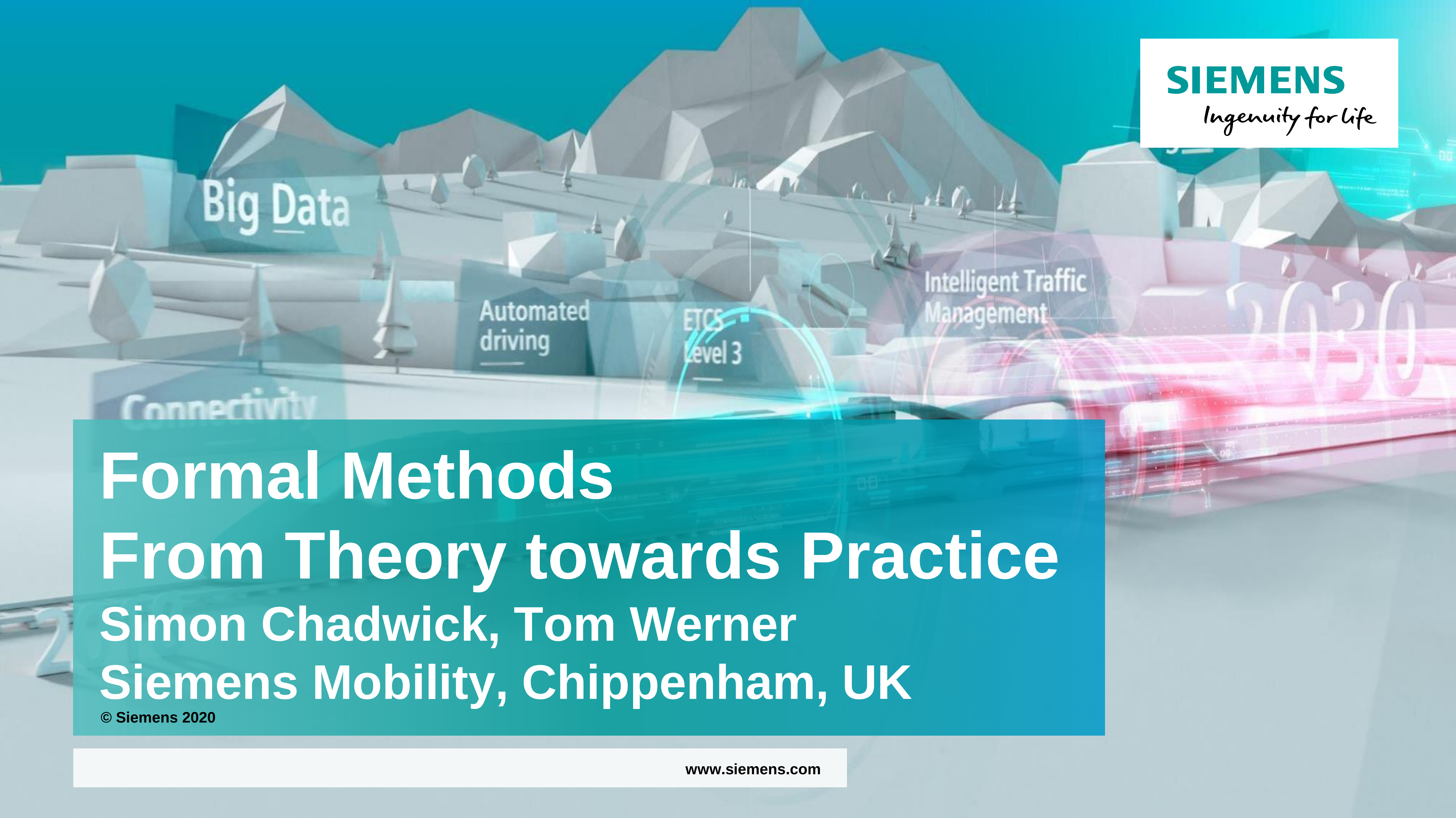
SIEMENS
Ingenuity for life



Formal Methods
From Theory towards Practice

Simon Chadwick, Tom Werner
Siemens Mobility, Chippenham, UK

www.siemens.com

Formal Verification – The Journey from Theory towards Practice
Table of contents

- The idea
- Background
- The idea re-visited
- …and then (1)
- …and then (2)
- …and then (3)
- Demonstration
- No such thing
- Completeness?

Formal Verification – The Journey from Theory towards Practice
The Idea

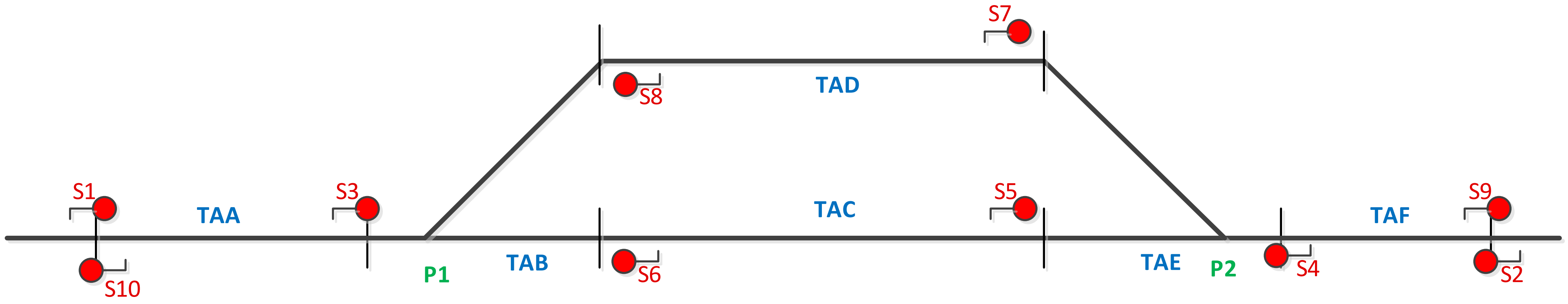
SIEMENS
Ingenuity for life

We can apply Formal Verification to Interlocking Logic

- Safety: Formal Verification of Safety Properties would improve safety
- Efficiency: Formal Verification of Safety Properties could reduce testing requirements

Formal Verification – The Journey from Theory towards Practice
Background

Railway signalling systems typically have a component called “Interlocking”.
Consider the passing loop below:



Some combinations of signals, points and track section occupancy are OK
Some combinations of signals, points and track section occupancy are not OK

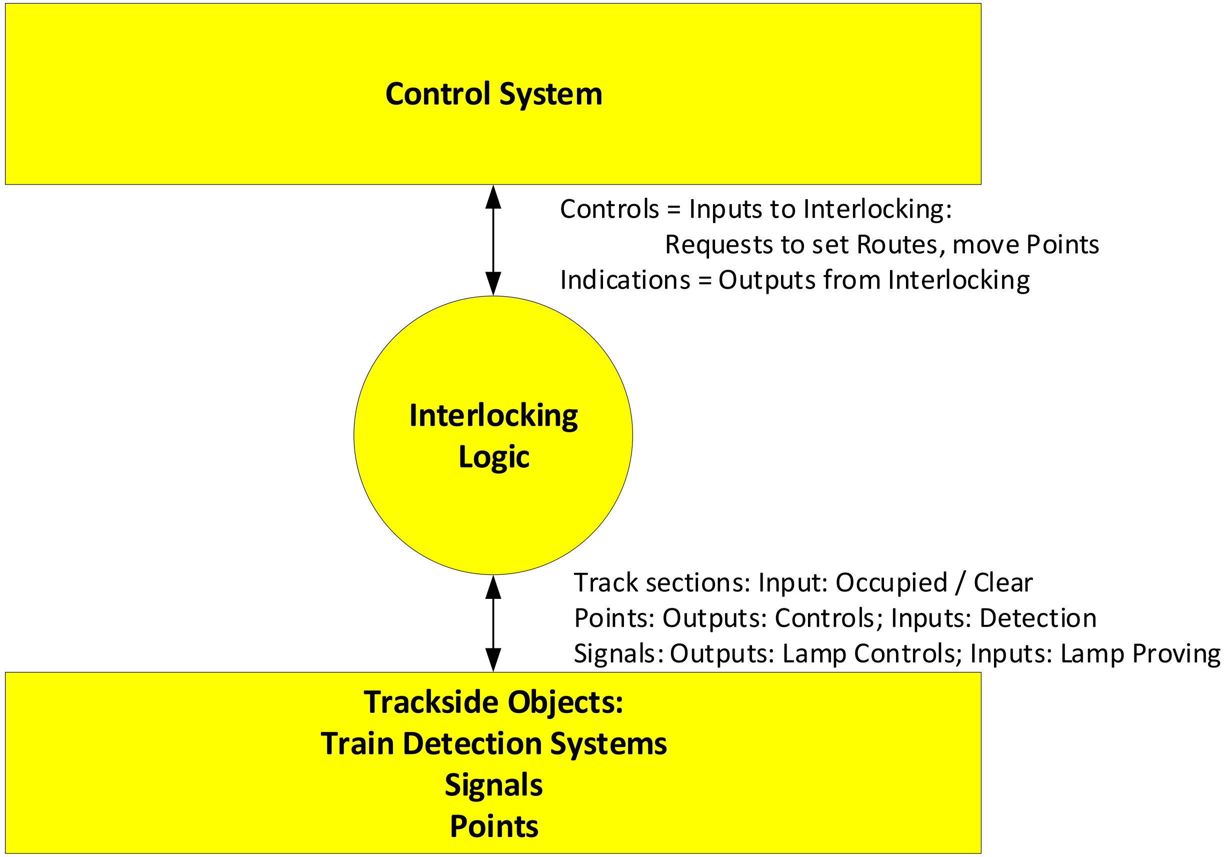
Formal Verification – The Journey from Theory towards Practice
Background

Different technologies have been used through the history of railway signalling.

