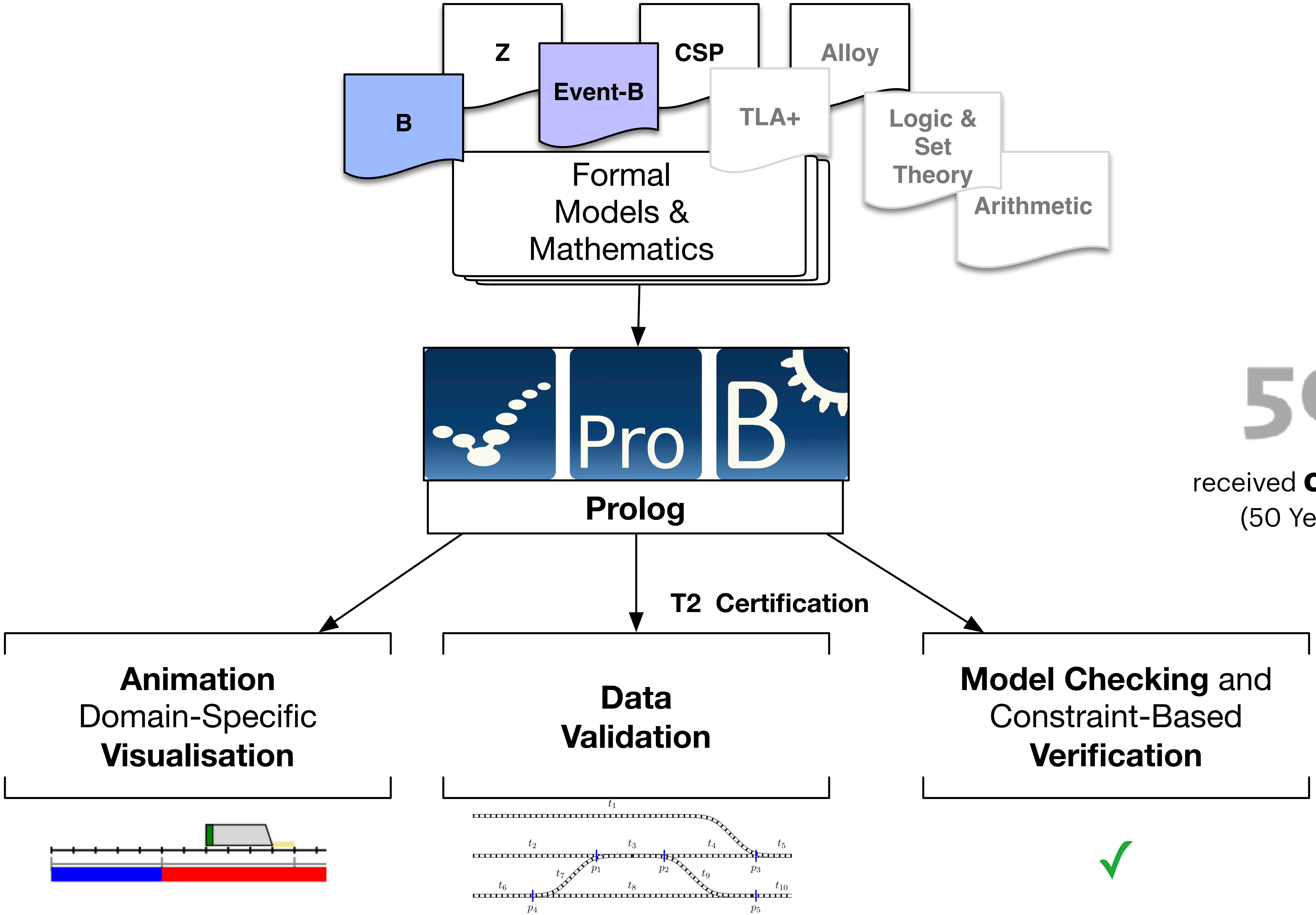
Real Animation of TLA+ Models

ProB for TLA+ and TLC for B

Michael Leuschel & Jan Gruteser, University of Düsseldorf, September 2024

hhu.

A Validation Tool for Formal Models



received **Colmerauer Prize**
(50 Years of Prolog)

Data Validation

Towards the limits

```
p_over := bool ( # ( over_track ) . ( ( over_track : seq ( t_block * t_direction ) & over_track /= {} & first ( over_track ) = p_X2MBlock |> p_X2MDir & ! ii . ( ii : 1 .. size ( over_track ) - 1 => (
over_track) ( ii ) : dom ( sidb_nextBlock ) ) & ! ii . ( ii : 1 .. size ( over_track ) => sidb_nextBlock ( ( over_track ) ( ii ) ) = ( over_track ) ( ii + 1 ) ) ) &( # ( over_res ) . ( ( over_res :
sidb_restrictionApplicable & ( # ii . ( ii : dom ( over_track ) & ( ( prj2 ( t_block , t_direction ) ( over_track ( ii ) ) ) = c_up => over_res : ran ( sgd_blockUpRestrictionSeq ( ( prj1 ( t_block , t_direction
) ( over_track ( ii ) ) ) ) ) ) & ( ( prj2 ( t_block , t_direction ) ( over_track ( ii ) ) ) = c_down => over_res : ran( sgd_blockDownRestrictionSeq ( ( prj1 ( t_block , t_direction ) ( over_track ( ii ) ) ) ) ) ) & (
ii = 1 => not ( over_res <= p_X2MRes ) ) & p_X2MSSWorst + p_X2MDSS + ( SIGMA( jj ) . ( jj : 1 .. ii | SIGMA ( pre_res ) . ( pre_res : t_restriction & ( ( prj2 ( t_block , t_direction ) ( over_track ( jj ) ) )
= c_up => pre_res : ran ( sgd_blockUpRestrictionSeq ( ( prj1 ( t_block, t_direction ) ( over_track ( jj ) ) ) ) ) ) & ( ( prj2 ( t_block , t_direction ) ( over_track ( jj ) ) ) = c_down => pre_res : ran (
sgd_blockDownRestrictionSeq ( ( prj1 ( t_block , t_direction ) ( over_track ( jj ) ) ) ) ) ) & ( jj = 1 => not ( pre_res <= p_X2MRes ) ) & ( jj = ii => not ( pre_res >= over_res ) ) |
sgd_restrictionDeltaSqSpeed ( pre_res ) ) ) ) > sgd_restrictionSquareSpeed ( over_res ) & ( over_res : sgd_restrictionFront => p_X2MResDist + ( ( SIGMA ( ti ) . ( ti : 1 .. ii | sgd_blockLength ( ( prj1
( t_block , t_direction )( ( over_track ) ( ti ) ) ) ) ) ) ({ c_down |>sgd_blockLength ( p_X2MBlock ) sgd_restrictionAbs ( p_X2MRes ) , c_up |>sgd_restrictionAbs ( p_X2MRes ) } ( p_X2MDir ) ) ({
c_down |>sgd_restrictionAbs ( over_res ) , c_up |>sgd_blockLength ( ( prj1 ( t_block , t_direction ) ( ( over_track ) ( ii ) ) ) ) sgd_restrictionAbs ( over_res ) } ( ( prj2 ( t_block ,t_direction ) ( (
over_track ) ( ii ) ) ) ) ) ) + sgd_restrictionLength ( over_res ) > loc_locationUncertainty + c_trainLength ) ) ) ) ) or ( # ( eoa_res , res_after_eoa , ii ) . ( eoa_res : t_restriction & res_after_eoa :
t_restriction & ii : dom ( over_track ) & p_EOABlock = ( prj1 ( t_block , t_direction )( over_track ( ii ) ) ) & ( ii = 1 => p_X2MRes <= eoa_res ) & ( ( prj2 ( t_block , t_direction ) ( over_track ( ii ) ) ) =
c_up => eoa_res : ran ( sgd_blockUpRestrictionSeq ( p_EOABlock ) ) & res_after_eoa : ran ( sgd_blockUpRestrictionSeq ( p_EOABlock ) ) & sgd_restrictionAbs ( eoa_res ) <= p_EOAAbs &
p_EOAAbs < sgd_restrictionAbs ( res_after_eoa ) & ! ri . ( ri : ran ( sgd_blockUpRestrictionSeq ( p_EOABlock ) ) => ri <= eoa_res or res_after_eoa <= ri ) ) & ( ( prj2 ( t_block , t_direction ) (
over_track ( ii ) ) ) = c_down => eoa_res : ran ( sgd_blockDownRestrictionSeq ( p_EOABlock ) ) & res_after_eoa : ran ( sgd_blockDownRestrictionSeq ( p_EOABlock ) ) & sgd_restrictionAbs (
eoa_res ) >= p_EOAAbs & p_EOAAbs > sgd_restrictionAbs ( res_after_eoa ) & ! ri . ( ri : ran ( sgd_blockDownRestrictionSeq ( p_EOABlock ) ) => ri <= eoa_res or res_after_eoa <= ri ) ) &
p_X2MSSWorst + p_X2MDSS + ( SIGMA ( jj ) . ( jj : 1 .. ii | SIGMA ( pre_res ) . ( pre_res : t_restriction & ( ( prj2 ( t_block , t_direction ) ( over_track ( jj ) ) ) = c_up => pre_res : ran (
sgd_blockUpRestrictionSeq ( ( prj1 ( t_block , t_direction ) ( over_track ( jj ) ) ) ) ) ) & ( ( prj2 ( t_block , t_direction ) ( over_track ( jj ) ) ) = c_down => pre_res : ran( sgd_blockDownRestrictionSeq (
( prj1 ( t_block , t_direction ) ( over_track ( jj ) ) ) ) ) ) & ( jj = 1 => not ( pre_res <= p_X2MRes ) ) & ( jj = ii => pre_res <= eoa_res ) | sgd_restrictionDeltaSqSpeed ( pre_res ) ) ) ) ) ({ c_up |>(
sgd_restrictionAccel ( eoa_res ) * ( ( sgd_restrictionAbs ( res_after_eoa ) p_EOAAbs ) / 1024 ) ) / 2 , c_down |>( sgd_restrictionAccel ( eoa_res ) * ( ( p_EOAAbs sgd_restrictionAbs (
res_after_eoa ) ) / 1024 ) ) / 2 } ( ( prj2 ( t_block , t_direction ) ( over_track ( ii ) ) ) ) ) > 0 ) ) or ( # ( eoa_res , ii ) . ( eoa_res : t_restriction & ii : dom ( over_track ) & ( ii = 1 => not ( eoa_res <=
p_X2MRes ) ) & p_EOABlock = ( prj1 ( t_block , t_direction ) ( over_track ( ii ) ) ) & ( ( prj2 ( t_block , t_direction ) ( over_track ( ii ) ) ) = c_up => eoa_res : ran ( sgd_blockUpRestrictionSeq (
p_EOABlock ) ) & eoa_res = last( sgd_blockUpRestrictionSeq ( p_EOABlock ) ) & sgd_restrictionAbs ( eoa_res ) <= p_EOAAbs ) & ( ( prj2 ( t_block , t_direction ) ( over_track ( ii ) ) ) = c_down =>
eoa_res : ran( sgd_blockDownRestrictionSeq ( p_EOABlock ) ) & eoa_res = last ( sgd_blockDownRestrictionSeq ( p_EOABlock ) ) & sgd_restrictionAbs ( eoa_res ) >= p_EOAAbs ) & p_X2MSSWorst
+ p_X2MDSS + ( SIGMA ( jj ) . ( jj : 1 .. ii | SIGMA ( pre_res ) . ( pre_res : t_restriction & ( ( prj2 ( t_block , t_direction ) ( over_track ( jj ) ) ) = c_up => pre_res : ran( sgd_blockUpRestrictionSeq ( (
prj1 ( t_block , t_direction ) ( over_track ( jj ) ) ) ) ) ) & ( ( prj2 ( t_block , t_direction ) ( over_track ( jj ) ) ) = c_down => pre_res : ran( sgd_blockDownRestrictionSeq ( ( prj1 ( t_block , t_direction ) (
over_track ( jj ) ) ) ) ) ) & ( jj = 1 => not ( pre_res <= p_X2MRes ) ) & ( jj = ii => not ( pre_res >= eoa_res ) ) | sgd_restrictionDeltaSqSpeed ( pre_res ) ) ) ) ) + ( { c_up |> ( sgd_restrictionAccel (
eoa_res ) * ( ( p_EOAAbs sgd_restrictionAbs ( eoa_res ) ) / 1024 ) ) / 2 , c_down |> ( sgd_restrictionAccel ( eoa_res ) * ( ( sgd_restrictionAbs ( eoa_res ) p_EOAAbs ) / 1024 ) ) / 2 } ( ( prj2 ( t_block
, t_direction ) ( over_track ( ii ) ) ) ) ) > 0 ) )
```

CLEARSY | Seminar Dagstuhl | Integrated Rigorous Analysis in Cyber-Physical Systems Engineering | P. 18

models with up to **10 million** lines of B

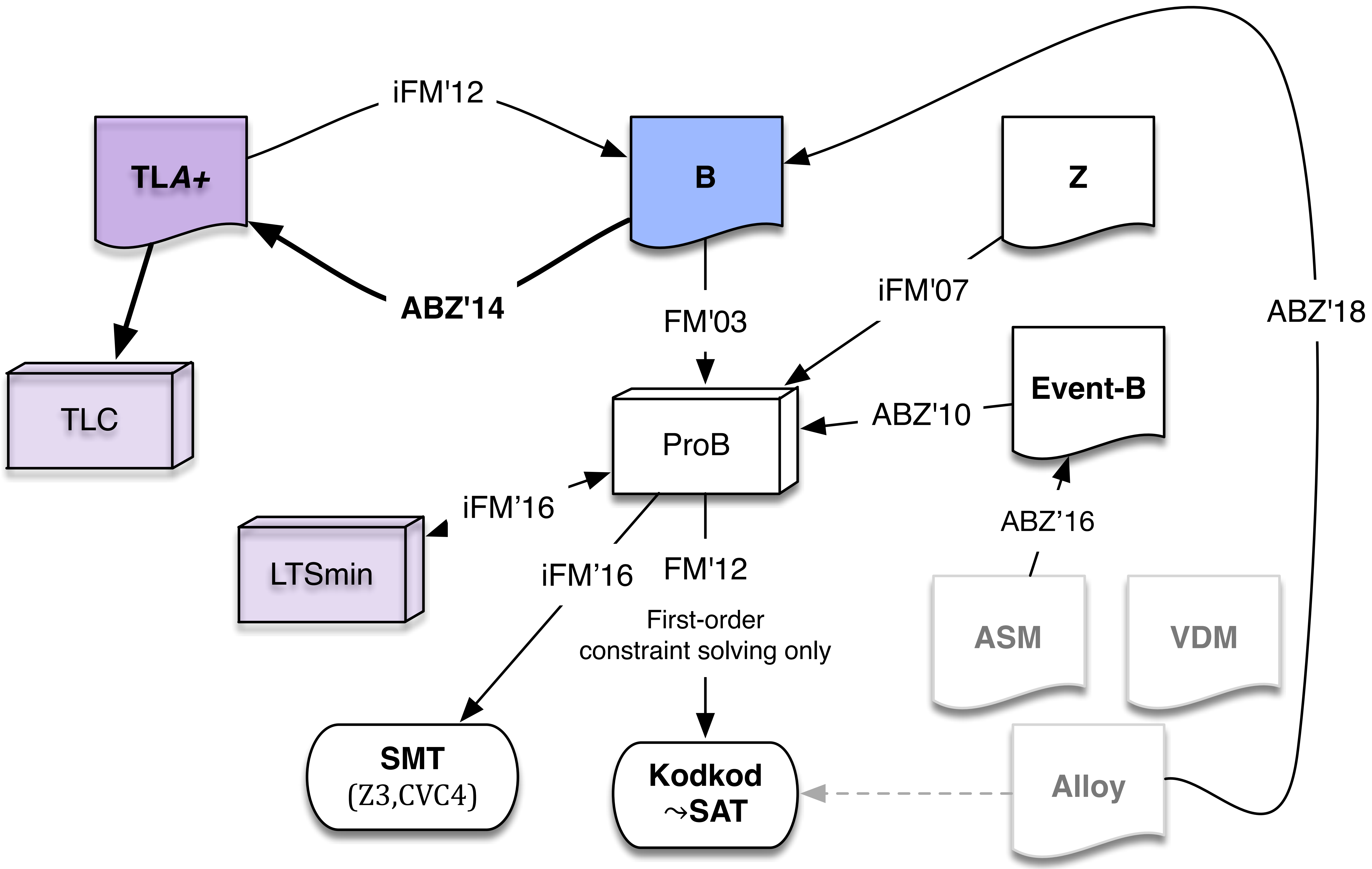
ProB

EN50128 Certification as T2 tool

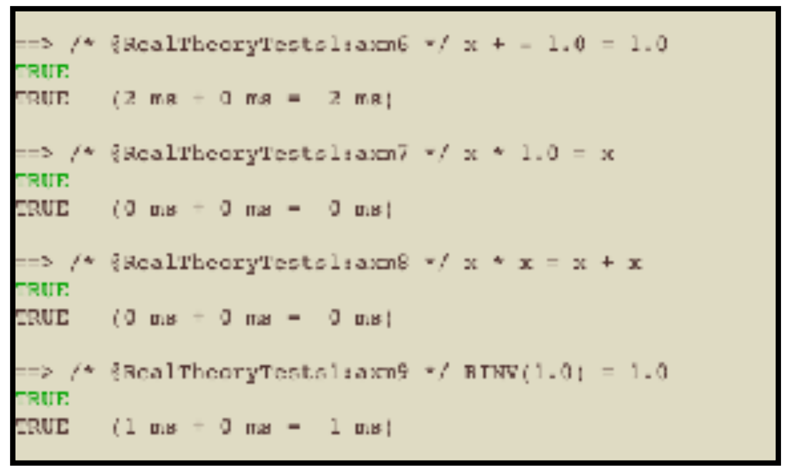
from Talk of Thierry Lecomte

Languages and Backends

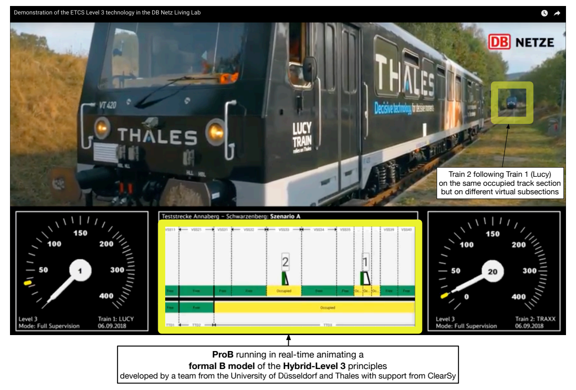
supported by ProB



ProB's User Interfaces



probcli
(Command-Line Interface)

