.03	A
A	A_additional_A_A_GSP_SignalReminder_group8	1.088	116.7	A	0.511	A	0.03	A
A	A_additional_A_A_GSP_SignalReminder_group9	1.133	82.08	A	0.376	A	0.03	A
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_group0	1.132	58.2	B	0.379	B	0.03	B
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_group1	1.037	56.09	A	0.379	E	0.03	E
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_group10	1.109	54.97	B	0.381	B	0.03	B
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_group11	1.091	58.83	B	0.377	B	0.03	B
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_group12	1.103	56.92	B	0.383	B	0.03	B
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_group13	1.106	57.1	B	0.378	B	0.031	B
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_group14	1.096	55.85	B	0.383	B	0.03	B
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_group15	1.094	57.35	B	0.379	B	0.03	B
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_group16	1.113	59.83	B	0.379	B	0.03	B
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_group17	1.098	55.82	B	0.382	B	0.03	B
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_group18	1.105	59	B	0.382	B	0.03	B
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_group19	1.098	57.7	B	0.382	B	0.031	B
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_group2	1.141	58.36	B	0.38	B	0.03	B
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_group20	1.126	56.49	B	0.385	B	0.03	B
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_group21	1.12	61.24	B	0.38	B	0.03	B
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_group22	1.1	56.74	B	0.383	B	0.03	B
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_group23	1.103	58.15	B	0.383	B	0.03	B
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_group24	1.111	59.23	B	0.383	B	0.03	B
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_group3	1.098	63.67	B	0.381	B	0.031	B
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_group4	1.108	59.25	B	0.38	B	0.03	B
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_group5	1.106	84.94	B	0.381	B	0.03	B
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_group6	1.115	56.35	B	0.382	B	0.03	B
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_group7	1.098	63.51	B	0.382	B	0.03	B
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_group8	1.099	56.09	B	0.38	B	0.03	B
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_group9	1.09	85.07	B	0.383	B	0.03	B
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_Fast_group0	1.054	44.53	A	0.398	A	0.031	A
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_Fast_group1	1.053	42.44	A	0.419	A	0.03	A
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_Fast_group10	1.029	42.81	A	0.541	A	0.03	A
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_Fast_group11	1.034	43.37	A	0.581	A	0.03	A
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_Fast_group12	1.016	43.97	A	0.541	A	0.03	A
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_Fast_group13	1.026	43.14	A	0.909	A	0.031	A
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_Fast_group14	1.019	29.38	A	0.62	A	0.03	A
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_Fast_group15	1.018	42.74	A	0.401	A	0.031	A
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_Fast_group16	1.024	45.11	A	0.42	A	0.031	A
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_Fast_group17	1.026	43.65	A	0.581	A	0.03	A
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_Fast_group18	1.024	43.26	A	0.735	A	0.03	A
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_Fast_group19	1.031	42.62	A	0.782	A	0.03	A
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_Fast_group2	1.013	44.31	A	0.396	A	0.031	A
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_Fast_group20	1.024	43.09	A	0.846	A	0.03	A
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_Fast_group21	1.063	43.07	A	0.582	A	0.03	A
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_Fast_group22	1.057	42.28	A	0.582	A	0.03	A
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_Fast_group23	1.037	43.99	A	0.421	A	0.03	A
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_Fast_group24	0.977	26.75	A	0.328	E	0.03	E

A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_Fast_group25	1.028	43.42	A	0.474	A	0.03	A
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_Fast_group26	0.97	26.74	A	0.329	E	0.03	E
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_Fast_group27	1.029	43.41	A	0.506	A	0.03	A
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_Fast_group28	1.027	43.96	A	0.465	A	0.031	A
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_Fast_group29	1.02	43.64	A	0.508	A	0.031	A
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_Fast_group3	1.024	45.28	A	0.585	A	0.031	A
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_Fast_group30	1.025	46.3	A	0.524	A	0.03	A
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_Fast_group31	1.027	43.05	A	0.465	A	0.03	A
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_Fast_group32	1.014	45.37	A	0.504	A	0.03	A
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_Fast_group33	1.019	43.82	A	0.528	A	0.03	A
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_Fast_group34	1.065	44.51	A	0.462	A	0.03	A
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_Fast_group35	1.051	42.43	A	0.464	A	0.03	A
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_Fast_group36	0.981	26.7	A	0.329	E	0.03	E
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_Fast_group4	1.03	43.91	A	0.472	A	0.03	A
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_Fast_group5	1.026	42.53	A	0.737	A	0.03	A
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_Fast_group6	1.022	44.04	A	0.506	A	0.03	A
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_Fast_group7	1.026	43.45	A	0.591	A	0.03	A
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_Fast_group8	1.025	44.26	A	0.554	A	0.031	A
A	A_additional_A_A_GSP_SubsequentRouteSectionRelease_Fast_group9	1.027	43.57	A	0.845	A	0.03	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group0	1.021	43.96	A	0.677	A	0.031	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group1	1.025	43.49	A	0.75	A	0.031	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group10	1.028	44.58	A	0.493	A	0.031	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group100	1.018	42.99	A	1.169	A	0.03	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group101	1.114	58.15	B	0.549	D	0.031	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group102	1.137	67.67	B	0.535	D	0.03	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group103	1.052	44.36	A	1.116	A	0.03	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group104	1.042	43.74	A	0.943	A	0.03	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group105	1.109	57.85	B	0.562	D	0.031	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group106	1.104	55.62	B	0.688	D	0.03	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group107	1.105	55.32	B	0.668	D	0.03	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group108	1.098	44.89	A	0.625	A	0.031	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group109	1.109	63.77	B	0.634	D	0.03	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group11	1.023	44.7	A	0.508	A	0.031	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group110	1.103	58.04	B	0.601	D	0.031	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group111	1.097	46.5	A	0.913	A	0.031	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group112	1.104	55.34	B	0.627	D	0.03	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group113	1.11	58.65	B	0.597	D	0.031	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group114	1.142	56.45	B	0.471	D	0.031	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group115	1.134	58.7	B	0.533	D	0.03	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group116	1.117	56	B	0.48	D	0.031	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group117	1.115	55.43	B	0.551	D	0.031	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group118	1.106	57	B	1.167	D	0.031	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group119	1.114	62.1	B	0.549	D	0.031	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group12	1.109	59.7	B	0.549	D	0.03	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group120	1.104	55.52	B	0.517	D	0.03	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group121	1.106	57.21	B	0.582	D	0.03	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group122	1.1	57.7	B	1.349	D	0.03	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group123	1.106	55.44	B	0.547	D	0.03	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group124	1.101	64.96	B	0.488	D	0.031	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group125	1.099	53.71	B	0.561	D	0.031	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group126	1.151	55.78	B	1.145	D	0.03	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group127	1.144	55.5	B	0.529	D	0.03	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group128	1.125	56.18	B	0.557	D	0.03	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group129	1.107	56.8	B	0.57	D	0.03	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group13	1.109	56.61	B	0.675	D	0.031	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group130	1.108	56.51	B	0.808	D	0.031	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group131	1.111	56.73	B	0.799	D	0.03	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group132	1.112	55.66	B	0.499	D	0.03	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group14	1.025	43.65	A	1.018	A	0.031	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group15	1.023	43.99	A	1.34	A	0.03	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group16	1.105	55.3	B	0.629	D	0.03	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group17	1.096	57.99	B	0.585	D	0.031	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group18	1.02	43.93	A	0.507	A	0.031	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group19	1.066	43.95	A	0.507	A	0.031	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group2	1.053	29.56	A	0.7	A	0.031	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group20	1.114	56.43	B	0.568	D	0.031	D

