Interactive Trace Replay for Event-B Models

Jan Gruteser, Michael Leuschel

12th Rodin Workshop
Düsseldorf, 10.06.2025

Heinrich Heine
University
Düsseldorf

prob.hhu.de
stups.hhu.de

ProB

hhu
Heinrich Heine
University Düsseldorf

- Model checker, animator, and constraint solver
- High-level formal specification languages (like Event-B and TLA+)

- Tooling:
  - CLI
  - ProB Java API
  - ProB Rodin Plugin
  - ProB2-UI: VisB, SimB, *trace replay*

- Developed by STUPS group at HHU Düsseldorf (open source)
- Focus of this talk: **Trace Export and Replay**

https://prob.hhu.de

Motivation

- ProB allows to store and replay *traces* (automatically)
  - sequence of transitions
  - validate the model's behaviour at different stages of its development

- But: often breaking changes between model versions
  - e.g. renamed variables/events
  - Old traces can only be partially replayed (or not at all)
  - Refactoring/repairing traces can be hard for complex models

- Idea: instead of trying to replay everything automatically, let the user decide interactively in case of conflicts

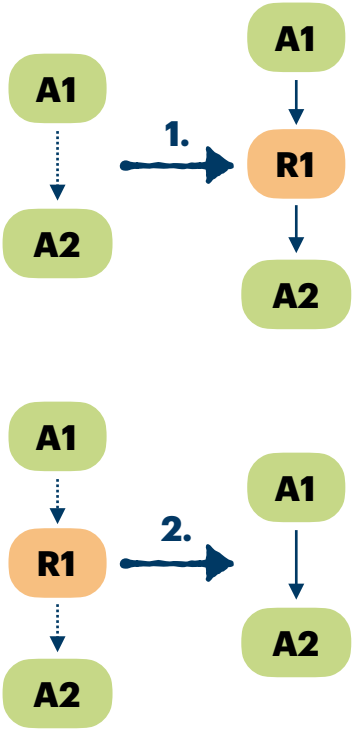
Position ▲	Transition
0	---root---
1	SETUP_CONSTANTS
2	INITIALISATION
3	close_elevator_door
4	move_to_floor(f=1)
5	open_elevator_door
6	close_elevator_door
7	move_to_floor(f=4)
8	open_elevator_door
9	close_elevator_door
10	move_to_floor(f=-1)
11	**open_elevator_door**

