UNIVERSITY OF Southampton
School of Electronics and Computer Science

UML-B

diagram-based refinement, verification and validation

Dr. Colin Snook

Contact: cfs@soton.ac.uk

UNIVERSITY OF Southampton
School of Electronics and Computer Science

UML-B

- A formal (mathematical) diagrammatic modelling language
  - Class diagrams, State-machines
  - `UML-like' - but has a different semantics
    - No run to completion (no trigger events)
- Refinement
  - Abstract model – simple to prove critical properties
  - Refine in detail – proven to preserve abstract properties
- Based on formal language - Event-B
  - Sets, predicates, guarded events
  - Tools : Rodin modelling platform – automatic theorem provers

UNIVERSITY OF Southampton
School of Electronics and Computer Science

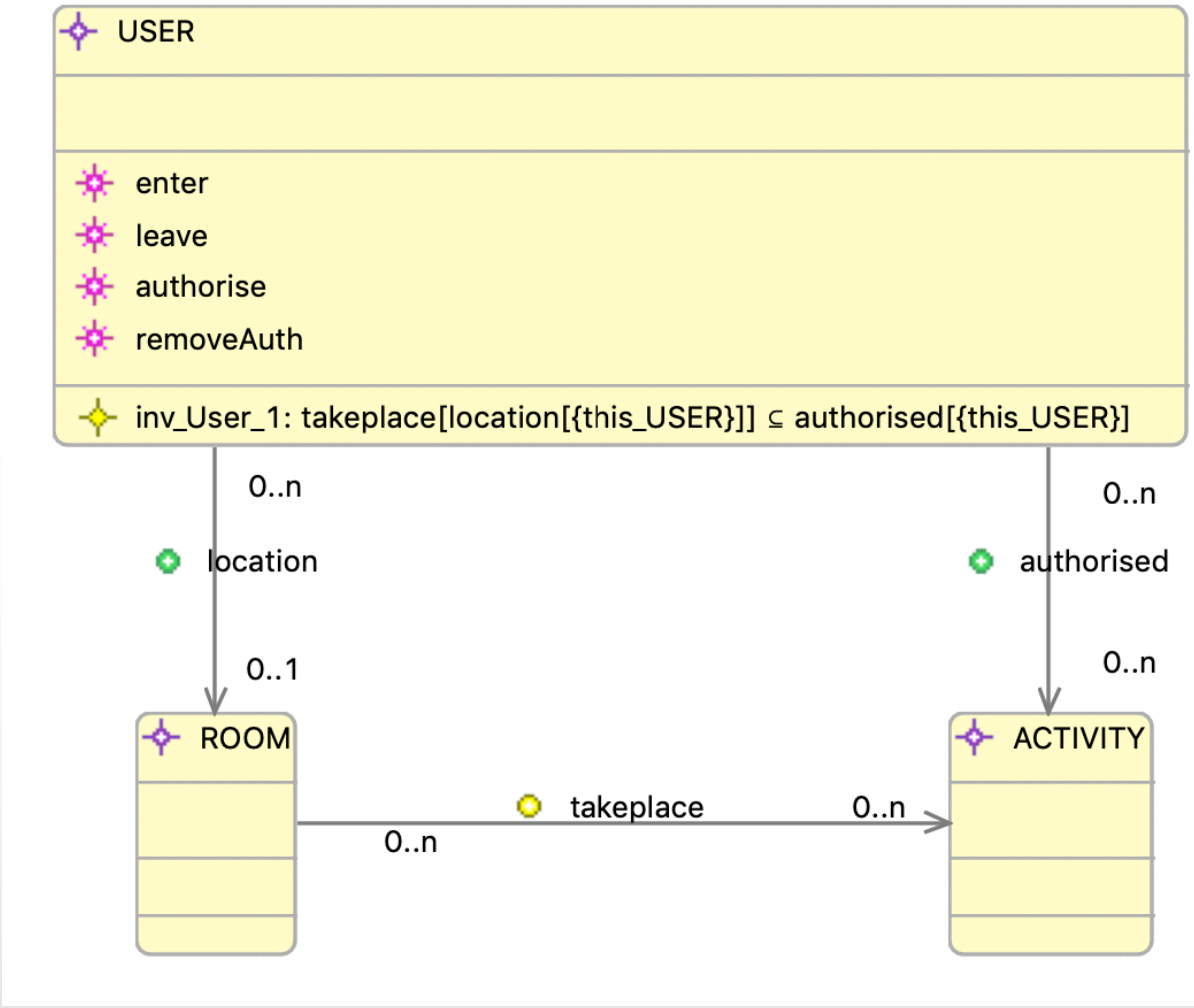
UML-B

- Verification by Proof
  - NO INSTANTIATION NEEDED
  - If not proven – use model checker to find counter-examples
- Validation
  - Animation + Visualisation
  - Scenarios