- Mechanical interlocking
- Relay interlocking
- Electronic interlocking

Formal Verification – The Journey from Theory towards Practice
Background

An abstraction of an electronic interlocking is:


Control System
Controls = Inputs to Interlocking:
Requests to set Routes, move Points
Indications = Outputs from Interlocking
Interlocking Logic
Track sections: Input: Occupied / Clear
Points: Outputs: Controls; Inputs: Detection
Signals: Outputs: Lamp Controls; Inputs: Lamp Proving
Trackside Objects:
Train Detection Systems
Signals
Points

Reminder:

- Combinations of inputs matter
- There are stored states, so sequences of combinations of inputs matter
- There are timers, so the durations of each step in a sequence of combinations of inputs matter

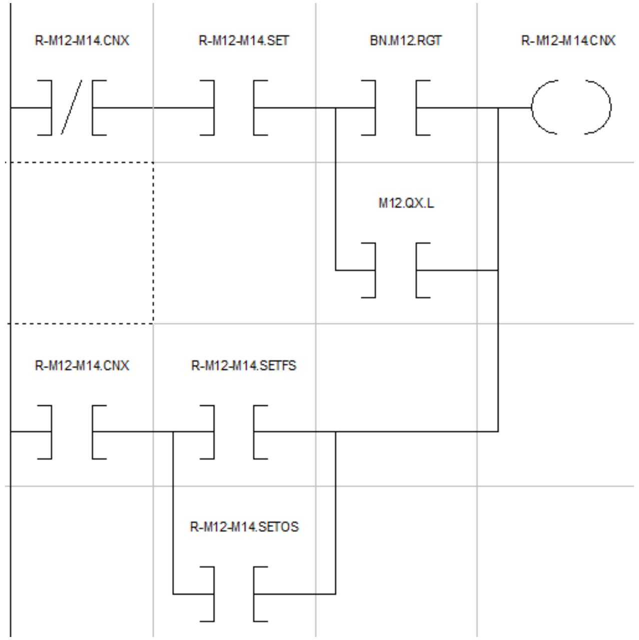
Some combinations of inputs and outputs are OK
Some combinations of inputs and outputs are not OK

Formal Verification – The Journey from Theory towards Practice
Background

Within Siemens, we have a specific interlocking product called “WESTRACE”.

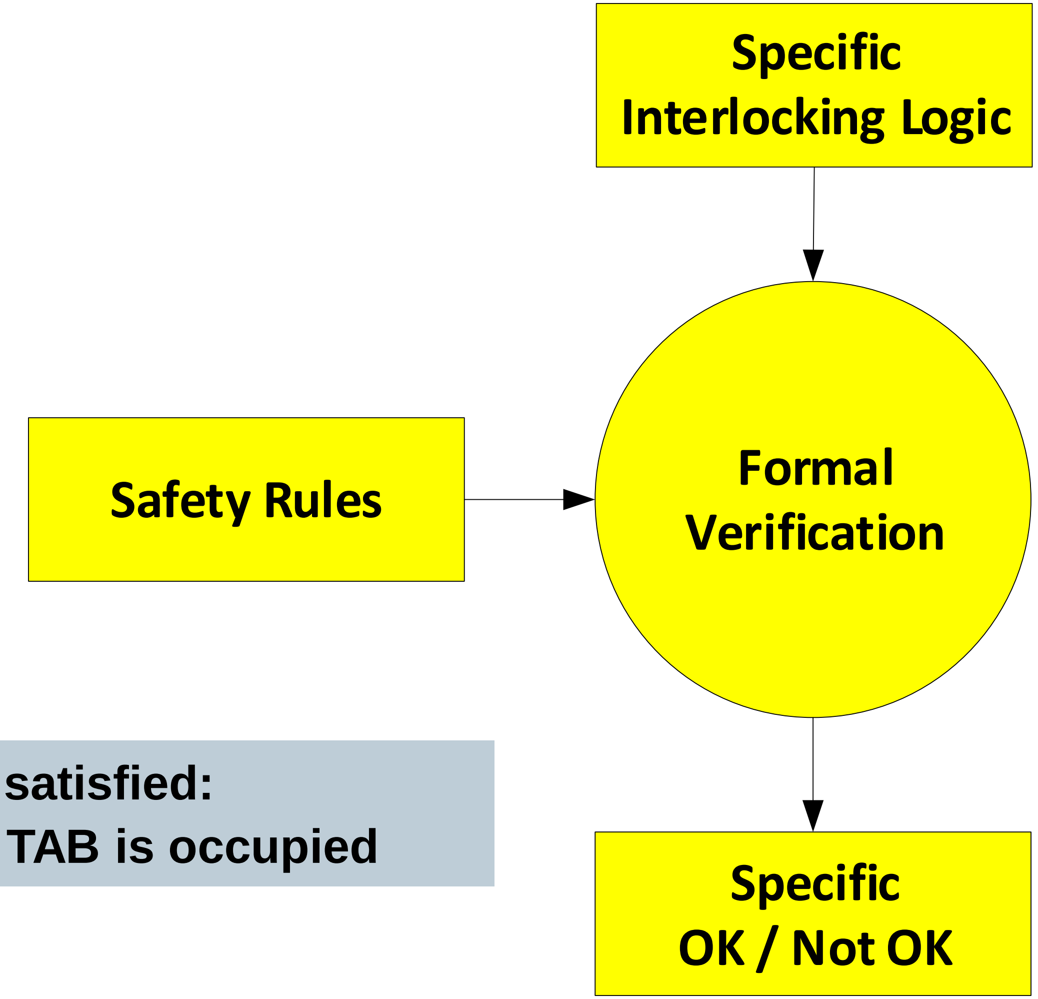
It is essentially an application-specific PLC.

It is programmed with a simplified form of Ladder Logic.



Formal Verification – The Journey from Theory towards Practice
The Idea Revisited

Now we can refine the idea:



Safety Rules define requirements which must be satisfied:
e.g. Point P1 must not be moved if Track Section TAB is occupied

Formal Verification – The Journey from Theory towards Practice
The Idea Revisited

SIEMENS
Ingenuity for life

An example safety rule:
Point P1 must not be moved if Track Section TAB is occupied

Point ‘P1’ commanded normal – “P1.NL”
Point ‘P1’ commanded reverse – “P1.RL”

$$(("\mathbf{P1.NL\_0}" \wedge "\mathbf{P1.RL\_1}") \vee ("\mathbf{P1.RL\_0}" \wedge "\mathbf{P1.NL\_1}")) \Rightarrow \neg("\mathbf{TAB.OCC(IL)\_1}")$$

Point Normal to Reverse   Point Reverse to Normal

If we don’t have a ‘point is moving’ state, then in order to model this property we need to be able to consider multiple states (0 / ‘current’ and 1 / ‘next’).

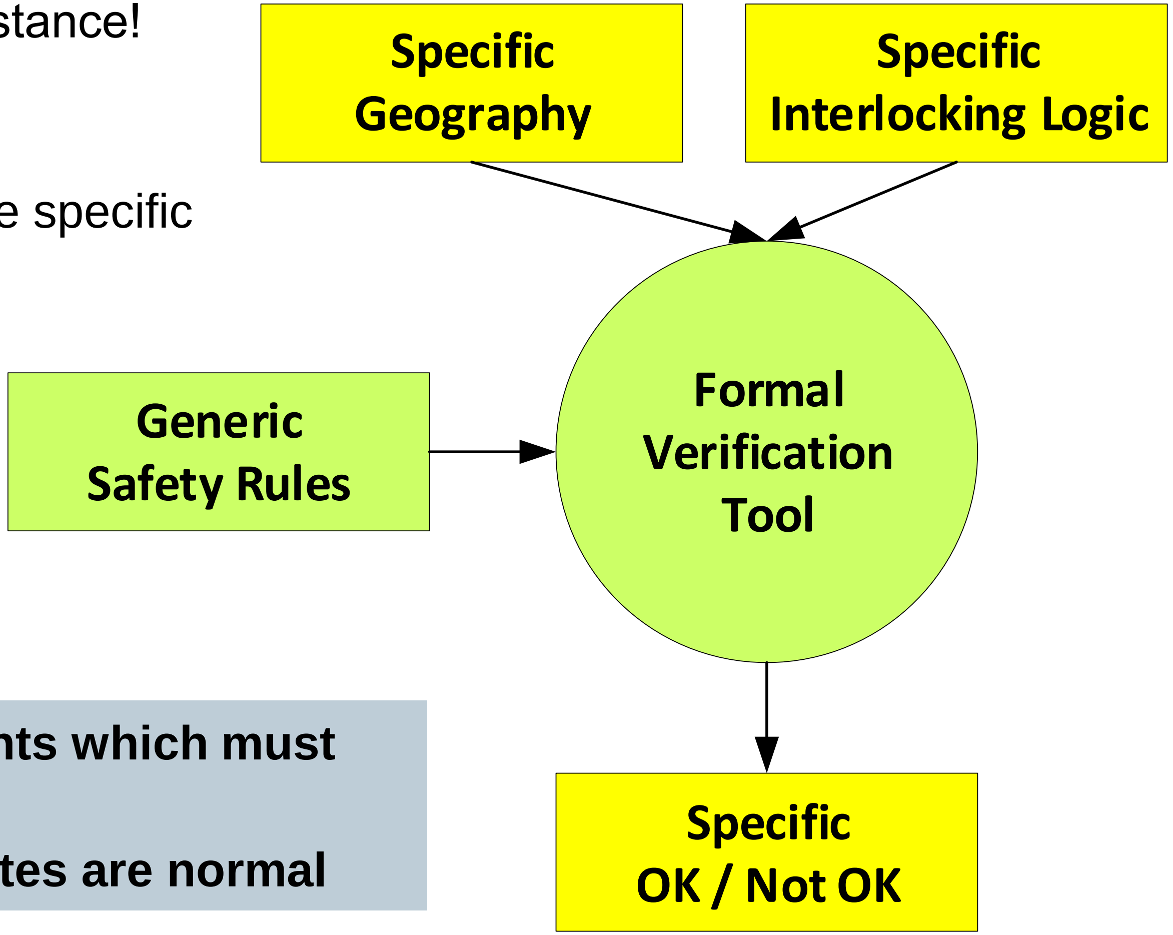
This works, but it is not enough…..

Formal Verification – The Journey from Theory towards Practice …and Then (1)

SIEMENS
Ingenuity for life

We don't want to write the Safety Rules for every instance!
We want a machine to help with that!

We need to find a machine readable way to input the specific geography



Generic Safety Rules define generic requirements which must be satisfied:
e.g. A route can only be set if all conflicting routes are normal

Formal Verification – The Journey from Theory towards Practice
Safety Requirements

SIEMENS
Ingenuity for life

Source:

2.1.3 Conflicting Routes

A route can only be set if all conflicting routes are normal.