Use Cases

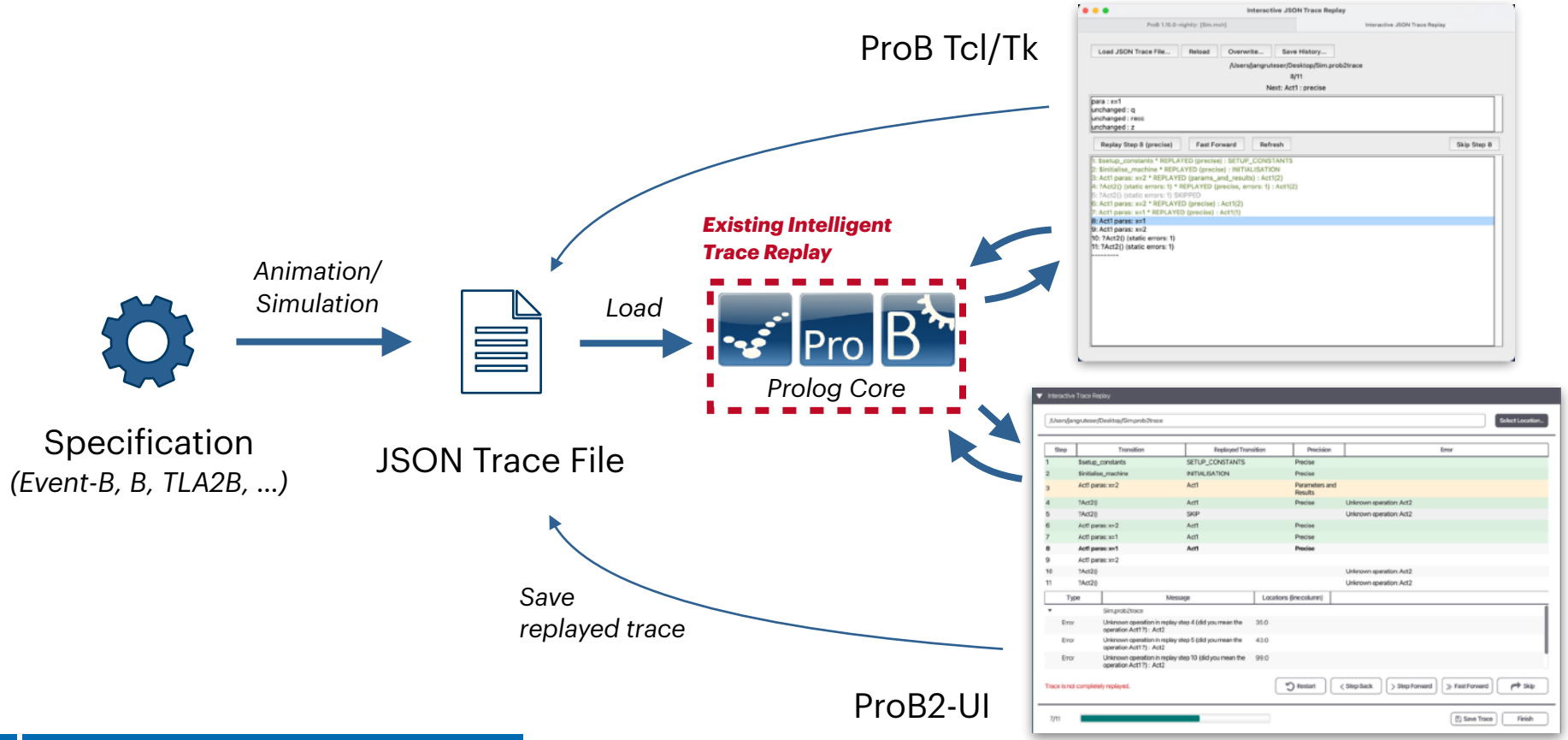
1. Refinement of Traces
   - Insert new trace steps between two abstract steps by manual animation of new or refined events

2. Abstraction of Traces
   - Skip steps with unavailable concrete events

3. Refactoring/Repairing of Traces
   - Refactor/repair traces of complex models after major changes



Overview


ProB Tcl/Tk
Existing Intelligent
Trace Replay
Animation/
Simulation
Load
Pro B
Prolog Core
Specification
(Event-B, B, TLA2B, ...)
JSON Trace File
Save
replayed trace
ProB2-UI

So far

Heinrich Heine University Düsseldorf

- "Intelligent" trace replay, automatic replay without user input
- Tries to resolve possible failure scenarios with different precision
  1. Perfect replay: *matching operation name, parameters and variable values after the trace step*
  2. Parameters: *matching operation name and parameters*
  3. Operation name: *matching operation name*
  4. Use another operation with the same effect (e.g. when an operation has been renamed)
  5. Skip steps where the original operation is unavailable

- This is not always possible.
- We combine the existing logic with the interactive replay
  - conflicts that cannot be handled can be resolved by the modeller
  - add / change / remove trace steps flexibly during replay