Example - Access Control

UNIVERSITY OF Southampton
School of Electronics and Computer Science

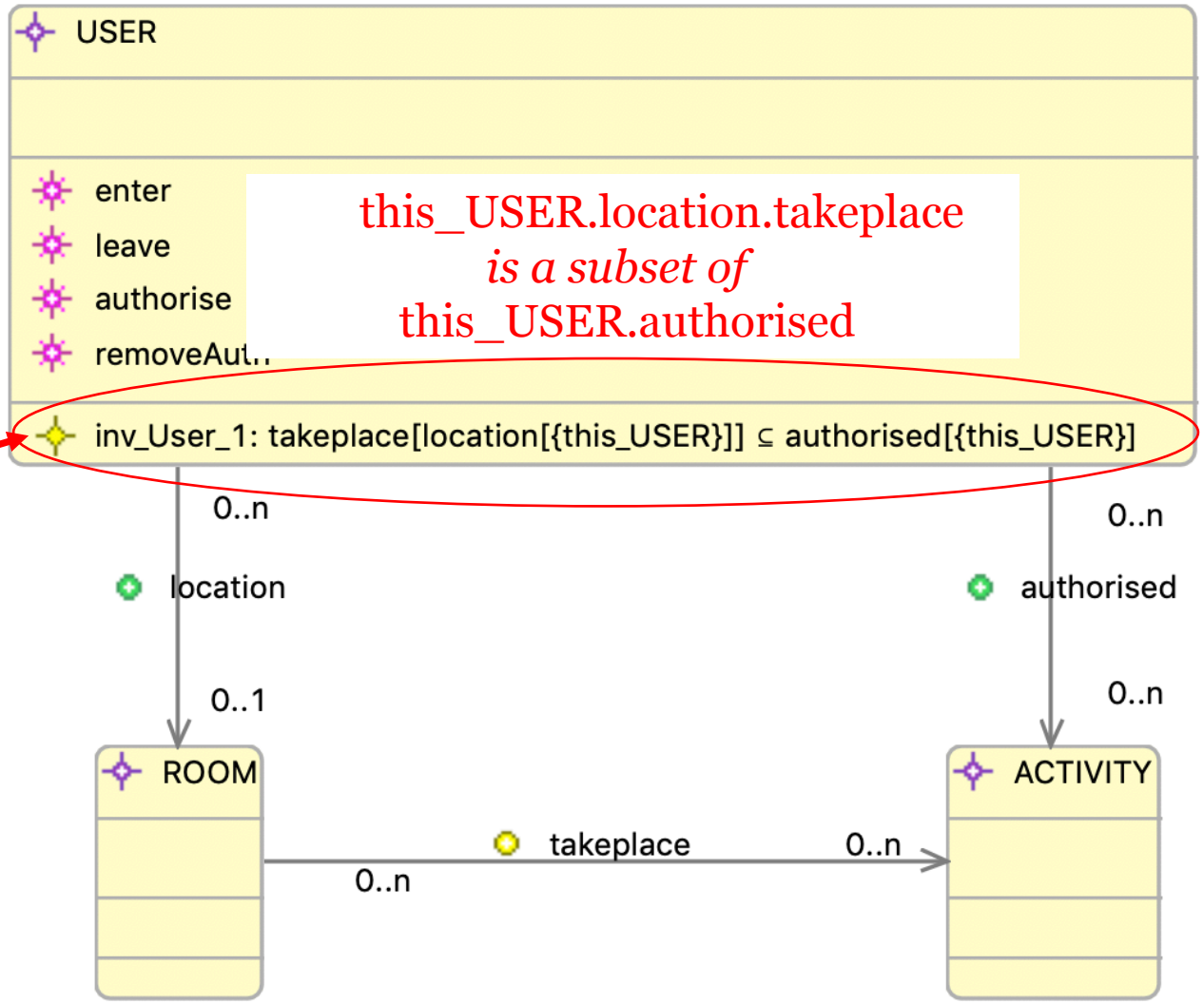
1. Activities take place in Rooms
2. Users are authorised to carry out activities
3. Users may enter Rooms only if they are authorised to carry out all the activities that take place in that room