A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group21	1.112	64.32	B	0.462	D	0.031	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group22	1.027	43.6	A	1.286	A	0.03	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group23	1.025	43.38	A	1.338	A	0.031	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group24	1.111	57.69	B	0.547	D	0.031	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group25	1.106	56.51	B	0.617	D	0.031	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group26	1.021	43.37	A	0.496	A	0.031	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group27	1.02	43.07	A	0.524	A	0.031	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group28	1.113	57.75	B	0.565	D	0.031	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group29	1.105	56.26	B	0.461	D	0.031	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group3	1.017	42.92	A	0.695	A	0.031	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group30	1.032	45.65	A	0.749	A	0.031	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group31	1.058	42.47	A	0.505	A	0.031	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group32	1.052	42.38	A	0.524	A	0.03	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group33	1.037	43.62	A	0.553	A	0.031	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group34	1.105	57.73	B	0.568	D	0.03	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group35	1.11	111.8	C	0.776	CT	0.031	CT
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group36	1.025	43.98	A	0.512	A	0.03	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group37	1.026	48.86	A	0.525	A	0.031	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group38	1.026	43.42	A	0.553	A	0.031	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group39	1.107	55.58	B	0.567	D	0.031	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group4	1.023	43.3	A	0.725	A	0.031	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group40	1.117	112.5	C	0.779	CT	0.031	CT
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group41	1.02	44.6	A	1.726	A	0.031	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group42	1.021	43.6	A	1.838	A	0.031	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group43	1.048	52.41	A	2.827	A	0.031	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group44	1.144	62.07	B	0.689	D	0.03	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group45	1.123	113	C	0.537	CT	0.031	CT
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group46	1.034	43.12	A	0.494	A	0.031	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group47	1.02	43.48	A	0.519	A	0.031	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group48	1.027	43.46	A	0.551	A	0.031	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group49	1.112	60.33	B	0.566	D	0.031	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group5	1.026	43	A	0.747	A	0.031	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group50	1.106	110.1	C	0.673	CT	0.031	CT
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group51	1.023	44.44	A	0.51	A	0.031	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group52	1.022	46.01	A	0.524	A	0.03	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group53	1.026	43.5	A	0.554	A	0.031	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group54	1.097	64.02	B	0.561	D	0.031	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group55	1.1	112.2	C	0.711	CT	0.03	CT
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group56	1.06	44.99	A	0.493	A	0.031	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group57	1.055	44.51	A	1.754	A	0.031	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group58	1.039	43.79	A	0.557	A	0.031	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group59	1.111	57.71	B	0.562	D	0.03	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group6	1.027	42.47	A	0.512	A	0.03	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group60	1.11	111.3	C	0.712	CT	0.03	CT
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group61	1.026	44.59	A	0.503	A	0.031	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group62	1.027	46.79	A	2.544	A	0.031	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group63	1.026	51.86	A	0.56	A	0.031	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group64	1.108	108.4	B	0.568	D	0.031	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group65	1.106	146.3	C	0.45	CT	0.03	CT
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group66	1.027	42.64	A	0.497	A	0.031	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group67	1.021	42.73	A	0.521	A	0.031	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group68	1.035	43.87	A	0.554	A	0.031	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group69	1.144	56.24	B	0.565	D	0.03	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group7	1.045	43.84	A	0.52	A	0.03	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group70	1.117	113.1	C	0.45	CT	0.031	CT
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group71	1.023	43.12	A	0.517	A	0.031	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group72	1.024	43.89	A	0.519	A	0.031	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group73	1.026	44	A	0.559	A	0.031	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group74	1.114	56.99	B	0.567	D	0.031	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group75	1.109	110.4	C	0.451	CT	0.03	CT
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group76	1.023	42.62	A	1.67	A	0.03	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group77	1.023	44.15	A	0.528	A	0.03	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group78	1.028	42.37	A	0.552	A	0.031	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group79	1.102	59.14	B	0.563	D	0.031	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group8	1.1	58.92	B	0.565	D	0.031	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group80	1.152	115.2	C	0.577	CT	0.031	CT

A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group81	1.05	44.57	A	0.509	A	0.031	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group82	1.036	43.75	A	0.526	A	0.031	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group83	1.031	42.63	A	0.553	A	0.03	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group84	1.107	56.86	B	0.566	D	0.031	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group85	1.111	143.5	C	0.565	CT	0.031	CT
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group86	1.027	42.87	A	0.508	A	0.03	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group87	1.024	43.24	A	0.526	A	0.031	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group88	1.025	45.75	A	0.558	A	0.031	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group89	1.106	57.86	B	0.563	D	0.031	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group9	1.106	58.5	B	0.62	D	0.031	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group90	1.108	114.8	C	0.452	CT	0.03	CT
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group91	1.024	43.1	A	0.911	A	0.031	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group92	1.035	46.49	A	0.554	A	0.031	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group93	1.152	57.42	B	0.565	D	0.031	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group94	1.127	64.04	B	0.537	D	0.031	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group95	1.037	43.49	A	1.09	A	0.03	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group96	1.022	43.3	A	1.176	A	0.031	A
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group97	1.113	57.31	B	0.547	D	0.031	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group98	1.107	55.82	B	0.536	D	0.031	D
A	A_additional_A_A_GSP_TrackSectionsInLineOfRoute_Fast_group99	1.026	45.41	A	0.496	A	0.031	A
B	B_additional_B_B_GSP_ConflictingRoutes_group0	1.11	56.21	B	0.324	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group10	1.113	56.34	B	0.324	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group100	1.113	56.31	B	0.322	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group101	1.108	56.37	B	0.323	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group102	1.114	56.43	B	0.324	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group103	1.112	56.38	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group104	1.11	56.46	B	0.323	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group105	1.129	56.23	B	0.323	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group106	1.153	56.28	B	0.323	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group107	1.128	56.26	B	0.324	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group108	1.123	56.18	B	0.323	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group109	1.118	56.24	B	0.323	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group11	1.112	56.3	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group110	1.115	56.28	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group111	1.114	56.34	B	0.322	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group112	1.113	56.44	B	0.324	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group113	1.113	56.27	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group114	1.115	56.27	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group115	1.11	56.38	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group116	1.11	56.47	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group117	1.144	56.32	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group118	1.133	56.39	B	0.323	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group119	1.129	56.37	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group12	1.12	56.32	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group120	1.115	56.26	B	0.324	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group121	1.114	56.42	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group122	1.114	56.31	B	0.324	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group123	1.116	56.38	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group124	1.111	56.34	B	0.324	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group125	1.114	56.21	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group126	1.115	56.42	B	0.324	B	0.031	B
B	B_additional_B_B_GSP_ConflictingRoutes_group127	1.109	56.39	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group128	1.111	56.33	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group129	1.157	56.42	B	0.324	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group13	1.144	56.3	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group130	1.123	56.42	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group131	1.117	56.39	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group14	1.118	56.42	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group15	1.114	56.42	B	0.324	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group16	1.115	56.36	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group17	1.116	56.39	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group18	1.11	56.38	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group19	1.111	56.37	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group2	1.115	56.28	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group20	1.108	56.47	B	0.327	B	0.031	B
B	B_additional_B_B_GSP_ConflictingRoutes_group21	1.128	56.41	B	0.326	B	0.03	B