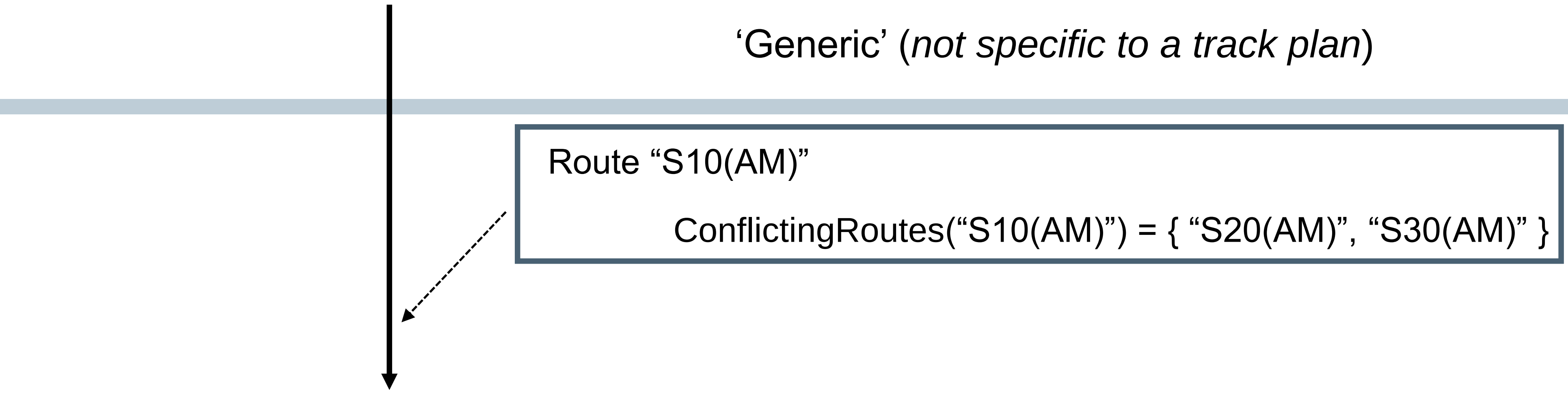
‘Generic’ (*not specific to a track plan*)

Formal Verification – The Journey from Theory towards Practice
Safety Requirements

Source:

2.1.3 Conflicting Routes

A route can only be set if all conflicting routes are normal.



‘Generic’ (*not specific to a track plan*)

Route “S10(AM)”

ConflictingRoutes(“S10(AM)”) = { “S20(AM)”, “S30(AM)” }

“S10(AM)” can only be set if “S20(AM)” and “S30(AM)” are normal.

‘Concrete’ (*specific to a track plan*)

Formal Verification – The Journey from Theory towards Practice
Safety Requirements

SIEMENS
Ingenuity for life

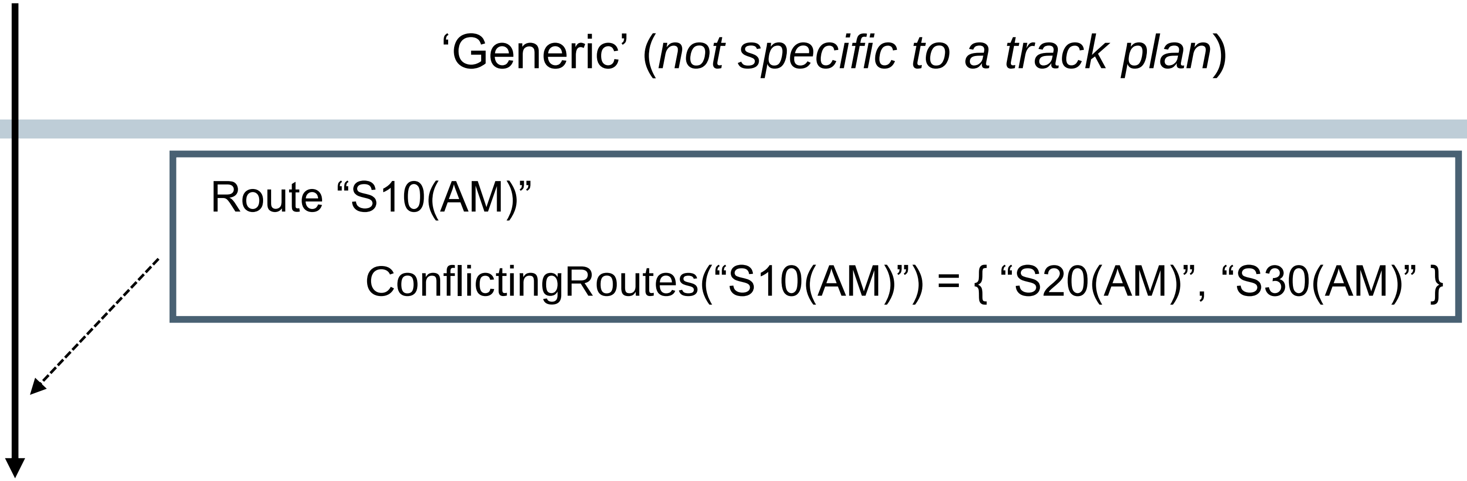
Source:

2.1.3 Conflicting Routes

A route can only be set if all conflicting routes are normal.

$$\begin{aligned}&\forall\, r1 \in Route\\ &\quad routeSet\,(r1) \;\Rightarrow\\ &\qquad \forall\, r2 \in ConflictingRoutes\,(r1)\\ &\qquad\quad routeNormal\,(r2)\end{aligned}$$

‘Generic’ (*not specific to a track plan*)



Route “S10(AM)”

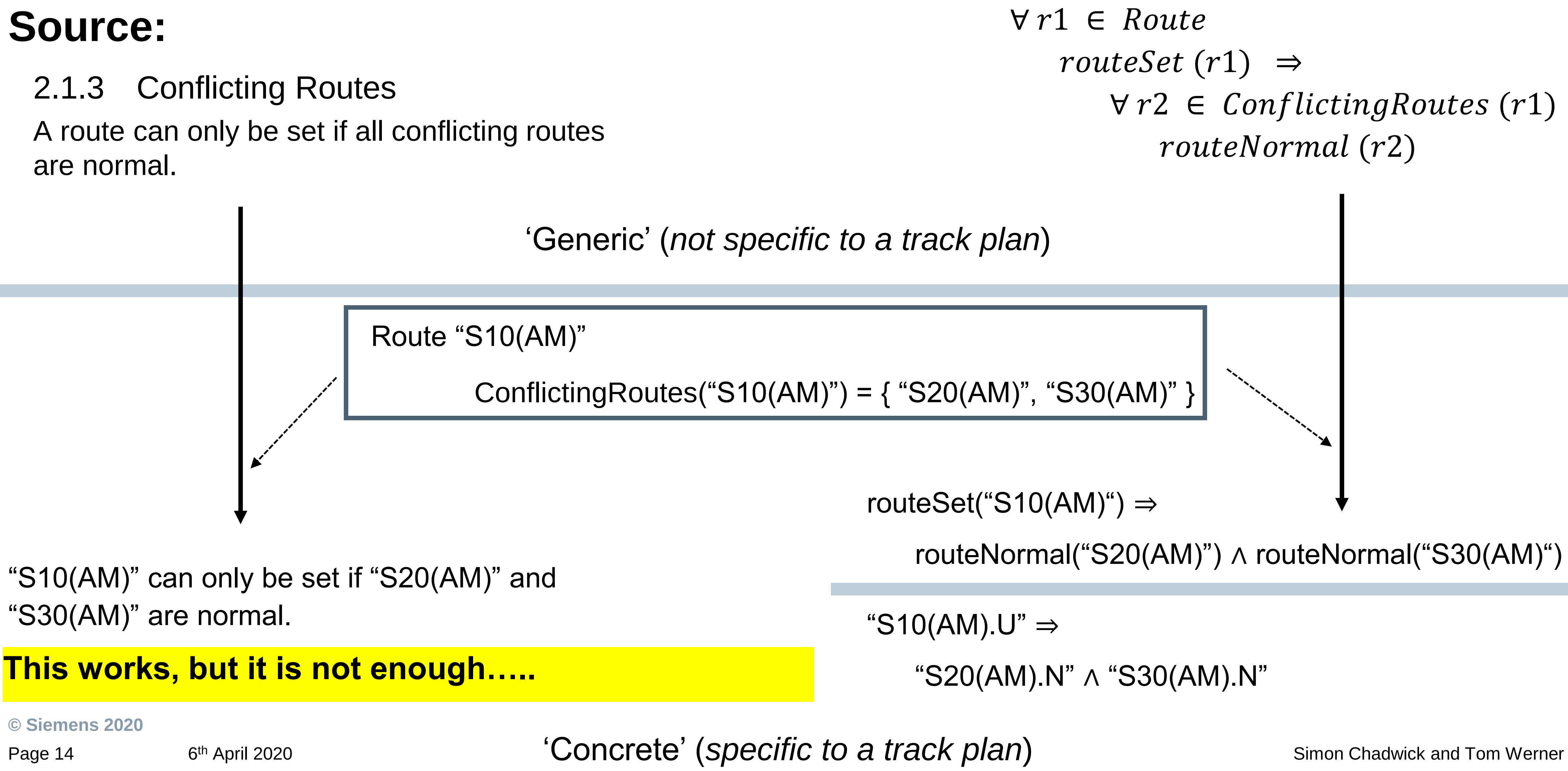
ConflictingRoutes(“S10(AM)”) = { “S20(AM)”, “S30(AM)” }

“S10(AM)” can only be set if “S20(AM)” and “S30(AM)” are normal.

‘Concrete’ (*specific to a track plan*)

Formal Verification – The Journey from Theory towards Practice
Safety Requirements

SIEMENS
Ingenuity for life



Source:

2.1.3 Conflicting Routes

A route can only be set if all conflicting routes are normal.

$$
\begin{aligned}
&\forall\, r1 \in Route \\
&\quad routeSet\,(r1) \;\Rightarrow \\
&\qquad \forall\, r2 \in ConflictingRoutes\,(r1) \\
&\qquad\quad routeNormal\,(r2)
\end{aligned}
$$

‘Generic’ (*not specific to a track plan*)

Route “S10(AM)”

ConflictingRoutes(“S10(AM)”) = { “S20(AM)”, “S30(AM)” }

routeSet(“S10(AM)“) ⇒

routeNormal(“S20(AM)”) ∧ routeNormal(“S30(AM)“)

“S10(AM)” can only be set if “S20(AM)” and “S30(AM)” are normal.

“S10(AM).U” ⇒

“S20(AM).N” ∧ “S30(AM).N”

This works, but it is not enough…..