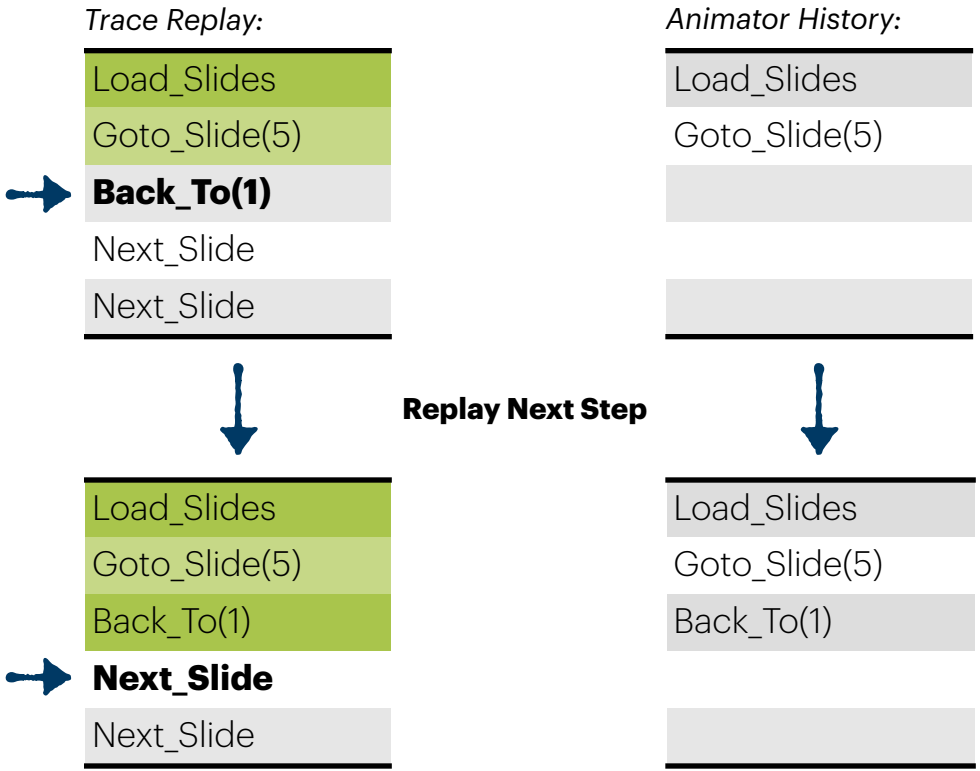
Current Control Options

Heinrich Heine University Düsseldorf

- **Two** states:
  - Current position in the replayed trace
  - State/history of the animator

- Based on this:
  - **Replay next step using a matching transition, if possible (selected by intelligent replay as before)**
  - "Fast Forward": automatic replay until the next step cannot be replayed (this is basically the previous implementation)
  - Skip the current trace step, always possible
  - Add manual animation steps anywhere in the trace using the animator
  - Undo the last replayed transition (from manual animation or replayed)

Current Control Options

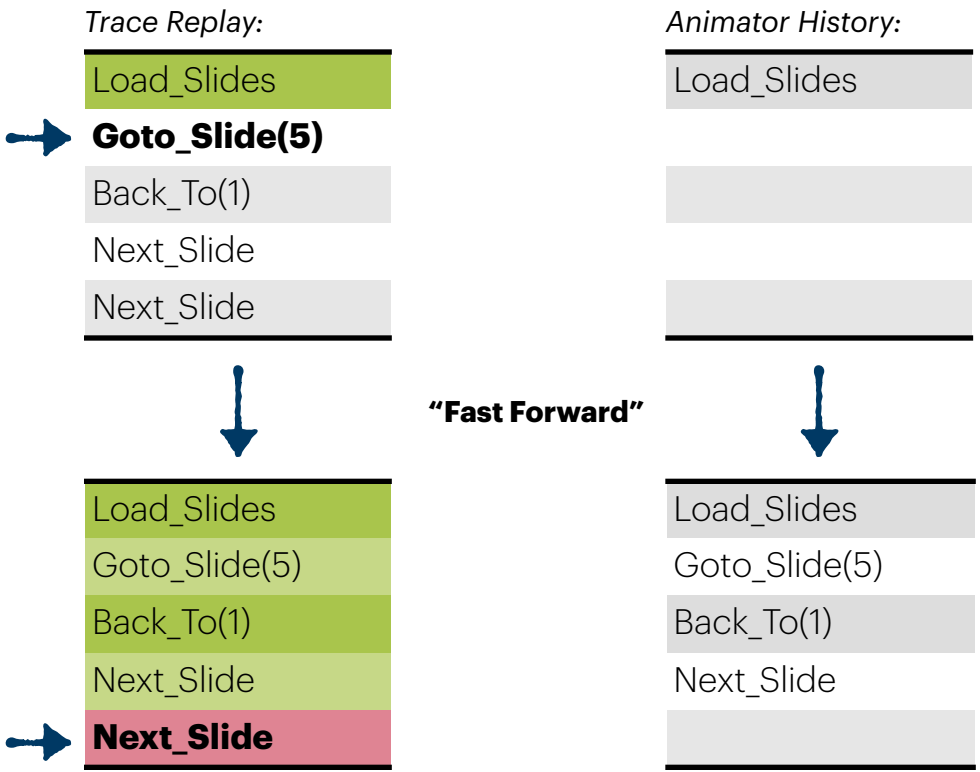


Current Control Options

- **Two** states:
  - Current position in the replayed trace
  - State/history of the animator