Example - Access Control

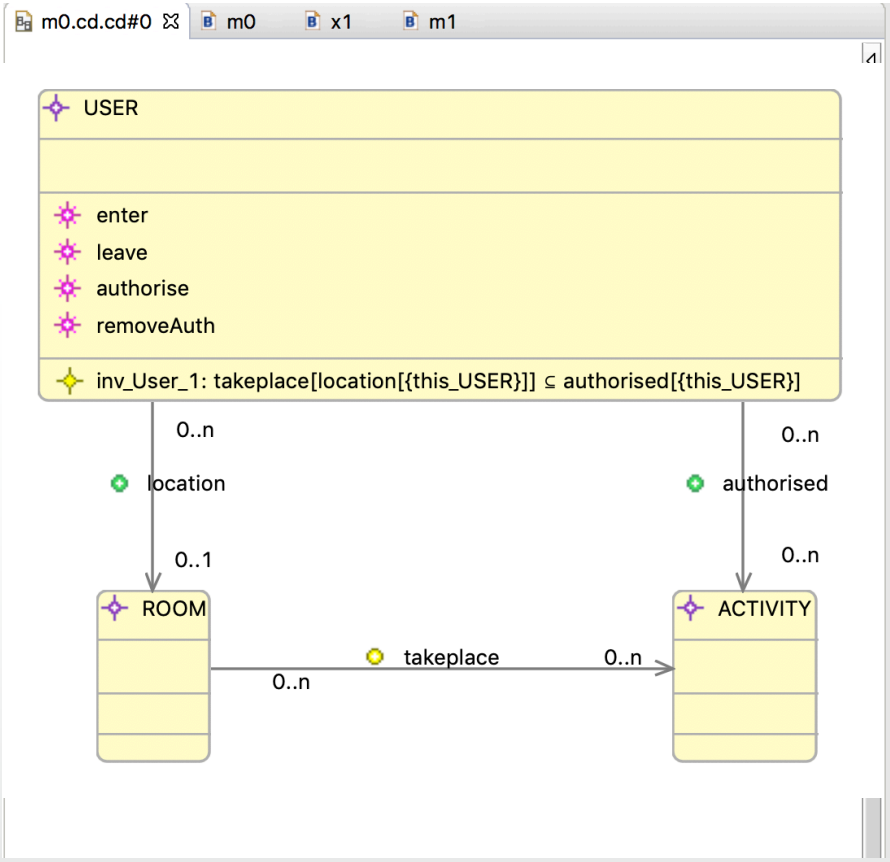
UNIVERSITY OF Southampton
School of Electronics and Computer Science

The security invariant



Example - Access Control

UNIVERSITY OF Southampton
School of Electronics and Computer Science



x0

```
CONTEXT
    x0 ›
SETS
○   USER ›
○   ROOM ›
○   ACTIVITY ›
CONSTANTS
○   takeplace ›
AXIOMS
○   assocMult_takeplace:  takeplace ∈ ROOM ↔ ACTIVITY not theorem ›
END
```

