

Interactive Proving with ProB



Katharina Engels, **Jan Gruteser**, Michael Leuschel

12th Rodin Workshop
Düsseldorf, 10.06.2025

hhu Heinrich Heine
University
Düsseldorf 

prob.hhu.de
stups.hhu.de

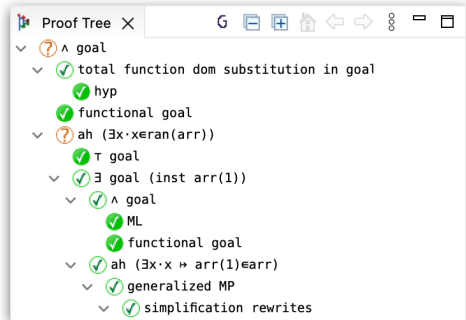
- Model checker, animator, and constraint solver
- High-level formal specification languages (like Event-B and TLA+)
- Tooling:
 - ▶ CLI
 - ▶ ProB Java API
 - ▶ **ProB Rodin Plugin**
 - ▶ **ProB Disprover**
 - ▶ ProB2-UI: VisB, SimB
- Developed by STUPS group at HHU Düsseldorf (open source)

1. Teaching

- In our course on Event-B modelling, we teach students how to prove proof obligations (POs)
- Rodin may not be a good choice for beginners
 - ▶ complex proof view
 - ▶ can be difficult to explore (and understand) single proof steps
- Idea: interactive proving with ProB together with visualisation + HTML export
 - ▶ Improve traceability and understanding of proofs
 - ▶ Comprehensible and explorable proof (tree)

2. Integration of Proof Rules in ProB (later)

- ProB already has a WD prover, which tries to prove as much as possible in a “big” step
- Idea: use the Prolog implementation of proof rules for (automatic?) proving of arbitrary POs in ProB



Export POs for ProB

Representation of
POs in Prolog

Core of this work

Proof Rules
in Prolog

ProB XTL Mode



Export



HTML Trace (*inspection*)



JSON Trace (*replay*)

Animation
+ Visualisation

$arr \in 1 \dots n \rightarrow \mathbb{N}_i$	$arr \in 1 \dots n \rightarrow \mathbb{N}_i$
$mxi \in 1 \dots n$	$mxi \in 1 \dots n$
$n \in \mathbb{N}_i$	$n \in \mathbb{N}_i$
$\forall j. (j \in 1 \dots n \Rightarrow arr(mxi) \geq arr(j))$	$\forall j. (j \in 1 \dots n \Rightarrow arr(mxi) \geq arr(j))$
$\top$	$1 \dots n \in FIN(1 \dots n)$

- contradict_1($\forall j. (j \in 1 \dots n \Rightarrow arr(mxi) \geq arr(j))$)
- contradict_1($arr \in 1 \dots n \rightarrow \mathbb{N}_i$)
- contradict_1($mxi \in 1 \dots n$)
- contradict_1($n \in \mathbb{N}_i$)
- contradict_r
- mon_deselect(1)
- mon_deselect(2)
- mon_deselect(3)
- mon_deselect(4)
- FIN_REL_RAN_R($ran(arr) \in FIN(ran(arr))$)
- DEF_IN_TFCT($arr \in 1 \dots n \rightarrow \mathbb{N}_i$)
- DEF_IN_UPTO($mxi \in 1 \dots n$)

1. Use Rodin to generate proof obligations (or create by hand)
2. Export them via `ProB Standalone > Export POs for ProB`
 - ▶ Prolog representation of all generated POs (originally for ProB Disprover)
3. Translate and normalise AST according to WD prover representation
 - ▶ Ignore position information
 - ▶ Allow later integration into ProB core

```
[member('$'d',natural_set),...,less_equal('$'n','$'d'),...]  
member(add('$'a',add('$'b','$'c')),natural_set)
```

thm1/THM

$d \in \mathbb{N}$
$n \leq d$
...

$c + b + a \in \mathbb{N}$

- We use the Rodin proof rules as a starting point

- ▶ Rewrite rules
- ▶ Inference rules

$$E \in \{F\} \hat{=} E = F \quad \text{SIMP_IN_SING}$$

$$\frac{H, P \vdash Q}{H \vdash P \Rightarrow Q} \quad \text{IMP_R}$$

- Turn mathematical definitions into executable Prolog rules that can be used in a prover:

```
trans(simplify_goal(Rule), sequent(Hyps,Goal,Cont), sequent(Hyps,NewGoal,Cont)) :-  
    simp_rule(Goal,NewGoal,Rule).
```

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

[...]

