An Event-B Sequent Prover in Prolog and ProB

Katharina Engels, Jan Gruteser, Michael Leuschel

13th Rodin Workshop
東京 (Hybrid), 18.05.2026

Heinrich Heine
University
Düsseldorf

prob.hhu.de
stups.hhu.de

Recap

Heinrich Heine University Düsseldorf

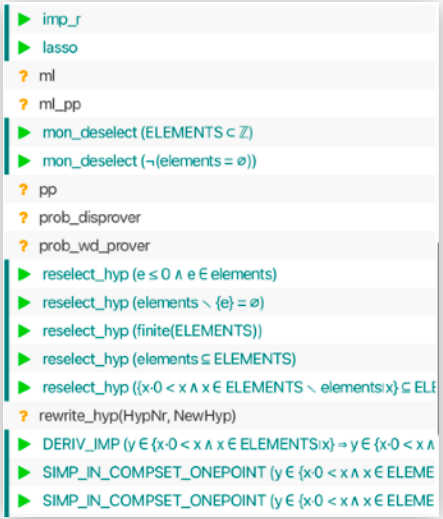
Last Year:

- We started implementing the Rodin proof rules in Prolog
- Initial idea: improve traceability and understanding of proofs in teaching
  - In Rodin, individual proofs steps can be difficult to explore (and understand)
- Turn mathematical definitions into executable Prolog rules that can be used in a prover

https://wiki.event-b.org/index.php/Inference_Rules
https://wiki.event-b.org/index.php/All_Rewrite_Rules

Animate the Proof

- ProB's sequent prover allows to *animate* the proof rules
  - Each PO is a start state
  - Based on the current sequent and the transition predicates, the ProB animator gives the applicable proof rules
  - The state is the current proof sequent (+ continuations)



- Prolog transition for the **`IMP_R`** rule:

```
trans(imp_r, sequent(Hyps, implication(P,Q),Cont),
             sequent(Hyps1,        Q        ,Cont)) :-
    add_hyp(P,Hyps,Hyps1).
```

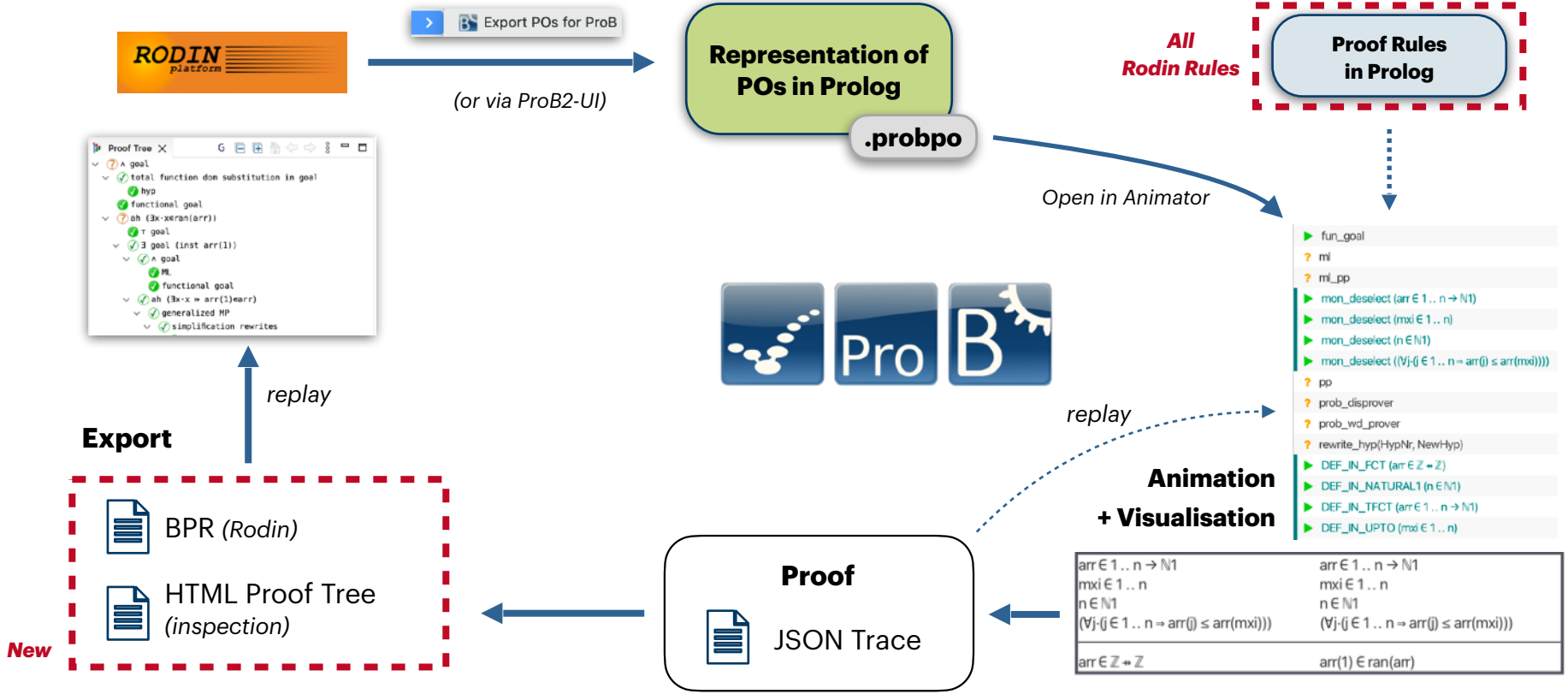
$$\frac{H, P \vdash Q}{H \vdash P \Rightarrow Q}$$