B	B_additional_B_B_GSP_ConflictingRoutes_group22	1.156	56.35	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group23	1.136	56.29	B	0.323	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group24	1.124	56.32	B	0.324	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group25	1.115	56.4	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group26	1.11	56.38	B	0.324	B	0.031	B
B	B_additional_B_B_GSP_ConflictingRoutes_group27	1.117	56.39	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group28	1.115	56.34	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group29	1.114	56.24	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group3	1.112	56.39	B	0.328	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group30	1.116	56.32	B	0.324	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group31	1.112	56.34	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group32	1.109	56.4	B	0.323	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group33	1.138	56.41	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group34	1.141	56.42	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group35	1.127	56.38	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group36	1.121	56.4	B	0.324	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group37	1.117	56.39	B	0.324	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group38	1.116	56.34	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group39	1.118	56.44	B	0.324	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group4	1.122	56.42	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group40	1.109	56.38	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group41	1.116	56.36	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group42	1.116	56.33	B	0.328	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group43	1.11	56.27	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group44	1.111	56.42	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group45	1.152	56.34	B	0.328	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group46	1.143	56.58	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group47	1.123	56.39	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group48	1.119	56.5	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group49	1.114	56.41	B	0.324	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group5	1.114	56.46	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group50	1.115	56.34	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group51	1.115	56.51	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group52	1.115	56.34	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group53	1.112	56.36	B	0.324	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group54	1.114	56.32	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group55	1.113	56.59	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group56	1.118	56.39	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group57	1.145	56.39	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group58	1.137	56.4	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group59	1.126	56.4	B	0.324	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group6	1.115	56.34	B	0.324	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group60	1.117	56.38	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group61	1.116	56.48	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group62	1.113	56.36	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group63	1.115	56.59	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group64	1.112	56.47	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group65	1.116	56.61	B	0.328	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group66	1.112	56.55	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group67	1.111	56.49	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group68	1.133	56.41	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group69	1.151	56.67	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group7	1.132	57.04	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group70	1.122	57.52	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group71	1.117	57.56	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group72	1.114	57.64	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group73	1.117	57.6	B	0.324	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group74	1.116	57.67	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group75	1.116	57.4	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group76	1.117	57.57	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group77	1.115	57.59	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group78	1.108	57.56	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group79	1.11	57.47	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group8	1.147	56.57	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group80	1.138	56.62	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group81	1.126	56.48	B	0.326	B	0.03	B

B	B_additional_B_B_GSP_ConflictingRoutes_group82	1.119	56.41	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group83	1.112	56.38	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group84	1.111	56.38	B	0.328	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group85	1.114	56.47	B	0.328	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group86	1.114	56.64	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group87	1.113	56.39	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group88	1.116	56.41	B	0.329	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group89	1.119	56.49	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group9	1.112	56.45	B	0.328	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group90	1.111	56.49	B	0.324	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group91	1.153	56.48	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group92	1.146	56.6	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group93	1.125	57.42	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group94	1.115	57.56	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group95	1.119	57.64	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group96	1.117	57.66	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group97	1.117	57.62	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group98	1.116	57.56	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_ConflictingRoutes_group99	1.114	57.58	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_FirstOverlapWithPreviousRouteSectionRel_group0	1.118	56.4	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_FirstOverlapWithPreviousRouteSectionRel_Fast_group0	1.121	69.15	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_FirstRouteSectionReleaseWithLocking_group0	1.111	56.16	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_FirstRouteSectionReleaseWithLocking_group1	1.125	56.07	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_FirstRouteSectionReleaseWithLocking_group10	1.148	56.22	B	0.329	B	0.03	B
B	B_additional_B_B_GSP_FirstRouteSectionReleaseWithLocking_group11	1.135	55.69	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_FirstRouteSectionReleaseWithLocking_group12	1.124	54.95	B	0.328	B	0.03	B
B	B_additional_B_B_GSP_FirstRouteSectionReleaseWithLocking_group13	1.111	55.04	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_FirstRouteSectionReleaseWithLocking_group2	1.113	55.02	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_FirstRouteSectionReleaseWithLocking_group3	1.113	55.06	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_FirstRouteSectionReleaseWithLocking_group4	1.12	54.94	B	0.33	B	0.03	B
B	B_additional_B_B_GSP_FirstRouteSectionReleaseWithLocking_group5	1.115	55.07	B	0.329	B	0.03	B
B	B_additional_B_B_GSP_FirstRouteSectionReleaseWithLocking_group6	1.113	54.94	B	0.328	B	0.03	B
B	B_additional_B_B_GSP_FirstRouteSectionReleaseWithLocking_group7	1.113	54.89	B	0.329	B	0.03	B
B	B_additional_B_B_GSP_FirstRouteSectionReleaseWithLocking_group8	1.112	54.94	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_FirstRouteSectionReleaseWithLocking_group9	1.112	55.05	B	0.328	B	0.03	B
B	B_additional_B_B_GSP_OverlapSectionsInLineOfRoute_group0	1.14	54.45	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_PointControlWhenFreeToMove_group0	1.152	55.29	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_PointControlWhenFreeToMove_group1	1.127	55.35	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_PointControlWhenFreeToMove_group2	1.116	55.22	B	0.33	B	0.03	B
B	B_additional_B_B_GSP_PointControlWhenFreeToMove_group3	1.118	55.28	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_PointControlWhenFreeToMove_group4	1.108	55.38	B	0.329	B	0.03	B
B	B_additional_B_B_GSP_PointControlWhenFreeToMove_group5	1.112	55.33	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_PointControlWhenFreeToMove_group6	1.116	55.41	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_PointControlWhenKeyed_group0	1.112	55.13	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_PointControlWhenKeyed_group1	1.112	55.1	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_PointControlWhenKeyed_group2	1.112	55.36	B	0.328	B	0.03	B
B	B_additional_B_B_GSP_PointControlWhenKeyed_group3	1.107	55.54	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_PointControlWhenKeyed_group4	1.106	55.53	B	0.328	B	0.03	B
B	B_additional_B_B_GSP_PointControlWhenKeyed_group5	1.146	55.68	B	0.329	B	0.03	B
B	B_additional_B_B_GSP_PointControlWhenKeyed_group6	1.14	55.28	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_PointControlWithSectionsOverPoints_group0	1.135	87.53	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_PointControlWithSectionsOverPoints_group1	1.129	87.45	B	0.329	B	0.03	B
B	B_additional_B_B_GSP_PointControlWithSectionsOverPoints_group10	1.122	87.35	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_PointControlWithSectionsOverPoints_group11	1.126	87.91	B	0.328	B	0.03	B
B	B_additional_B_B_GSP_PointControlWithSectionsOverPoints_group12	1.125	88.9	B	0.328	B	0.03	B
B	B_additional_B_B_GSP_PointControlWithSectionsOverPoints_group13	1.125	89	B	0.329	B	0.03	B
B	B_additional_B_B_GSP_PointControlWithSectionsOverPoints_group2	1.126	88.91	B	0.328	B	0.03	B
B	B_additional_B_B_GSP_PointControlWithSectionsOverPoints_group3	1.125	88.91	B	0.328	B	0.03	B
B	B_additional_B_B_GSP_PointControlWithSectionsOverPoints_group4	1.126	88.55	B	0.328	B	0.03	B
B	B_additional_B_B_GSP_PointControlWithSectionsOverPoints_group5	1.117	88.84	B	0.329	B	0.03	B
B	B_additional_B_B_GSP_PointControlWithSectionsOverPoints_group6	1.124	89.12	B	0.329	B	0.03	B
B	B_additional_B_B_GSP_PointControlWithSectionsOverPoints_group7	1.158	89.04	B	0.329	B	0.03	B
B	B_additional_B_B_GSP_PointControlWithSectionsOverPoints_group8	1.145	88.96	B	0.328	B	0.03	B
B	B_additional_B_B_GSP_PointControlWithSectionsOverPoints_group9	1.132	88.95	B	0.33	B	0.03	B
B	B_additional_B_B_GSP_PointControlWithTechniciansControls_group0	1.113	58.07	B	0.329	B	0.03	B
B	B_additional_B_B_GSP_PointControlWithTechniciansControls_group1	1.111	58.08	B	0.331	B	0.03	B