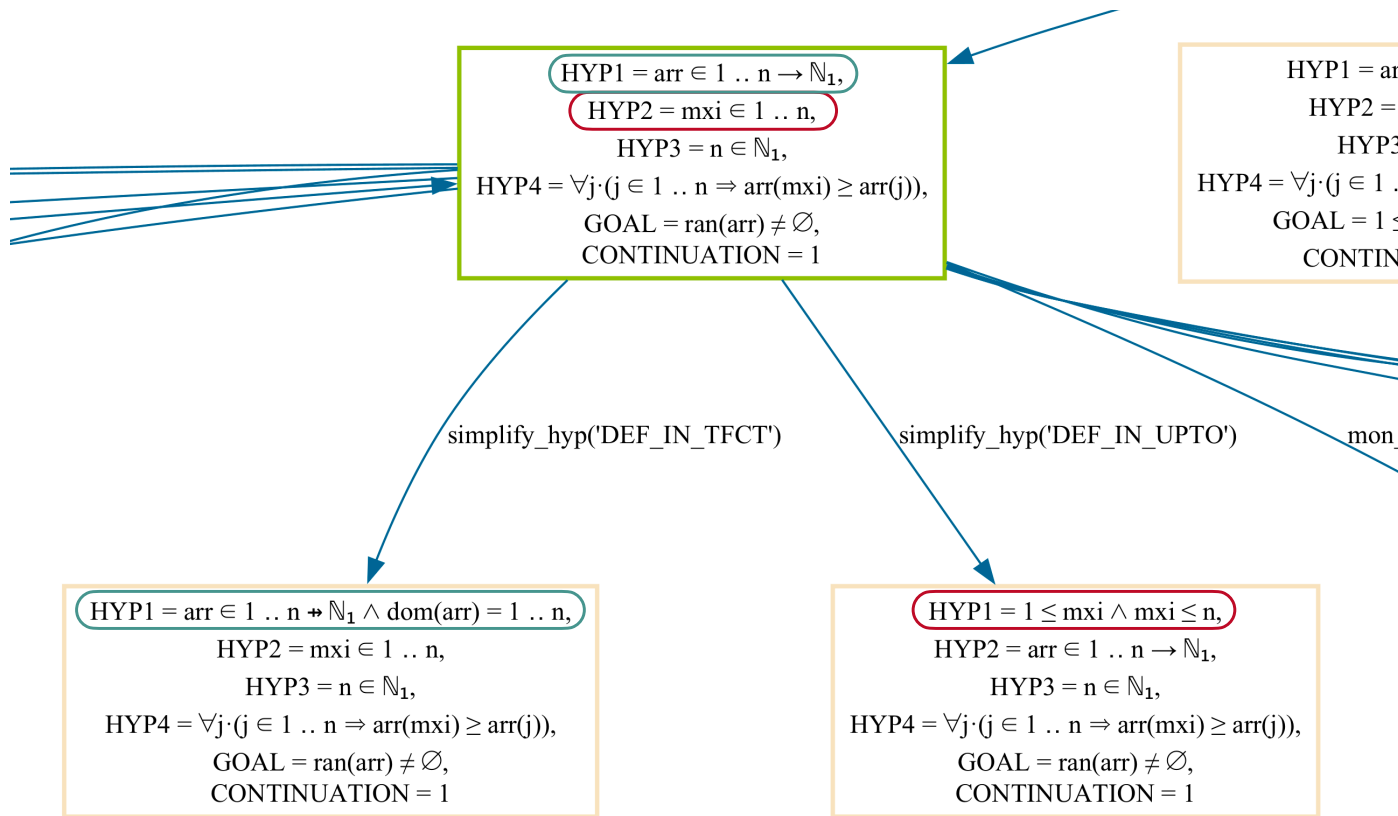
```
simp_rule(member(E,SetF), equal(E,F), 'SIMP_IN_SING') :-  
    singleton_set(SetF,F).
```