New Features

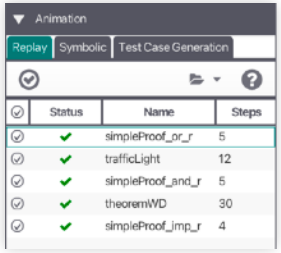
hhu
Heinrich Heine
University Düsseldorf

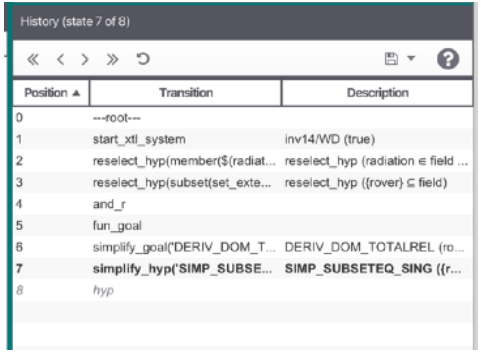
- Allmost all rules implemented (≈ 600) and integrated into the ProB core
- Interactive proving with ProB together with visualisation
- New exports:
  - Interactive HTML proof tree
  - Proof export as Rodin BPR file (experimental)
- First version of an automated prover

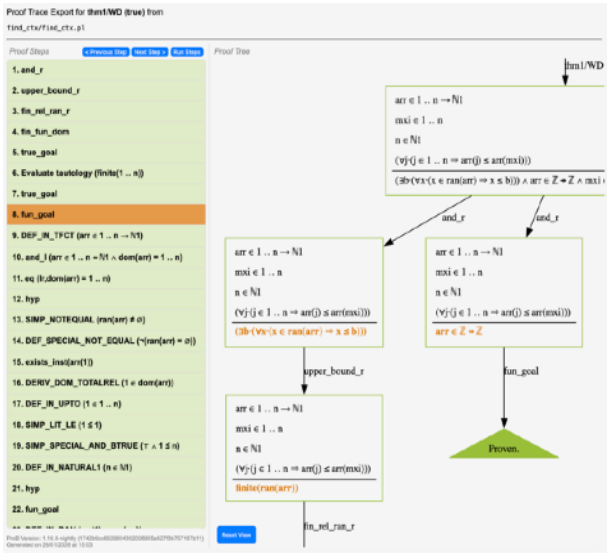
Overview


RODIN platform
Export POs for ProB
(or via ProB2-UI)
Representation of POs in Prolog
.probpo
All Rodin Rules
Proof Rules in Prolog
Open in Animator
Proof Tree
A goal
total function dom substitution in goal
hyp
functional goal
ah (∃x·xeran(arr))
T goal
∃ goal (inst arr(1))
∧ goal
ML
functional goal