- Based on this:
  - Replay next step using a matching transition, if possible (selected by intelligent replay as before)
  - **"Fast Forward": automatic replay until the next step cannot be replayed (this is basically the previous implementation)**
  - Skip the current trace step, always possible
  - Add manual animation steps anywhere in the trace using the animator
  - Undo the last replayed transition (from manual animation or replayed)

Current Control Options

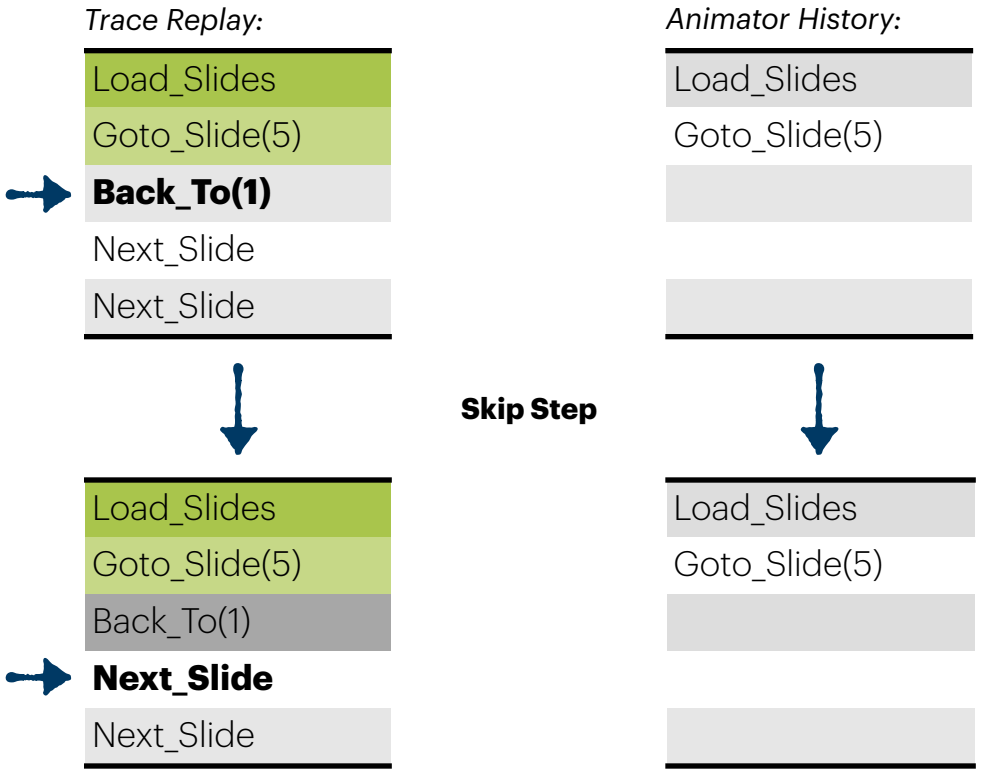


Current Control Options

- **Two** states:
  - Current position in the replayed trace
  - State/history of the animator

- Based on this:
  - Replay next step using a matching transition, if possible (selected by intelligent replay as before)
  - "Fast Forward": automatic replay until the next step cannot be replayed (this is basically the previous implementation)
  - **Skip the current trace step, always possible**
  - Add manual animation steps anywhere in the trace using the animator
  - Undo the last replayed transition (from manual animation or replayed)