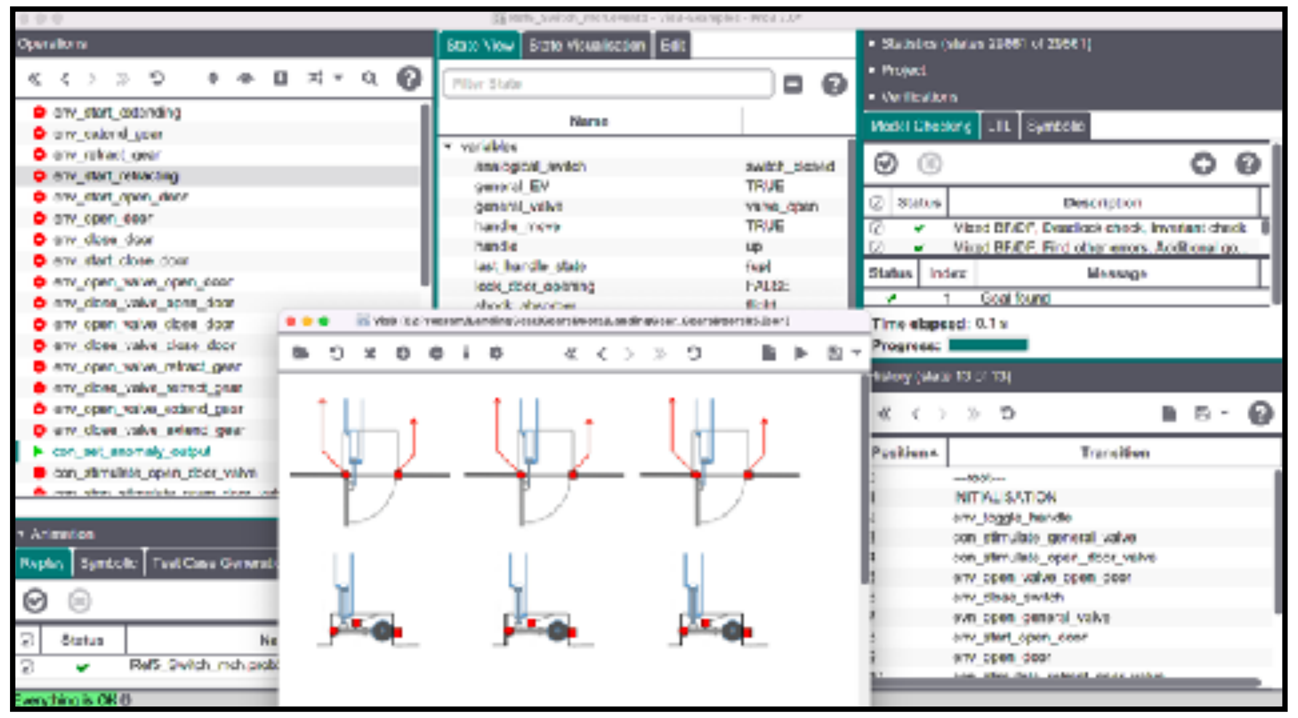


ProB embedded @ runtime

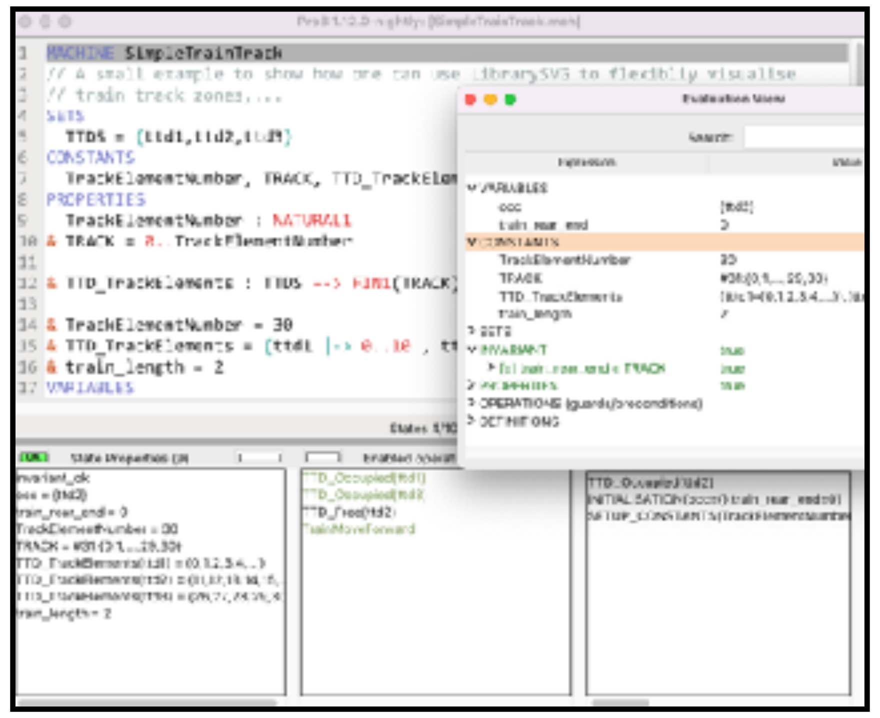
ProB Jupyter Kernel
(Notebook interface)

all share the same Prolog core

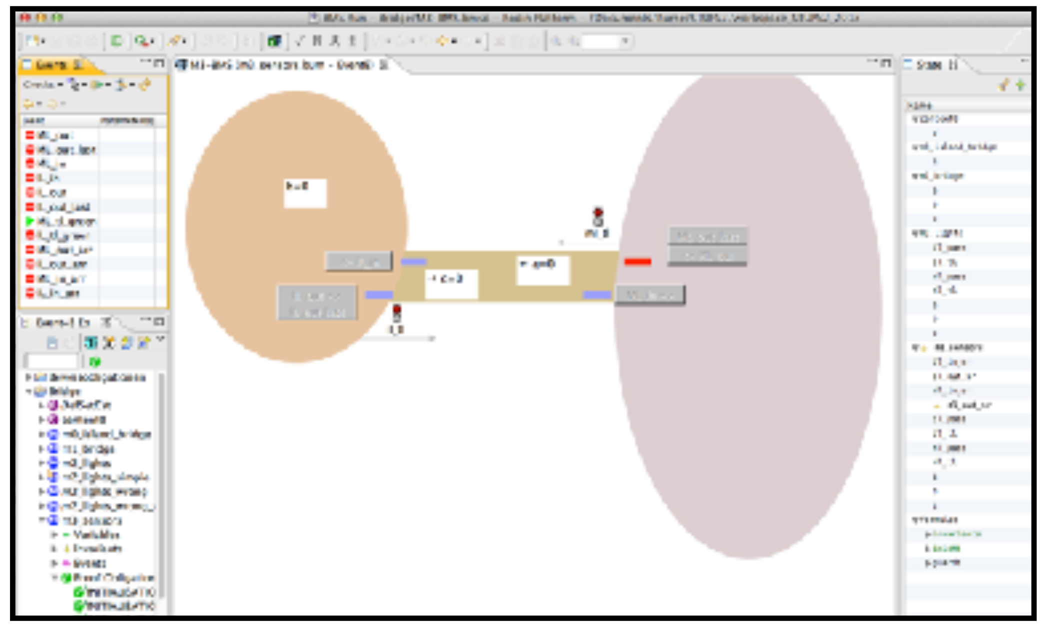
ProB2-UI



ProB Tcl/Tk



ProB Rodin (Eclipse) Plugin



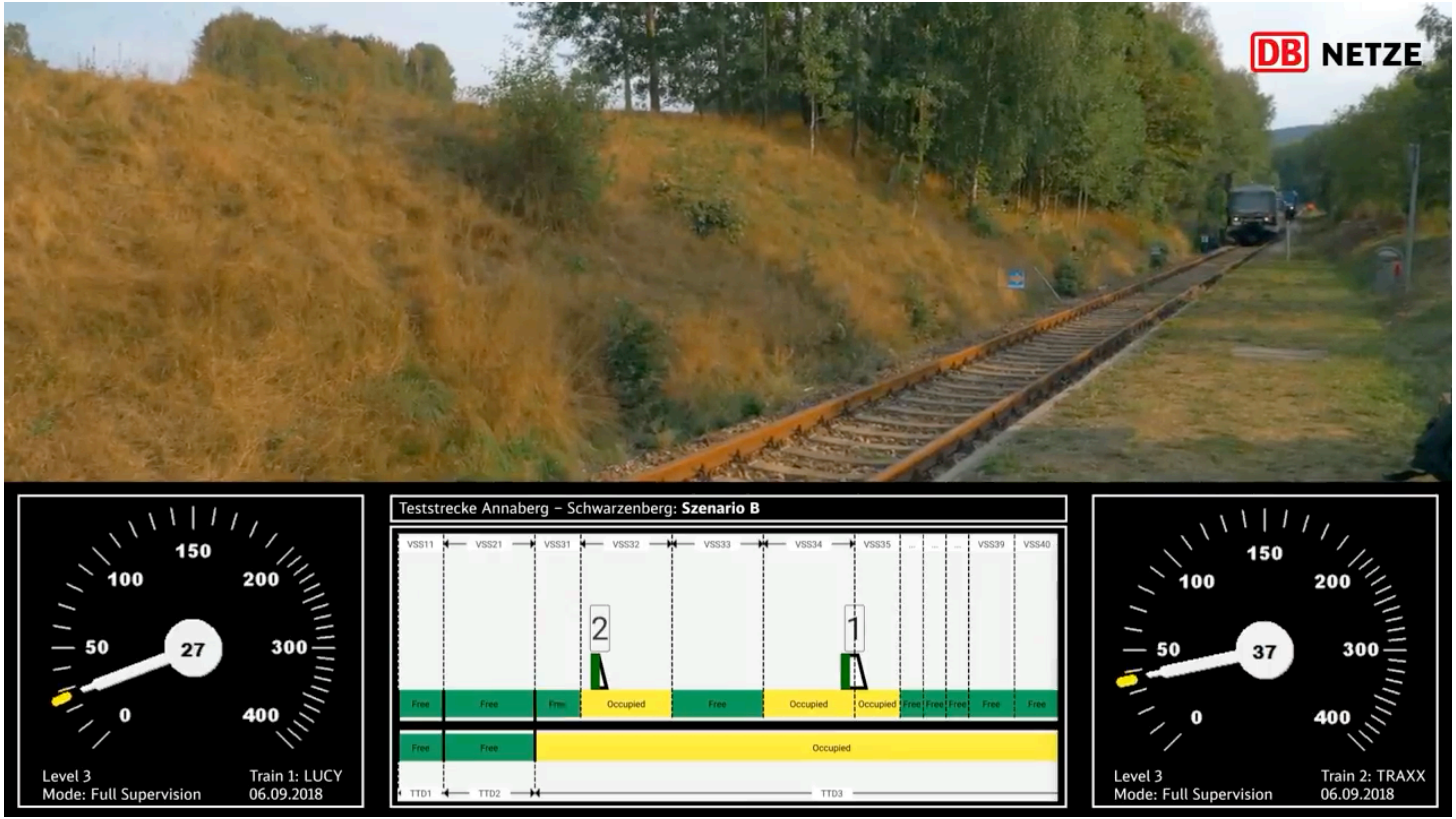
including Disprover

ProB's Solver in Action

ProB's Solver in Action

```
Restored session: Mon  7 Nov 2022 19:30:29 CET
```

ProB running in SICStus **Prolog** in real-time
executing a formal B model of the Hybrid-Level 3 principles



Train 2 following Train 1 (Lucy) on the same occupied track section, but on different virtual subsections

Source: https://www.youtube.com/watch?v=FjKnugbmrP4

B Logical Foundations

- Typed first-order **predicate logic** with equality
  - Well-Definedness Conditions to stay in two-valued logic
- **Arithmetic** over mathematical integers and implementable integers
- **Set theory**
  - Sets, Relations, Functions, Sequences
  - including higher-order functions
- B is simpler than its predecessor Z
- and provides structuring and refinement for proving and code generation

related state-based formal methods:
Z, TLA+, Alloy, VDM, ASM

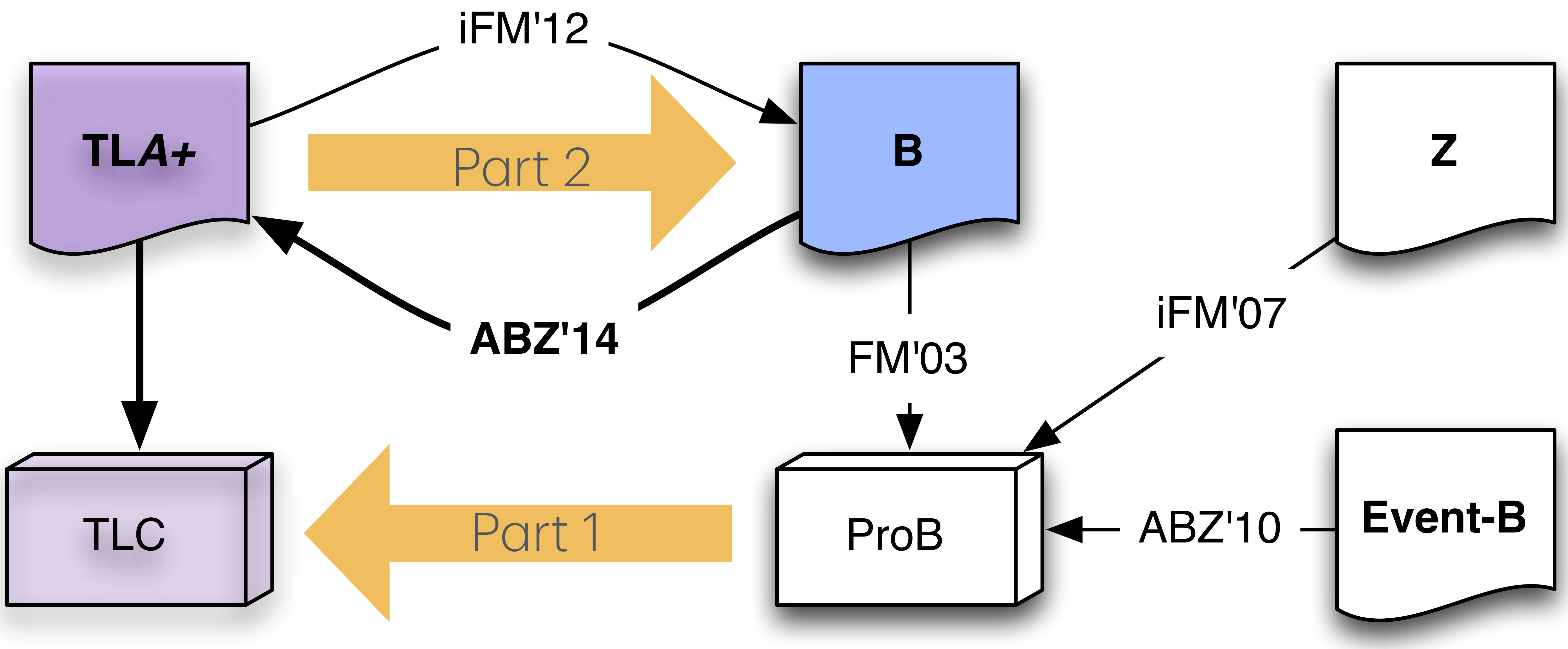
```
p∈dom(a)⤖dom(a) ∧ ∀i·(i∈1‥(size(a)−1) ⇒ p(a(i)) < p(a(i+1)))
```

TLA+ vs B

	TLA$^+$	B-Method
Invented by	Leslie Lamport	J.R. Abrial
State-based	√	√
Typed	X	√
Set theory	√	√
Predicate logic	√	√
Arithmetic	√	√
Temporal formulas	√	X
State transition	Before-after predicate	Generalised substitutions
Model checker	TLC	ProB
Prover support	TLAPS	AtelierB