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

- ProB's XTL mode allows to *animate* the proof rules specified in Prolog
 - ▶ `start(InitialState)`
 - ▶ `trans(Label, StateBefore, StateAfter)`
- Labelled transition system:
 - ▶ Each PO is a start state, proof rules are transitions
 - ▶ The state is the current proof sequent:
`sequent([Hyps], Goal, Continuations)`
 - ▶ Continuations: branches of the proof tree that are not yet proven
- Based on the current sequent and the transition predicates, the ProB animator gives the applicable proof rules
 - ▶ we can provide more readable descriptions for transitions/rules

```
▶ INITIALISATION/inv5/INV
▶ ML_out/inv1/INV
▶ ML_out/inv4/INV
▶ ML_out/inv5/INV
▶ ML_out/grd1/GRD
▶ IL_in/inv1/INV
```

```
▶ contradict_l( $\forall j. j \in 1 \dots n \rightarrow \text{arr}(\text{mxi}) \geq \text{arr}(j)$ )
▶ contradict_l( $\text{arr} \in 1 \dots n \rightarrow \mathbb{N}_t$ )
▶ contradict_l( $\text{mxi} \in 1 \dots n$ )
▶ contradict_l( $n \in \mathbb{N}_t$ )
▶ contradict_r
▶ mon_deselect(1)
▶ mon_deselect(2)
▶ mon_deselect(3)
▶ mon_deselect(4)
▶ FIN_REL_RAN_R( $\text{ran}(\text{arr}) \in \text{FIN}(\text{ran}(\text{arr}))$ )
▶ DEF_IN_TFCT( $\text{arr} \in 1 \dots n \rightarrow \mathbb{N}_t$ )
▶ DEF_IN_UPTO( $\text{mxi} \in 1 \dots n$ )
```



ProB2-UI as an Interactive Prover

**Applicable
Proof Rules**

**Replay Traces
of Proofs**

sequent_prover.P - Sequent Prover - ProB 2.0

Operations: Filter State

Raw State

Name	Value	Previous Value
XTL Property 1	$XTL_range(\$arr), greater_equal(\$b, \$x)) success))$	$XTL_range(\$arr), greater_equal(\$b, \$x)) success))$
XTL Property 2	$HYP1=arr \in 1..n \rightarrow \%$	$HYP1=arr \in 1..n \rightarrow \%$
XTL Property 3	$HYP2=msl \in 1..n$	$HYP2=msl \in 1..n$
XTL Property 4	$HYP3=msl \in \%$	$HYP3=msl \in \%$
XTL Property 5	$HYP4=\forall j \in 1..n \rightarrow arr(mx) \geq arr(j)$	$HYP4=\forall j \in 1..n \rightarrow arr(mx) \geq arr(j)$
XTL Property 6	$GOAL=arr \in 1..n \rightarrow \%$	$GOAL=arr \in 1..n \rightarrow \%$
XTL Property 7	$CONTINUATION=3$	$CONTINUATION=2$
XTL Property 8		
XTL Property 9		
XTL Property 10		
XTL Property 11		
XTL Property 12		
XTL Property 13		
XTL Property 14		
XTL Property 15		
XTL Property 16		
XTL Property 17		
XTL Property 18		
XTL Property 19		
XTL Property 20		

Current State/Sequent

State Visualisation (+ Rule Selection by Click)

Current State

$$arr \in 1..n \rightarrow \%, msl \in 1..n, \forall j \in 1..n \rightarrow arr(mx) \geq arr(j), arr \in 1..n \rightarrow \%, msl \in \text{dom}(arr), ran(arr) \neq \emptyset, 3b \cdot (\forall x \in ran(arr) \rightarrow b \geq x)$$

Previous State

$$arr \in 1..n \rightarrow \%, msl \in 1..n, \forall j \in 1..n \rightarrow arr(mx) \geq arr(j), arr \in 1..n \rightarrow \%, msl \in \text{dom}(arr), ran(arr) \neq \emptyset, 3b \cdot (\forall x \in ran(arr) \rightarrow b \geq x)$$

Applied Proof Steps

Position	Transition	Descr...
0	---root---	
1	start_xt_system	SKS ...
2	and_r	
3	and_r	
4	and_r	
5	fun_goal(\$arr)	
6	simplify_hyp(DEF_IN_TFCT)	
7	and_f	
8	eq(tr, domain(\$arr), interval(1, \$n))	
9	hyp	
10	simplify_goal('SIMP_NOTEQUAL')	
11	simplify_goal('DEF_SPECIAL_NO...')	
12	exists_inst(function(\$arr, [1])	
13	simplify_goal('DEF_IN_RAN')	
14	exists_inst(1)	
15	simplify_goal('Evaluate tautology')	
16	true_goal	
17	true_goal	
18	and_r	