Current Control Options

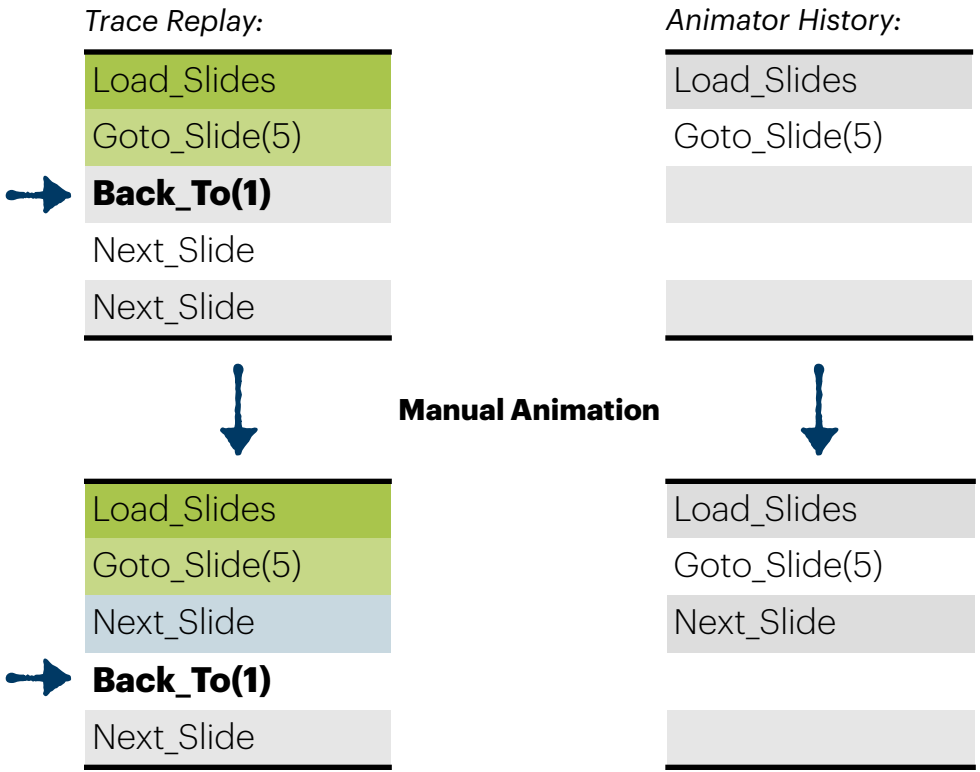


Current Control Options

- **Two** states:
  - Current position in the replayed trace
  - State/history of the animator

- Based on this:
  - Replay next step using a matching transition, if possible (selected by intelligent replay as before)
  - "Fast Forward": automatic replay until the next step cannot be replayed (this is basically the previous implementation)
  - Skip the current trace step, always possible
  - **Add manual animation steps anywhere in the trace using the animator**
  - Undo the last replayed transition (from manual animation or replayed)

Current Control Options

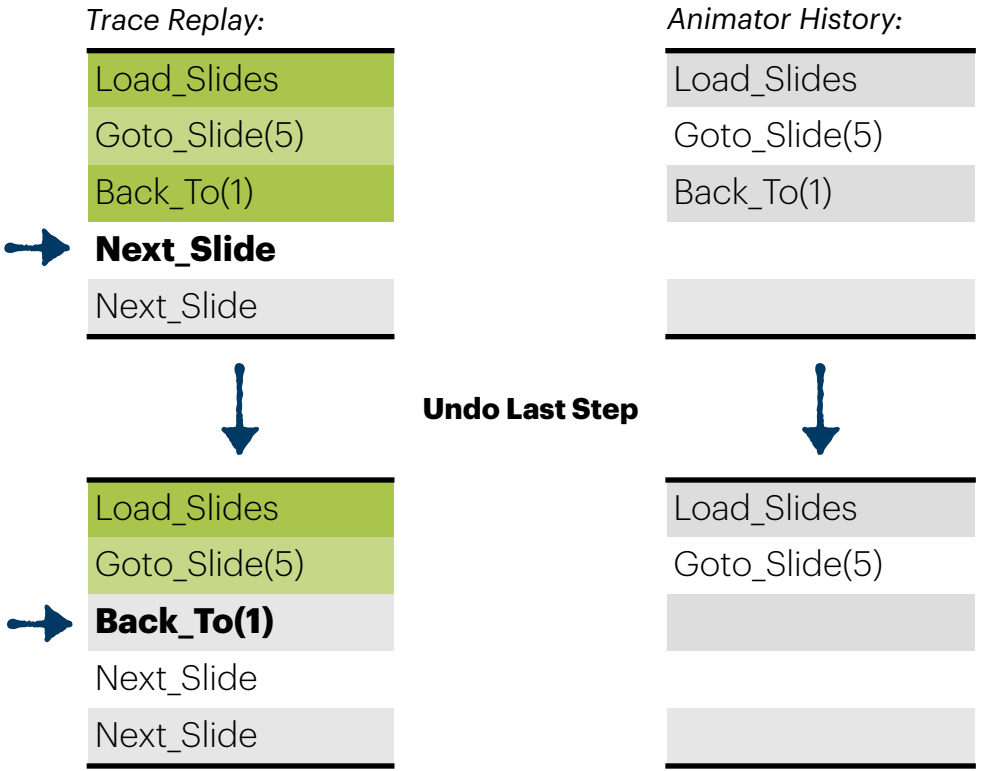


Current Control Options

- **Two** states:
  - Current position in the replayed trace
  - State/history of the animator