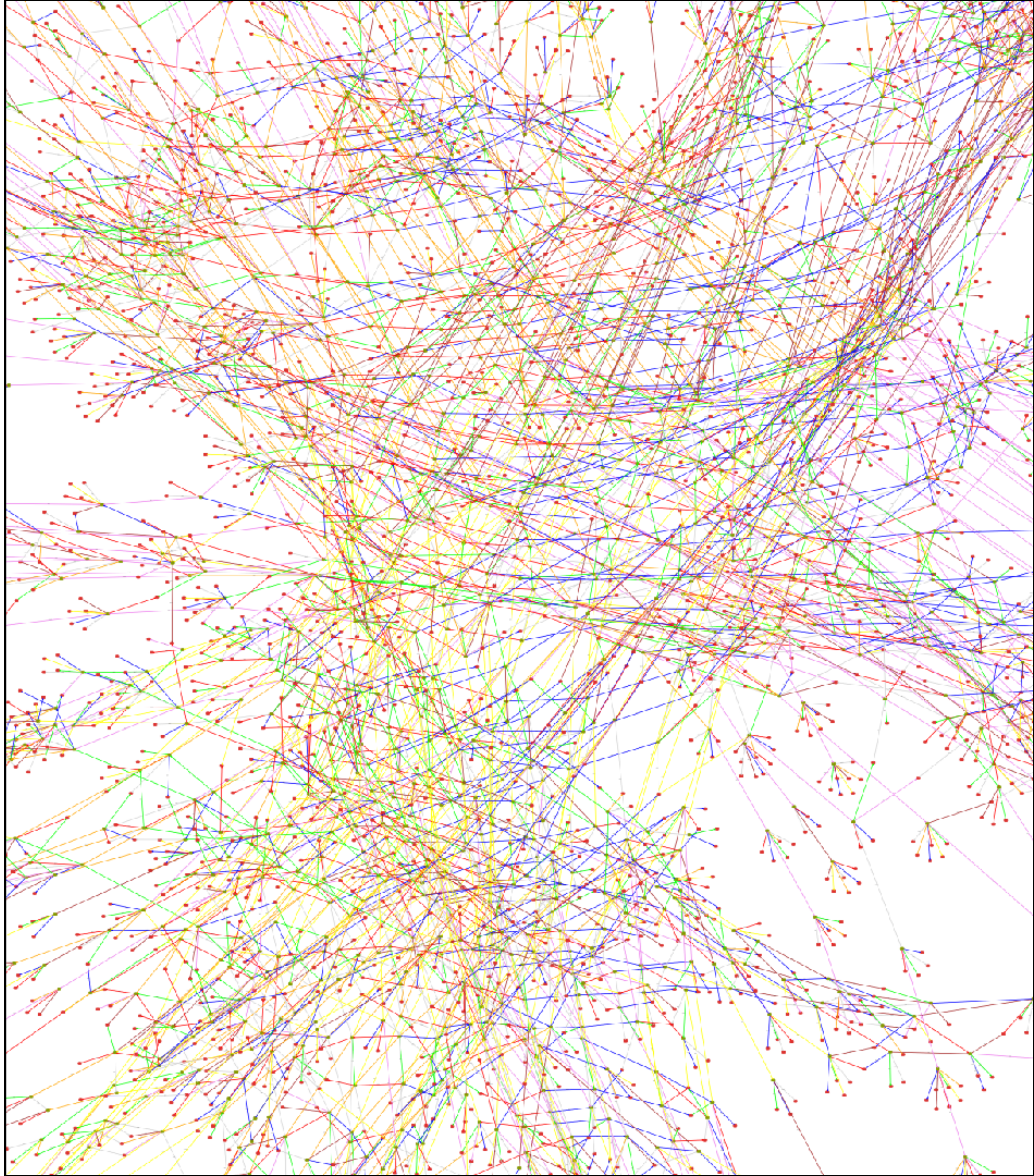
This talk

ProB and TLA+



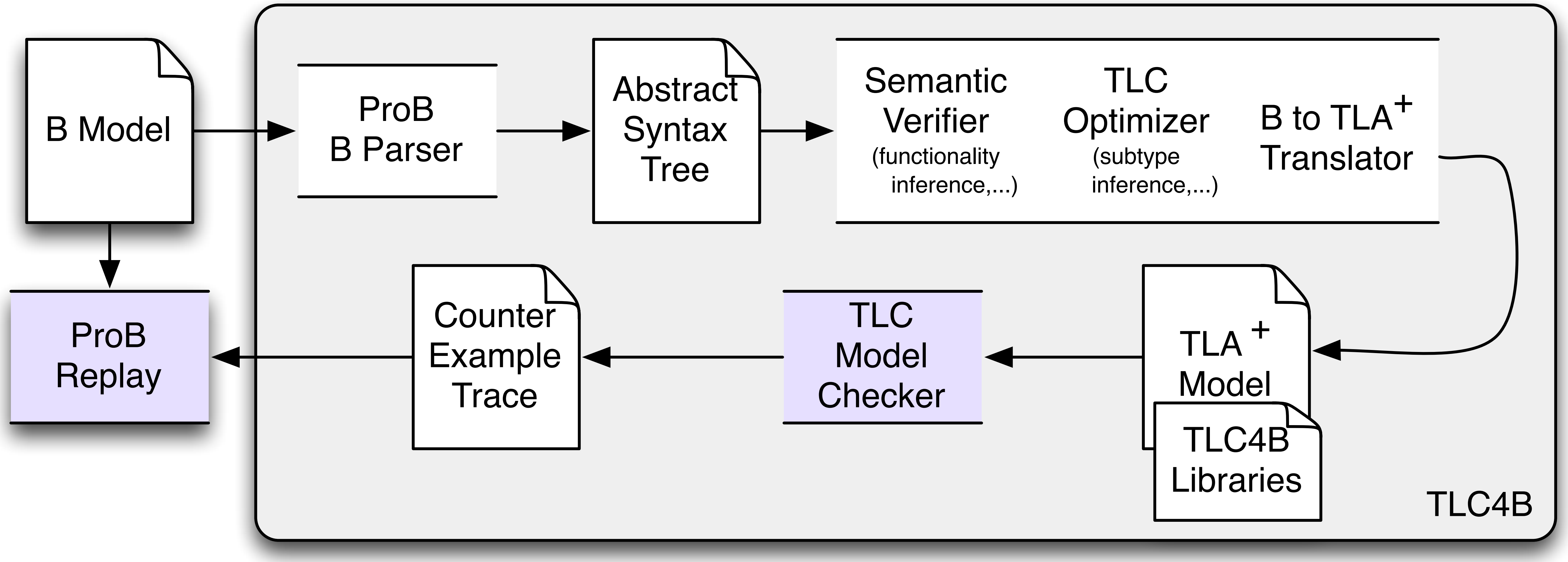
TLC for B/ProB

Part 1 of Talk



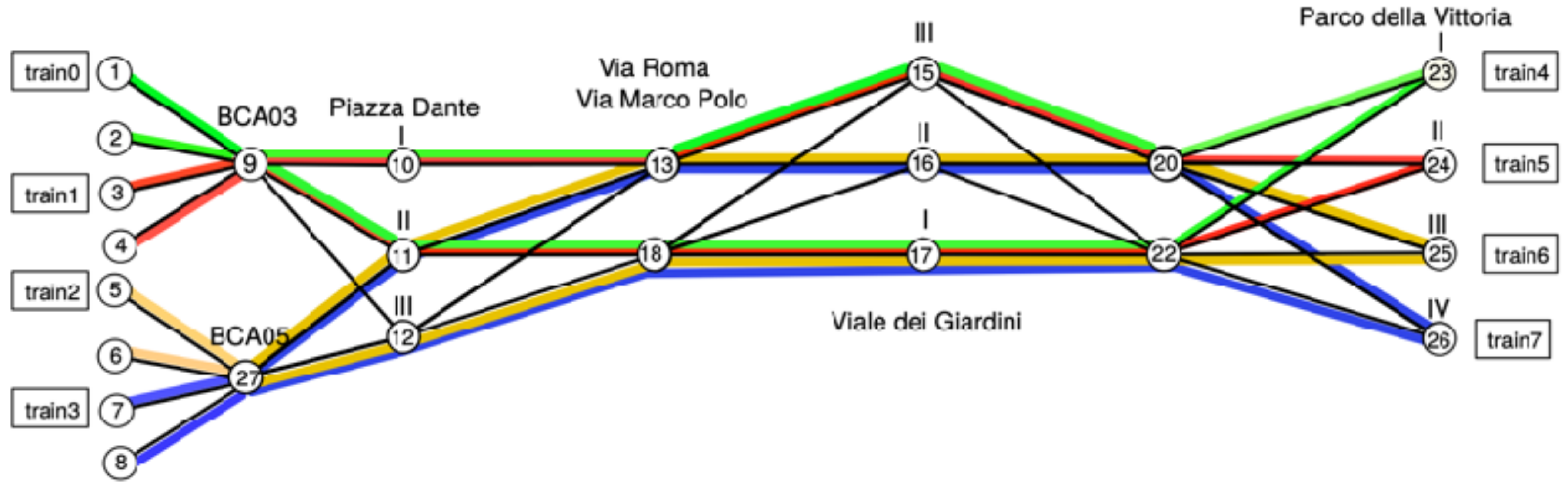
TLC for B

- Based on a translation of B to TLA+ with some special modules for relations and functions
  - Functions in B are sets of tuples and we can apply set and relation operators on them
- Motivation of TLC4B: for low-level models TLC is much faster than ProB:
  - no constraint solving overhead
  - Java vs Prolog?
  - ignoring hash collisions in TLC
  - parallel



Signalling Example

Benchmark



- Benchmark from Mars 2018 (Models for Formal Analysis of Real Systems)
- Using TLC4B required extension to handle limited sequential composition used

Communications-based Train Control (CBTC) systems are metro signalling platforms, which coordinate and protect the movements of trains within the tracks of a station, and between different stations. In CBTC platforms, a prominent role is played by the Automatic Train Supervision (ATS) system, which automatically dispatches and routes trains within the metro network. Among the various functions, an ATS needs to avoid deadlock situations, i.e., cases in which a group of trains block each other. In the context of a technology transfer study, we designed an algorithm for deadlock avoidance in train scheduling. In this paper, we present a case study in which the algorithm has been applied. The case study has been encoded using ten different formal verification environments, namely UMC, SPIN, NuSMV/nuXmv, mCRL2, CPN Tools, FDR4, CADP, TLA+, UPPAAL and ProB. Based on our experience, we observe commonalities and differences among the modelling languages consid- ered, and we highlight the impact of the specific characteristics of each language on the presented models.

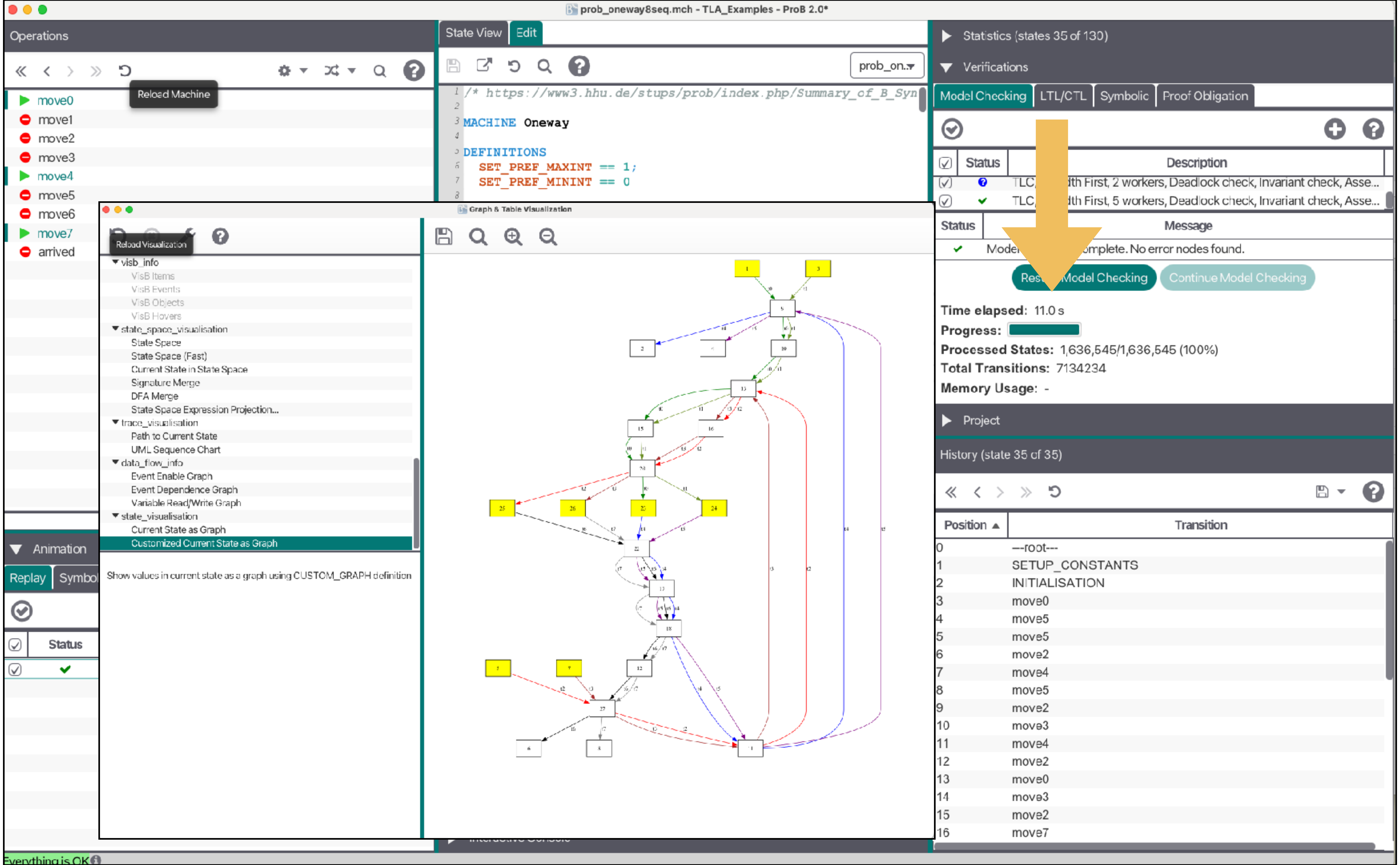
Table 3: Indicative Summary of Evaluation Times

Framework	Range of evalution times
UMC	38 - 86 seconds
SPIN	13 - 47 seconds
NuSMV/nuXMV	2.9 - 43 seconds
CADP	29 seconds
UPPAAL	16 seconds
TLA+	3 minutes
ProB	32 minutes
mCRL2	2 minutes -19 minutes
FDR4	15 seconds - 20 minutes
CPN	unable to deal with the state-space size

http://mars-workshop.org/repository/020-CBTC.html

https://arxiv.org/pdf/1803.10324

Model Check
ProB
TLC
Search Strategy:
Breadth First
Search for Errors and Properties:
Find Deadlocks
Find Invariant Violations
Find B ASSERTION Violations
Check welldefinedness
Check ASSERTLTL assertions
Check LTL formula
Search for GOAL
TLC preferences:
Setup constants using ProB
Number of workers
2
Proof guided model checking
Use symmetry
Save generated files
/Users/leuschel/git_root/prob_examples/public_
...
Validation task ID:
(optional)
Model Check
Cancel

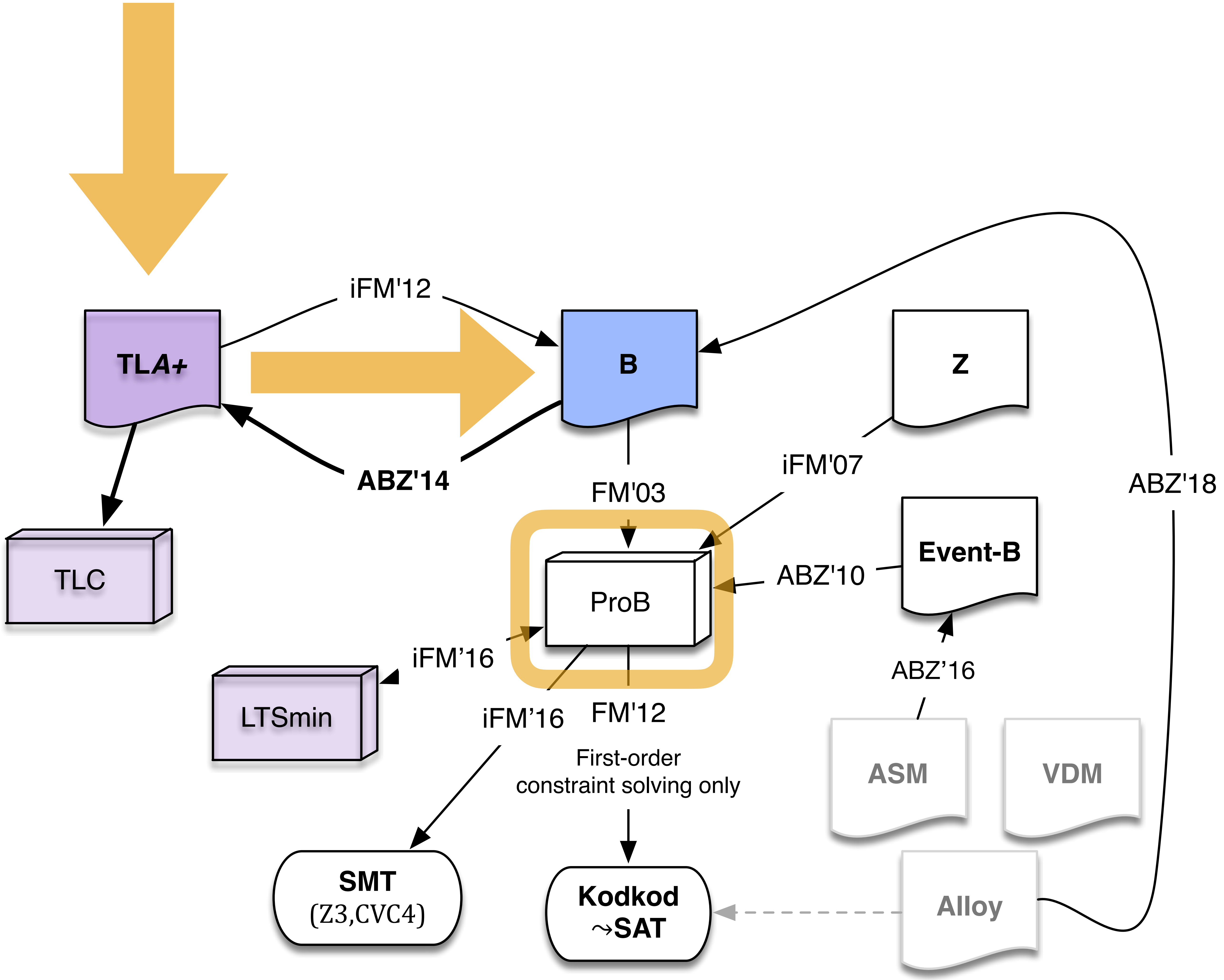

prob_oneway8seq.mch - TLA_Examples - ProB 2.0*
Operations
Reload Machine
move0
move1
move2
move3
move4
move5
move6
move7
arrived
State View
Edit
prob_on...
/* https://www3.hhu.de/stups/prob/index.php/Summary_of_B_Syn
MACHINE Oneway
DEFINITIONS
SET_PREF_MAXINT == 1;
SET_PREF_MININT == 0
Graph & Table Visualization
Reload Visualization
visb_info
VisB Items
VisB Events
VisB Objects
VisB Hovers
state_space_visualisation
State Space
State Space (Fast)
Current State in State Space
Signature Merge
DFA Merge
State Space Expression Projection...
trace_visualisation
Path to Current State
UML Sequence Chart
data_flow_info
Event Enable Graph
Event Dependence Graph
Variable Read/Write Graph
state_visualisation
Current State as Graph
Customized Current State as Graph
Show values in current state as a graph using CUSTOM_GRAPH definition
Animation
Replay
Symbol
Status
Statistics (states 35 of 130)
Verifications
Model Checking
LTL/CTL
Symbolic
Proof Obligation
Status
Description
TLC, ...th First, 2 workers, Deadlock check, Invariant check, Asse...
TLC, ...th First, 5 workers, Deadlock check, Invariant check, Asse...
Status
Message
Model ... omplete. No error nodes found.
Res... Model Checking
Continue Model Checking
Time elapsed: 11.0 s
Progress:
Processed States: 1,636,545/1,636,545 (100%)
Total Transitions: 7134234
Memory Usage: -
Project
History (state 35 of 35)
Position
Transition
0 ---root---
1 SETUP_CONSTANTS
2 INITIALISATION
3 move0
4 move5
5 move5
6 move2
7 move4
8 move5
9 move2
10 move3
11 move4
12 move2
13 move0
14 move3
15 move2
16 move7
Everything is OK