Everything is OK

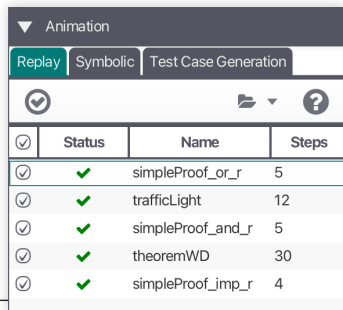
thm1/THM

- Simple example: thm1/THM
 - ▶ can be proven easily with EQL_LR for n , then HYP
- More advanced example:
 - ▶ 30 proof steps

$$\begin{array}{l} n = c + b + a \\ n \in \mathbb{N} \\ \dots \\ \hline c + b + a \in \mathbb{N} \end{array}$$

$$\begin{array}{l} \text{arr} \in 1 \dots n \rightarrow \mathbb{N}_1 \\ \text{mxi} \in 1 \dots n \\ n \in \mathbb{N}_1 \\ \forall j. (j \in 1 \dots n \Rightarrow \text{arr}(\text{mxi}) \geq \text{arr}(j)) \\ \hline \text{arr} \in 1 \dots n \rightarrow \mathbb{N}_1 \wedge \text{mxi} \in \text{dom}(\text{arr}) \wedge \text{ran}(\text{arr}) \neq \emptyset \wedge \exists b. (\forall x. (x \in \text{ran}(\text{arr}) \Rightarrow b \geq x)) \end{array}$$

- Enabled JSON Trace Export for ProB's XTL mode
- Save sequence of applied proof rules for
 - ▶ later replay of the proof
 - ▶ continuation of the work on a partial proof



```

arr ∈ 1 .. n → N1
mxi ∈ 1 .. n
n ∈ N1
∀j. (j ∈ 1 .. n ⇒ arr(mxi) ≥ arr(j))
-----
∃b. (∀x. (x ∈ ran(arr) ⇒ b ≥ x))
    
```

25. upper_bound_1

```

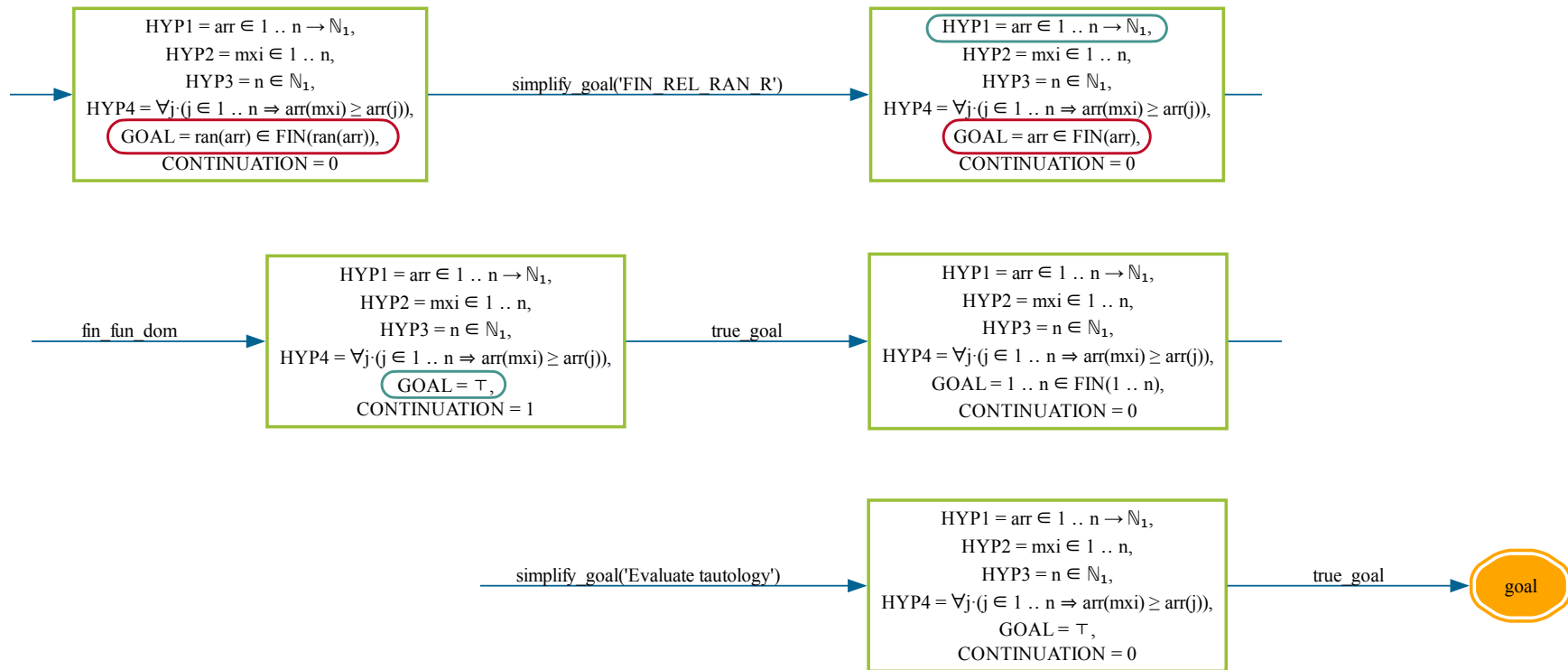
arr ∈ 1 .. n → N1
mxi ∈ 1 .. n
n ∈ N1
∀j. (j ∈ 1 .. n ⇒ arr(mxi) ≥ arr(j))
-----
ran(arr) ∈ FIN(ran(arr))
    
```

