ProB
Harnessing the Power of Prolog to Bring Formal Models and Mathematics to Life

Michael Leuschel and the STUPS Team
University of Düsseldorf
What’s in a name?
Prolog B
A validation tool for the formal method: B Prolog

Built on a constraint solver for predicate logic, set theory and arithmetic, written in SICStus Prolog
ProB’s Solver in Action

Restored session: Mon  7 Nov 2022 19:30:29 CET
ProB's Prolog Constraint Solver

Demo using Jupyter

Prolog Day 2022

We highlight some of the features of ProB's constraint solving kernel written in Prolog, dealing with unbounded arithmetic, higher-order and infinite sets:

Some Features

Automatically detecting infinite sets:
What is the B formal method?
Formal Methods

- Mathematical techniques to produce correct software and systems: B, TLA+, Z, Alloy, CSP, …

- Highly recommended for safety critical applications, e.g. for SIL3/SIL4 railway applications by norm EN50128
B Formal Method

Industrial Applications

Tool Support

Mathematical Foundation

Figure of L14: Wikipedia, CC BY-SA 3.0
B was used for software for L14 and L1. ProB was used to validate configuration of L1.
Origins of B

• Train protection system SACEM for Paris RER Line A, sketch of the B-Method by Jean-Raymond Abrial, 1989 project by Alstom, RATP, SNCF to develop tools and train engineers

• Paris Metro Line 14 contract won by Matra Transport (now Siemens)
  - 1995: B tools industrialised by Digilog (now CLEARSY) leading to Atelier-B
  - ready by end 1998: 110 kLOC B model
  - Still in version 1.0, “no single issue caused by software”
• B for **Software**:

  - about 30% of CBTC systems worldwide employ the B formal method
  - Urbalis 400, Alstom, over 100 metro lines worldwide, 25% of worldwide CBTC market

Figure: from ClearSy
Validation with ProB
B Development Process

High-Level Formal Model

Refinement

Code Correct by Construction

- Code Generator A
- Code Generator B

Refinement Proof

Invariant Proof

Implementations

Imports

Refinement Proof

Proof

Proof

Invariant

High-Level Formal Model
But who guarantees the correctness of the high-level specification?
ProB2-UI

Operations View for interactive animation

State View to inspect current and preceding state

Project View for models and preferences

Replay View for automatic trace replay

Console (REPL) for interactive exploration

History View to inspect and navigating current animation trace

VisB View SVG-based visualization of current state

https://prob.hhu.de
Demo
Can we use the high-level specification before a complete implementation?

High-Level Formal Model

Model Checking

Animation

Visualisation

ProB

Refinement

Execution: Mathematical Model as Runtime Artefact

Code Correct by Construction
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
But who guarantees the correct configuration of the final system?
Data Validation at Siemens
One man-month of work done automatically in 3 minutes with ProB and undetected issues found
The work done with ProB is a great success for Siemens:

- Thanks to the automatization and ProB, the wayside data validation is quicker, more complete and easier than AtelierB.
- The on-board data validation, which was not formally proven, is now proven with ProB, so the use of formal methods has been extended.
- The B experts are only required when problems have been found, whereas before highly skilled people were required for long and fastidious B modification and proof.
- The validation of ProB itself enables a use in a SIL4 development.

Jérôme Falampin, SAS Siemens

cf. Deploy Deliverable D41, June 2011
ProB Uses for Data Validation

- RDV (Siemens; L1, Barcelona,...): used as primary toolchain
- Ovado, Ovado2 (Systerel, RATP): used as secondary toolchain
- DTVT (Clearsy, Alstom),
- Dave (ClearSy, G&E),
- Olaf (ClearSy, SNCF), ... : used as primary tool chain
- Caval (ClearSy), Rubin (Thales): currently used as only tool chain
A lot of (other) tools build upon ProB
Model checking, simulation, test-case generation, …

ClearSy Data Solver (Caval), …
Thanks to Prolog and Logic Programming

- Convenient to express semantics of specification languages
- Possible to write domain specific solvers, via Prolog’s flexible computation rule
- Compact encoding of analysis, optimisation, type checking and verification rules
- Very fast and robust Prolog systems like SICStus Prolog, with efficient co-routines and fast CLP(FD) library
Conclusion: Prolog has enabled a tool that

- brings formal mathematics to life
- helps find bugs in safety critical systems
- helps users visualise and understand their models
- is used in industrial applications
- is used for teaching
- is used in research (> 1000 citations)
- is the foundation for many other tools
Thanks for the Support

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

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

**SICStus Prolog** (Mats Carlsson, Per Mildner)

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

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