Recap: TLC for B

- Improved TLC support for B models
  - Latest version of TLC
  - More options (disable coverage, inspect TLA+ translation), support in ProB2-UI
  - Limited support for sequential composition (but not full support, no refinement support, no WHILE, ...)
- For MARS 18 model: combined ProB+TLC faster than either alone
  - ProB generates solutions for constants / initial state (via solver)
  - TLC checks large state space of simple B operations
  - Performance similar to Spin!

ProB for TLA+

Part 2 of Talk



ProB for TLA+

Translation of TLA+ to B AST

- Several values exist in both B and TLA+: strings, integers, records, sets, sequences,...
- B is typed (with type inference)
- Some slight differences (modulo, division)
-

MODULE *HourClock*

EXTENDS *Naturals*
CONSTANTS *start*
VARIABLES *hr*
ASSUME $start \in 0 \,..\, 12$

$Inv \triangleq hr \in 0 \,..\, 12$
$Init \triangleq hr = start$
$Inc \triangleq hr < 12 \land hr' = hr + 1$
$Reset \triangleq hr = 12 \land hr' = 1$
$Next \triangleq Inc \lor Reset$

MACHINE *HourClock*
CONSTANTS *start*
VARIABLES *hr*
PROPERTIES $start \in 0 \,..\, 12$
INVARIANT $hr \in 0 \,..\, 12$
INITIALISATION $hr : (hr = start)$
OPERATIONS
  $Inc\_Op =$ ANY hr_n
    WHERE $hr < 12 \land hr\_n = hr + 1$
    THEN $hr := hr\_n$ END

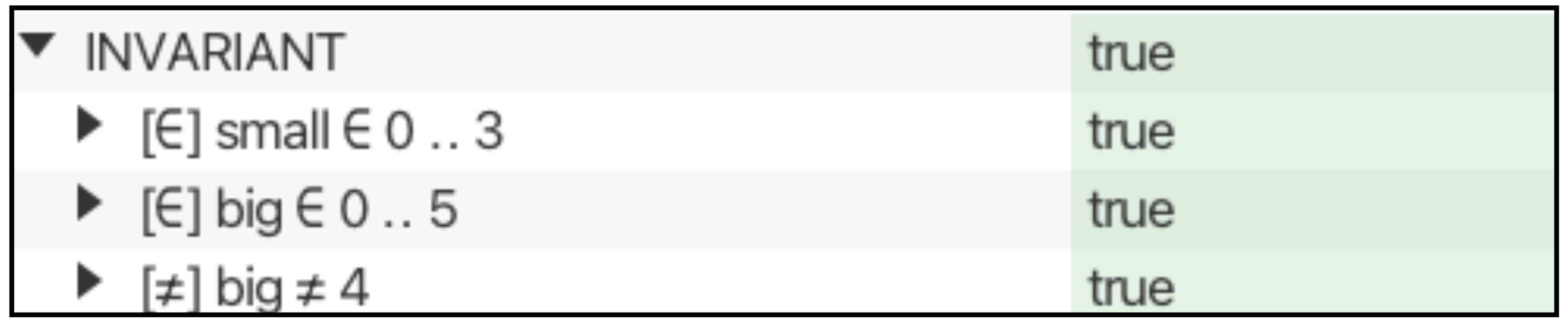
  $Reset\_Op =$ ANY hr_n
    WHERE $hr = 12 \land hr\_n = 1$
    THEN $hr := hr\_n$ END
END

Why ProB for TLA+

- Interactive animation
- Constraint solving capabilities
- Visualisation: state space projection, individual states using GraphViz, SVG-based interactive visualisation
- Storing & replaying traces, VO manager
- Model checking: other algorithms available: symmetry, operation caching, POR, ...

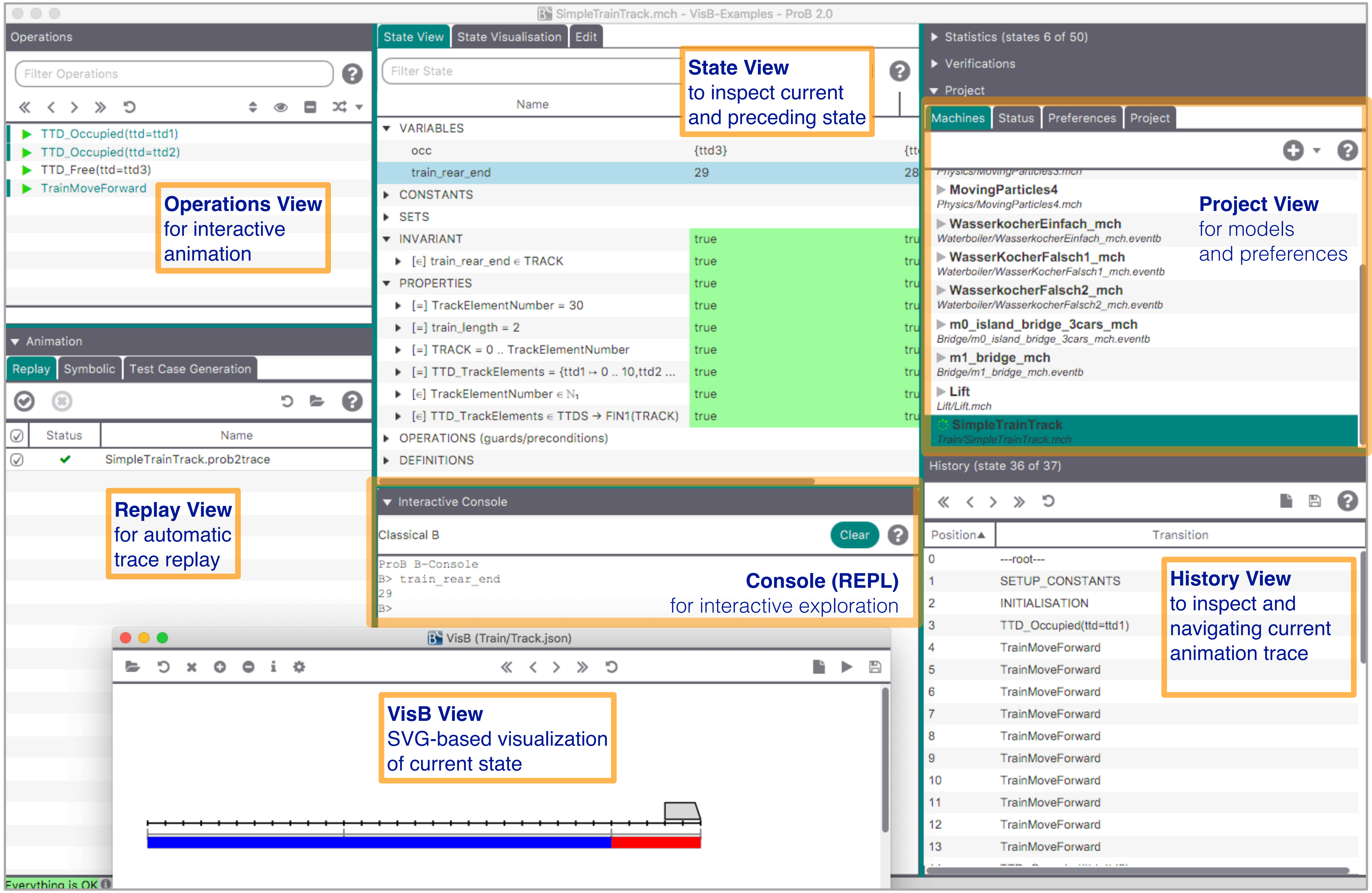
Why not ProB for TLA+

- Model checking can be much slower
- Only typed specifications are accepted
- Not all features of TLA+ supported
- Tool may show B formulas (even though use of Unicode overcomes part of the hurdle)



ProB2-UI

https://prob.hhu.de

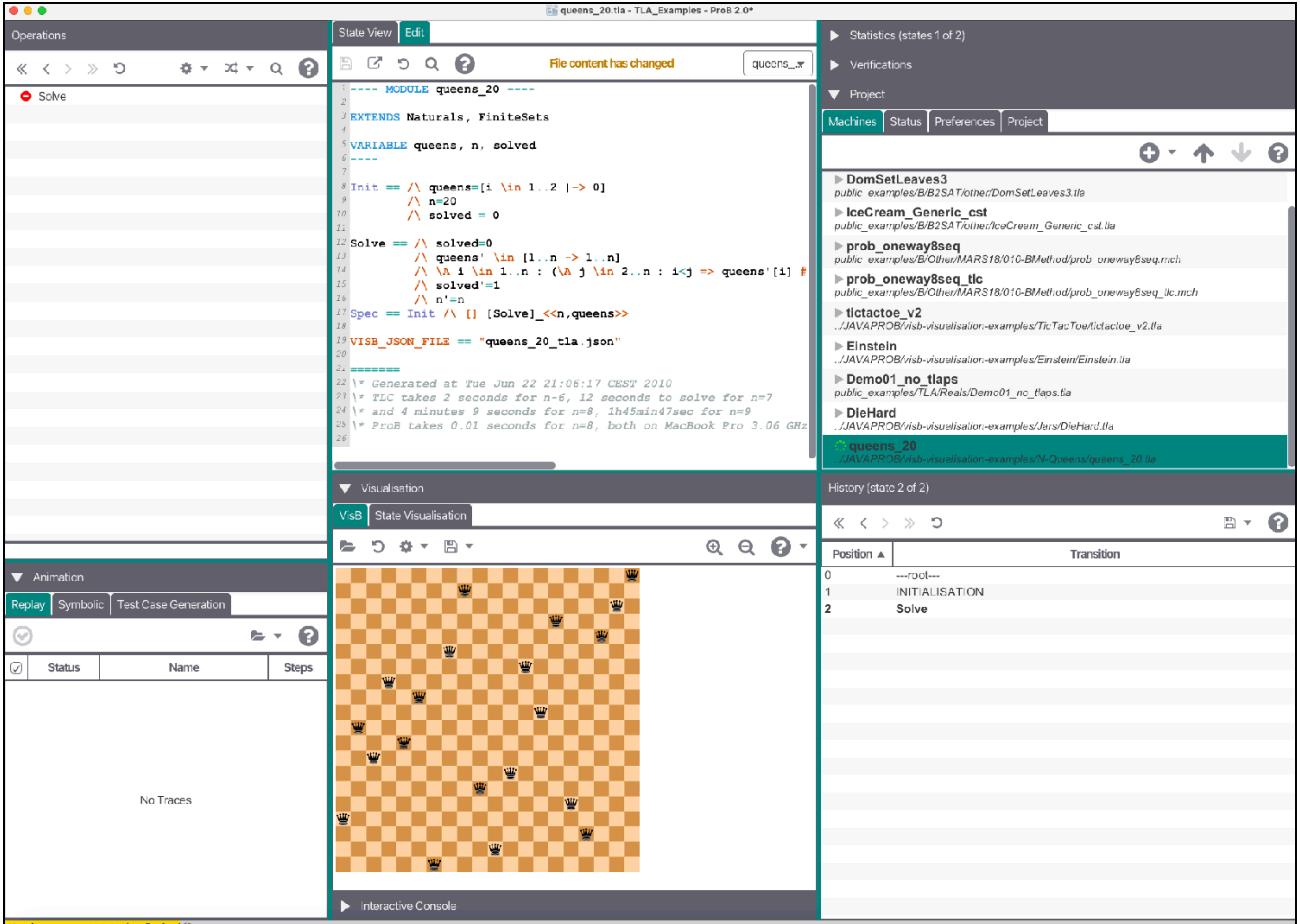


Constraint Solving Example

N-Queens

```
---- MODULE queens_20 ----

EXTENDS Naturals, FiniteSets

VARIABLE queens, n, solved
----

Init == /\ queens=[i \in 1..2 |-> 0]
       /\ n=20
       /\ solved = 0

Solve == /\ solved=0
        /\ queens' \in [1..n -> 1..n]
        /\ \A i \in 1..n : (\A j \in 2..n : i<j => queens'[i] # queens'[j] /\
                          queens'[i]+i-j # queens'[j] /\ queens'[i]-i+j # queens'[j])
        /\ solved'=1
        /\ n'=n
Spec == Init /\ [] [Solve]_<<n,queens>>
```

MacBook Air M2 **n**	**ProB 1.13.1 (MC Time)**	**TLC 2.19 Toolbox 1.7.4 (MC Time)**
6	0.003	2.408
7	0.003	5.953
8	0.003	77.590
9	0.004	35min 23 sec
20	0.018	?

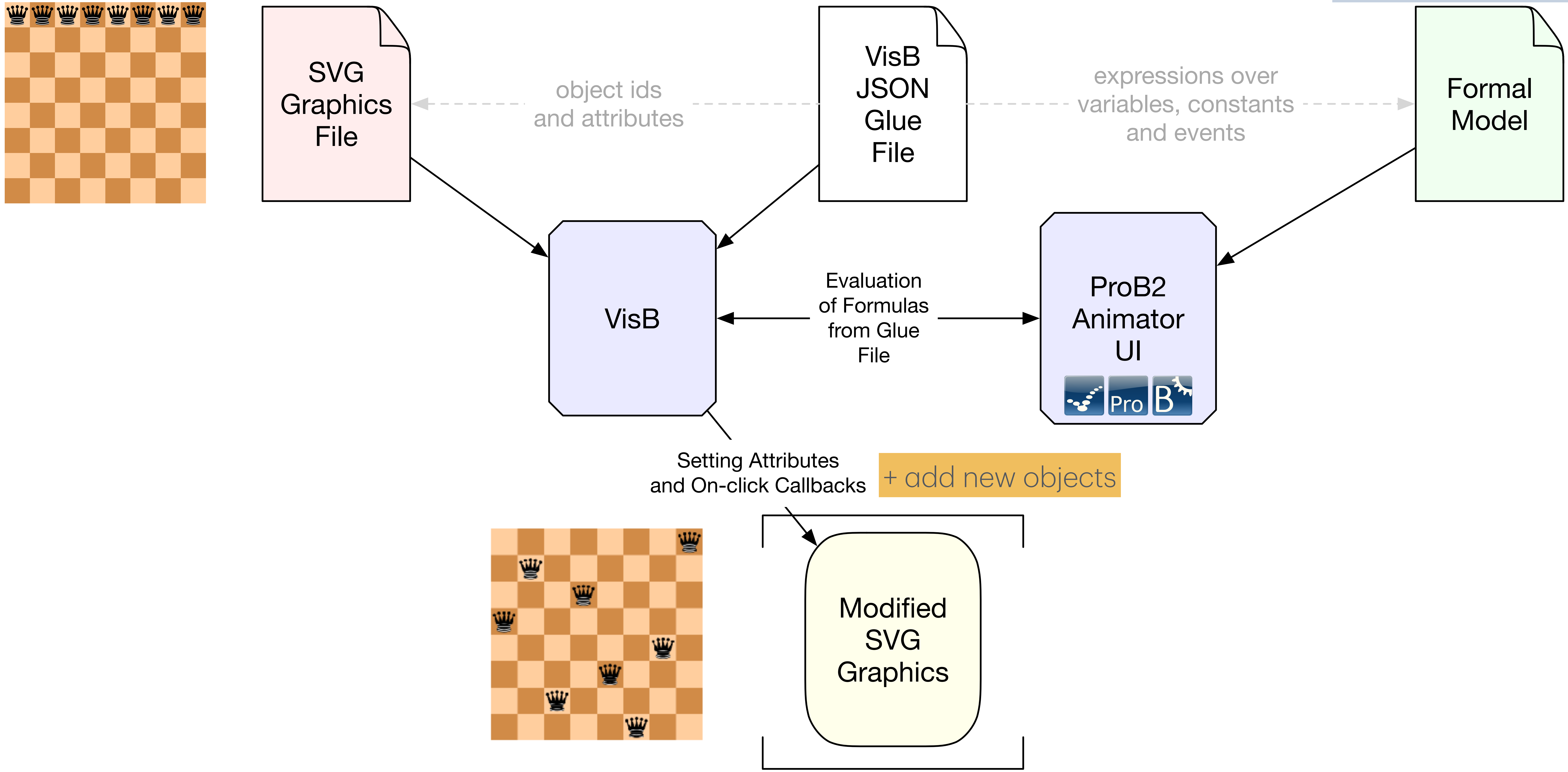

queens_20.tla - TLA_Examples - ProB 2.0*
Operations
Solve
State View
Edit
File content has changed
queens_
---- MODULE queens_20 ----
EXTENDS Naturals, FiniteSets
VARIABLE queens, n, solved