ah (∃x·x ↦ arr(1)earr)
generalized MP
simplification rewrites
ProB
replay
Export
BPR (Rodin)
HTML Proof Tree (inspection)
New
Proof
JSON Trace
replay
Animation + Visualisation
fun_goal
mi
mi_pp
mon_deselect (arr ∈ 1 .. n → ℕ1)
mon_deselect (mxi ∈ 1 .. n)
mon_deselect (n ∈ ℕ1)
mon_deselect ((∀j·j ∈ 1 .. n ⇒ arr(j) ≤ arr(mxi)))
pp
prob_disprover
prob_wd_prover
rewrite_hyp(HypNr, NewHyp)
DEF_IN_FCT (arr ∈ ℤ ↔ ℤ)
DEF_IN_NATURAL1 (n ∈ ℕ1)
DEF_IN_TFCT (arr ∈ 1 .. n → ℕ1)
DEF_IN_UPTO (mxi ∈ 1 .. n)
arr ∈ 1 .. n → ℕ1
mxi ∈ 1 .. n
n ∈ ℕ1
(∀j·j ∈ 1 .. n ⇒ arr(j) ≤ arr(mxi))
arr ∈ ℤ ↔ ℤ
arr(1) ∈ ran(arr)

Proof Exports from ProB





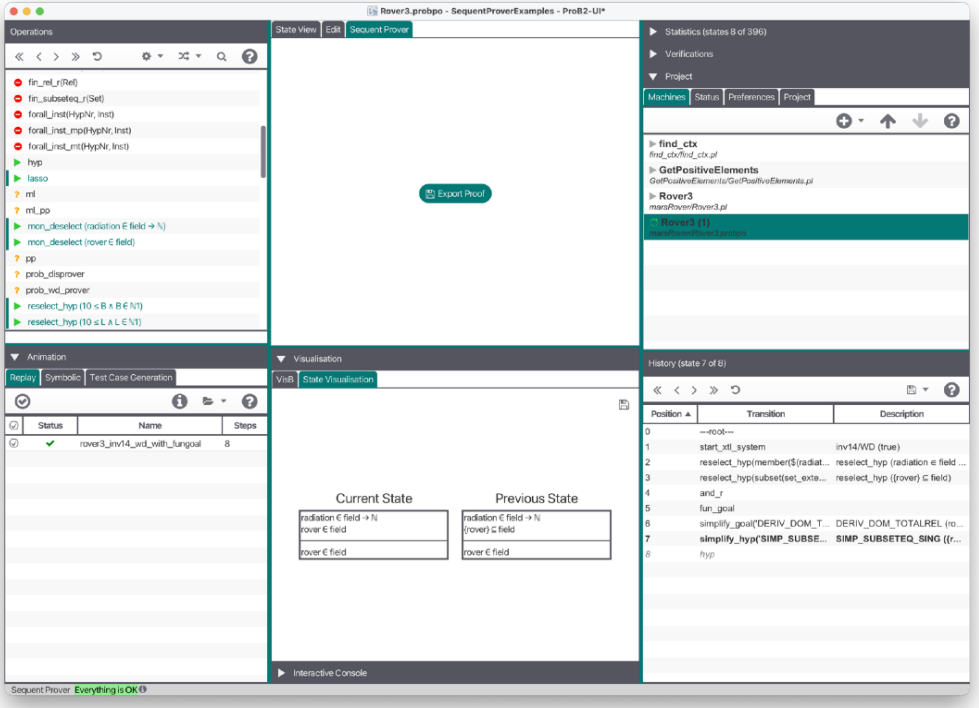


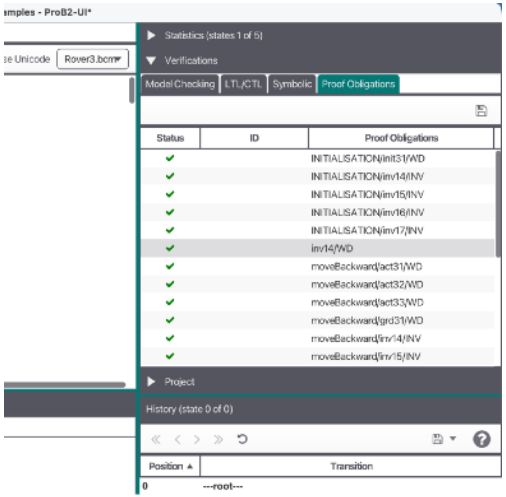
JSON Trace of Proof Steps
replay with ProB