B	B_additional_B_B_GSP_PointControlWithTechniciansControls_group2	1.11	58.05	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_PointControlWithTechniciansControls_group3	1.113	57.9	B	0.329	B	0.03	B
B	B_additional_B_B_GSP_PointControlWithTechniciansControls_group4	1.114	58	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_PointControlWithTechniciansControls_group5	1.113	58.01	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_PointControlWithTechniciansControls_group6	1.113	58.04	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_PointReminderDevice_group0	1.106	58.41	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_PointReminderDevice_group1	1.104	58.4	B	0.329	B	0.03	B
B	B_additional_B_B_GSP_PointReminderDevice_group2	1.126	57.32	B	0.329	B	0.03	B
B	B_additional_B_B_GSP_PointReminderDevice_group3	1.146	57.33	B	0.328	B	0.03	B
B	B_additional_B_B_GSP_PointReminderDevice_group4	1.126	57.3	B	0.328	B	0.03	B
B	B_additional_B_B_GSP_PointReminderDevice_group5	1.119	57.18	B	0.33	B	0.03	B
B	B_additional_B_B_GSP_PointReminderDevice_group6	1.108	57.24	B	0.33	B	0.03	B
B	B_additional_B_B_GSP_PointsCalledNormal_group0	1.118	56.86	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_PointsCalledNormal_group1	1.115	56.86	B	0.328	B	0.03	B
B	B_additional_B_B_GSP_PointsCalledNormal_group2	1.119	56.75	B	0.329	B	0.03	B
B	B_additional_B_B_GSP_PointsCalledNormal_group3	1.114	56.85	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_PointsCalledNormal_group4	1.118	56.89	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_PointsCalledNormal_group5	1.118	56.79	B	0.328	B	0.03	B
B	B_additional_B_B_GSP_PointsCalledNormal_group6	1.118	56.77	B	0.329	B	0.03	B
B	B_additional_B_B_GSP_PointsCalledReverse_group0	1.118	56.79	B	0.328	B	0.03	B
B	B_additional_B_B_GSP_PointsCalledReverse_group1	1.153	56.78	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_PointsCalledReverse_group2	1.144	56.83	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_PointsCalledReverse_group3	1.132	56.83	B	0.328	B	0.03	B
B	B_additional_B_B_GSP_PointsCalledReverse_group4	1.126	56.89	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_PointsCalledReverse_group5	1.118	56.78	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_PointsCalledReverse_group6	1.121	56.88	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_PointsInTheRoute_group0	1.113	53.95	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_PointsInTheRoute_group1	1.116	54.05	B	0.331	B	0.03	B
B	B_additional_B_B_GSP_PointsInTheRoute_group10	1.11	53.83	B	0.325	B	0.031	B
B	B_additional_B_B_GSP_PointsInTheRoute_group11	1.113	54.05	B	0.329	B	0.03	B
B	B_additional_B_B_GSP_PointsInTheRoute_group12	1.114	53.85	B	0.329	B	0.03	B
B	B_additional_B_B_GSP_PointsInTheRoute_group13	1.11	53.88	B	0.326	B	0.031	B
B	B_additional_B_B_GSP_PointsInTheRoute_group14	1.107	53.96	B	0.325	B	0.031	B
B	B_additional_B_B_GSP_PointsInTheRoute_group15	1.154	54.02	B	0.327	B	0.031	B
B	B_additional_B_B_GSP_PointsInTheRoute_group16	1.14	53.85	B	0.327	B	0.031	B
B	B_additional_B_B_GSP_PointsInTheRoute_group17	1.125	54.07	B	0.329	B	0.031	B
B	B_additional_B_B_GSP_PointsInTheRoute_group18	1.115	53.86	B	0.329	B	0.031	B
B	B_additional_B_B_GSP_PointsInTheRoute_group19	1.115	53.9	B	0.327	B	0.031	B
B	B_additional_B_B_GSP_PointsInTheRoute_group2	1.113	53.89	B	0.326	B	0.031	B
B	B_additional_B_B_GSP_PointsInTheRoute_group20	1.12	54.38	B	0.329	B	0.031	B
B	B_additional_B_B_GSP_PointsInTheRoute_group21	1.116	54.27	B	0.33	B	0.03	B
B	B_additional_B_B_GSP_PointsInTheRoute_group22	1.116	54.25	B	0.33	B	0.031	B
B	B_additional_B_B_GSP_PointsInTheRoute_group23	1.114	54.24	B	0.326	B	0.031	B
B	B_additional_B_B_GSP_PointsInTheRoute_group24	1.118	54.03	B	0.325	B	0.031	B
B	B_additional_B_B_GSP_PointsInTheRoute_group25	1.108	54.72	B	0.328	B	0.03	B
B	B_additional_B_B_GSP_PointsInTheRoute_group26	1.122	54.13	B	0.328	B	0.03	B
B	B_additional_B_B_GSP_PointsInTheRoute_group27	1.153	54.27	B	0.327	B	0.031	B
B	B_additional_B_B_GSP_PointsInTheRoute_group28	1.134	54.58	B	0.328	B	0.031	B
B	B_additional_B_B_GSP_PointsInTheRoute_group29	1.124	54.15	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_PointsInTheRoute_group3	1.114	54.28	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_PointsInTheRoute_group30	1.111	54.37	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_PointsInTheRoute_group31	1.113	54.46	B	0.328	B	0.031	B
B	B_additional_B_B_GSP_PointsInTheRoute_group32	1.116	54.6	B	0.328	B	0.031	B
B	B_additional_B_B_GSP_PointsInTheRoute_group33	1.111	54.19	B	0.327	B	0.031	B
B	B_additional_B_B_GSP_PointsInTheRoute_group34	1.109	54.49	B	0.328	B	0.031	B
B	B_additional_B_B_GSP_PointsInTheRoute_group35	1.113	54.39	B	0.329	B	0.031	B
B	B_additional_B_B_GSP_PointsInTheRoute_group36	1.115	54.26	B	0.326	B	0.031	B
B	B_additional_B_B_GSP_PointsInTheRoute_group37	1.107	54.48	B	0.327	B	0.031	B
B	B_additional_B_B_GSP_PointsInTheRoute_group38	1.132	54.49	B	0.329	B	0.031	B
B	B_additional_B_B_GSP_PointsInTheRoute_group39	1.151	54.5	B	0.327	B	0.031	B
B	B_additional_B_B_GSP_PointsInTheRoute_group4	1.129	54.21	B	0.326	B	0.031	B
B	B_additional_B_B_GSP_PointsInTheRoute_group40	1.118	54.32	B	0.328	B	0.03	B
B	B_additional_B_B_GSP_PointsInTheRoute_group41	1.114	54.52	B	0.326	B	0.031	B
B	B_additional_B_B_GSP_PointsInTheRoute_group5	1.111	54.29	B	0.329	B	0.031	B
B	B_additional_B_B_GSP_PointsInTheRoute_group6	1.118	54.67	B	0.326	B	0.031	B
B	B_additional_B_B_GSP_PointsInTheRoute_group7	1.118	54.21	B	0.327	B	0.03	B