Init == /\ queens=[i \in 1..2 |-> 0]
/\ n=20
/\ solved = 0
Solve == /\ solved=0
/\ queens' \in [1..n -> 1..n]
/\ \A i \in 1..n : (\A j \in 2..n : i<j => queens'[i] #
/\ solved'=1
/\ n'=n
Spec == Init /\ [] [Solve]_<<n,queens>>
VISB_JSON_FILE == "queens_20_tla.json"
=======
* Generated at Tue Jun 22 21:06:17 CEST 2010
* TLC takes 2 seconds for n=6, 12 seconds to solve for n=7
* and 4 minutes 9 seconds for n=8, 1h45min47sec for n=9
* ProB takes 0.01 seconds for n=8, both on MacBook Pro 3.06 GHz
Statistics (states 1 of 2)
Verifications
Project
Machines
Status
Preferences
Project
DomSetLeaves3
public_examples/B/B2SAT/other/DomSetLeaves3.tla
IceCream_Generic_cst
public_examples/B/B2SAT/other/IceCream_Generic_cst.tla
prob_oneway8seq
public_examples/B/Other/MARS18/010-BMethod/prob_oneway8seq.mch
prob_oneway8seq_tlc
public_examples/B/Other/MARS18/010-BMethod/prob_oneway8seq_tlc.mch
tictactoe_v2
../JAVAPROB/visb-visualisation-examples/TicTacToe/tictactoe_v2.tla
Einstein
../JAVAPROB/visb-visualisation-examples/Einstein/Einstein.tla
Demo01_no_tlaps
public_examples/TLA/Reals/Demo01_no_tlaps.tla
DieHard
../JAVAPROB/visb-visualisation-examples/Jars/DieHard.tla
queens_20
../JAVAPROB/visb-visualisation-examples/N-Queens/queens_20.tla
Visualisation
VisB
State Visualisation
History (state 2 of 2)
Position
Transition
0
---root---
1
INITIALISATION
2
Solve
Animation
Replay
Symbolic
Test Case Generation
Status
Name
Steps
No Traces
Interactive Console

VisB Architecture

How to visualise formal models

```
---- MODULE queens_20 ----
EXTENDS Naturals, FiniteSets
VARIABLE queens, n, solved
----
Init == /\ queens=[i \in 1..2 |-> 0]
       /\ n=20
       /\ solved = 0
Solve == /\ solved=0
        /\ queens' \in [1..n -> 1..n]
        /\ \A i \in 1..n : (\A j \in 2..n : i<j => queens'[i] #
queens'[j] /\
                          queens'[i]+i-j # queens'[j] /\
queens'[i]-i+j # queens'[j])
        /\ solved'=1
        /\ n'=n
Spec == Init /\ [] [Solve]_<<n,queens>>
```



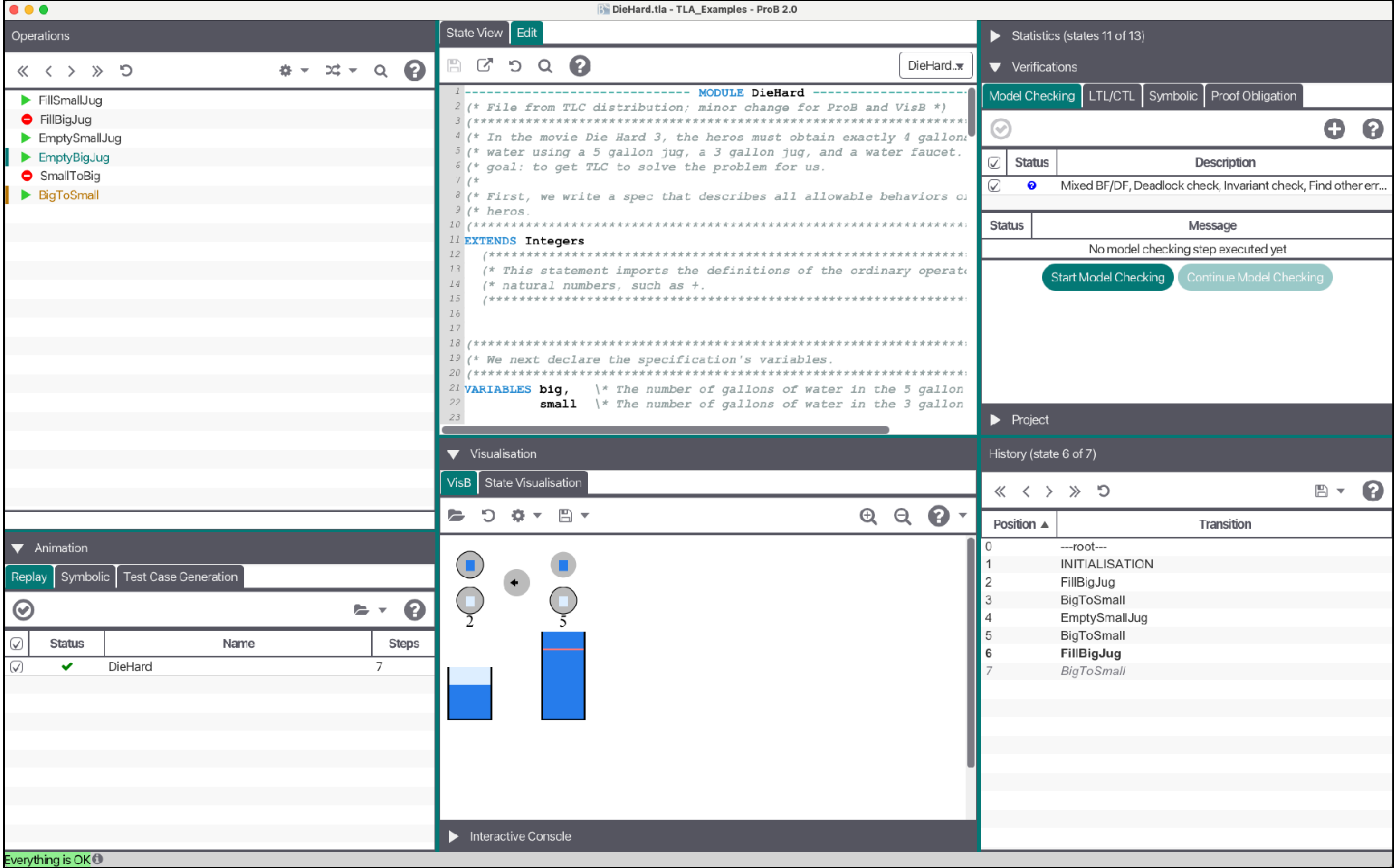
Another Example

Die Hard Jugs Puzzle

- Interactive visualisation
- Note: instead of putting VisB infos in JSON file, one can now also provide definitions in TLA+ syntax in the .tla file
  - VISB_SVG_OBJECTS == ...

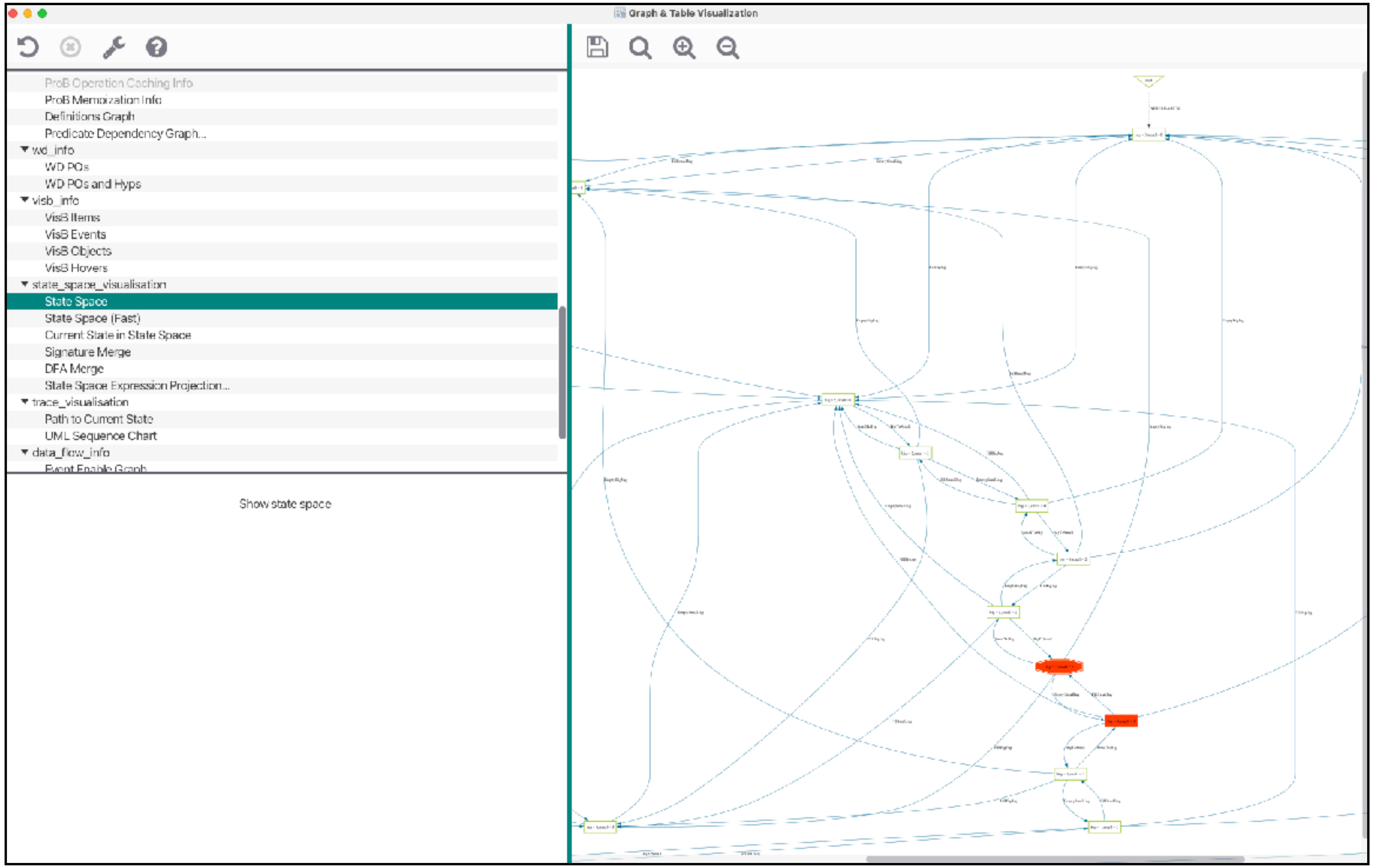
```
------------------------------ MODULE DieHard ------------------------------
(* File from TLC distribution; minor changes for ProB and VisB *)
(***************************************************************************)
(* In the movie Die Hard 3, the heroes must obtain exactly 4 gallons of*)
(* water using a 5 gallon jug, a 3 gallon jug, and a water faucet.  Our  *)
(* goal: to get TLC to solve the problem for us.                          *)
(*                                                                         *)
(* First, we write a spec that describes all allowable behaviors of our    *)
(* heros.                                                                  *)
(***************************************************************************)
...
TypeOK == /\ small \in 0..3
          /\ big   \in 0..5
...
(***************************************************************************)
Next ==  \/ FillSmallJug
         \/ FillBigJug
         \/ EmptySmallJug
         \/ EmptyBigJug
         \/ SmallToBig
         \/ BigToSmall
...
```

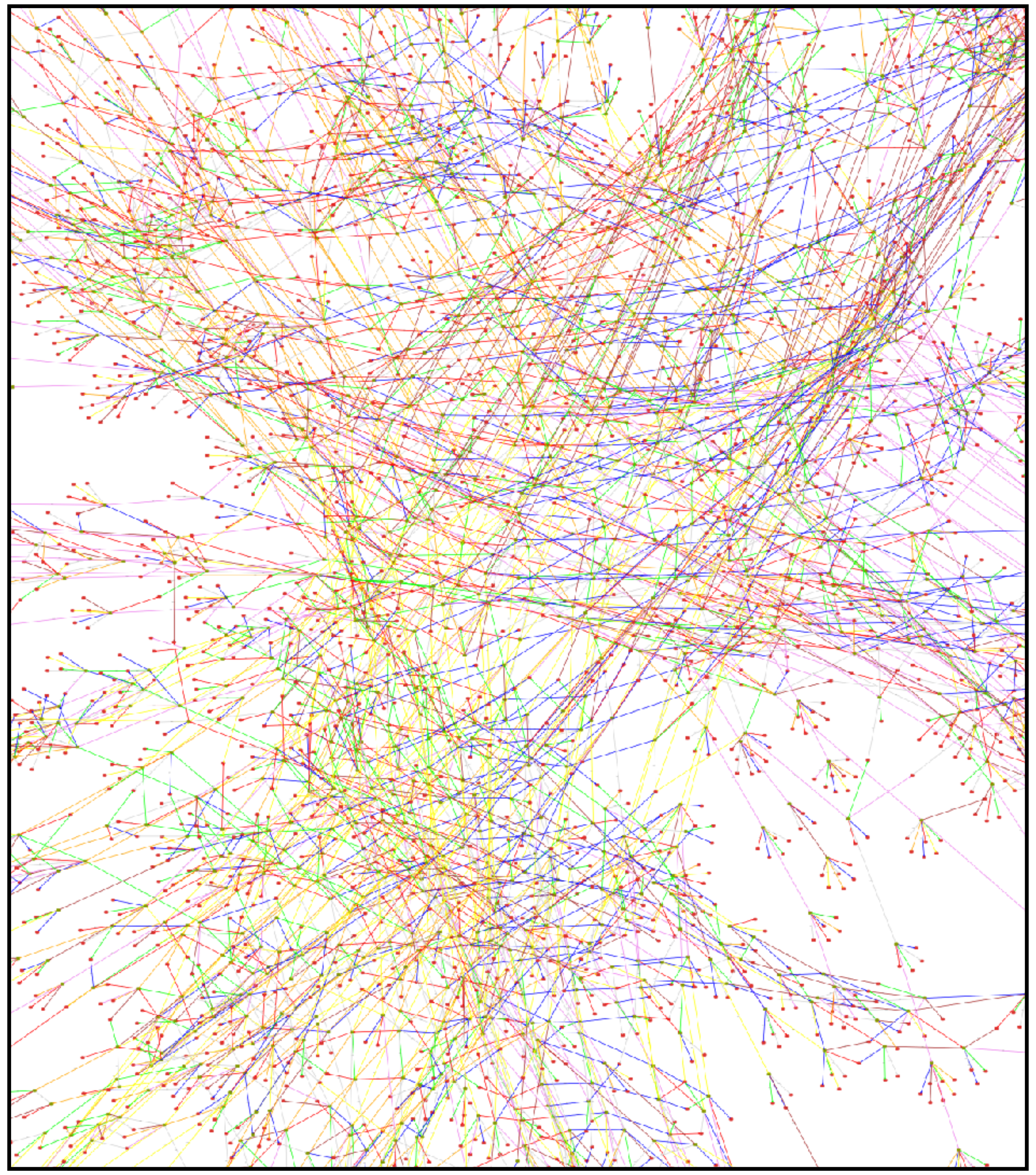
```
VISB_JSON_FILE == "DieHard_tla.json" \* addition for ProB-VisB
GOAL == (big=4) \* for ProB; not really required; config file has invariant
```


DieHard.tla - TLA_Examples - ProB 2.0
Operations
FillSmallJug
FillBigJug
EmptySmallJug
EmptyBigJug
SmallToBig
BigToSmall
State View
Edit
DieHard.tla
------------------------------ MODULE DieHard --------------------
(* File from TLC distribution; minor change for ProB and VisB *)
(* In the movie Die Hard 3, the heros must obtain exactly 4 gallons
(* water using a 5 gallon jug, a 3 gallon jug, and a water faucet.
(* goal: to get TLC to solve the problem for us.
(* First, we write a spec that describes all allowable behaviors o
(* heros.
EXTENDS Integers
(* This statement imports the definitions of the ordinary operat
(* natural numbers, such as +.
(* We next declare the specification's variables.
VARIABLES big, * The number of gallons of water in the 5 gallon
small * The number of gallons of water in the 3 gallon
Statistics (states 11 of 13)
Verifications
Model Checking
LTL/CTL
Symbolic
Proof Obligation
Status
Description
Mixed BF/DF, Deadlock check, Invariant check, Find other err...
Status
Message
No model checking step executed yet
Start Model Checking
Continue Model Checking
Project
Visualisation
VisB
State Visualisation
2
5
Interactive Console
History (state 6 of 7)
Position
Transition
0 ---root---
1 INITIALISATION
2 FillBigJug
3 BigToSmall
4 EmptySmallJug
5 BigToSmall
6 FillBigJug
7 BigToSmall
Animation
Replay
Symbolic
Test Case Generation
Status
Name
Steps
DieHard
7
Everything is OK