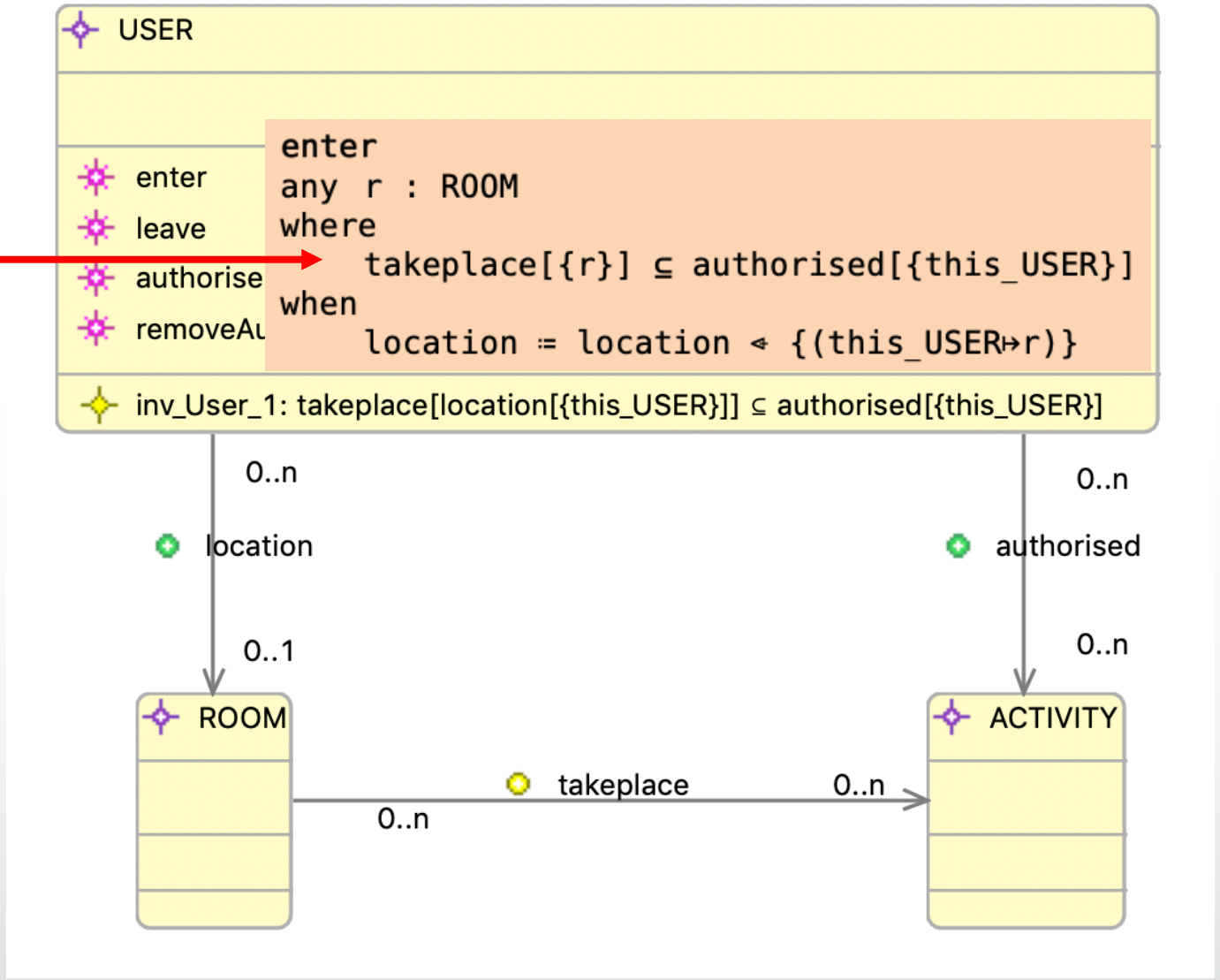
m0

```
MACHINE
    m0 ›
SEES
○   x0
VARIABLES
○   location ›
○   authorised ›
INVARIANTS
○   inv_User_1: ∀this_USER·this_USER∈USER ⇒ (takeplace[location[{this_U
○   assocMult_location: location ∈ USER ⇸ ROOM not theorem ›
○   assocMult_authorised:  authorised ∈ USER ↔ ACTIVITY not theorem ›
EVENTS
```