B	B_additional_B_B_GSP_PointsInTheRoute_group8	1.114	54.48	B	0.328	B	0.03	B
B	B_additional_B_B_GSP_PointsInTheRoute_group9	1.114	54.34	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_ProceedWithExitSignal_group0	1.12	56.12	B	0.326	B	0.031	B
B	B_additional_B_B_GSP_ProceedWithOverlapTracksClear_group0	1.118	56.19	B	0.327	B	0.031	B
B	B_additional_B_B_GSP_ProceedWithTracksClear_group0	1.111	55.89	B	0.326	B	0.031	B
B	B_additional_B_B_GSP_ProceedWithTracksClear_group1	1.152	56.28	B	0.327	B	0.031	B
B	B_additional_B_B_GSP_ProceedWithTracksClear_group10	1.147	55.89	B	0.326	B	0.031	B
B	B_additional_B_B_GSP_ProceedWithTracksClear_group11	1.129	56.1	B	0.325	B	0.031	B
B	B_additional_B_B_GSP_ProceedWithTracksClear_group12	1.121	56.02	B	0.326	B	0.031	B
B	B_additional_B_B_GSP_ProceedWithTracksClear_group13	1.115	56.39	B	0.329	B	0.031	B
B	B_additional_B_B_GSP_ProceedWithTracksClear_group14	1.119	55.95	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_ProceedWithTracksClear_group15	1.12	56.28	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_ProceedWithTracksClear_group16	1.121	55.88	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_ProceedWithTracksClear_group17	1.117	55.93	B	0.325	B	0.031	B
B	B_additional_B_B_GSP_ProceedWithTracksClear_group18	1.115	56.03	B	0.324	B	0.031	B
B	B_additional_B_B_GSP_ProceedWithTracksClear_group19	1.122	56.25	B	0.325	B	0.031	B
B	B_additional_B_B_GSP_ProceedWithTracksClear_group2	1.112	55.9	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_ProceedWithTracksClear_group3	1.117	56	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_ProceedWithTracksClear_group4	1.154	56.17	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_ProceedWithTracksClear_group5	1.141	55.93	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_ProceedWithTracksClear_group6	1.127	56.12	B	0.326	B	0.031	B
B	B_additional_B_B_GSP_ProceedWithTracksClear_group7	1.119	56.24	B	0.327	B	0.031	B
B	B_additional_B_B_GSP_ProceedWithTracksClear_group8	1.122	55.88	B	0.325	B	0.031	B
B	B_additional_B_B_GSP_ProceedWithTracksClear_group9	1.117	55.73	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_RouteCancellation_group0	1.092	54.96	B	0.349	B	0.031	B
B	B_additional_B_B_GSP_RouteCancellation_group1	1.09	55.07	B	0.351	B	0.03	B
B	B_additional_B_B_GSP_RouteCancellation_group10	1.091	55.46	B	0.351	B	0.03	B
B	B_additional_B_B_GSP_RouteCancellation_group11	1.092	55.11	B	0.352	B	0.03	B
B	B_additional_B_B_GSP_RouteCancellation_group12	1.093	55.17	B	0.351	B	0.031	B
B	B_additional_B_B_GSP_RouteCancellation_group13	1.088	55.21	B	0.353	B	0.031	B
B	B_additional_B_B_GSP_RouteCancellation_group2	1.1	55.23	B	0.351	B	0.03	B
B	B_additional_B_B_GSP_RouteCancellation_group3	1.126	55.22	B	0.35	B	0.031	B
B	B_additional_B_B_GSP_RouteCancellation_group4	1.108	54.99	B	0.351	B	0.031	B
B	B_additional_B_B_GSP_RouteCancellation_group5	1.097	55.23	B	0.35	B	0.031	B
B	B_additional_B_B_GSP_RouteCancellation_group6	1.092	54.86	B	0.35	B	0.031	B
B	B_additional_B_B_GSP_RouteCancellation_group7	1.091	54.84	B	0.35	B	0.031	B
B	B_additional_B_B_GSP_RouteCancellation_group8	1.092	54.96	B	0.349	B	0.03	B
B	B_additional_B_B_GSP_RouteCancellation_group9	1.091	55.01	B	0.35	B	0.031	B
B	B_additional_B_B_GSP_RouteSectionLockedAheadOfTrain_group0	1.118	55.52	B	0.329	B	0.03	B
B	B_additional_B_B_GSP_RouteSectionLockedAheadOfTrain_group1	1.111	55.97	B	0.327	B	0.031	B
B	B_additional_B_B_GSP_RouteSectionLockedAheadOfTrain_group10	1.112	60.88	B	0.327	B	0.031	B
B	B_additional_B_B_GSP_RouteSectionLockedAheadOfTrain_group11	1.115	60.18	B	0.33	B	0.031	B
B	B_additional_B_B_GSP_RouteSectionLockedAheadOfTrain_group12	1.105	60.65	B	0.33	B	0.031	B
B	B_additional_B_B_GSP_RouteSectionLockedAheadOfTrain_group2	1.133	55.6	B	0.333	B	0.031	B
B	B_additional_B_B_GSP_RouteSectionLockedAheadOfTrain_group3	1.149	56.29	B	0.326	B	0.031	B
B	B_additional_B_B_GSP_RouteSectionLockedAheadOfTrain_group4	1.135	59.92	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_RouteSectionLockedAheadOfTrain_group5	1.12	60.9	B	0.329	B	0.031	B
B	B_additional_B_B_GSP_RouteSectionLockedAheadOfTrain_group6	1.121	57.77	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_RouteSectionLockedAheadOfTrain_group7	1.113	54.3	B	0.326	B	0.031	B
B	B_additional_B_B_GSP_RouteSectionLockedAheadOfTrain_group8	1.114	57.73	B	0.325	B	0.031	B
B	B_additional_B_B_GSP_RouteSectionLockedAheadOfTrain_group9	1.116	60.08	B	0.329	B	0.031	B
B	B_additional_B_B_GSP_RouteSectionLockedAheadOfTrainAlt_group0	1.115	55.68	B	0.327	B	0.031	B
B	B_additional_B_B_GSP_RouteSectionLockedAheadOfTrainAlt_group1	1.113	56.25	B	0.325	B	0.031	B
B	B_additional_B_B_GSP_RouteSectionLockedAheadOfTrainAlt_group10	1.114	60.79	B	0.327	B	0.031	B
B	B_additional_B_B_GSP_RouteSectionLockedAheadOfTrainAlt_group11	1.111	60.25	B	0.327	B	0.031	B
B	B_additional_B_B_GSP_RouteSectionLockedAheadOfTrainAlt_group12	1.106	60.93	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_RouteSectionLockedAheadOfTrainAlt_group2	1.147	55.56	B	0.327	B	0.031	B
B	B_additional_B_B_GSP_RouteSectionLockedAheadOfTrainAlt_group3	1.133	56.14	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_RouteSectionLockedAheadOfTrainAlt_group4	1.13	60	B	0.326	B	0.031	B
B	B_additional_B_B_GSP_RouteSectionLockedAheadOfTrainAlt_group5	1.117	60.81	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_RouteSectionLockedAheadOfTrainAlt_group6	1.114	58.39	B	0.326	B	0.031	B
B	B_additional_B_B_GSP_RouteSectionLockedAheadOfTrainAlt_group7	1.116	54.64	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_RouteSectionLockedAheadOfTrainAlt_group8	1.113	57.91	B	0.325	B	0.031	B
B	B_additional_B_B_GSP_RouteSectionLockedAheadOfTrainAlt_group9	1.118	60.39	B	0.328	B	0.03	B
B	B_additional_B_B_GSP_RouteSectionLockedOverPoints_group0	1.11	57.45	B	0.323	B	0.031	B
B	B_additional_B_B_GSP_RouteSectionLockedOverPoints_group1	1.109	57.21	B	0.325	B	0.03	B