Full State Space Visualisation

sometimes beautiful, but seldom informative

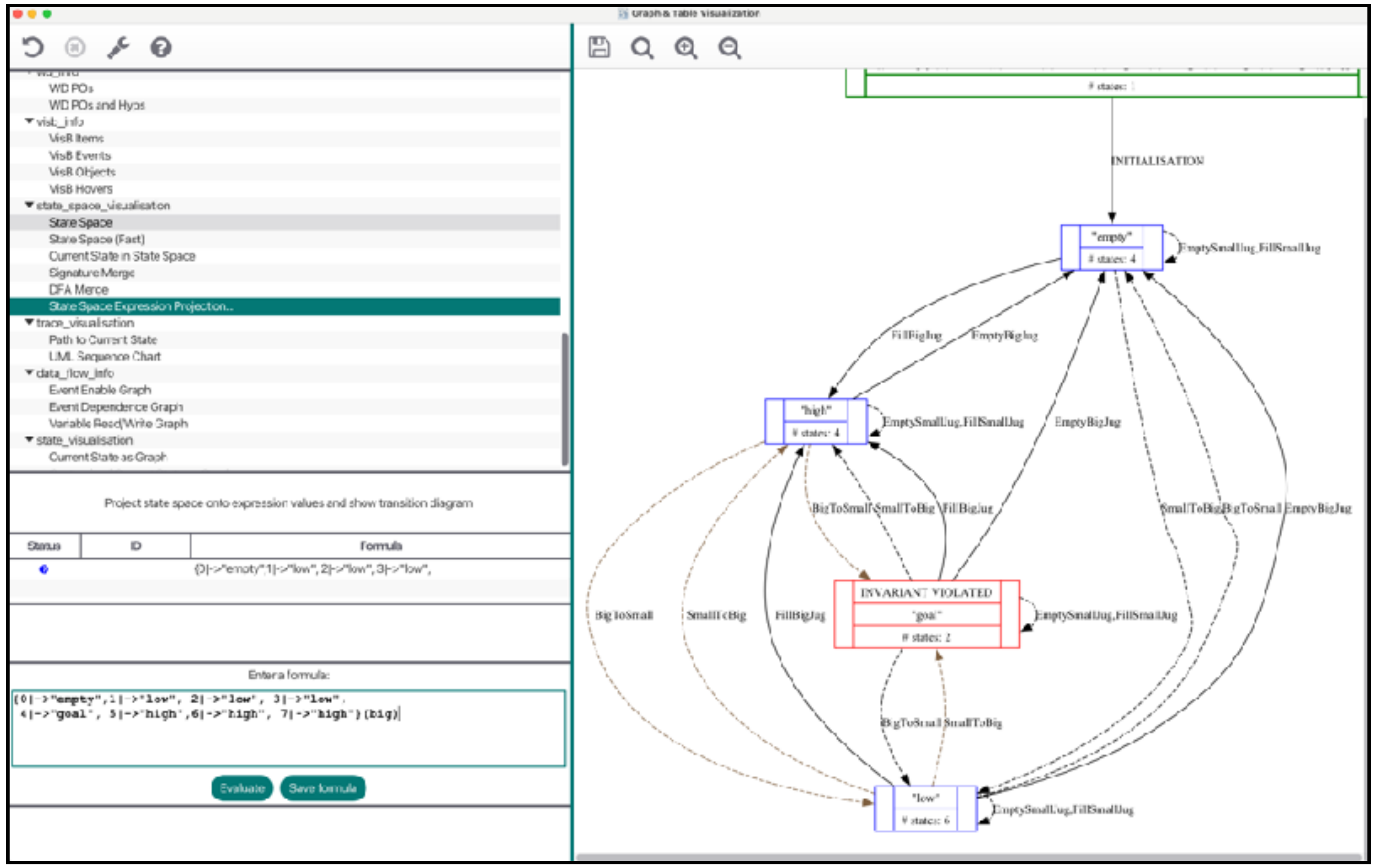


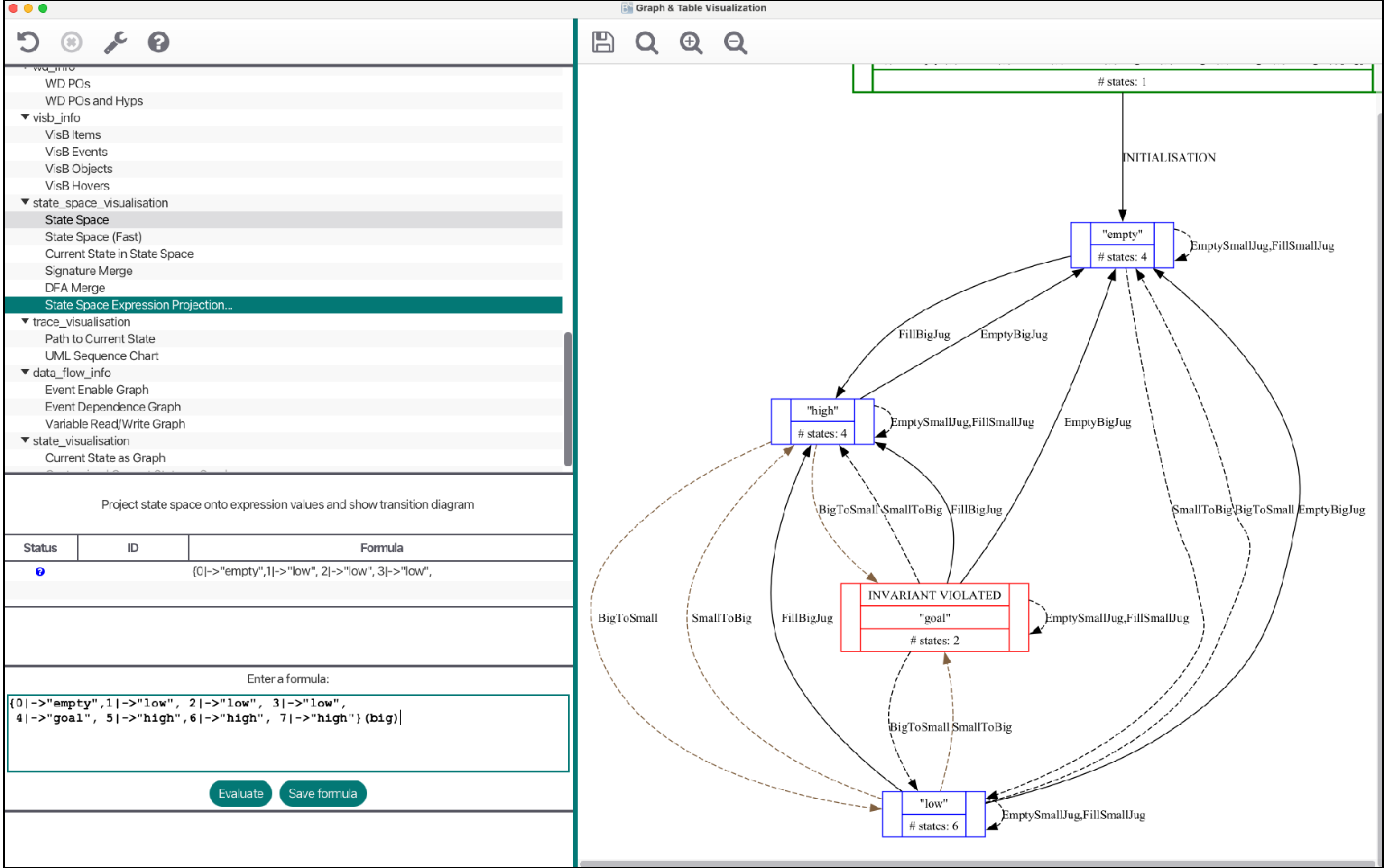


State Space Visualisation

Projection Diagrams

- Provide an expression, like
  - big
  - (big mod 2, small mod 2)
  - {0|->"empty",1|->"low", 2|->"low", 3|->"low", 4|->"goal", 5|->"high",6|->"high", 7|->”high"}(big)
- and state space will be projected onto possible values of the expression




Graph & Table Visualization
WD POs
WD POs and Hyps
visb_info
VisB Items
VisB Events
VisB Objects
VisB Hovers
state_space_visualisation
State Space
State Space (Fast)
Current State in State Space
Signature Merge
DFA Merge
State Space Expression Projection...
trace_visualisation
Path to Current State
UML Sequence Chart
data_flow_info
Event Enable Graph
Event Dependence Graph
Variable Read/Write Graph
state_visualisation
Current State as Graph
Project state space onto expression values and show transition diagram
Status
ID
Formula
{0|->"empty",1|->"low", 2|->"low", 3|->"low",
Enter a formula:
{0|->"empty",1|->"low", 2|->"low", 3|->"low",
4|->"goal", 5|->"high",6|->"high", 7|->"high"}(big)
Evaluate
Save formula
states: 1
INITIALISATION
"empty"
states: 4
EmptySmallJug,FillSmallJug
FillBigJug
EmptyBigJug
"high"
states: 4
EmptySmallJug,FillSmallJug
EmptyBigJug
BigToSmall
SmallToBig
FillBigJug
SmallToBig
BigToSmall
EmptyBigJug
INVARIANT VIOLATED
"goal"
states: 2
EmptySmallJug,FillSmallJug
BigToSmall
SmallToBig
FillBigJug
BigToSmall
SmallToBig
"low"
states: 6
EmptySmallJug,FillSmallJug

New B2SAT backend of ProB

cf talk at FM'24 tomorrow

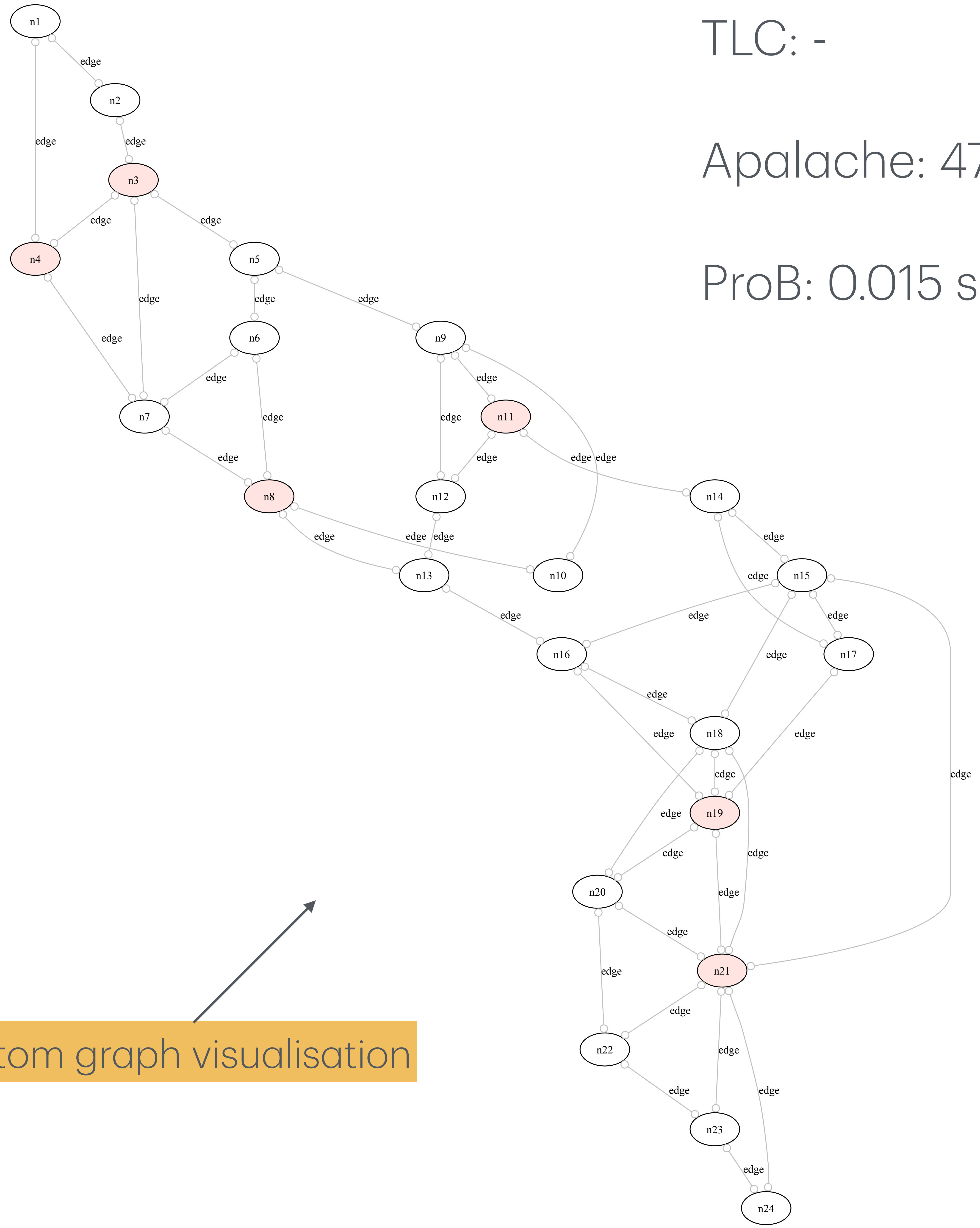
- ProB has various constraint solvers
  - default Prolog solver based on CLP(FD) solver and custom boolean, set, relation solvers
  - Kodkod: translation to SAT via Kodkod relational logic API (cf Alloy)
  - SMT: CVC4/Z3 translations (axiomatic and constructive)
  - new B2SAT direct translation to SAT
- All solvers also in principle available for TLA+ models

IceCream_Generic_cst.tla - TLA_Examples - ProB 2.0

State View | Edit

IceCream_...

```
---- MODULE IceCream_Generic_cst ----
\* Dominating set puzzle:
\* place ice cream vans so that every house (node) is
\* at most one block away from a van
EXTENDS Naturals, FiniteSets
CONSTANTS n1, n2, n3, n4, n5, n6, n7, n8, n9, n10,
          n11, n12, n13, n14, n15, n16, n17, n18, n19, n20,
          n21, n22, n23, n24,
          ice
VARIABLES vans
NODES == {n1, n2, n3, n4, n5, n6, n7, n8, n9, n10,
          n11, n12, n13, n14, n15, n16, n17, n18, n19, n20,
          n21, n22, n23, n24}
edge == {<<n1, n2>>, <<n1, n4>>, <<n2, n3>>,
         <<n3, n4>>, <<n3, n5>>, <<n3, n7>>,
         <<n4, n7>>, <<n5, n6>>, <<n5, n9>>,
         <<n6, n7>>, <<n6, n8>>, <<n7, n8>>,
         <<n8, n10>>, <<n8, n13>>,
         <<n9, n10>>, <<n9, n11>>, <<n9, n12>>,
         <<n11, n12>>, <<n11, n14>>, <<n12, n13>>,
         <<n13, n16>>, <<n14, n15>>, <<n14, n17>>,
         <<n15, n16>>, <<n15, n17>>, <<n15, n18>>, <<n15, n21>>,
         <<n16, n18>>, <<n16, n19>>, <<n17, n19>>,
         <<n18, n19>>, <<n18, n20>>, <<n18, n21>>,
         <<n19, n20>>, <<n19, n21>>, <<n20, n21>>, <<n20, n22>>,
         <<n21, n22>>, <<n21, n23>>, <<n21, n24>>,
         <<n22, n23>>, <<n21, n24>>, <<n23, n24>>}
DomSet == (\A x \in NODES :
            (ice[x] = TRUE
             \/ (\E nbour \in NODES :
                   (  <<nbour,x>> \in edge
                    \/ <<x,nbour>> \in edge)
                  /\ ice[nbour] = TRUE)))
ASSUME ice \in [NODES -> BOOLEAN]
       /\ DomSet /\ Cardinality({x \in (NODES): ice[x] = TRUE}) =< 6
Init ==
        vans = Cardinality({x\in NODES : ice[x]=TRUE})
Invariant ==
        vans \in (0 .. 10)
Next == UNCHANGED <<ice, vans>>

SET_PREF_TIME_OUT == 1500
SET_PREF_SOLVER_FOR_PROPERTIES == "sat"
CUSTOM_GRAPH == [layout |-> "dot", rankdir |-> "TB",
  nodes |-> {[value |-> j, style |-> "filled",
             fillcolor |-> (IF ice[j] = TRUE THEN "mistyrose" ELSE "white")]: j \in NODE
  edges |-> [color |-> "gray", arrowhead |-> "odot",
            arrowtail |-> "odot", dir |-> "both",
            label |-> "edge",
            edges |-> edge]]
====
```



Custom graph visualisation

TLC: -

Apalache: 47 secs

ProB: 0.015 secs

Playing Games

Tic-Tac-Toe TLA+ model

- VisB interactive visualisation
- MCTS Auto-Play

```
\* additions for ProB:
VISB_JSON_FILE == "tictactoe_visb.json"
GOAL == Won("O")
\* the following Invariant is violated by this model
INVARIANT == ~Won("O") \/ ~Won("X")
\* additions for ProB so that we can apply MCTS auto play:
GAME_MCTS_RUNS == 400
GAME_PLAYER == IF nextTurn = "X" THEN "max" ELSE "min"
GAME_OVER == IF Won("X") \/ Won("O") THEN TRUE ELSE FALSE
GAME_VALUE == IF Won("X") THEN 1 ELSE 0
```