Example - Access Control

UNIVERSITY OF Southampton
School of Electronics and Computer Science

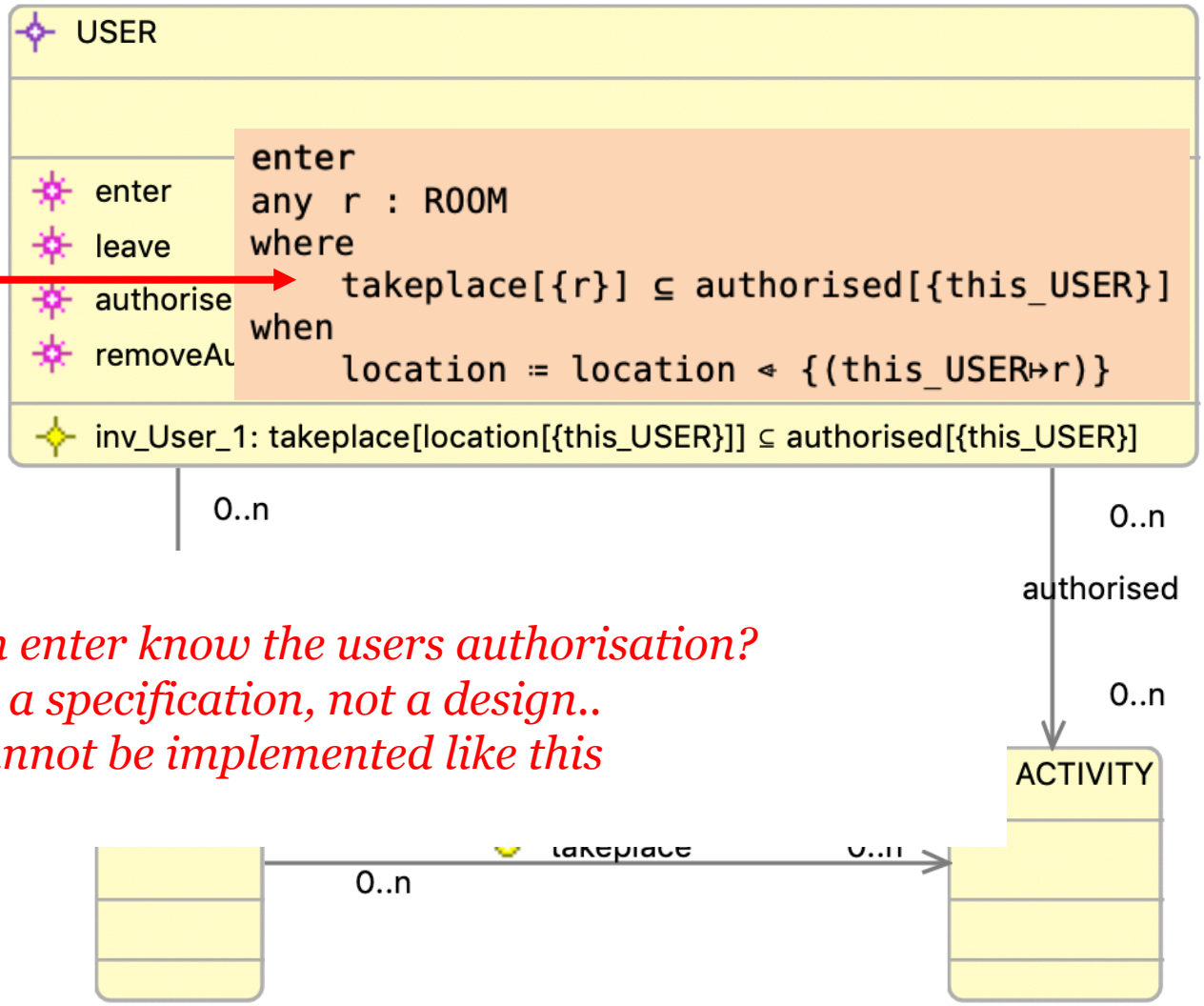
The guard achieves the security invariant…



Example - Access Control

UNIVERSITY OF Southampton
School of Electronics and Computer Science

The guard achieves the security invariant…



But how can enter know the users authorisation?
It is a specification, not a design..
… cannot be implemented like this