‘Concrete’ (*specific to a track plan*)

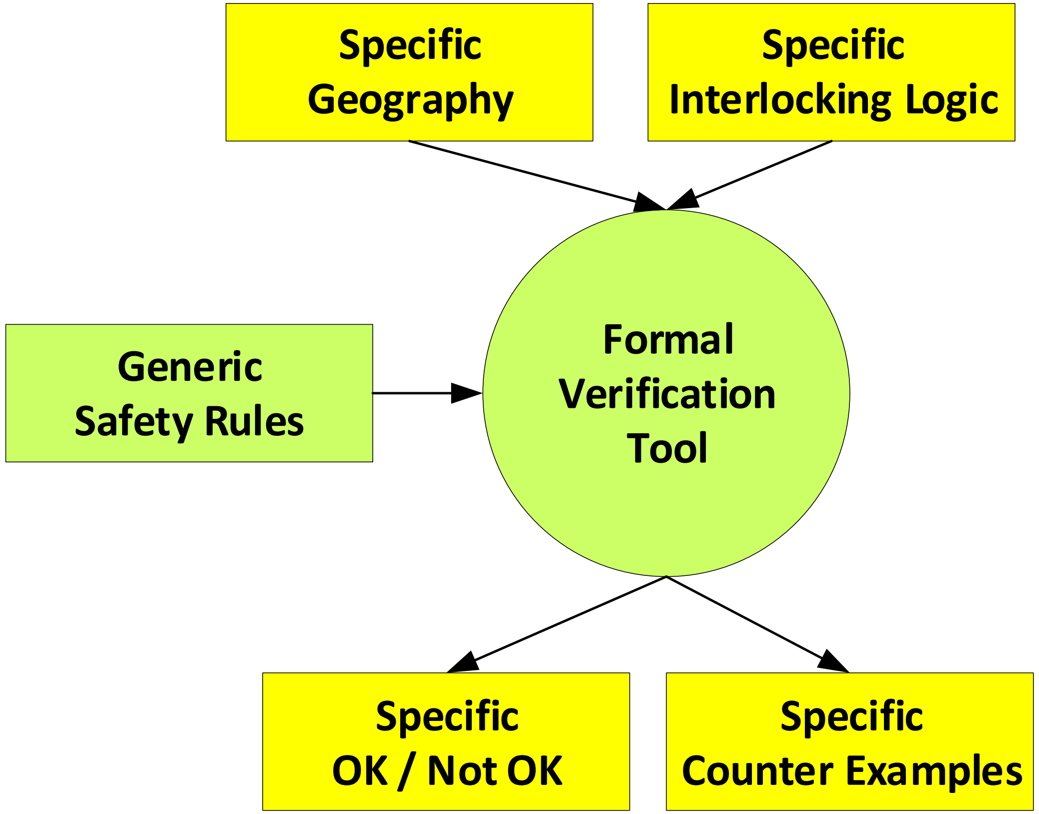
Formal Verification – The Journey from Theory towards Practice …and Then (2)

SIEMENS
Ingenuity for life

We have to be able to understand what it means if the computer says "No"!

There can be many counter examples!

Counter examples should present the steps from initialisation to the invalid state

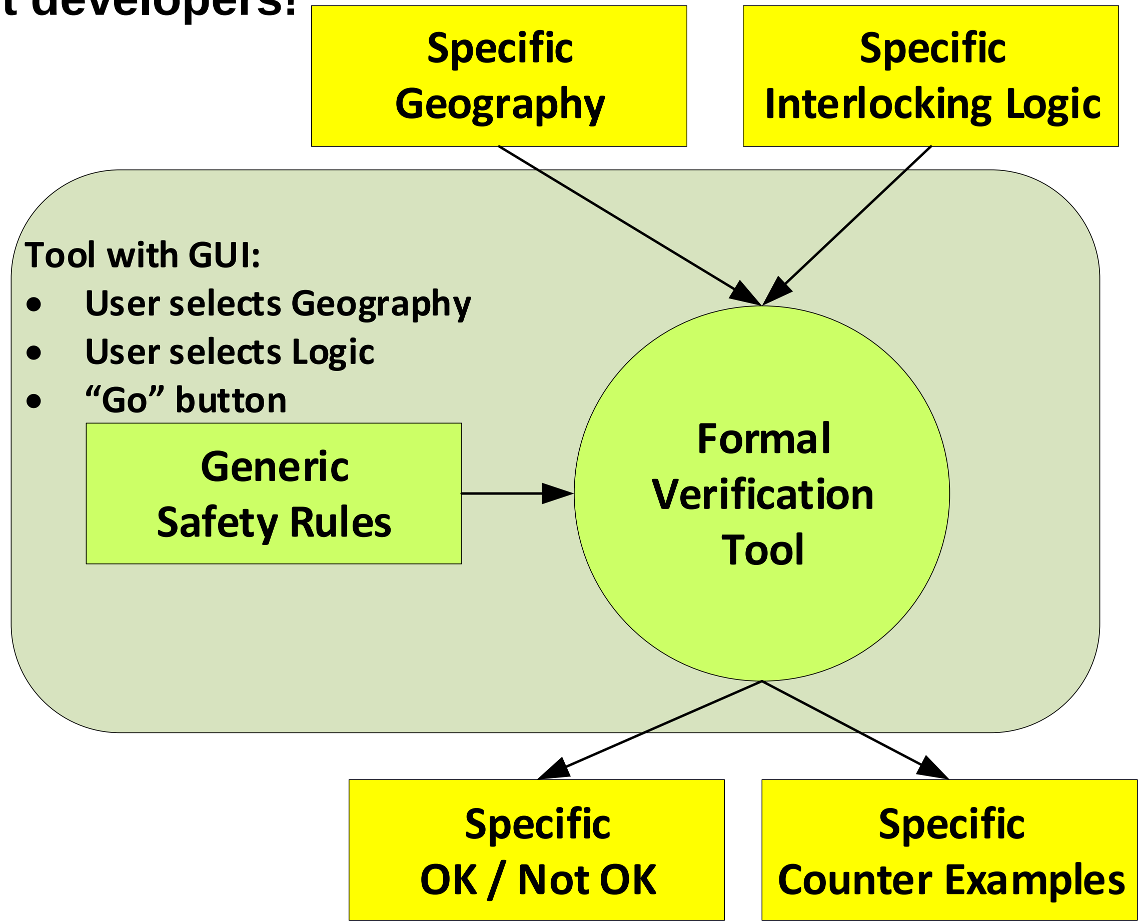


Getting there, but it is still not enough…..

Formal Verification – The Journey from Theory towards Practice …and Then (3)

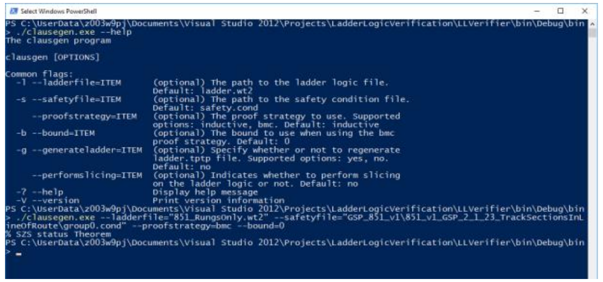
SIEMENS
Ingenuity for life

We need to provide a neat package for users, not developers!



Formal Verification – The Journey from Theory towards Practice
Stages of tool development

Stage 1 (Swansea - original)



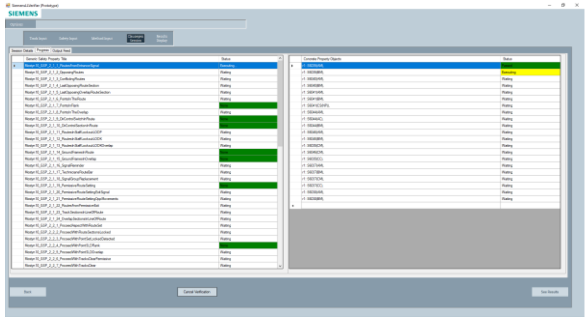
```
PS C:\UserData\z003w9pj\Documents\Visual Studio 2012\Projects\LadderLogicVerification\LLVerifier\bin\Debug\bin
> ./clausegen.exe --help
The clausgen program

clausgen [OPTIONS]

Common flags:
  -l --ladderfile=ITEM      (optional) The path to the ladder logic file.
                            Default: ladder.wt2
  -s --safetyfile=ITEM      (optional) The path to the safety condition file.
                            Default: safety.cond
     --proofstrategy=ITEM   (optional) The proof strategy to use. Supported
                            options: inductive, bmc. Default: inductive
  -b --bound=ITEM           (optional) The bound to use when using the bmc
                            proof strategy. Default: 0
  -g --generateladder=ITEM  (optional) Specify whether or not to regenerate
                            ladder.tptp file. Supported options: yes, no.
                            Default: no
     --performslicing=ITEM  (optional) Indicates whether to perform slicing
                            on the ladder logic or not. Default: no
  -? --help                 Display help message
  -V --version              Print version information
PS C:\UserData\z003w9pj\Documents\Visual Studio 2012\Projects\LadderLogicVerification\LLVerifier\bin\Debug\bin
> ./clausegen.exe --ladderfile="851_RungsOnly.wt2" --safetyfile="GSP_851_v1\851_v1_GSP_2_1_23_TrackSectionsInL
ineOfRoute\group0.cond" --proofstrategy=bmc --bound=0
% SZS status Theorem
PS C:\UserData\z003w9pj\Documents\Visual Studio 2012\Projects\LadderLogicVerification\LLVerifier\bin\Debug\bin
> _
```

- Command-line tool.
- Safety properties manually written in propositional logic for a particular track plan.

