Integrating ProB into the TLA Toolbox

Dominik Hansen, Jens Bendisposto, Michael Leuschel

Institut für Informatik
Heinrich-Heine-Universität Düsseldorf
{hansen, bendisposto, leuschel}@cs.uni-duesseldorf.de

TLA$^+$ Workshop at ABZ 2014, Toulouse
03.06.2014

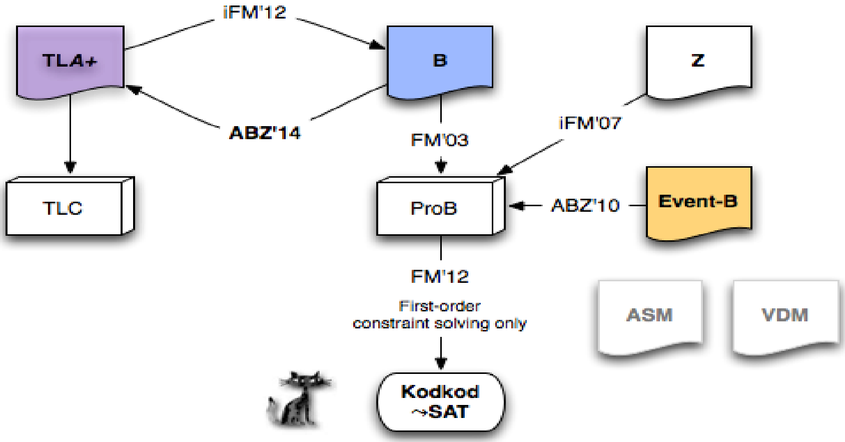
Toolbox

Demo

What is ProB?

- Model checker, animator and constraint solver
- Other features
  - Visualization (Model, Statespace, Formulas, Value over time)
  - LTL model checker
  - Profiler (Event coverage, value coverage, ...)
  - Disprover
- ProB kernel is written in prolog
- Originally designed to validate Classical B specifications
- Other supported formal languages:
  - EventB, Z, ...
  - TLA^{+}

Supported Formal Languages



iFM'12

TLA+

B

Z

ABZ'14

FM'03

iFM'07

TLC

ProB

ABZ'10

Event-B

FM'12

First-order
constraint solving only

ASM

VDM

Kodkod
$\leadsto$SAT

TLA2B translator

- Full automatic translation tool
- Build upon SANY
- Type inference algorithm
- Uses a TLC run configuration
- New version
  - Creating the extended B abstract syntax tree
  - No renaming phase
  - No need to extend the B language

Supported subset of TLA^{+}

- Data values
  - Integers, boolean values, strings, sets, functions, records
  - TLC's model values
- Operators
  - All non temporal built-in operators
  - Standard modules (Naturals, Integers, Sequences, ...)
  - User-defined operators
- Recursive Functions
- Extends & Instance

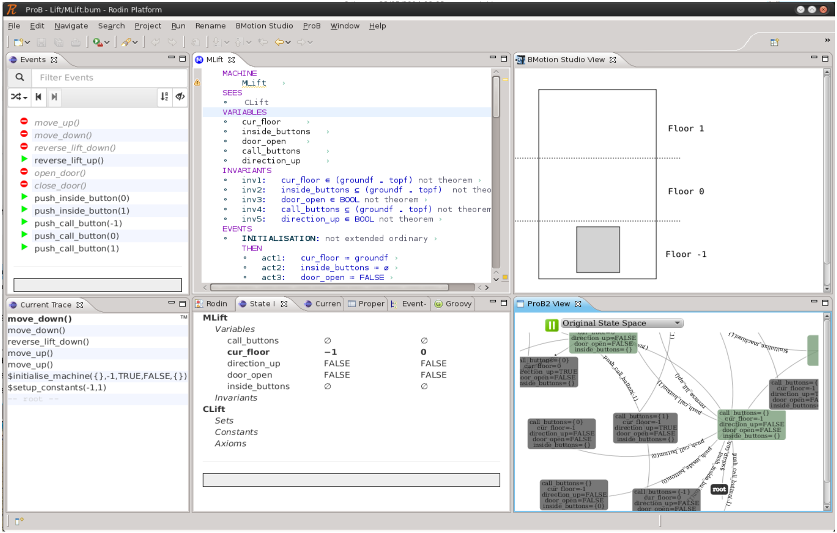
Restrictions

- Restriction caused by the B type system
  - Only values of the same type can be mixed in a set
  - Model values can not be compared to other values
  - Variables and constants must have a fixed type
- Temporal operators are not supported
- Recursive definitions are not supported yet

Toolbox Integration

- Existing plugin for the Eclipse based RODIN platform
- Almost independent from RODIN
- Reuse of all UI elements
- Toolbox plugin
  - UI bindings for the Toolbox
  - Code for loading TLA$^+$ models
  - Small changes to the product and target definitions of the toolbox

RODIN Integration



Events:

- move_up()
- move_down()
- reverse_lift_down()
- reverse_lift_up()
- open_door()
- close_door()
- push_inside_button(0)
- push_inside_button(1)
- push_call_button(-1)
- push_call_button(0)
- push_call_button(1)

```
MACHINE
    MLift
SEES
  ∘   CLift
VARIABLES
  ∘   cur_floor
  ∘   inside_buttons
  ∘   door_open
  ∘   call_buttons
  ∘   direction_up
INVARIANTS
  ∘   inv1:   cur_floor ∈ (groundf ‥ topf)  not theorem
  ∘   inv2:   inside_buttons ⊆ (groundf ‥ topf)  not theo
  ∘   inv3:   door_open ∈ BOOL  not theorem
  ∘   inv4:   call_buttons ⊆ (groundf ‥ topf)  not theorem
  ∘   inv5:   direction_up ∈ BOOL  not theorem
EVENTS
  ∘   INITIALISATION:  not extended ordinary
      THEN
        ∘   act1:   cur_floor ≔ groundf
        ∘   act2:   inside_buttons ≔ ∅
        ∘   act3:   door_open ≔ FALSE
```

Current Trace:

```
move_down()
move_down()
reverse_lift_down()
move_up()
move_up()
$initialise_machine({},-1,TRUE,FALSE,{})
$setup_constants(-1,1)
-- root --
```

MLift		
Variables		
call_buttons	∅	∅
cur_floor	**−1**	**0**
direction_up	FALSE	FALSE
door_open	FALSE	FALSE
inside_buttons	∅	∅
Invariants		
CLift		
Sets		
Constants		
Axioms		

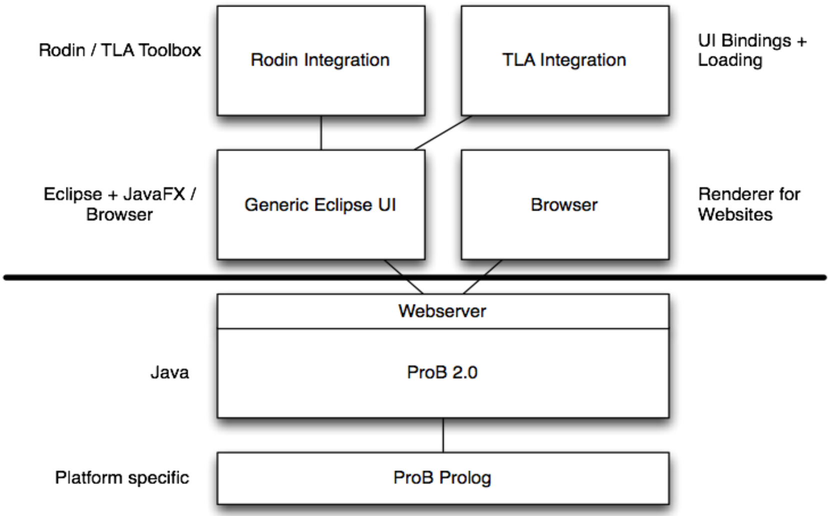
Current & Future Work

- Extending the translation
  - Recursive operators
  - Temporal formulas
- Interaction of ProB and TLC
  - Using ProB to setup the constants
  - Replaying traces produced by TLC in the animator
- Toolbox plugin
  - TLA$^+$ syntax in the evaluation console
  - Back-translation of B expression
  - Perspective & Views
  - Release of a stand-alone version

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

Questions?

ProB Architecture



B-Method & TLA^{+}

	TLA^{+}	B-Method
Invented by	Leslie Lamport	J.R. Abrial
State-based	√	√
Set theory	√	√
Predicate logic	√	√
Arithmetic	√	√
Temporal formulas	√	X
Type system	X	√
State transitions	Before-after predicates	Generalised substitutions
Model checker	TLC	ProB
Prove support	TLAPS	AtelierB

ProB & TLC

	ProB	TLC
Animation	√	
Model Checking	√(Symmetry reduction, Partial order Reduction)	√(Symmetry reduction)
Disk-Based		√
Parallelisation	√	√
Temporal Properties	√	√
Constraint Solving	√ (Inductive Inv. check, Disprover, ...)	
Graphical Visualization	√ (State, Formulas, Traces, Statespace)	
Coverage	√ (Profiling, Event coverage, Value Coverage, ...)	√ (Action coverage)

Translation of an example

MODULE *HourClock*

EXTENDS *Naturals*

CONSTANTS *start*

VARIABLES *hr*

ASSUME $start \in 0 \,..\, 12$

$$\begin{aligned}
Inv &\triangleq hr \in 0 \,..\, 12 \\
Init &\triangleq hr = start \\
Inc &\triangleq hr < 12 \land hr' = hr + 1 \\
Reset &\triangleq hr = 12 \land hr' = 1 \\
Next &\triangleq Inc \lor Reset
\end{aligned}$$

MACHINE *HourClock*

CONSTANTS *start*

VARIABLES *hr*

PROPERTIES $start \in 0 \,..\, 12$

INVARIANT $hr \in 0 \,..\, 12$

INITIALISATION $hr : (hr = start)$

OPERATIONS

$Inc =$ ANY *hr_n*

WHERE $hr < 12 \land hr\_n = hr + 1$

THEN $hr := hr\_n$ END

$Reset =$ ANY *hr_n*

WHERE $hr = 12 \land hr\_n = 1$

THEN $hr := hr\_n$ END

END