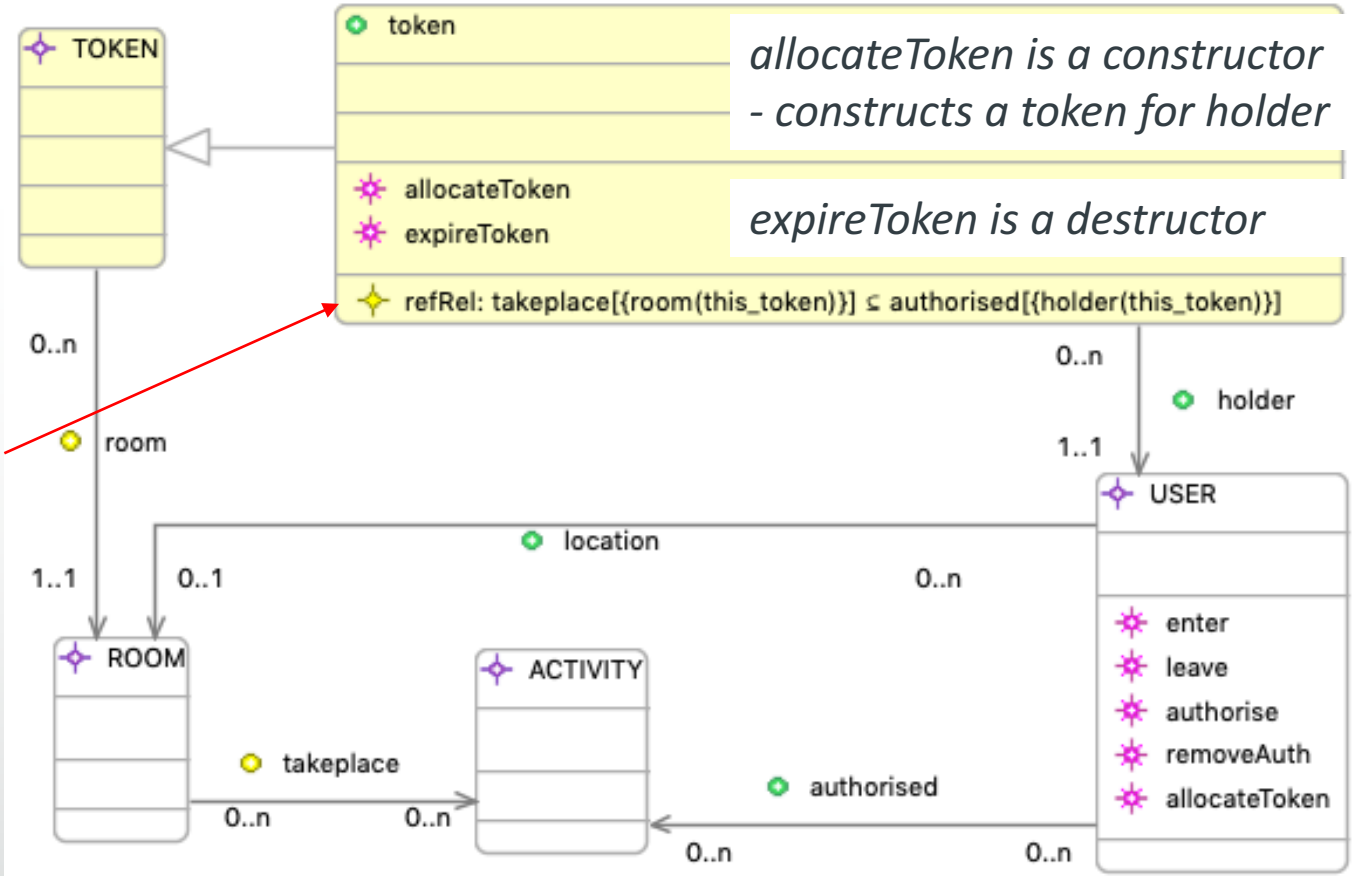
Example - Access Control

UNIVERSITY OF Southampton
School of Electronics and Computer Science

4. Users are allocated Tokens
5. Holding a Token enables access to Rooms