Stage 2 (Siemens - current)



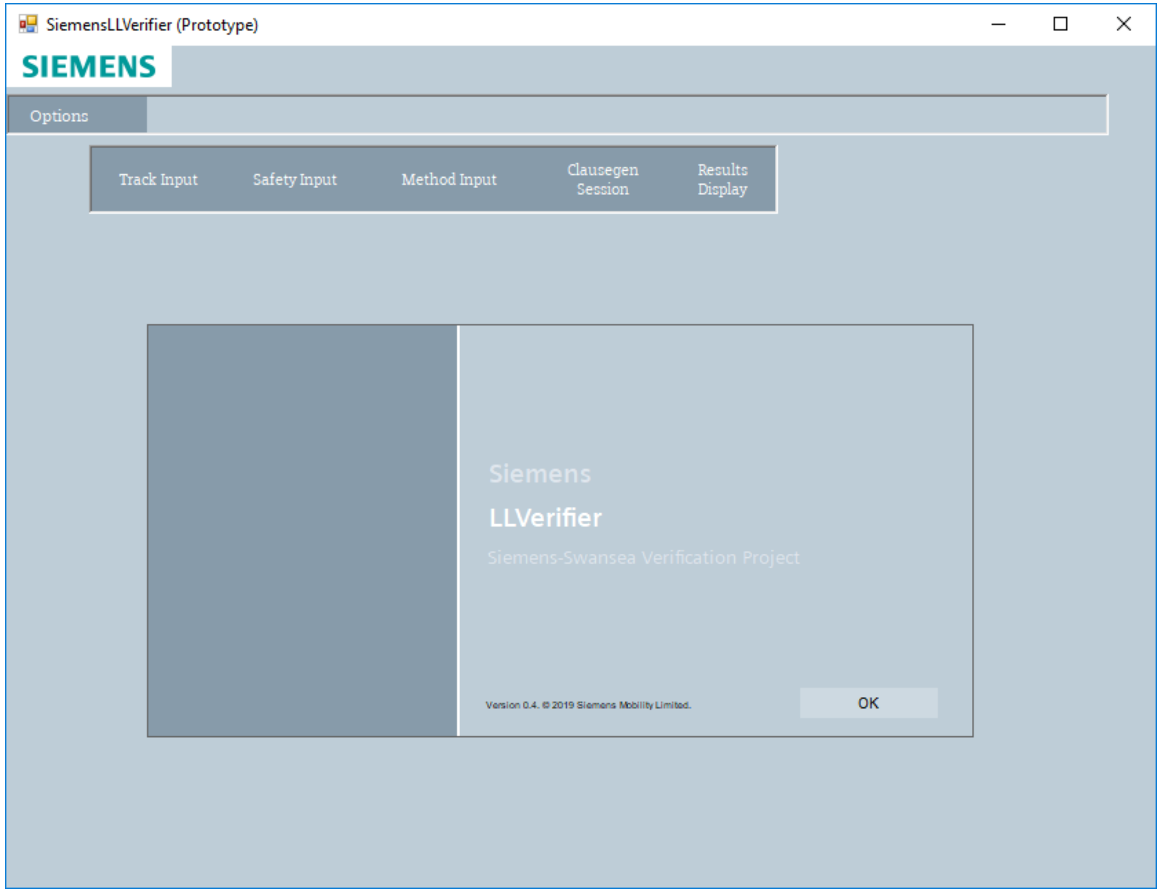
- Basic graphical front-end.
- Local windows application.
- Pre-modelled list of safety properties for particular signalling rules.
- Automatic generation of track-specific properties.

Stage 3 (Siemens - Future)

?

- Local front-end interfacing to external server for execution.
- Embedded into existing tools.
- Handling of verification outputs (reports, etc).

Formal Verification – The Journey from Theory towards Practice
Demonstration



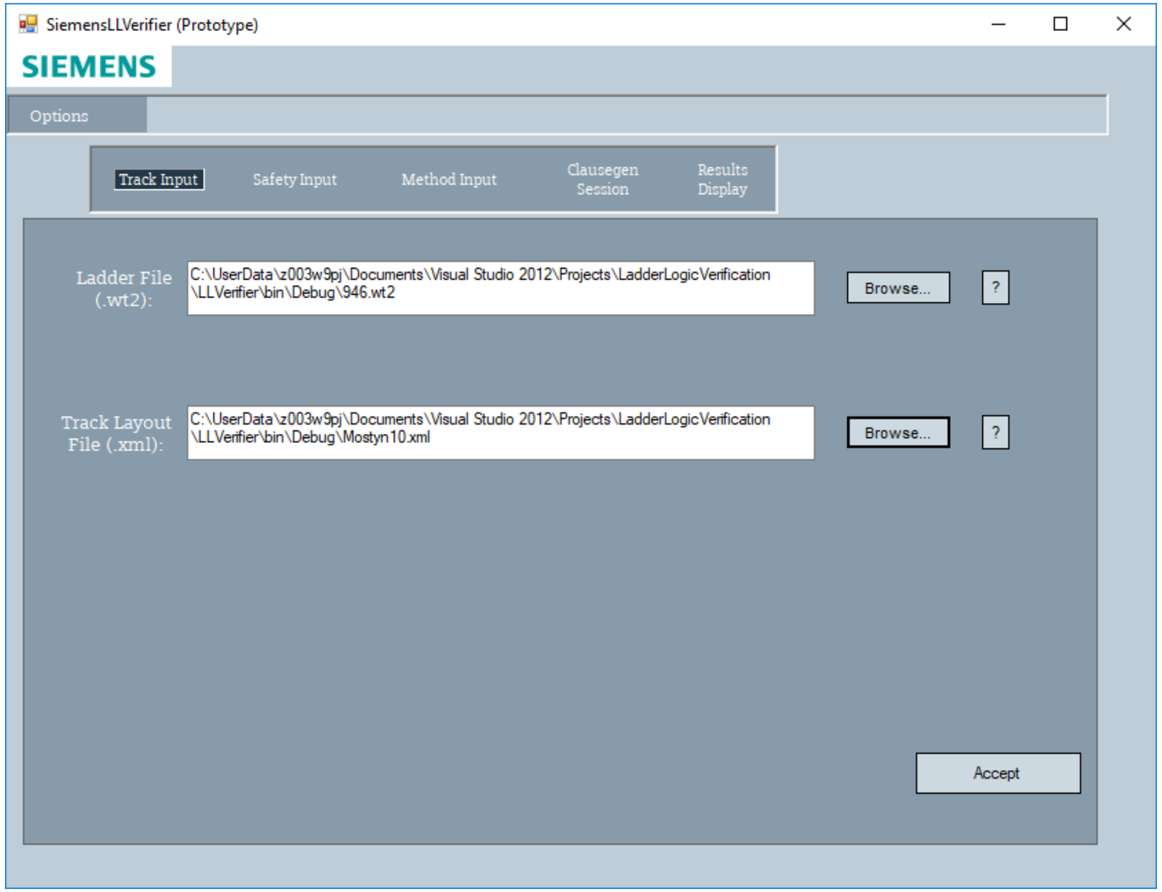
Siemens

LLVerifier

Siemens-Swansea Verification Project

Version 0.4. © 2019 Siemens Mobility Limited.

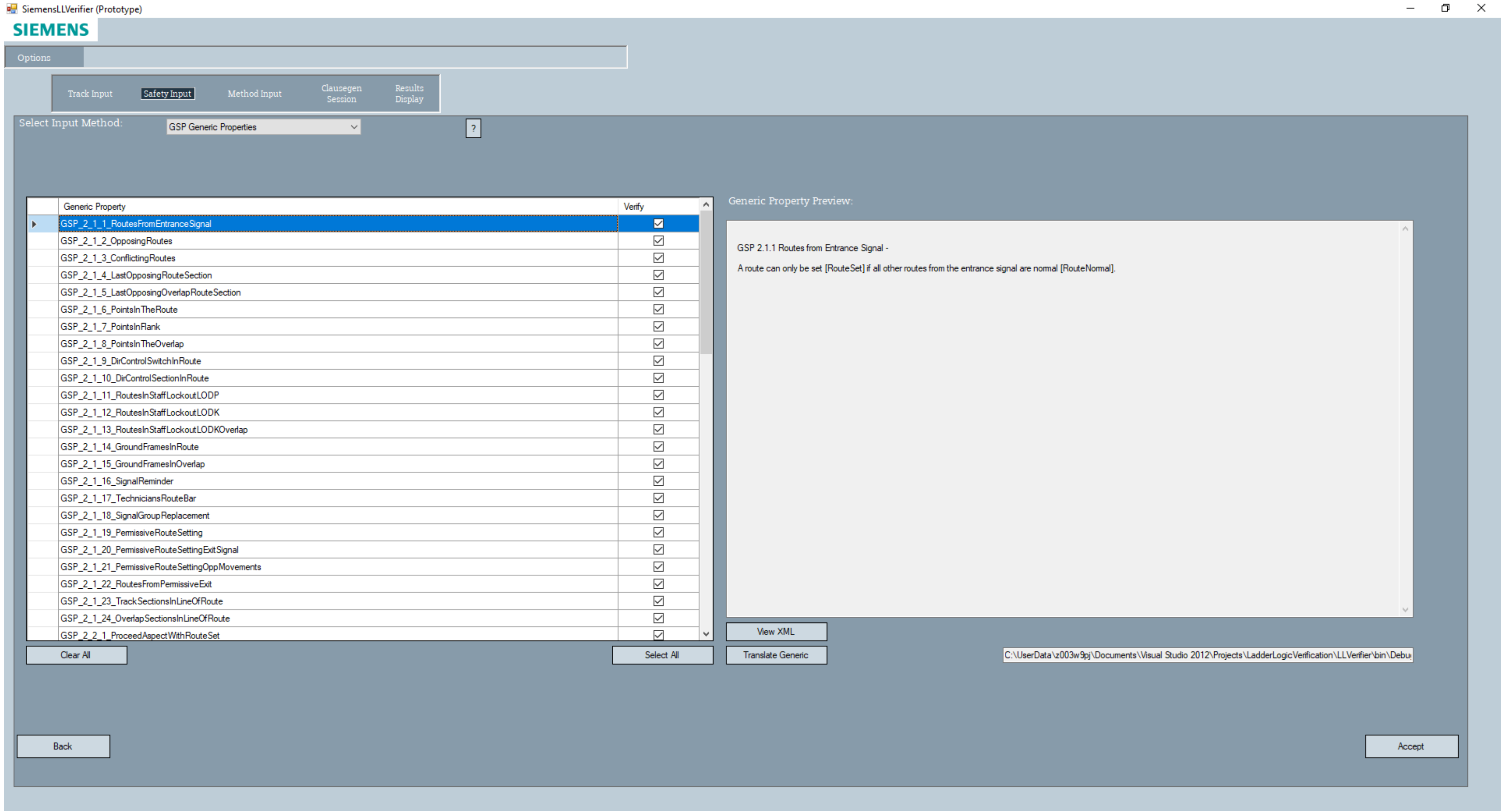
Formal Verification – The Journey from Theory towards Practice
Demonstration: Ladder logic and layout



Ladder File (.wt2): C:\UserData\z003w9pj\Documents\Visual Studio 2012\Projects\LadderLogicVerification\LLVerifier\bin\Debug\946.wt2