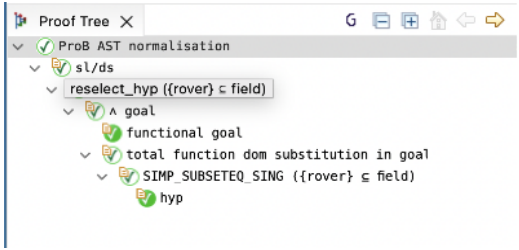
Interactive HTML Document
inspection of the proof tree

BPR Proof File
replay with Rodin
(2nd chain)

Demo: ProB2-UI



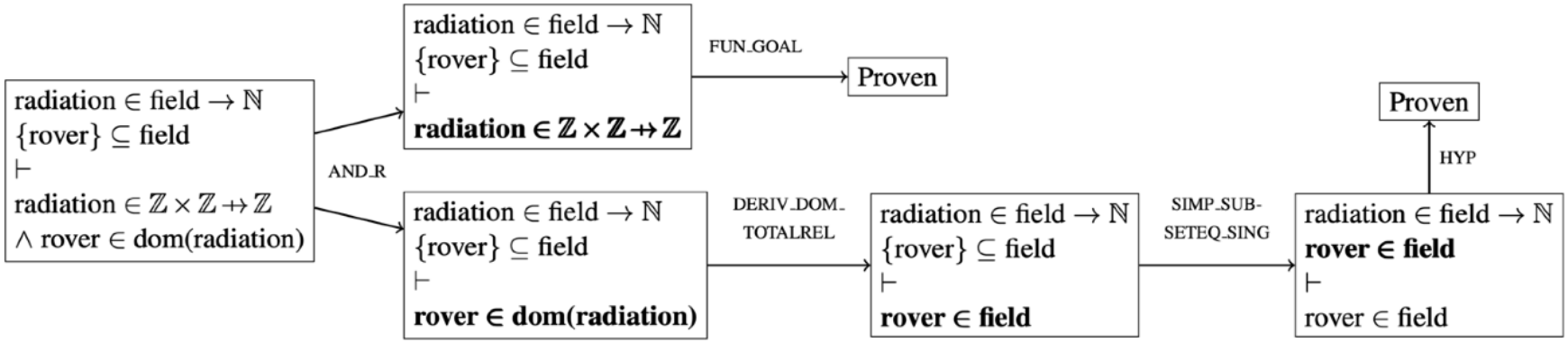




Demo: Mars-Rover

Heinrich Heine University Düsseldorf

FUN_GOAL $\dfrac{}{\mathbf{H},\ f \in E\ op\ F \ \vdash\ f \in T_1 \nrightarrow T_2}$



AND_R $\dfrac{\mathbf{H} \vdash \mathbf{P} \qquad \mathbf{H} \vdash \mathbf{Q}}{\mathbf{H} \vdash \mathbf{P} \wedge \mathbf{Q}}$

DERIV_DOM_TOTALREL $\mathrm{dom}(r) \mathrel{\widehat{=}} E$

SIMP_SUBSETEQ_SING $\{E\} \subseteq S \mathrel{\widehat{=}} E \in S$

Special Transitions

Rules with User Input:

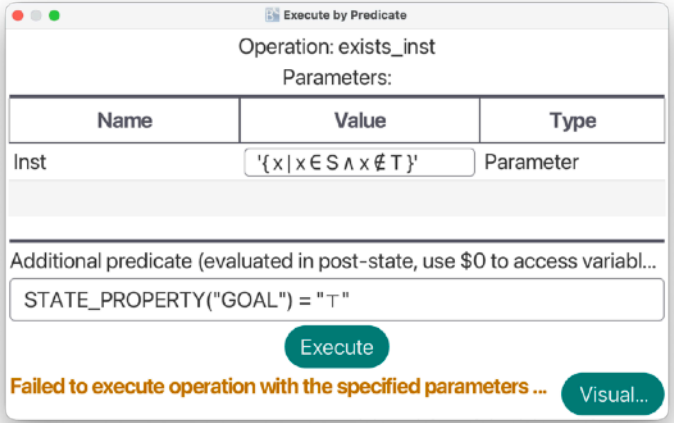
- „execute by predicate“ mechanism
  - transitions arguments provided by the user