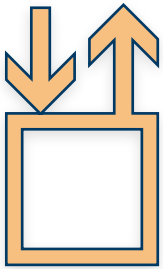
- Based on this:
  - Replay next step using a matching transition, if possible (selected by intelligent replay as before)
  - "Fast Forward": automatic replay until the next step cannot be replayed (this is basically the previous implementation)
  - Skip the current trace step, always possible
  - Add manual animation steps anywhere in the trace using the animator
  - **Undo the last replayed transition (from manual animation or replayed)**

Current Control Options



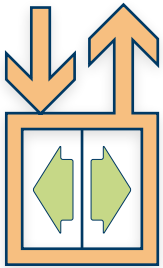
Example: Lift

- Model of a simple lift
- Consider abstraction and first refinement
  - Abstraction has only one event: `move_to_floor`
  - Refinement adds the cabin door: `close_door`, `open_door`



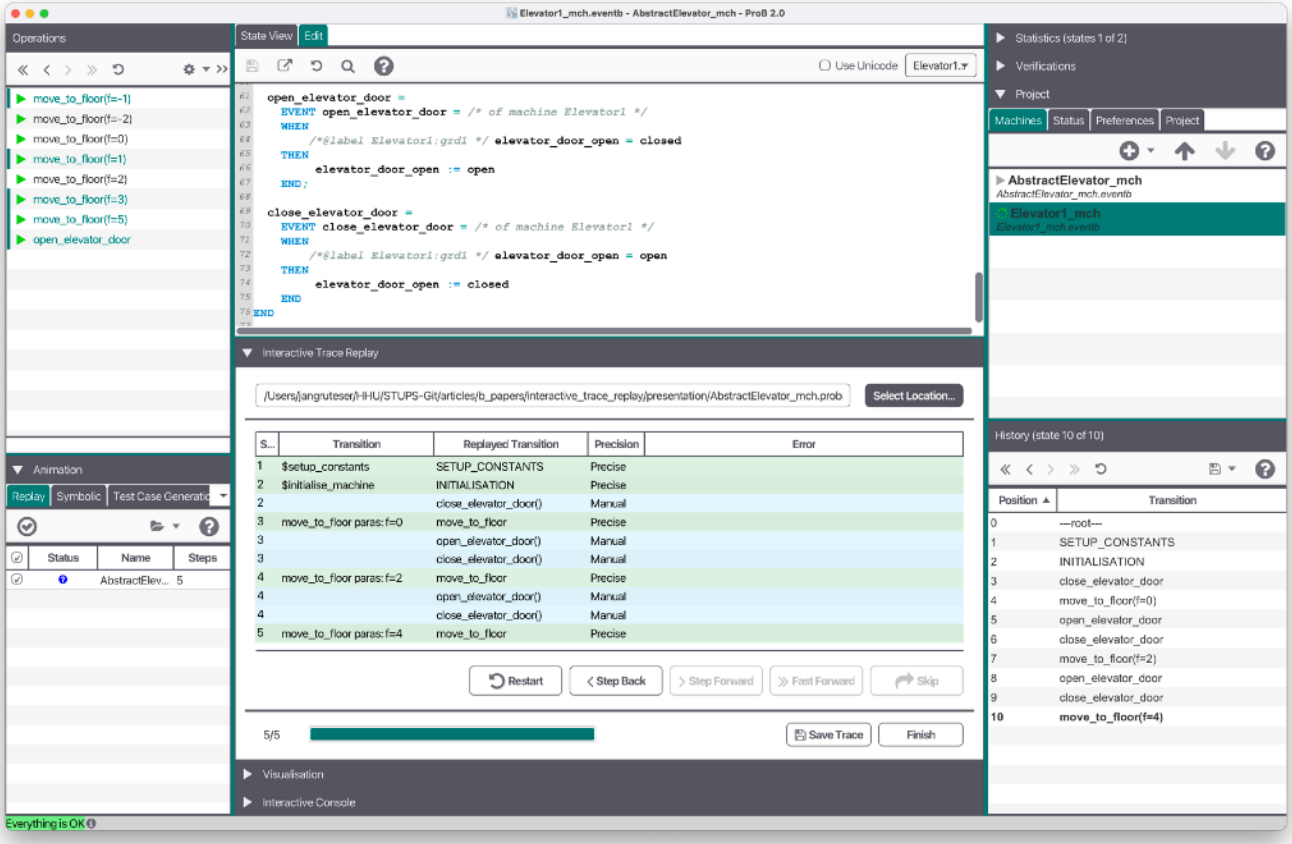
Lift0

- We want to replay a trace:
  - created for Lift0 on Lift1 (refinement of a trace)
  - created for Lift1 on Lift0 (abstraction of a trace)