Track Layout File (.xml): C:\UserData\z003w9pj\Documents\Visual Studio 2012\Projects\LadderLogicVerification\LLVerifier\bin\Debug\Mostyn10.xml

Formal Verification – The Journey from Theory towards Practice
Demonstration: Selection of safety rules



Select Input Method: GSP Generic Properties

	Generic Property	Verify
▶	GSP_2_1_1_RoutesFromEntranceSignal	☑
	GSP_2_1_2_OpposingRoutes	☑
	GSP_2_1_3_ConflictingRoutes	☑
	GSP_2_1_4_LastOpposingRouteSection	☑
	GSP_2_1_5_LastOpposingOverlapRouteSection	☑
	GSP_2_1_6_PointsInTheRoute	☑
	GSP_2_1_7_PointsInFlank	☑
	GSP_2_1_8_PointsInTheOverlap	☑
	GSP_2_1_9_DirControlSwitchInRoute	☑
	GSP_2_1_10_DirControlSectionInRoute	☑
	GSP_2_1_11_RoutesInStaffLockoutLODP	☑
	GSP_2_1_12_RoutesInStaffLockoutLODK	☑
	GSP_2_1_13_RoutesInStaffLockoutLODKOverlap	☑
	GSP_2_1_14_GroundFramesInRoute	☑
	GSP_2_1_15_GroundFramesInOverlap	☑
	GSP_2_1_16_SignalReminder	☑
	GSP_2_1_17_TechniciansRouteBar	☑
	GSP_2_1_18_SignalGroupReplacement	☑
	GSP_2_1_19_PermissiveRouteSetting	☑
	GSP_2_1_20_PermissiveRouteSettingExitSignal	☑
	GSP_2_1_21_PermissiveRouteSettingOppMovements	☑
	GSP_2_1_22_RoutesFromPermissiveExit	☑
	GSP_2_1_23_TrackSectionsInLineOfRoute	☑
	GSP_2_1_24_OverlapSectionsInLineOfRoute	☑
	GSP_2_2_1_ProceedAspectWithRouteSet	☑

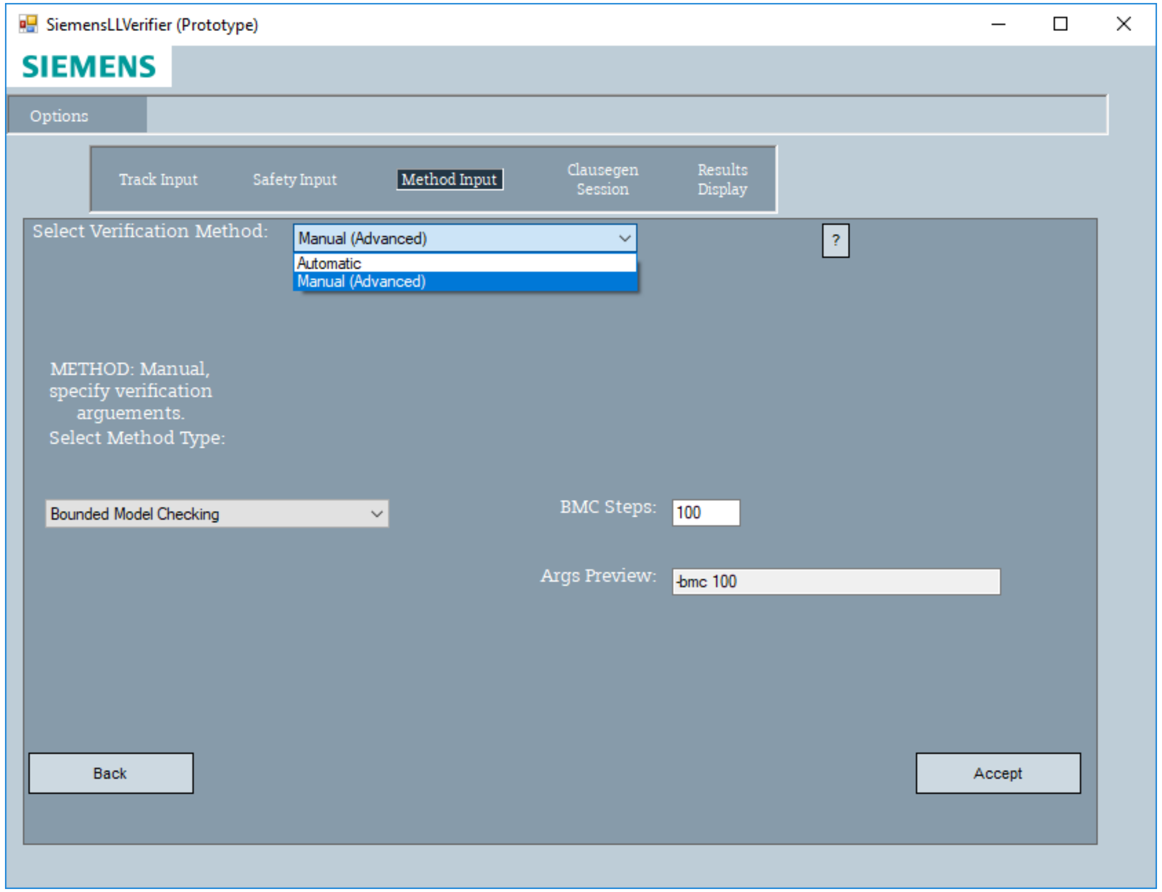
Generic Property Preview:

GSP 2.1.1 Routes from Entrance Signal -

A route can only be set [RouteSet] if all other routes from the entrance signal are normal [RouteNormal].

C:\UserData\z003w9pj\Documents\Visual Studio 2012\Projects\LadderLogicVerification\LLVerifier\bin\Debu

Formal Verification – The Journey from Theory towards Practice
Demonstration: Selection of verification method



Select Verification Method: Manual (Advanced)

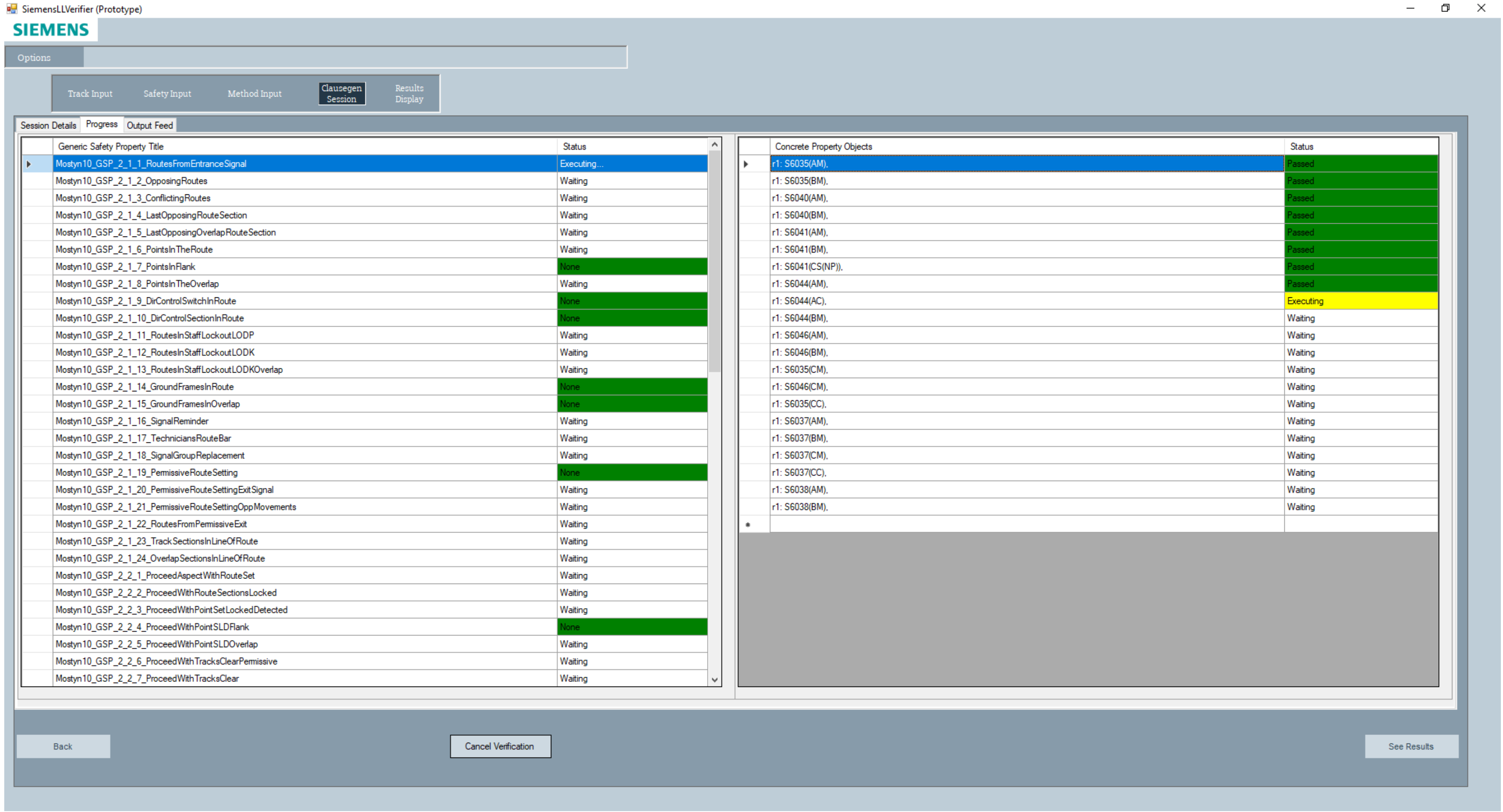
METHOD: Manual, specify verification arguements.