B	B_additional_B_B_GSP_RouteSectionLockedOverPoints_group10	1.114	57.15	B	0.323	B	0.031	B
B	B_additional_B_B_GSP_RouteSectionLockedOverPoints_group11	1.105	57.36	B	0.325	B	0.031	B
B	B_additional_B_B_GSP_RouteSectionLockedOverPoints_group12	1.106	57.62	B	0.324	B	0.03	B
B	B_additional_B_B_GSP_RouteSectionLockedOverPoints_group13	1.15	57.41	B	0.324	B	0.031	B
B	B_additional_B_B_GSP_RouteSectionLockedOverPoints_group2	1.134	57.25	B	0.325	B	0.031	B
B	B_additional_B_B_GSP_RouteSectionLockedOverPoints_group3	1.12	57.3	B	0.325	B	0.031	B
B	B_additional_B_B_GSP_RouteSectionLockedOverPoints_group4	1.11	57.67	B	0.325	B	0.031	B
B	B_additional_B_B_GSP_RouteSectionLockedOverPoints_group5	1.112	57.2	B	0.324	B	0.03	B
B	B_additional_B_B_GSP_RouteSectionLockedOverPoints_group6	1.112	56.99	B	0.325	B	0.031	B
B	B_additional_B_B_GSP_RouteSectionLockedOverPoints_group7	1.116	57.3	B	0.323	B	0.03	B
B	B_additional_B_B_GSP_RouteSectionLockedOverPoints_group8	1.112	57.11	B	0.325	B	0.031	B
B	B_additional_B_B_GSP_RouteSectionLockedOverPoints_group9	1.111	57.21	B	0.324	B	0.03	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group0	1.114	0.6	B	0.323	B	0.031	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group1	1.115	0.602	B	0.325	B	0.031	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group10	1.109	0.602	B	0.324	B	0.03	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group100	1.12	0.604	B	0.324	B	0.031	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group101	1.153	0.601	B	0.324	B	0.031	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group102	1.133	0.605	B	0.325	B	0.031	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group103	1.122	0.599	B	0.324	B	0.031	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group104	1.113	0.601	B	0.324	B	0.031	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group105	1.112	0.604	B	0.324	B	0.031	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group106	1.115	0.604	B	0.323	B	0.03	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group107	1.112	0.602	B	0.324	B	0.03	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group108	1.114	0.602	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group109	1.11	0.605	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group11	1.112	0.602	B	0.324	B	0.03	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group110	1.11	0.601	B	0.324	B	0.03	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group111	1.109	0.602	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group112	1.128	0.603	B	0.326	B	0.031	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group113	1.149	0.603	B	0.326	B	0.031	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group114	1.128	0.602	B	0.324	B	0.03	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group115	1.121	0.604	B	0.323	B	0.03	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group116	1.115	0.601	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group117	1.112	0.602	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group118	1.112	0.602	B	0.325	B	0.031	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group119	1.117	0.602	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group12	1.112	0.603	B	0.325	B	0.031	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group120	1.113	0.599	B	0.323	B	0.03	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group121	1.113	0.602	B	0.324	B	0.031	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group122	1.108	0.601	B	0.324	B	0.03	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group123	1.108	0.603	B	0.323	B	0.03	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group124	1.151	0.605	B	0.323	B	0.03	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group125	1.137	0.6	B	0.326	B	0.031	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group126	1.122	0.601	B	0.324	B	0.03	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group127	1.116	0.6	B	0.325	B	0.031	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group128	1.113	0.602	B	0.326	B	0.031	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group129	1.111	0.603	B	0.323	B	0.03	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group13	1.114	0.602	B	0.324	B	0.031	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group130	1.111	0.603	B	0.324	B	0.03	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group131	1.112	0.603	B	0.323	B	0.03	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group14	1.113	0.603	B	0.325	B	0.031	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group15	1.114	0.599	B	0.323	B	0.031	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group16	1.106	0.605	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group17	1.129	0.603	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group18	1.145	0.599	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group19	1.129	0.599	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group2	1.121	0.601	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group20	1.112	0.603	B	0.324	B	0.03	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group21	1.116	0.603	B	0.323	B	0.03	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group22	1.114	0.603	B	0.324	B	0.03	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group23	1.117	0.602	B	0.326	B	0.031	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group24	1.114	0.604	B	0.324	B	0.031	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group25	1.112	0.604	B	0.325	B	0.031	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group26	1.116	0.606	B	0.325	B	0.031	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group27	1.11	0.628	B	0.326	B	0.031	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group28	1.112	0.611	B	0.325	B	0.03	B

B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group29	1.154	0.603	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group3	1.142	0.593	B	0.324	B	0.031	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group30	1.125	0.588	B	0.326	B	0.031	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group31	1.118	0.585	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group32	1.114	0.581	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group33	1.115	0.581	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group34	1.115	0.58	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group35	1.117	0.581	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group36	1.113	0.581	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group37	1.11	0.582	B	0.326	B	0.031	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group38	1.108	0.579	B	0.325	B	0.031	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group39	1.11	0.58	B	0.327	B	0.031	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group4	1.14	0.579	B	0.324	B	0.031	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group40	1.14	0.577	B	0.326	B	0.031	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group41	1.125	0.579	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group42	1.118	0.58	B	0.326	B	0.031	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group43	1.113	0.578	B	0.333	B	0.03	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group44	1.114	0.58	B	0.326	B	0.031	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group45	1.113	0.575	B	0.326	B	0.031	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group46	1.113	0.578	B	0.324	B	0.031	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group47	1.108	0.576	B	0.326	B	0.031	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group48	1.113	0.577	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group49	1.111	0.581	B	0.326	B	0.031	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group5	1.111	0.579	B	0.323	B	0.031	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group50	1.116	0.583	B	0.325	B	0.031	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group51	1.112	0.579	B	0.324	B	0.031	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group52	1.109	0.58	B	0.325	B	0.031	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group53	1.124	0.578	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group54	1.147	0.581	B	0.324	B	0.031	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group55	1.131	0.581	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group56	1.124	0.579	B	0.328	B	0.031	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group57	1.112	0.582	B	0.324	B	0.03	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group58	1.114	0.581	B	0.323	B	0.031	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group59	1.114	0.582	B	0.325	B	0.031	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group6	1.113	0.581	B	0.325	B	0.031	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group60	1.111	0.583	B	0.324	B	0.03	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group61	1.113	0.578	B	0.326	B	0.031	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group62	1.114	0.58	B	0.328	B	0.031	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group63	1.114	0.585	B	0.324	B	0.031	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group64	1.117	0.577	B	0.324	B	0.031	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group65	1.119	0.581	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group66	1.112	0.581	B	0.323	B	0.03	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group67	1.109	0.578	B	0.324	B	0.031	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group68	1.152	0.582	B	0.324	B	0.031	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group69	1.142	0.581	B	0.325	B	0.031	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group7	1.126	0.581	B	0.326	B	0.031	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group70	1.114	0.58	B	0.324	B	0.03	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group71	1.116	0.581	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group72	1.115	0.584	B	0.327	B	0.031	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group73	1.113	0.58	B	0.329	B	0.03	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group74	1.113	0.582	B	0.326	B	0.031	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group75	1.115	0.581	B	0.324	B	0.031	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group76	1.113	0.582	B	0.325	B	0.031	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group77	1.113	0.584	B	0.326	B	0.031	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group78	1.115	0.581	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group79	1.115	0.581	B	0.326	B	0.031	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group8	1.114	0.578	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group80	1.113	0.583	B	0.327	B	0.031	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group81	1.14	0.58	B	0.327	B	0.031	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group82	1.136	0.58	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group83	1.125	0.581	B	0.325	B	0.031	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group84	1.117	0.58	B	0.326	B	0.031	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group85	1.111	0.583	B	0.326	B	0.031	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group86	1.114	0.58	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group87	1.115	0.58	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group88	1.116	0.584	B	0.326	B	0.031	B