6. Tokens expire

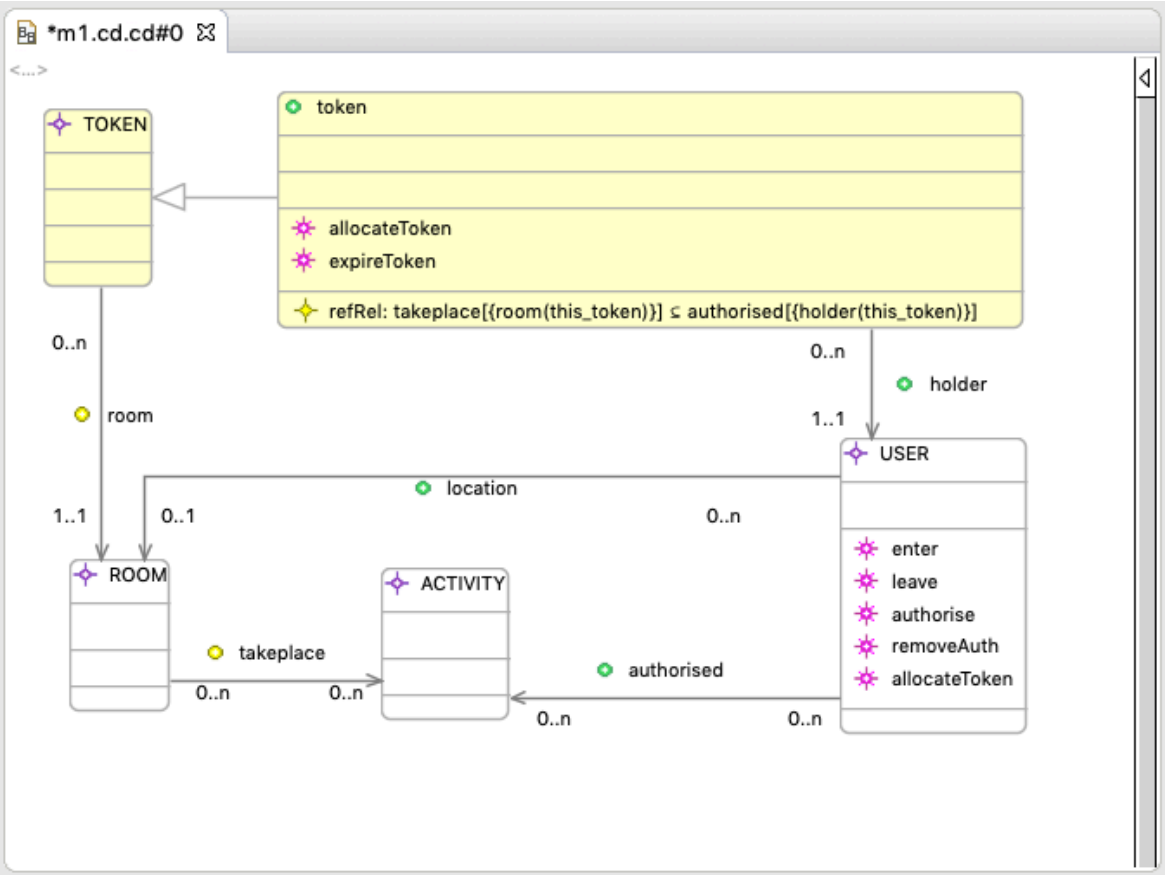
The design assumption.

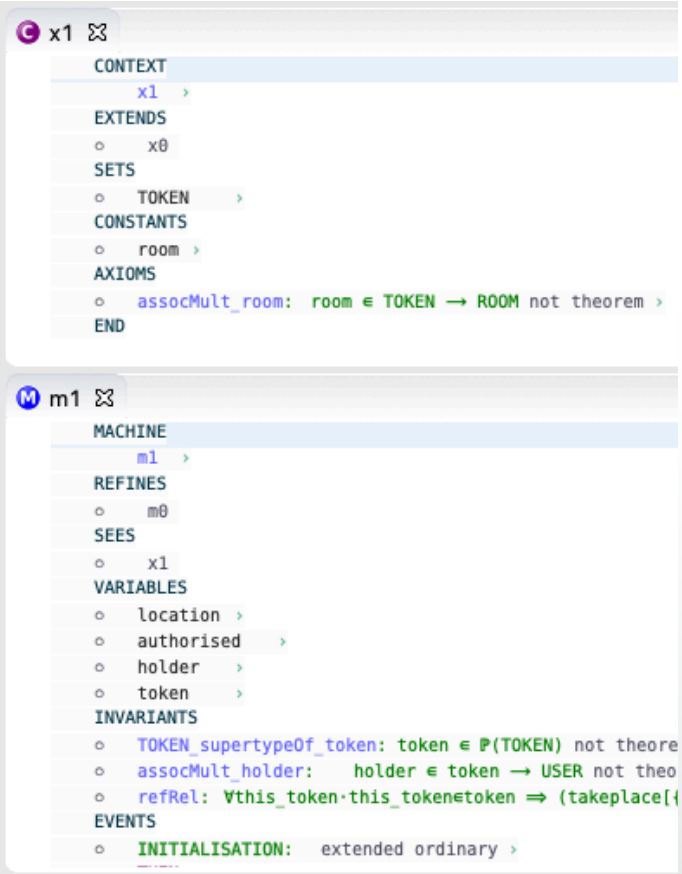
If true the refinement preserves the security invariant