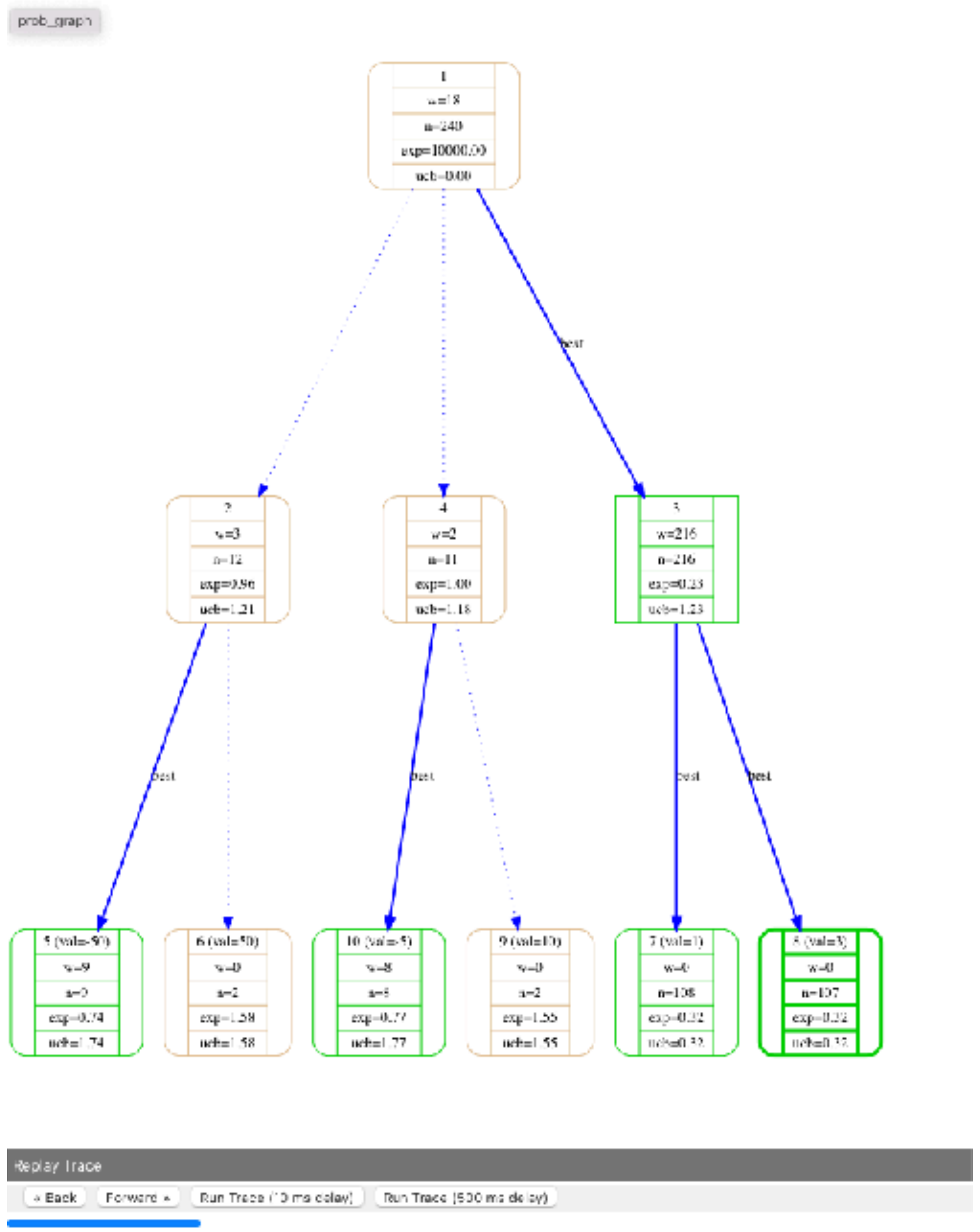
From:
https://elliotswart.github.io/pragmaticformalmodeling/

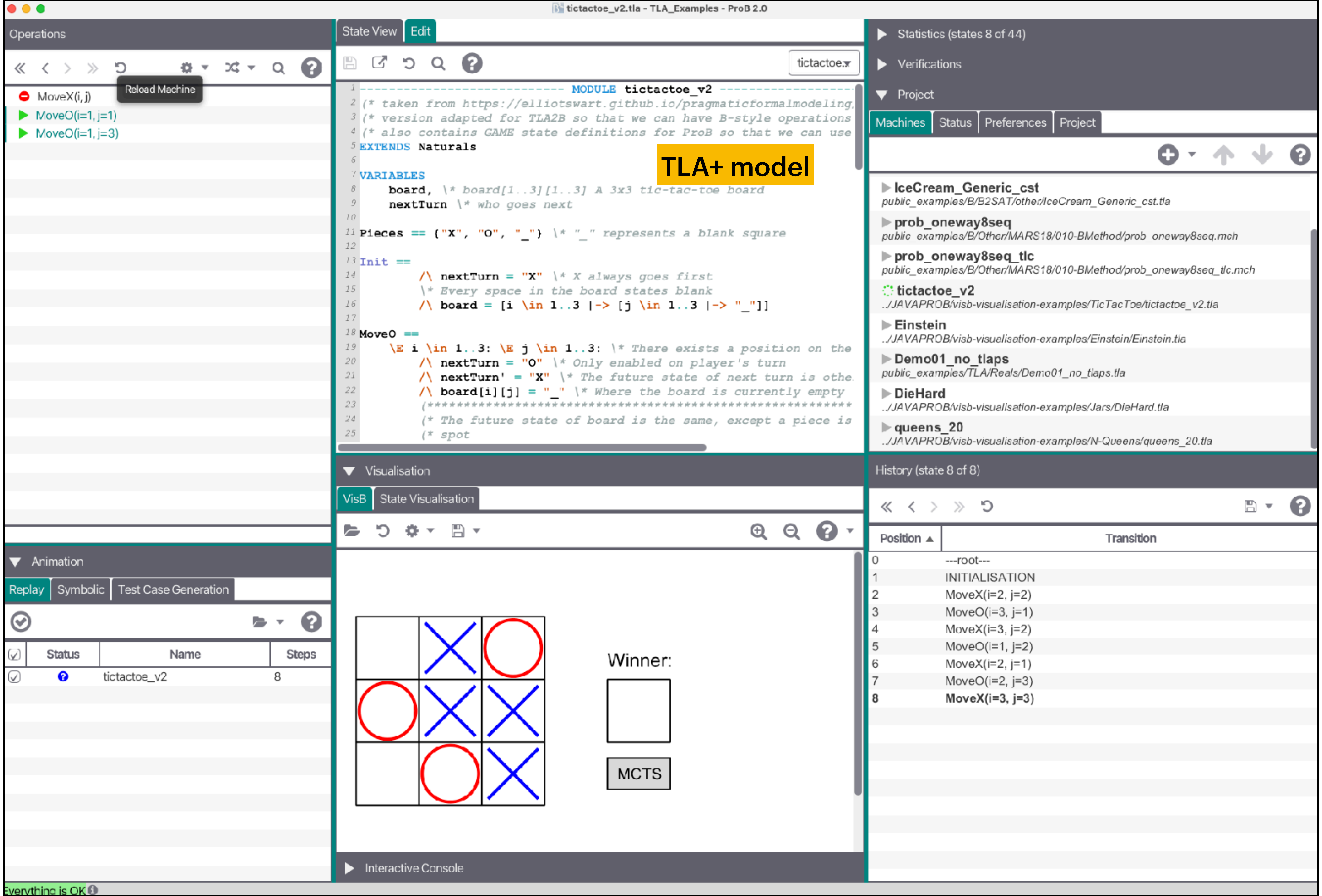
```
...
VARIABLES
   board, \* board[1..3][1..3] A 3x3 tic-tac-toe board
   nextTurn \* who goes next
Pieces == {"X", "O", "_"} \* "_" represents a blank square
Init ==
      /\ nextTurn = "X" \* X always goes first
      \* Every space in the board states blank
      /\ board = [i \in 1..3 |-> [j \in 1..3 |-> "_"]]
MoveO ==
   \E i \in 1..3: \E j \in 1..3: \* There exists a position on the board
      /\ nextTurn = "O" \* Only enabled on player's turn
      /\ nextTurn' = "X" \* The future state of next turn is other player
      /\ board[i][j] = "_" \* Where the board is currently empty
      (********************************************************************)
      (* The future state of board is the same, except a piece is in that *)
      (* spot                                                             *)
      (********************************************************************)
      /\ board' = [board EXCEPT
                  ![i][j] = "O"]
...
```

MCTS Game Play

Monte-Carlo Tree Search

- You can ask ProB to choose next action based on MCTS
  - GAME_MCTS_RUNS == 100
  - GAME_PLAYER == IF nextTurn = "X" THEN "max" ELSE "min"
  - GAME_OVER == IF Won("X") \/ Won("O") THEN TRUE ELSE FALSE
  - GAME_VALUE == IF Won("X") THEN 1 ELSE 0




tictactoe_v2.tla - TLA_Examples - ProB 2.0
Operations
Reload Machine
MoveX(i,j)
MoveO(i=1, j=1)
MoveO(i=1, j=3)
State View
Edit
tictactoe
---------------------------- MODULE tictactoe_v2 ----------------------------
(* taken from https://elliotswart.github.io/pragmaticformalmodeling
(* version adapted for TLA2B so that we can have B-style operations
(* also contains GAME state definitions for ProB so that we can use
EXTENDS Naturals
VARIABLES
board, * board[1..3][1..3] A 3x3 tic-tac-toe board
nextTurn * who goes next
Pieces == {"X", "O", "_"} * "_" represents a blank square
Init ==
/\ nextTurn = "X" * X always goes first
* Every space in the board states blank
/\ board = [i \in 1..3 |-> [j \in 1..3 |-> "_"]]
MoveO ==
\E i \in 1..3: \E j \in 1..3: * There exists a position on the
/\ nextTurn = "O" * Only enabled on player's turn
/\ nextTurn' = "X" * The future state of next turn is othe
/\ board[i][j] = "_" * Where the board is currently empty
(***
(* The future state of board is the same, except a piece is
(* spot
TLA+ model
Statistics (states 8 of 44)
Verifications
Project
Machines
Status
Preferences
Project
IceCream_Generic_cst
public_examples/B/B2SAT/other/IceCream_Generic_cst.tla
prob_oneway8seq
public_examples/B/Other/MARS18/010-BMethod/prob_oneway8seq.mch
prob_oneway8seq_tlc
public_examples/B/Other/MARS18/010-BMethod/prob_oneway8seq_tlc.mch
tictactoe_v2
../JAVAPROB/visb-visualisation-examples/TicTacToe/tictactoe_v2.tla
Einstein
../JAVAPROB/visb-visualisation-examples/Einstein/Einstein.tla
Demo01_no_tlaps
public_examples/TLA/Reals/Demo01_no_tlaps.tla
DieHard
../JAVAPROB/visb-visualisation-examples/Jars/DieHard.tla
queens_20
../JAVAPROB/visb-visualisation-examples/N-Queens/queens_20.tla
Visualisation
VisB
State Visualisation
Winner:
MCTS
Interactive Console
History (state 8 of 8)
Position
Transition
0 ---root---
1 INITIALISATION
2 MoveX(i=2, j=2)
3 MoveO(i=3, j=1)
4 MoveX(i=3, j=2)
5 MoveO(i=1, j=2)
6 MoveX(i=2, j=1)
7 MoveO(i=2, j=3)
8 MoveX(i=3, j=3)
Animation
Replay
Symbolic
Test Case Generation
Status
Name
Steps
tictactoe_v2
8
Everything is OK

Reals/Floats in ProB

Syntax in B

- New classical B keywords:
  - ℝ → ℤ: **ceiling**(.), **floor**(.)
  - ℤ → ℝ: **real**(.)
  - and real literals
- Existing B operators work for ℝ:
  - +, -, *, /, **max**, **min**, ∑, ∏
- LibraryReals.def provides many functions:
  - RADD, RMINUS, ..., RSIN, RCOS, ..., RSQRT, RPOW, RLOG,
  - RPI, REULER, RONE, RZERO

```
>>>> SIGMA(x).(x:1..100|1.0/real(x))
 5.187377517639621
>>>> RSIN(RADIANS(90.0))
 1.0
```

ProB Float Support

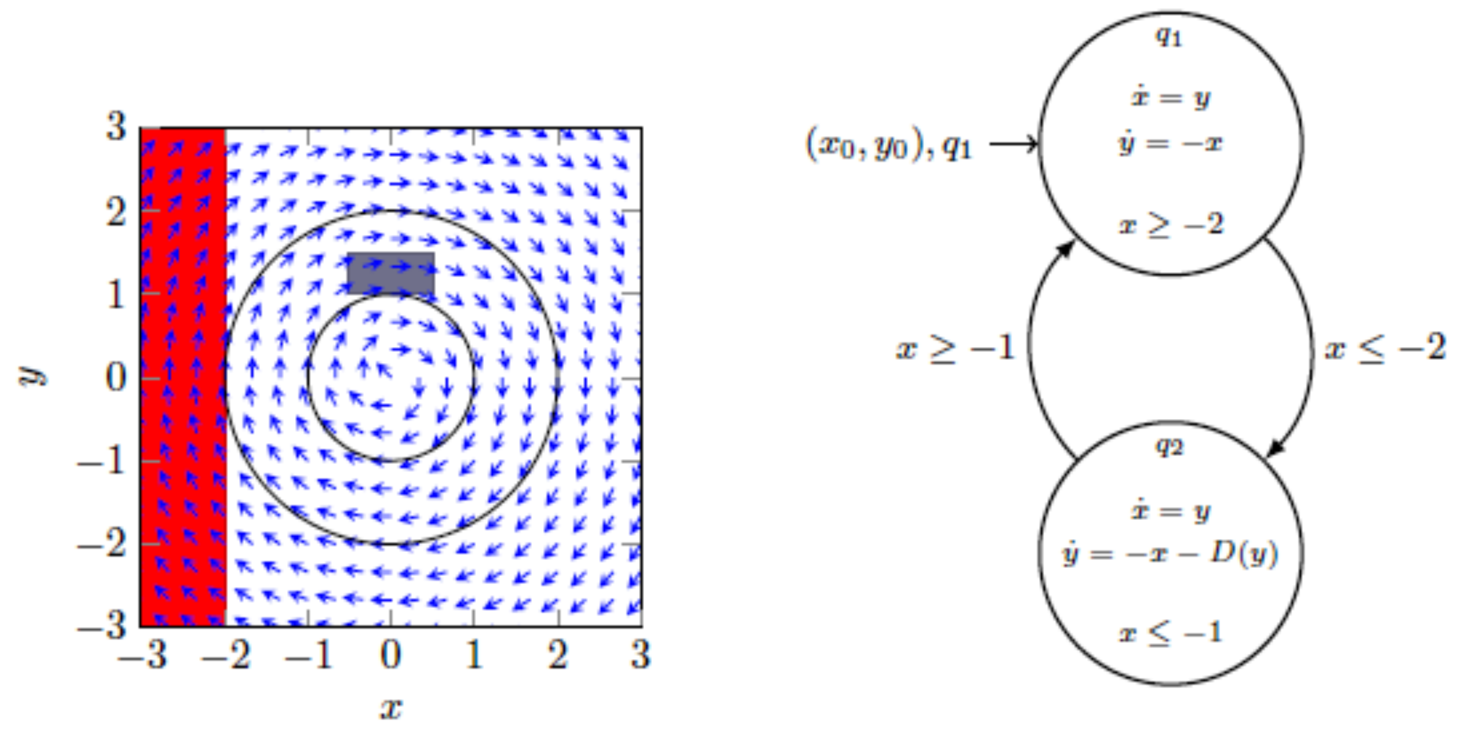
- Currently internally only floats supported
- Useful for VisB, Simulation, Controllers with floats and for “approximate” validation of models with reals
- preference REAL_SOLVER:
  - aggressive_float_solver,float_solver,none,**precise_float_solver**
  - precise_float_solver: default, tries to find exact solutions for 64-bit floats
  - aggressive_float_solver: does not check that solution is exact or the only one (similar to how CLP(Real) works in Prolog)
- Future: CLP(Q), Z3 support, real interval solver, ...