B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group89	1.107	0.582	B	0.324	B	0.03	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group9	1.114	0.583	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group90	1.113	0.583	B	0.326	B	0.031	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group91	1.115	0.583	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group92	1.113	0.585	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group93	1.113	0.586	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group94	1.109	0.582	B	0.328	B	0.03	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group95	1.134	0.581	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group96	1.149	0.584	B	0.326	B	0.031	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group97	1.13	0.583	B	0.325	B	0.031	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group98	1.121	0.581	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_RoutesFromEntranceSignal_group99	1.116	0.584	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_SignalGroupReplacement_group0	1.109	57.01	B	0.326	B	0.031	B
B	B_additional_B_B_GSP_SignalGroupReplacement_group1	1.113	57.22	B	0.325	B	0.031	B
B	B_additional_B_B_GSP_SignalGroupReplacement_group10	1.112	57.48	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_SignalGroupReplacement_group11	1.106	57.36	B	0.327	B	0.031	B
B	B_additional_B_B_GSP_SignalGroupReplacement_group12	1.11	57.26	B	0.326	B	0.031	B
B	B_additional_B_B_GSP_SignalGroupReplacement_group13	1.113	57.67	B	0.324	B	0.03	B
B	B_additional_B_B_GSP_SignalGroupReplacement_group2	1.107	57.69	B	0.324	B	0.031	B
B	B_additional_B_B_GSP_SignalGroupReplacement_group3	1.105	57.51	B	0.324	B	0.03	B
B	B_additional_B_B_GSP_SignalGroupReplacement_group4	1.152	57.36	B	0.323	B	0.031	B
B	B_additional_B_B_GSP_SignalGroupReplacement_group5	1.141	57.6	B	0.326	B	0.031	B
B	B_additional_B_B_GSP_SignalGroupReplacement_group6	1.121	57.36	B	0.323	B	0.031	B
B	B_additional_B_B_GSP_SignalGroupReplacement_group7	1.114	57.42	B	0.324	B	0.03	B
B	B_additional_B_B_GSP_SignalGroupReplacement_group8	1.115	57.36	B	0.324	B	0.031	B
B	B_additional_B_B_GSP_SignalGroupReplacement_group9	1.117	57.51	B	0.324	B	0.03	B
B	B_additional_B_B_GSP_SignalReminder_group0	1.12	57.99	B	0.324	B	0.03	B
B	B_additional_B_B_GSP_SignalReminder_group1	1.124	57.88	B	0.326	B	0.031	B
B	B_additional_B_B_GSP_SignalReminder_group10	1.121	57.89	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_SignalReminder_group11	1.118	58.15	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_SignalReminder_group12	1.115	57.88	B	0.324	B	0.031	B
B	B_additional_B_B_GSP_SignalReminder_group13	1.114	58.03	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_SignalReminder_group2	1.142	57.99	B	0.326	B	0.031	B
B	B_additional_B_B_GSP_SignalReminder_group3	1.145	57.82	B	0.325	B	0.031	B
B	B_additional_B_B_GSP_SignalReminder_group4	1.134	57.7	B	0.324	B	0.031	B
B	B_additional_B_B_GSP_SignalReminder_group5	1.122	57.67	B	0.325	B	0.031	B
B	B_additional_B_B_GSP_SignalReminder_group6	1.121	57.77	B	0.326	B	0.031	B
B	B_additional_B_B_GSP_SignalReminder_group7	1.122	58.2	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_SignalReminder_group8	1.122	57.94	B	0.325	B	0.031	B
B	B_additional_B_B_GSP_SignalReminder_group9	1.119	57.78	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_SubsequentRouteSectionRelease_group0	1.116	55.53	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_SubsequentRouteSectionRelease_group1	1.117	55.45	B	0.325	B	0.031	B
B	B_additional_B_B_GSP_SubsequentRouteSectionRelease_group10	1.12	55.32	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_SubsequentRouteSectionRelease_group11	1.114	56.21	B	0.326	B	0.031	B
B	B_additional_B_B_GSP_SubsequentRouteSectionRelease_group12	1.132	56.32	B	0.325	B	0.031	B
B	B_additional_B_B_GSP_SubsequentRouteSectionRelease_group13	1.158	56.36	B	0.325	B	0.031	B
B	B_additional_B_B_GSP_SubsequentRouteSectionRelease_group14	1.137	56.49	B	0.326	B	0.031	B
B	B_additional_B_B_GSP_SubsequentRouteSectionRelease_group15	1.13	56.32	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_SubsequentRouteSectionRelease_group2	1.12	56.12	B	0.325	B	0.031	B
B	B_additional_B_B_GSP_SubsequentRouteSectionRelease_group3	1.118	55.59	B	0.325	B	0.031	B
B	B_additional_B_B_GSP_SubsequentRouteSectionRelease_group4	1.121	59.18	B	0.328	B	0.031	B
B	B_additional_B_B_GSP_SubsequentRouteSectionRelease_group5	1.119	55.86	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_SubsequentRouteSectionRelease_group6	1.124	59.24	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_SubsequentRouteSectionRelease_group7	1.115	55.54	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_SubsequentRouteSectionRelease_group8	1.125	56.56	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_SubsequentRouteSectionRelease_group9	1.115	55.53	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_SubsequentRouteSectionRelease_Fast_group0	1.119	60.38	B	0.328	B	0.031	B
B	B_additional_B_B_GSP_SubsequentRouteSectionRelease_Fast_group1	1.163	58.7	B	0.328	B	0.03	B
B	B_additional_B_B_GSP_SubsequentRouteSectionRelease_Fast_group10	1.151	55.56	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_SubsequentRouteSectionRelease_Fast_group11	1.13	55.79	B	0.325	B	0.031	B
B	B_additional_B_B_GSP_SubsequentRouteSectionRelease_Fast_group12	1.122	56.35	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_SubsequentRouteSectionRelease_Fast_group13	1.123	59.8	B	0.328	B	0.03	B
B	B_additional_B_B_GSP_SubsequentRouteSectionRelease_Fast_group14	1.122	56.05	B	0.328	B	0.031	B
B	B_additional_B_B_GSP_SubsequentRouteSectionRelease_Fast_group15	1.122	60.07	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_SubsequentRouteSectionRelease_Fast_group16	1.125	56.11	B	0.328	B	0.03	B
B	B_additional_B_B_GSP_SubsequentRouteSectionRelease_Fast_group2	1.122	60.08	B	0.329	B	0.031	B

B	B_additional_B_B_GSP_SubsequentRouteSectionRelease_Fast_group3	1.122	58.46	B	0.331	B	0.03	B
B	B_additional_B_B_GSP_SubsequentRouteSectionRelease_Fast_group4	1.123	59.65	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_SubsequentRouteSectionRelease_Fast_group5	1.118	55.91	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_SubsequentRouteSectionRelease_Fast_group6	1.151	56.07	B	0.324	B	0.03	B
B	B_additional_B_B_GSP_SubsequentRouteSectionRelease_Fast_group7	1.15	56.24	B	0.324	B	0.031	B
B	B_additional_B_B_GSP_SubsequentRouteSectionRelease_Fast_group8	1.13	55.39	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_SubsequentRouteSectionRelease_Fast_group9	1.123	55.44	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_TechniciansRouteBar_group0	1.118	55.54	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_TechniciansRouteBar_group1	1.119	55.73	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_TechniciansRouteBar_group10	1.117	55.69	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_TechniciansRouteBar_group11	1.117	55.44	B	0.324	B	0.03	B
B	B_additional_B_B_GSP_TechniciansRouteBar_group12	1.115	55.58	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_TechniciansRouteBar_group13	1.118	55.65	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_TechniciansRouteBar_group2	1.119	55.28	B	0.324	B	0.03	B
B	B_additional_B_B_GSP_TechniciansRouteBar_group3	1.116	55.33	B	0.324	B	0.03	B
B	B_additional_B_B_GSP_TechniciansRouteBar_group4	1.128	55.58	B	0.325	B	0.031	B
B	B_additional_B_B_GSP_TechniciansRouteBar_group5	1.154	55.41	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_TechniciansRouteBar_group6	1.139	55.27	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_TechniciansRouteBar_group7	1.131	55.66	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_TechniciansRouteBar_group8	1.117	55.74	B	0.324	B	0.03	B
B	B_additional_B_B_GSP_TechniciansRouteBar_group9	1.12	55.47	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group0	1.116	57.39	B	0.324	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group1	1.118	57.28	B	0.328	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group10	1.118	57.18	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group11	1.114	57.27	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group12	1.121	57.22	B	0.324	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group13	1.113	57.45	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group14	1.113	57.15	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group15	1.165	57.03	B	0.324	B	0.031	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group16	1.143	57.16	B	0.324	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group17	1.13	57.25	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group18	1.122	57.6	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group19	1.116	57.09	B	0.333	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group2	1.115	57.22	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group20	1.117	57.36	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group21	1.115	57.03	B	0.324	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group22	1.117	56.98	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group23	1.118	57.08	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group24	1.116	57.58	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group25	1.114	57.51	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group26	1.141	57.64	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group27	1.146	57.17	B	0.324	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group28	1.131	57.29	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group29	1.123	57.4	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group3	1.119	57.52	B	0.33	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group30	1.118	57.43	B	0.329	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group31	1.119	57.36	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group32	1.121	57.4	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group33	1.114	57.38	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group34	1.119	56.99	B	0.328	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group35	1.116	57.07	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group36	1.11	57.26	B	0.328	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group37	1.124	57.15	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group38	1.154	56.98	B	0.328	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group39	1.139	57.22	B	0.329	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group4	1.126	57.3	B	0.329	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group40	1.116	57.13	B	0.328	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group41	1.118	57.26	B	0.328	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group42	1.118	57.32	B	0.329	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group43	1.119	57.24	B	0.329	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group44	1.12	57.42	B	0.328	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group45	1.118	57.13	B	0.328	B	0.031	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group46	1.116	57.13	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group47	1.111	57.06	B	0.329	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group48	1.11	57.11	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group49	1.153	57.39	B	0.329	B	0.03	B