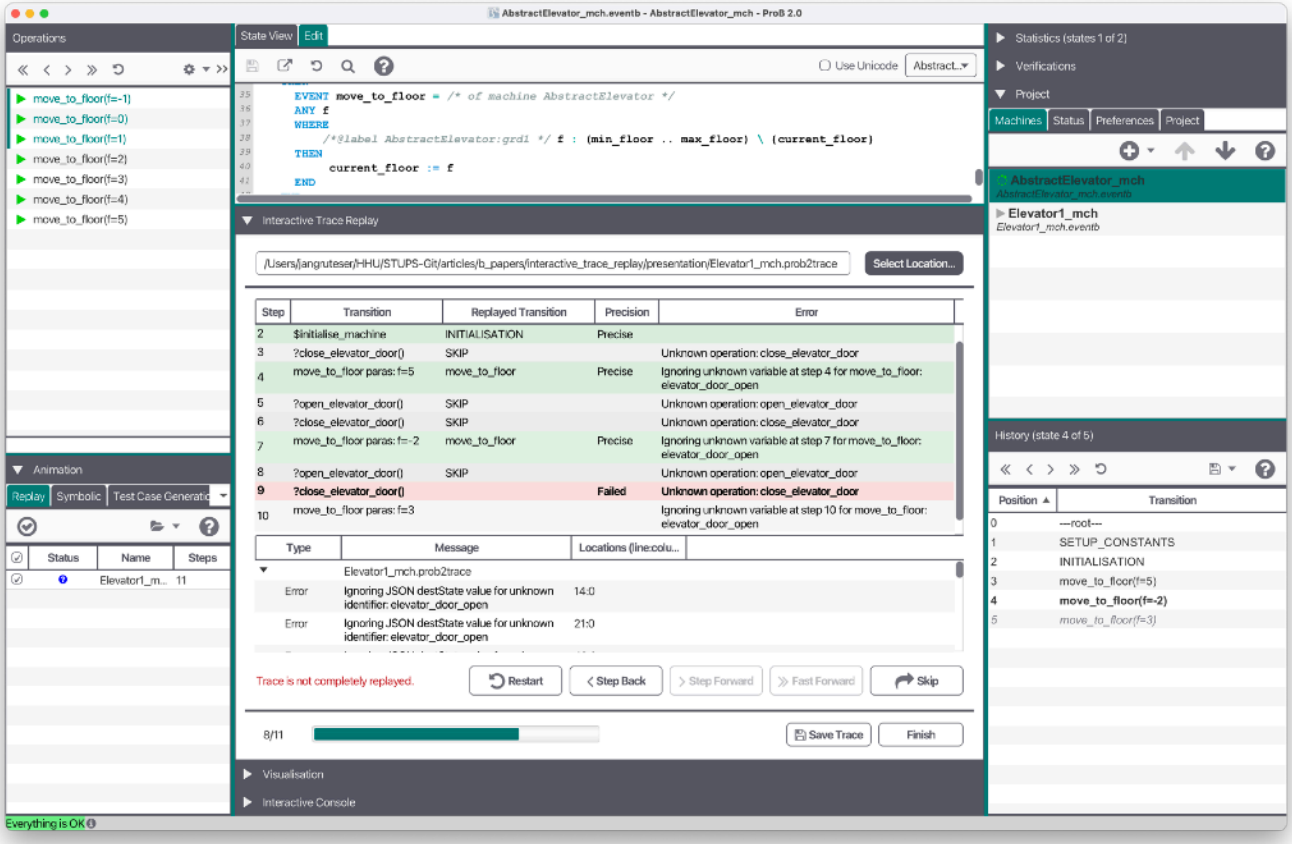
Lift1

ProB2-UI Example: Refinement of a Trace



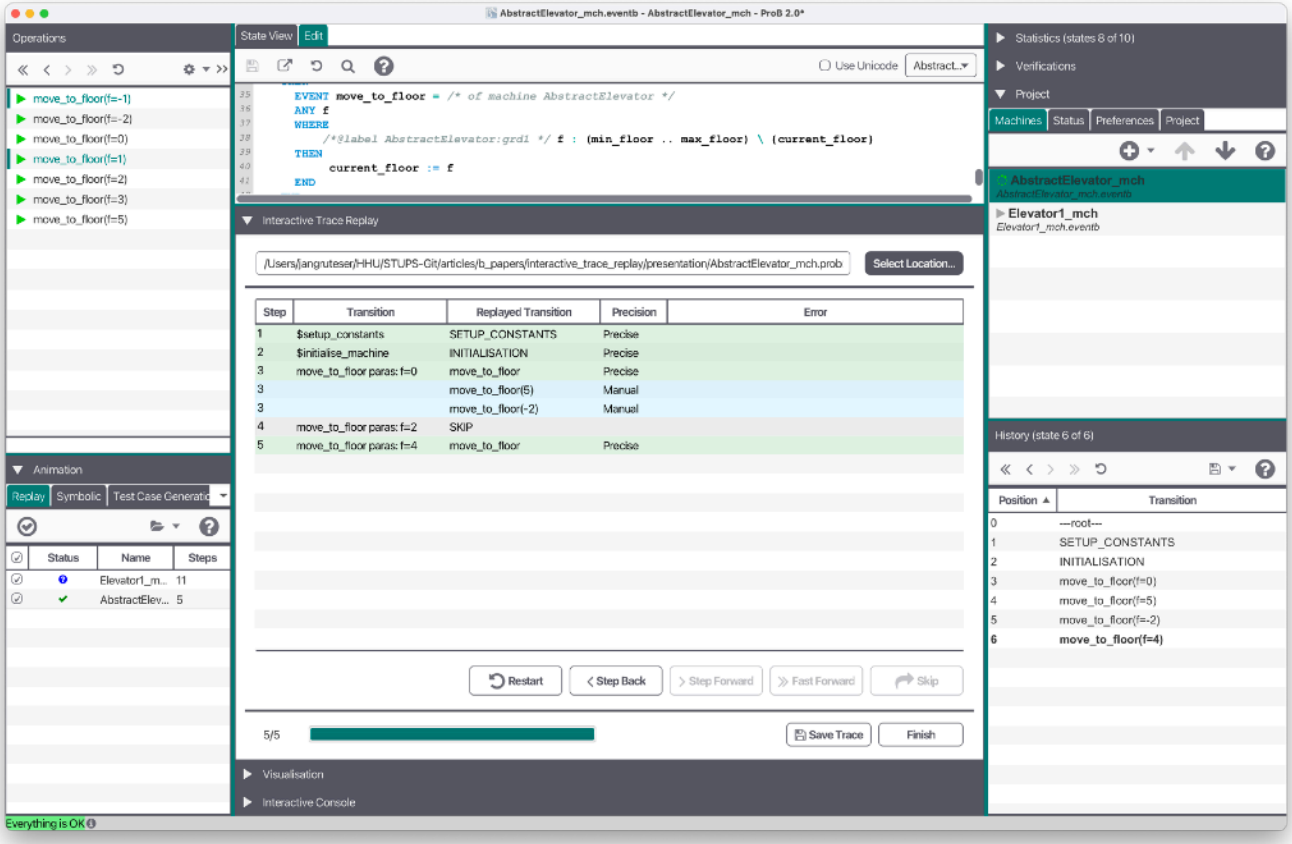
ProB2-UI Example: Abstraction of a Trace

Heinrich Heine University Düsseldorf



ProB2-UI Example: Refactoring of a Trace

Heinrich Heine University Düsseldorf



Conclusion

Heinrich Heine University Düsseldorf

- Interactive trace replay initially motivated by refactoring of trace for complex model
  - Also useful for further applications: abstraction and refinement of traces

- Future Plans:
  - Provide more options how the next transition should be selected
  - Allow the user to select the next replayed transitions explicitly
  - Replace not available operations in the entire trace with another one, e.g. if a operation has been renamed
  - Restrict the replay precision, e.g. allowing only precise replay

- Comments, questions? What other functionality/application would be interesting?