Jupyter Notebook Demo

TLA+ Example with Reals

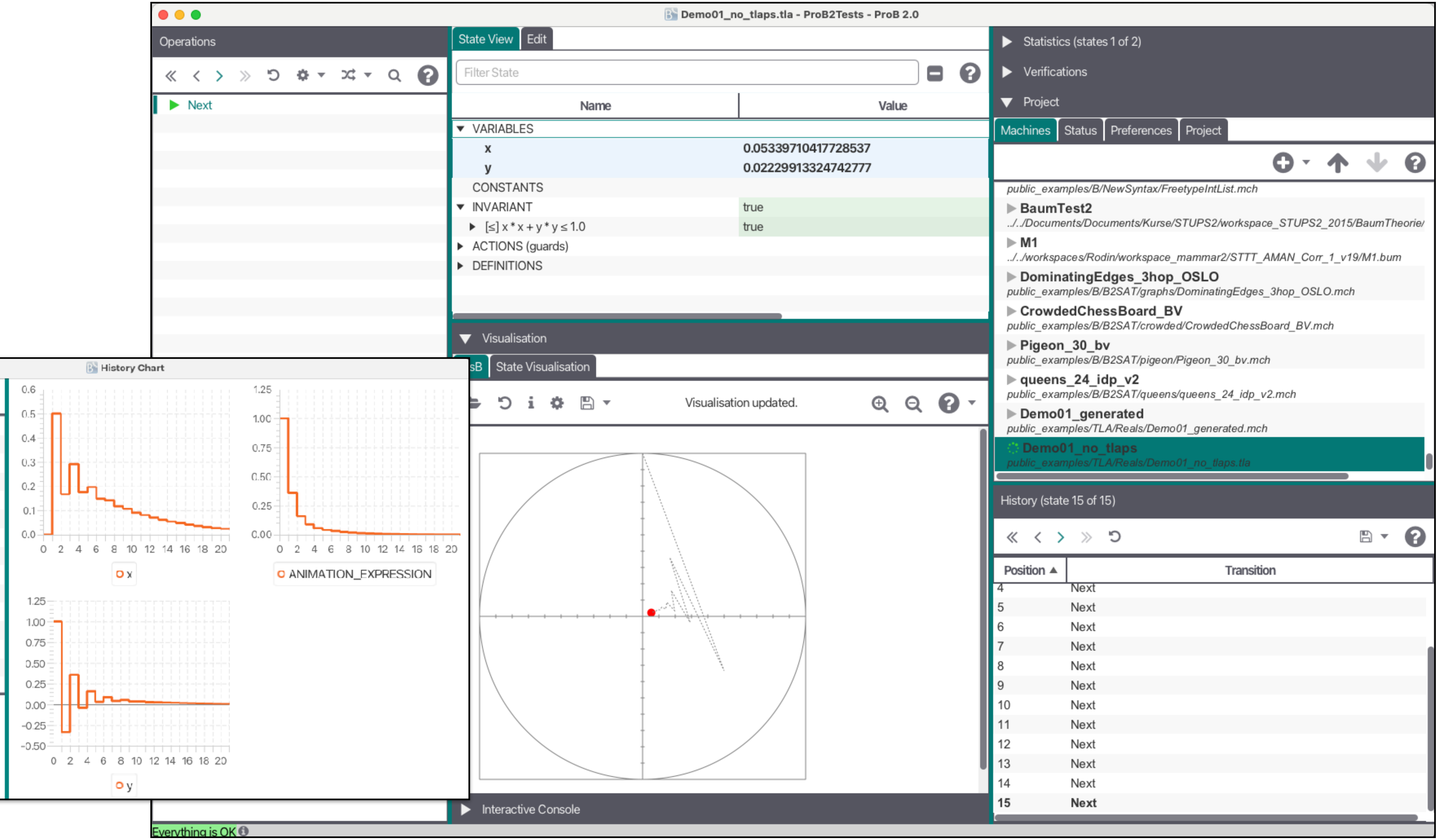
- From NFM'24 article "Real Arithmetic in TLAPM" by Gunasekera et al.

```
...
VARIABLES x, y
vars == << x, y >>

TypeInvariant == /\ x \in Real
                 /\ y \in Real

\* Initialise variables: x(0)^2 + y(0)^2 <= 1
Init == /\ x = 0.0
        /\ y = 1.0

Next == /\ x' = (2.0 / 3.0) * x  +  0.5 * y
        /\ y' = 0.5 * x - (1.0 / 3.0) * y

Spec == Init /\ [][Next]_vars\* /\ WF_vars(Next)
...
```



(a) Vector field (mode q_1) (b) Hybrid automaton

Fig. 2: Hybrid system model of an oscillator


Demo01_no_tlaps.tla - ProB2Tests - ProB 2.0
Operations
Next
State View
Edit
Filter State
Name
Value
VARIABLES
x
0.053397104177285 37
y
0.022299133247427 77
CONSTANTS
INVARIANT
true
[≤] x * x + y * y ≤ 1.0
true
ACTIONS (guards)
DEFINITIONS
Visualisation
State Visualisation
Visualisation updated.
Interactive Console
Statistics (states 1 of 2)
Verifications
Project
Machines
Status
Preferences
Project
public_examples/B/NewSyntax/FreetypeIntList.mch
BaumTest2
../../Documents/Documents/Kurse/STUPS2/workspace_STUPS2_2015/BaumTheorie/
M1
../../workspaces/Rodin/workspace_mammar2/STTT_AMAN_Corr_1_v19/M1.bum
DominatingEdges_3hop_OSLO
public_examples/B/B2SAT/graphs/DominatingEdges_3hop_OSLO.mch
CrowdedChessBoard_BV
public_examples/B/B2SAT/crowded/CrowdedChessBoard_BV.mch
Pigeon_30_bv
public_examples/B/B2SAT/pigeon/Pigeon_30_bv.mch
queens_24_idp_v2
public_examples/B/B2SAT/queens/queens_24_idp_v2.mch
Demo01_generated
public_examples/TLA/Reals/Demo01_generated.mch
Demo01_no_tlaps
public_examples/TLA/Reals/Demo01_no_tlaps.tla
History (state 15 of 15)
Position
Transition
4 Next
5 Next
6 Next
7 Next
8 Next
9 Next
10 Next
11 Next
12 Next
13 Next
14 Next
15 Next
History Chart
x
ANIMATION_EXPRESSION
y
Everything is OK

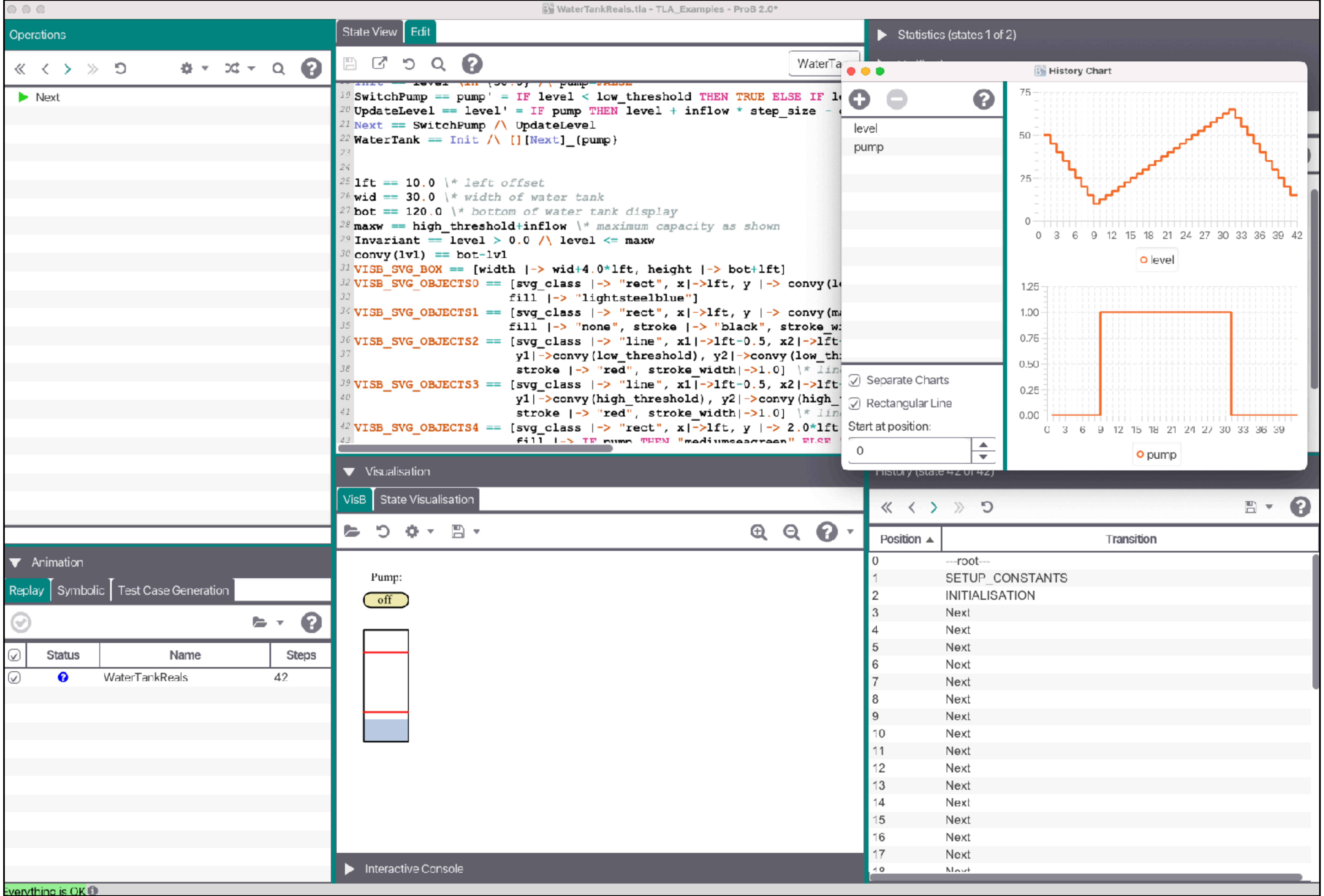
Another Example

Adapted Water Tank

- Using reals instead of integers
- With inlined VisB visualisation (no JSON file)

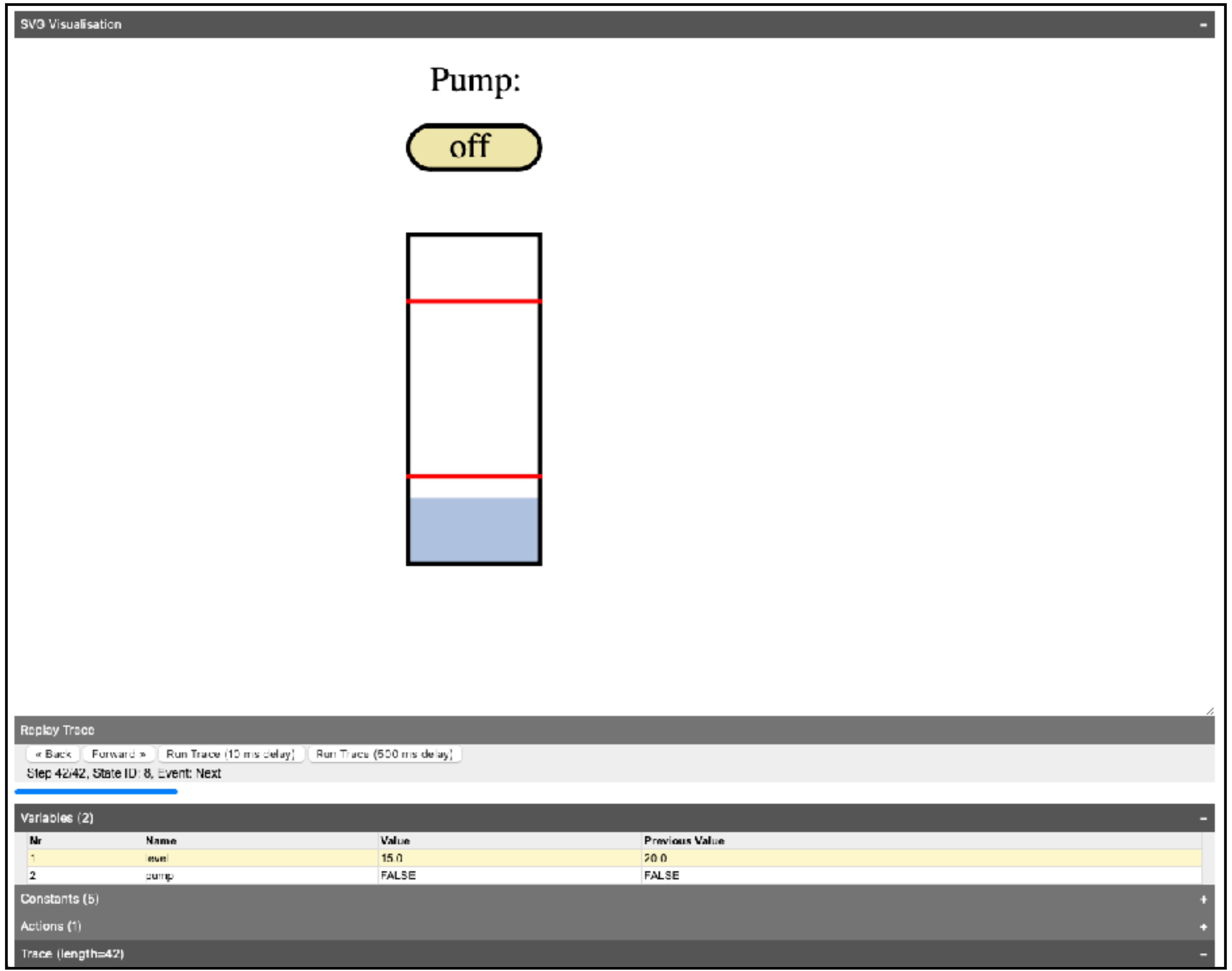
```
lft == 10.0 \* left offset
wid == 30.0 \* width of water tank
bot == 120.0 \* bottom of water tank display
maxw == high_threshold+inflow \* maximum capacity
Invariant == level > 0.0 /\ level <= maxw
convy(lvl) == bot-lvl
VISB_SVG_BOX == [width |-> wid+4.0*lft, height |-> bot+lft]
VISB_SVG_OBJECTS0 == [svg_class |-> "rect", x|->lft,
                y |-> convy(level), height |-> level, width |-> wid,
                fill |-> “lightsteelblue"]
...
```

```
--------------------- MODULE WaterTankReals ---------------------
EXTENDS Naturals, Reals
CONSTANTS
  low_threshold,
  high_threshold,
  (*@ unit s *) step_size,
  (*@ unit m**3 / s *) outflow,
  inflow
ASSUME
  /\ low_threshold = 20.0
  /\ high_threshold = 60.0
  /\ outflow = 10.0
  /\ inflow = 15.0
  /\ step_size = 0.5
VARIABLES
  pump,
  level
Init == level \in {50.0} /\ pump=FALSE
SwitchPump == pump' = IF level < low_threshold THEN TRUE ELSE
                          IF level > high_threshold THEN FALSE ELSE pump
UpdateLevel == level' = IF pump THEN level + inflow * step_size - outflow * step_size
                        ELSE level - outflow * step_size
Next == SwitchPump /\ UpdateLevel
WaterTank == Init /\ [][Next]_{pump}
```


WaterTankReals.tla - TLA_Examples - ProB 2.0*
Operations
Next
State View
Edit
WaterTa
SwitchPump == pump' = IF level < low_threshold THEN TRUE ELSE IF l
UpdateLevel == level' = IF pump THEN level + inflow * step_size -
Next == SwitchPump /\ UpdateLevel
WaterTank == Init /\ [][Next]_(pump)
lft == 10.0 * left offset
wid == 30.0 * width of water tank
bot == 120.0 * bottom of water tank display
maxw == high_threshold+inflow * maximum capacity as shown
Invariant == level > 0.0 /\ level <= maxw
convy(lvl) == bot-lvl
VISB_SVG_BOX == [width |-> wid+4.0*lft, height |-> bot+lft]
VISB_SVG_OBJECTS0 == [svg_class |-> "rect", x|->lft, y |-> convy(l
fill |-> "lightsteelblue"]
VISB_SVG_OBJECTS1 == [svg_class |-> "rect", x|->lft, y |-> convy(m
fill |-> "none", stroke |-> "black", stroke_w
VISB_SVG_OBJECTS2 == [svg_class |-> "line", x1|->lft-0.5, x2|->lft
y1|->convy(low_threshold), y2|->convy(low_th
stroke |-> "red", stroke_width|->1.0]
VISB_SVG_OBJECTS3 == [svg_class |-> "line", x1|->lft-0.5, x2|->lft
y1|->convy(high_threshold), y2|->convy(high_
stroke |-> "red", stroke_width|->1.0]
VISB_SVG_OBJECTS4 == [svg_class |-> "rect", x|->lft, y |-> 2.0*lft
Statistics (states 1 of 2)
History Chart
level
pump
Separate Charts
Rectangular Line
Start at position:
0
Visualisation
VisB
State Visualisation
Pump:
off
Animation
Replay
Symbolic
Test Case Generation
Status
Name
Steps
WaterTankReals
42
History (state 42 of 42)
Position
Transition
0 ---root---
1 SETUP_CONSTANTS
2 INITIALISATION
3 Next
4 Next
5 Next
6 Next
7 Next
8 Next
9 Next
10 Next
11 Next
12 Next
13 Next
14 Next
15 Next
16 Next
17 Next
Interactive Console
Everything is OK

HTML Export of Trace

stand-alone HTML file, can be opened without ProB in browser



Conclusion

ProB and TLA+

- TLC available as improved backend for B models
- TLA+ support in ProB
  - improvements: new SANY version, REAL, add CUSTOM_GRAPH/VISB definitions
  - Interactive Animation
  - Visualisation
  - Constraint Solving
  - Looking for ways to extend support for larger subset of TLA+

https://prob.hhu.de/w/index.php?title=TLA

Download Snapshot version for all of today's features

Thanks for the Support

Alstom (F. Mejia,…)
ClearSy (T Lecomte, R. Lapostelle, E. Mottin,…)
Siemens
Systerel
Thales/Hitachi (N. Nayeri, G. Hemzal,…)

DFG (Gepavas I+II, IVOIRE)
EU (Rodin, Deploy, Advance)

SICStus Prolog (Mats Carlsson, Per Mildner)

hhu.

STUPS Team & Friends

Jens Bendisposto
Carl Friedrich Bolz
Michael Butler
Joy Clark
Ivo Dobrikov
Jannik Dunkelau
Nadine Elbeshausen
Fabian Fritz
Marc Fontaine
Marc Frappier
David Geleßus
Jan Gruteser
Stefan Hallerstede
Dominik Hansen
Christoph Heinzen
Yumiko Jansing
Michael Jastram
Philipp Körner
Sebastian Krings

Lukas Ladenberger
Li Luo
Thierry Massart
Daniel Plagge
Antonia Pütz
Jan Roßbach
Mireille Samia
Joshua Schmidt
David Schneider
Sherin Schneider
Corinna Spermann
Sebastian Stock
Yumiko Takahashi
Edd Turner
Miles Vella
Fabian Vu
Michelle Werth
Dennis Winter