Additional Provers:

- Symbolic transitions
  - can only be executed manually
  - call external provers
  - e.g. AtelierB ML/PP, ProB WD/Disprover

EXISTS_INST

$$\frac{\mathbf{H} \vdash WD(E) \qquad \mathbf{H} \vdash \mathbf{P}(E)}{\mathbf{H} \vdash \exists x \cdot \mathbf{P}(x)}$$



Prolog vs Java Encoding of Proof Rules

FIN_SETMINUS_R

$$\frac{\mathbf{H} \vdash \text{finite}(S)}{\mathbf{H} \vdash \text{finite}(S \setminus T)}$$

```
trans_desc_wo_info(
  fin_setminus_r,
  sequent(Hyps,finite(set_subtraction(S,_)),Cont),
  sequent(Hyps,finite(S)                   ,Cont),
  'finite of \'
).
```

```
public class FiniteSetMinus extends EmptyInputReasoner {

        public static final String REASONER_ID = SequentProver.PLUGIN_ID + ".finiteSetMinus";

        @Override
        public String getReasonerID() {
                return REASONER_ID;
        }

        @ProverRule("FIN_SETMINUS_R")
        protected IAntecedent[] getAntecedents(IProverSequent seq) {
                Predicate goal = seq.goal();

                // goal should have the form finite(S \ T)
                if (!Lib.isFinite(goal))
                        return null;
                SimplePredicate sPred = (SimplePredicate) goal;
                if (!Lib.isSetMinus(sPred.getExpression()))
                        return null;

                // There will be 1 antecidents
                IAntecedent[] antecidents = new IAntecedent[1];

                BinaryExpression aExp = (BinaryExpression) sPred.getExpression();

                Expression S = aExp.getLeft();
                final FormulaFactory ff = seq.getFormulaFactory();
                Predicate newGoal = ff.makeSimplePredicate(Predicate.KFINITE, S, null);

                antecidents[0] = ProverFactory.makeAntecedent(newGoal);
                return antecidents;
        }

        protected String getDisplayName() {
                return "finite of \ ";
        }

        @Override
        public IReasonerOutput apply(IProverSequent seq, IReasonerInput input,
                        IProofMonitor pm) {
                IAntecedent[] antecidents = getAntecedents(seq);
                if (antecidents == null)
                        return ProverFactory.reasonerFailure(this, input,
                                        "Inference " + getReasonerID() + " is not applicable");

                // Generate the successful reasoner output
                return ProverFactory.makeProofRule(this, input, seq.goal(),
                                getDisplayName(), antecidents);
        }
}
```

Prolog vs Java Encoding of Proof Rules

Heinrich Heine University Düsseldorf

```
simp_rule_with_hyps(domain(R),Dom,'DERIV_DOM_TOTALREL',Hyps) :-
    member(member(R,RType),Hyps), is_rel(RType,total,Dom,_).
simp_rule_with_hyps(range(R),Ran,'DERIV_RAN_SURJREL',Hyps) :-
    member(member(R,RType),Hyps), is_surj(RType,_,Ran).
```

- Java (Rodin): class with more than 200 lines for first rule, second rule not implemented
- Prolog (ProB): add second rule with 1-2 lines of code

Tool	Language	Code for Sequent Prover				Missing Rules	
		Files	LOC	SLOC	Person Years	Rewrite	Inference
Rodin	Java	320	50182	27580	6.51	55/536	16/123
ProB	Prolog	4	4208	3705	0.79	12/536	8/123

The columns SLOC (Source Lines of Code) and the estimated development effort in person years were generated using David A. Wheeler's 'SLOCCount'.

Conclusion and Outlook

- Implemented almost all of Rodin proof rules in Prolog
  - new Sequent Prover mode of ProB
  - HTML: interactive proof tree export — also useful for teaching
  - BPR: proof replay in Rodin — use of ProB prover as second chain
- Ideas for future work:
  - continue and improve automated prover, integrate as Rodin Plugin?
  - PO generation with ProB
  - AI-guided proof
- More ideas, questions?

https://prob.hhu.de/w/index.php?title=Sequent_Prover
https://stups.hhu-hosting.de/models/sequent_prover