Select Method Type: Bounded Model Checking

BMC Steps: 100

Args Preview: -bmc 100

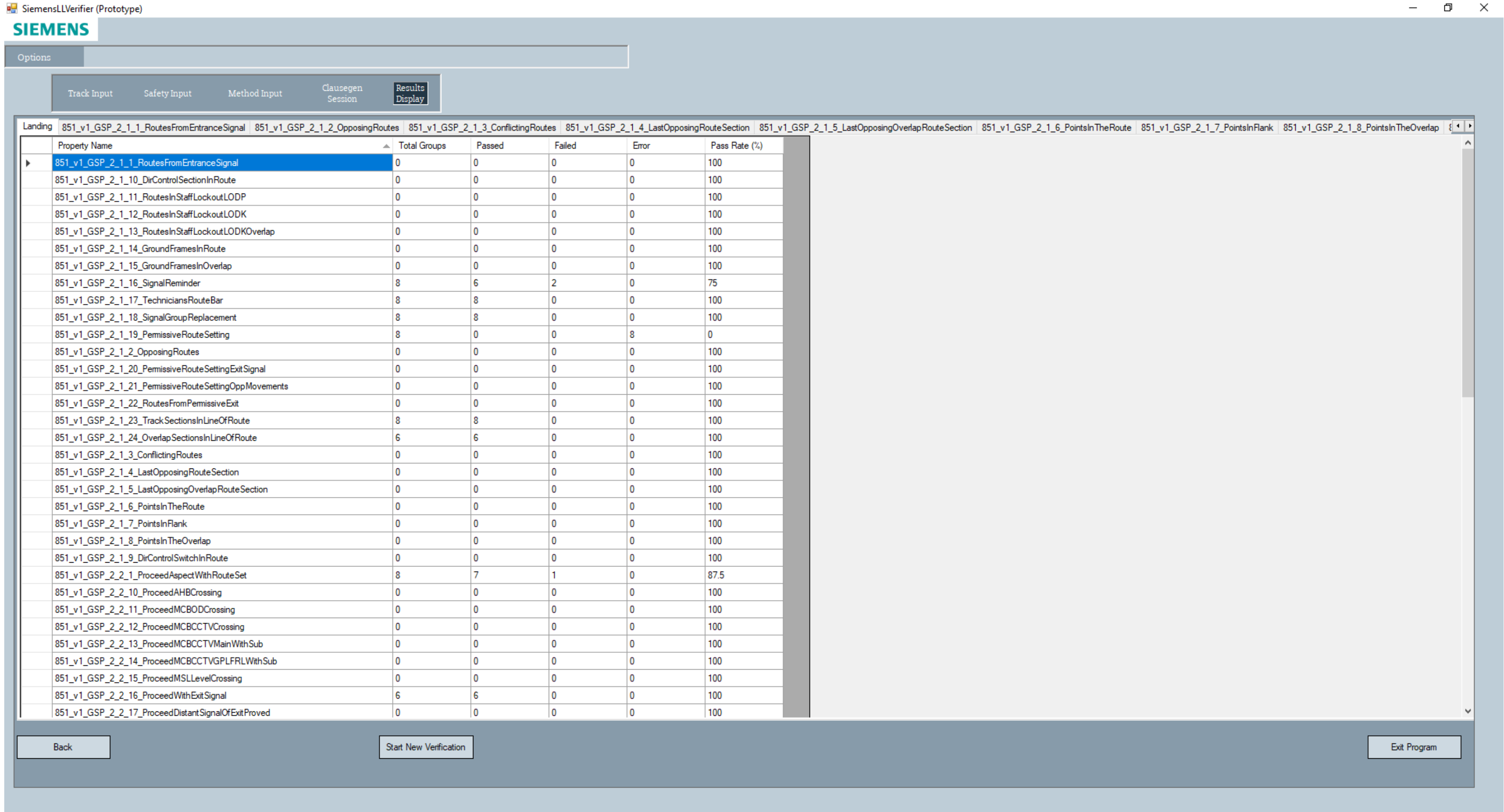
Formal Verification – The Journey from Theory towards Practice
Demonstration: Verification progress



	Generic Safety Property Title	Status
▶	Mostyn10_GSP_2_1_1_RoutesFromEntranceSignal	Executing...
	Mostyn10_GSP_2_1_2_OpposingRoutes	Waiting
	Mostyn10_GSP_2_1_3_ConflictingRoutes	Waiting
	Mostyn10_GSP_2_1_4_LastOpposingRouteSection	Waiting
	Mostyn10_GSP_2_1_5_LastOpposingOverlapRouteSection	Waiting
	Mostyn10_GSP_2_1_6_PointsInTheRoute	Waiting
	Mostyn10_GSP_2_1_7_PointsInFlank	None
	Mostyn10_GSP_2_1_8_PointsInTheOverlap	Waiting
	Mostyn10_GSP_2_1_9_DirControlSwitchInRoute	None
	Mostyn10_GSP_2_1_10_DirControlSectionInRoute	None
	Mostyn10_GSP_2_1_11_RoutesInStaffLockoutLODP	Waiting
	Mostyn10_GSP_2_1_12_RoutesInStaffLockoutLODK	Waiting
	Mostyn10_GSP_2_1_13_RoutesInStaffLockoutLODKOverlap	Waiting
	Mostyn10_GSP_2_1_14_GroundFramesInRoute	None
	Mostyn10_GSP_2_1_15_GroundFramesInOverlap	None
	Mostyn10_GSP_2_1_16_SignalReminder	Waiting
	Mostyn10_GSP_2_1_17_TechniciansRouteBar	Waiting
	Mostyn10_GSP_2_1_18_SignalGroupReplacement	Waiting
	Mostyn10_GSP_2_1_19_PermissiveRouteSetting	None
	Mostyn10_GSP_2_1_20_PermissiveRouteSettingExitSignal	Waiting
	Mostyn10_GSP_2_1_21_PermissiveRouteSettingOppMovements	Waiting
	Mostyn10_GSP_2_1_22_RoutesFromPermissiveExit	Waiting
	Mostyn10_GSP_2_1_23_TrackSectionsInLineOfRoute	Waiting
	Mostyn10_GSP_2_1_24_OverlapSectionsInLineOfRoute	Waiting
	Mostyn10_GSP_2_2_1_ProceedAspectWithRouteSet	Waiting
	Mostyn10_GSP_2_2_2_ProceedWithRouteSectionsLocked	Waiting
	Mostyn10_GSP_2_2_3_ProceedWithPointSetLockedDetected	Waiting
	Mostyn10_GSP_2_2_4_ProceedWithPointSLDFlank	None
	Mostyn10_GSP_2_2_5_ProceedWithPointSLDOverlap	Waiting
	Mostyn10_GSP_2_2_6_ProceedWithTracksClearPermissive	Waiting
	Mostyn10_GSP_2_2_7_ProceedWithTracksClear	Waiting

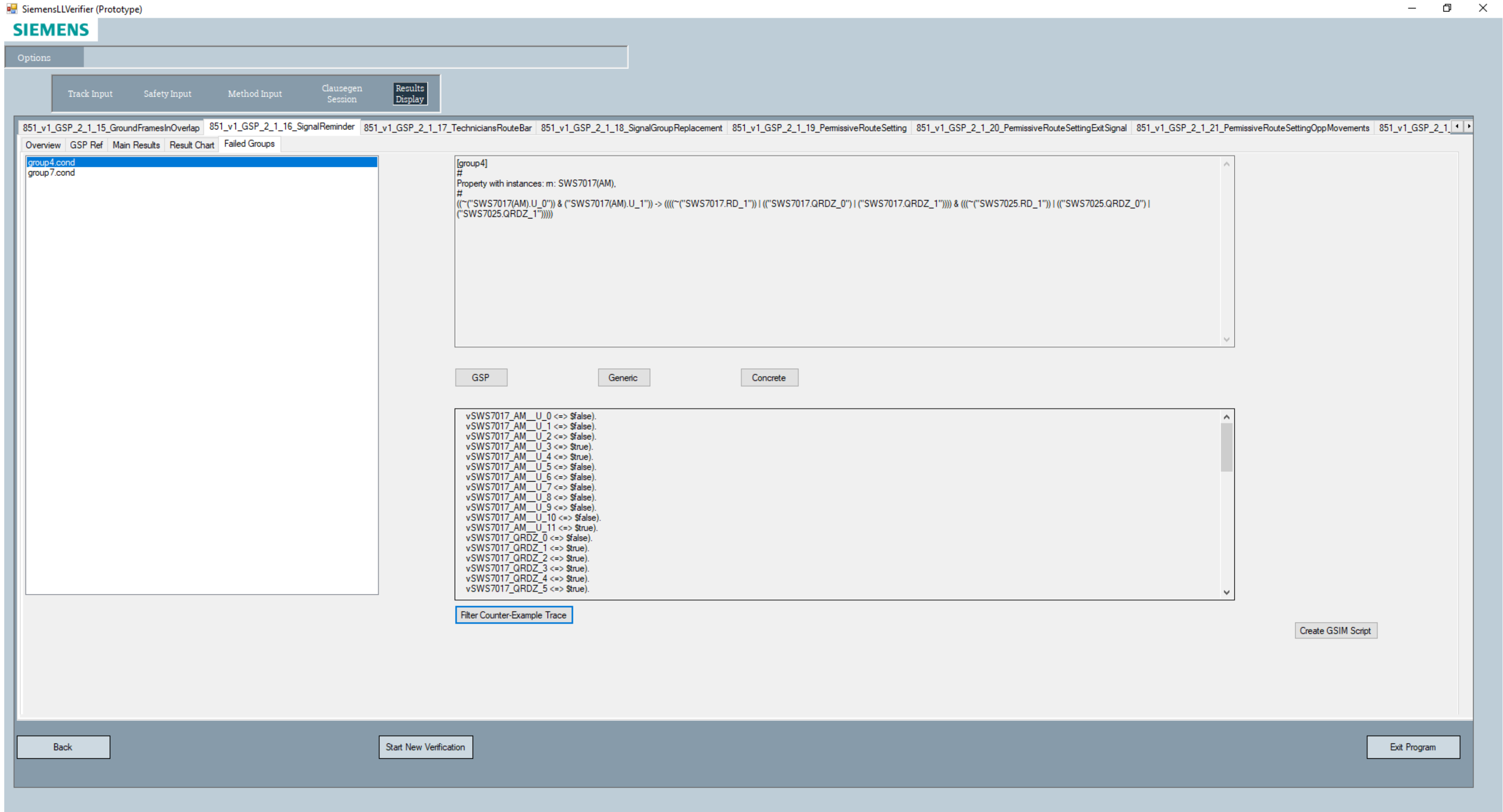
	Concrete Property Objects	Status
▶	r1: S6035(AM),	Passed
	r1: S6035(BM),	Passed
	r1: S6040(AM),	Passed
	r1: S6040(BM),	Passed
	r1: S6041(AM),	Passed
	r1: S6041(BM),	Passed
	r1: S6041(CS(NP)),	Passed
	r1: S6044(AM),	Passed
	r1: S6044(AC),	Executing
	r1: S6044(BM),	Waiting
	r1: S6046(AM),	Waiting
	r1: S6046(BM),	Waiting
	r1: S6035(CM),	Waiting
	r1: S6046(CM),	Waiting
	r1: S6035(CC),	Waiting
	r1: S6037(AM),	Waiting
	r1: S6037(BM),	Waiting
	r1: S6037(CM),	Waiting
	r1: S6037(CC),	Waiting
	r1: S6038(AM),	Waiting
	r1: S6038(BM),	Waiting
*		