B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group5	1.139	57.12	B	0.328	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group50	1.123	57.18	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group51	1.122	57.21	B	0.328	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group52	1.114	57.3	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group53	1.116	57.26	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group54	1.117	57.06	B	0.328	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group55	1.12	57.16	B	0.329	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group56	1.111	57.03	B	0.328	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group57	1.115	57.03	B	0.328	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group58	1.113	57.35	B	0.328	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group59	1.113	57.53	B	0.329	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group6	1.135	57.03	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group60	1.153	57.28	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group61	1.131	57.35	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group62	1.124	57.2	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group63	1.116	57.4	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group64	1.119	57.26	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group65	1.119	57.01	B	0.329	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group66	1.12	57.05	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group67	1.115	57.1	B	0.328	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group7	1.116	57.11	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group8	1.116	57.2	B	0.326	B	0.031	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_group9	1.109	57.35	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_Fast_group0	1.114	55.7	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_Fast_group1	1.156	55.67	B	0.329	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_Fast_group10	1.148	55.79	B	0.33	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_Fast_group11	1.132	59.17	B	0.328	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_Fast_group12	1.119	55.59	B	0.328	B	0.031	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_Fast_group13	1.121	55.62	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_Fast_group14	1.119	56.74	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_Fast_group15	1.123	56.15	B	0.328	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_Fast_group16	1.119	56.05	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_Fast_group17	1.122	55.94	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_Fast_group18	1.121	56	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_Fast_group19	1.119	56.05	B	0.328	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_Fast_group2	1.12	59.94	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_Fast_group20	1.156	56.2	B	0.328	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_Fast_group21	1.145	55.99	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_Fast_group22	1.129	55.87	B	0.328	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_Fast_group23	1.121	56.53	B	0.329	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_Fast_group24	1.12	55.96	B	0.329	B	0.031	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_Fast_group25	1.121	55.89	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_Fast_group26	1.122	55.99	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_Fast_group27	1.118	55.43	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_Fast_group28	1.114	55.69	B	0.328	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_Fast_group29	1.12	56	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_Fast_group3	1.116	55.48	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_Fast_group30	1.115	55.45	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_Fast_group31	1.136	55.48	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_Fast_group32	1.164	56.24	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_Fast_group33	1.136	55.67	B	0.327	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_Fast_group34	1.125	55.59	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_Fast_group4	1.12	55.61	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_Fast_group5	1.118	59.06	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_Fast_group6	1.116	55.62	B	0.325	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_Fast_group7	1.121	55.56	B	0.328	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_Fast_group8	1.122	59.04	B	0.326	B	0.03	B
B	B_additional_B_B_GSP_TrackSectionsInLineOfRoute_Fast_group9	1.122	55.4	B	0.325	B	0.03	B

C.3 Runtimes of classifying the Siemens Interlocking Testbed grouped by Category

	LLV			IC3ref			ABC		
	A	B	C	A	B	CT	A	B	CT
Range	0.1313	51.7657	407.6185	2.8967	5.7795	3.8308	0.0010	0.0011	0.0007
Standard Deviation	0.0470	11.7204	95.9976	1.0590	1.6153	1.6618	0.0002	0.0003	0.0002
2_1_11		74.8510			0.7400			0.0299	
2_1_12	1.4237			0.6081			0.0299		
2_1_13	1.4220			0.6086			0.0300		
2_1_16	1.5298	81.7734		0.7668	0.6814		0.0301	0.0300	
2_1_17	1.4210			0.7210			0.0303		
2_1_18	1.4715			0.5750			0.0303		
2_1_1			69.0110			0.3956			0.0302
2_1_20	1.5144			0.6796			0.0302		
2_1_21		78.0895	79.2970		0.6825	0.7490		0.0303	0.0304
2_1_22			170.3198			0.8284			0.0304
2_1_23	1.4758	84.1228		0.7408	0.6753		0.0305	0.0305	
2_1_24	1.4945	72.1710		0.9701	0.7390		0.0305	0.0308	
2_1_2			142.4809			0.3964			0.0307
2_1_3			474.2771			0.4014			0.0306
2_1_4	1.5367	90.9371	162.0728	0.7258	0.7315	0.7006	0.0307	0.0306	0.0306
2_1_5	1.5367	74.1630	151.8910	0.6940	0.7380	0.6883	0.0305	0.0306	0.0306
2_1_6			168.3713			0.3971			0.0307
2_1_8			303.1003			0.3978			0.0307
2_2_12	1.5063			2.0365			0.0302		
2_2_16	1.4246		102.1254	2.7334		3.8683	0.0305		0.0302
2_2_1	1.4258			2.9052			0.0303		
2_2_21	1.4436			2.9822			0.0303		
2_2_22	1.5192			2.7066			0.0303		
2_2_23		82.3959			0.5693			0.0303	
2_2_24	1.5256	75.3998	107.6927	2.8764	2.6555	3.8074	0.0304	0.0304	0.0305
2_2_25	1.5250	73.8108	96.4184	2.5512	3.0788	3.6151	0.0307	0.0305	0.0306
2_2_26		74.5433			3.3143			0.0305	
2_2_27	1.5304	82.8434		2.5512	3.4774		0.0305	0.0306	
2_2_28	1.4183			0.4826			0.0306		
2_2_2	1.4483	81.6921	121.5478	2.4660	4.7086	3.7865	0.0307	0.0305	0.0306
2_2_30	1.4251			0.3973			0.0305		
2_2_31	1.4811			2.0333			0.0305		
2_2_32	1.4458	74.7750		2.7572	1.9710		0.0306	0.0305	
2_2_34		76.4785			0.3995			0.0306	
2_2_3	1.5040		115.9169	2.7638		3.7197	0.0306		0.0306
2_2_43	1.4207			3.0052			0.0305		
2_2_48	1.4220	72.6770	66.6586	3.2940	1.9700	3.9062	0.0303	0.0310	0.0308
2_2_50		123.7370			6.1790			0.0303	
2_2_51		84.3319			3.0452			0.0305	
2_2_52		71.9713			3.1217			0.0305	
2_2_53	1.5240			3.1023			0.0305		

2_2_54	1.4206		103.0845	2.7080		3.6782	0.0306		0.0306
2_2_55	1.4053			0.7176			0.0306		
2_2_5			113.3165			3.4378			0.0306
2_2_6	1.5163			2.8767			0.0309		
2_2_7	1.5192	93.3562	102.7491	2.5692	3.0056	3.6319	0.0302	0.0301	0.0302
2_2_8	1.4195		88.5988	2.8878		4.2264	0.0302		0.0301