26. simplify_goal('FIN_REL_RAN_R')

```

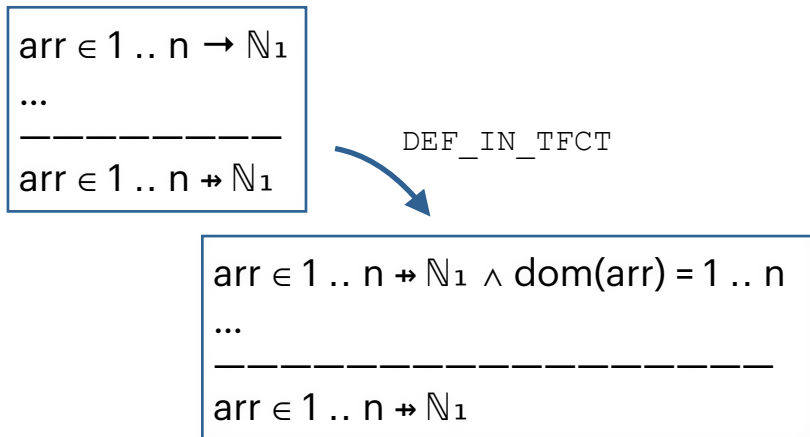
arr ∈ 1 .. n → N1
mxi ∈ 1 .. n
n ∈ N1
∀j. (j ∈ 1 .. n ⇒ arr(mxi) ≥ arr(j))
-----
arr ∈ FIN(arr)
    
```

- Human-readable HTML export of a trace together with visualisation
 - ▶ standalone document to inspect the proof



Again: Interactive Trace Replay

- Can be used to refactor a sequence of proof steps
 - ▶ Replace and skip rules / parts of the proof
- Example: replace FUN_GOAL by manual proof steps:



Step	Transition	Replayed Transition	Precision
1	start_xtl_system	start_xtl_system	Precise
2	and_r	and_r	Precise
3	and_r	and_r	Precise
4	and_r	and_r	Precise
4		simplify_hyp('DEF_IN_TFCT')	Manual
4		and_l()	Manual
4		hyp()	Manual
5	fun_goal	SKIP	
6	simplify_hyp	simplify_hyp	Precise
7	and_l	and_l	Precise
8	eq	eq	Precise
9	hyp		

- Implemented a subset of Rodin proof rules in Prolog
 - ▶ Simple POs generated by Rodin can already be proven
 - ▶ (Graph) visualisation and HTML export for traceability
 - ▶ *still in development*
- There are a lot of ideas for future work:
 - ▶ Implement more rules
 - ▶ Allow to recognise user input (add hypotheses, instantiations of $\exists, \forall$)
 - ▶ Allow to add custom proof rules
 - ▶ More advanced visualisation using VisB, improving user interaction
 - ▶ Apply automatic provers as proof step (as in Rodin), or even try model checking
- More ideas, inspirations, questions?