Formal Verification – The Journey from Theory towards Practice
Demonstration: Results (1)


SiemensLLVerifier (Prototype)
SIEMENS
Options
Track Input
Safety Input
Method Input
Clausegen Session
Results Display
Landing
851_v1_GSP_2_1_1_RoutesFromEntranceSignal
851_v1_GSP_2_1_2_OpposingRoutes
851_v1_GSP_2_1_3_ConflictingRoutes
851_v1_GSP_2_1_4_LastOpposingRouteSection
851_v1_GSP_2_1_5_LastOpposingOverlapRouteSection
851_v1_GSP_2_1_6_PointsInTheRoute
851_v1_GSP_2_1_7_PointsInFlank
851_v1_GSP_2_1_8_PointsInTheOverlap
Back
Start New Verification
Exit Program

Property Name	Total Groups	Passed	Failed	Error	Pass Rate (%)
851_v1_GSP_2_1_1_RoutesFromEntranceSignal	0	0	0	0	100
851_v1_GSP_2_1_10_DirControlSectionInRoute	0	0	0	0	100
851_v1_GSP_2_1_11_RoutesInStaffLockoutLODP	0	0	0	0	100
851_v1_GSP_2_1_12_RoutesInStaffLockoutLODK	0	0	0	0	100
851_v1_GSP_2_1_13_RoutesInStaffLockoutLODKOverlap	0	0	0	0	100
851_v1_GSP_2_1_14_GroundFramesInRoute	0	0	0	0	100
851_v1_GSP_2_1_15_GroundFramesInOverlap	0	0	0	0	100
851_v1_GSP_2_1_16_SignalReminder	8	6	2	0	75
851_v1_GSP_2_1_17_TechniciansRouteBar	8	8	0	0	100
851_v1_GSP_2_1_18_SignalGroupReplacement	8	8	0	0	100
851_v1_GSP_2_1_19_PermissiveRouteSetting	8	0	0	8	0
851_v1_GSP_2_1_2_OpposingRoutes	0	0	0	0	100
851_v1_GSP_2_1_20_PermissiveRouteSettingExitSignal	0	0	0	0	100
851_v1_GSP_2_1_21_PermissiveRouteSettingOppMovements	0	0	0	0	100
851_v1_GSP_2_1_22_RoutesFromPermissiveExit	0	0	0	0	100
851_v1_GSP_2_1_23_TrackSectionsInLineOfRoute	8	8	0	0	100
851_v1_GSP_2_1_24_OverlapSectionsInLineOfRoute	6	6	0	0	100
851_v1_GSP_2_1_3_ConflictingRoutes	0	0	0	0	100
851_v1_GSP_2_1_4_LastOpposingRouteSection	0	0	0	0	100
851_v1_GSP_2_1_5_LastOpposingOverlapRouteSection	0	0	0	0	100
851_v1_GSP_2_1_6_PointsInTheRoute	0	0	0	0	100
851_v1_GSP_2_1_7_PointsInFlank	0	0	0	0	100
851_v1_GSP_2_1_8_PointsInTheOverlap	0	0	0	0	100
851_v1_GSP_2_1_9_DirControlSwitchInRoute	0	0	0	0	100
851_v1_GSP_2_2_1_ProceedAspectWithRouteSet	8	7	1	0	87.5
851_v1_GSP_2_2_10_ProceedAHBCrossing	0	0	0	0	100
851_v1_GSP_2_2_11_ProceedMCBODCrossing	0	0	0	0	100
851_v1_GSP_2_2_12_ProceedMCBCCTVCrossing	0	0	0	0	100
851_v1_GSP_2_2_13_ProceedMCBCCTVMainWithSub	0	0	0	0	100
851_v1_GSP_2_2_14_ProceedMCBCCTVGPLFRLWithSub	0	0	0	0	100
851_v1_GSP_2_2_15_ProceedMSLLevelCrossing	0	0	0	0	100
851_v1_GSP_2_2_16_ProceedWithExitSignal	6	6	0	0	100
851_v1_GSP_2_2_17_ProceedDistantSignalOfExitProved	0	0	0	0	100

Formal Verification – The Journey from Theory towards Practice
Demonstration: Results (2)


SiemensLLVerifier (Prototype)
SIEMENS
Options
Track Input
Safety Input
Method Input
Clausegen Session
Results Display
851_v1_GSP_2_1_15_GroundFramesInOverlap
851_v1_GSP_2_1_16_SignalReminder
851_v1_GSP_2_1_17_TechniciansRouteBar
851_v1_GSP_2_1_18_SignalGroupReplacement
851_v1_GSP_2_1_19_PermissiveRouteSetting
851_v1_GSP_2_1_20_PermissiveRouteSettingExitSignal
851_v1_GSP_2_1_21_PermissiveRouteSettingOppMovements
851_v1_GSP_2_1_
Overview
GSP Ref
Main Results
Result Chart
Failed Groups
group4.cond
group7.cond
GSP
Generic
Concrete
Filter Counter-Example Trace
Create GSIM Script
Back
Start New Verification
Exit Program

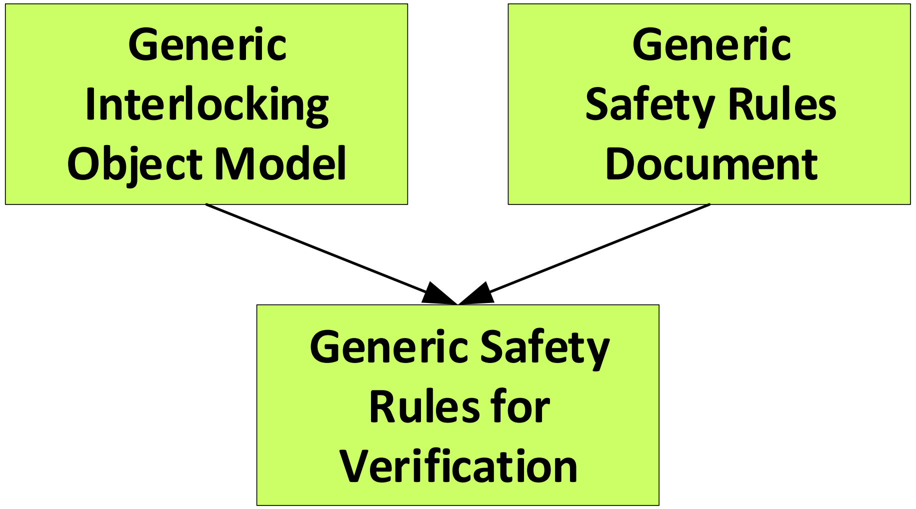
```
[group4]
#
Property with instances: m: SWS7017(AM),
#
((~("SWS7017(AM).U_0")) & ("SWS7017(AM).U_1")) -> ((((~("SWS7017.RD_1")) | (("SWS7017.QRDZ_0") | ("SWS7017.QRDZ_1")))) & (((~("SWS7025.RD_1")) | (("SWS7025.QRDZ_0") |
("SWS7025.QRDZ_1")))))
```

```
vSWS7017_AM__U_0 <=> $false).
vSWS7017_AM__U_1 <=> $false).
vSWS7017_AM__U_2 <=> $false).
vSWS7017_AM__U_3 <=> $true).
vSWS7017_AM__U_4 <=> $true).
vSWS7017_AM__U_5 <=> $false).
vSWS7017_AM__U_6 <=> $false).
vSWS7017_AM__U_7 <=> $false).
vSWS7017_AM__U_8 <=> $false).
vSWS7017_AM__U_9 <=> $false).
vSWS7017_AM__U_10 <=> $false).
vSWS7017_AM__U_11 <=> $true).
vSWS7017_QRDZ_0 <=> $false).
vSWS7017_QRDZ_1 <=> $true).
vSWS7017_QRDZ_2 <=> $true).
vSWS7017_QRDZ_3 <=> $true).
vSWS7017_QRDZ_4 <=> $true).
vSWS7017_QRDZ_5 <=> $true).
```

Formal Verification – The Journey from Theory towards Practice
No such thing…

…as a free lunch!

- To set up any automation requires some effort!
- This applies to automated data generation, automated testing, formal verification
- We have done this first in document form, second in machine-readable form:


Generic Interlocking Object Model

Generic Safety Rules Document

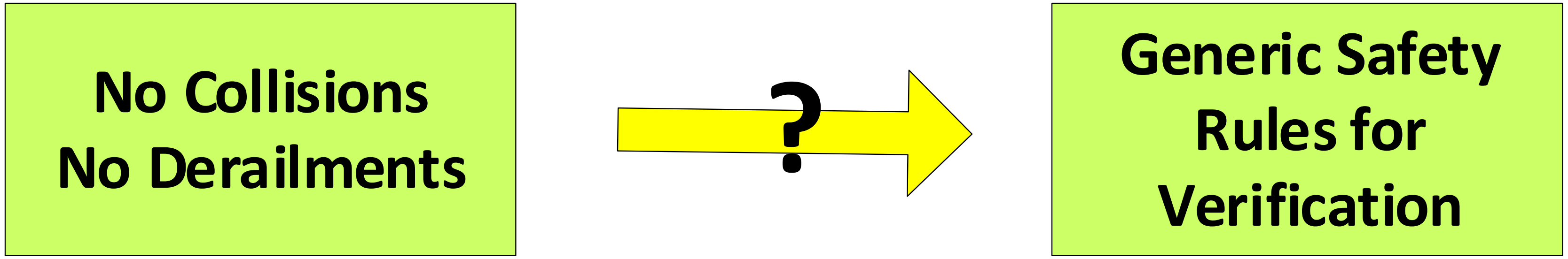
Generic Safety Rules for Verification

This has to be repeated for each set of signalling rules!
Some standards are available to help

Non-trivial effort!

Formal Verification – The Journey from Theory towards Practice Completeness?

There is a completeness problem.
We have no way to demonstrate that the set of rules is complete.



No Collisions
No Derailments

?

Generic Safety Rules for Verification

Only the review by signaling experts
Review based on standards which have evolved over a long time

Thankyou!

Simon Chadwick, Tom Werner
Siemens Mobility, Chippenham, UK

www.siemens.com