Example - Access Control

UNIVERSITY OF Southampton
School of Electronics and Computer Science

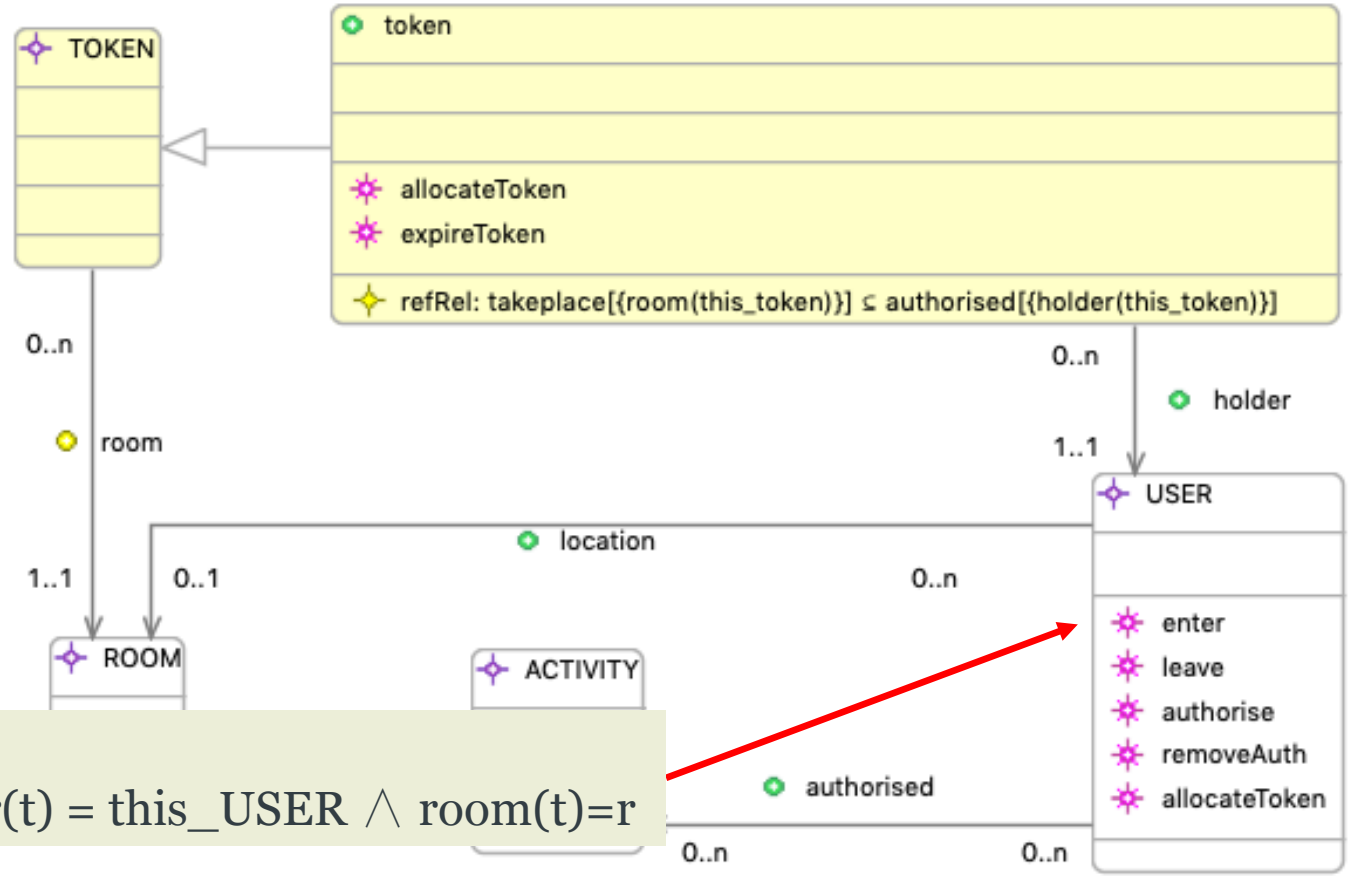




Example - Access Control

UNIVERSITY OF Southampton
School of Electronics and Computer Science

4. Users are allocated Tokens
5. Holding a Token enables access to Rooms
6. Tokens expire



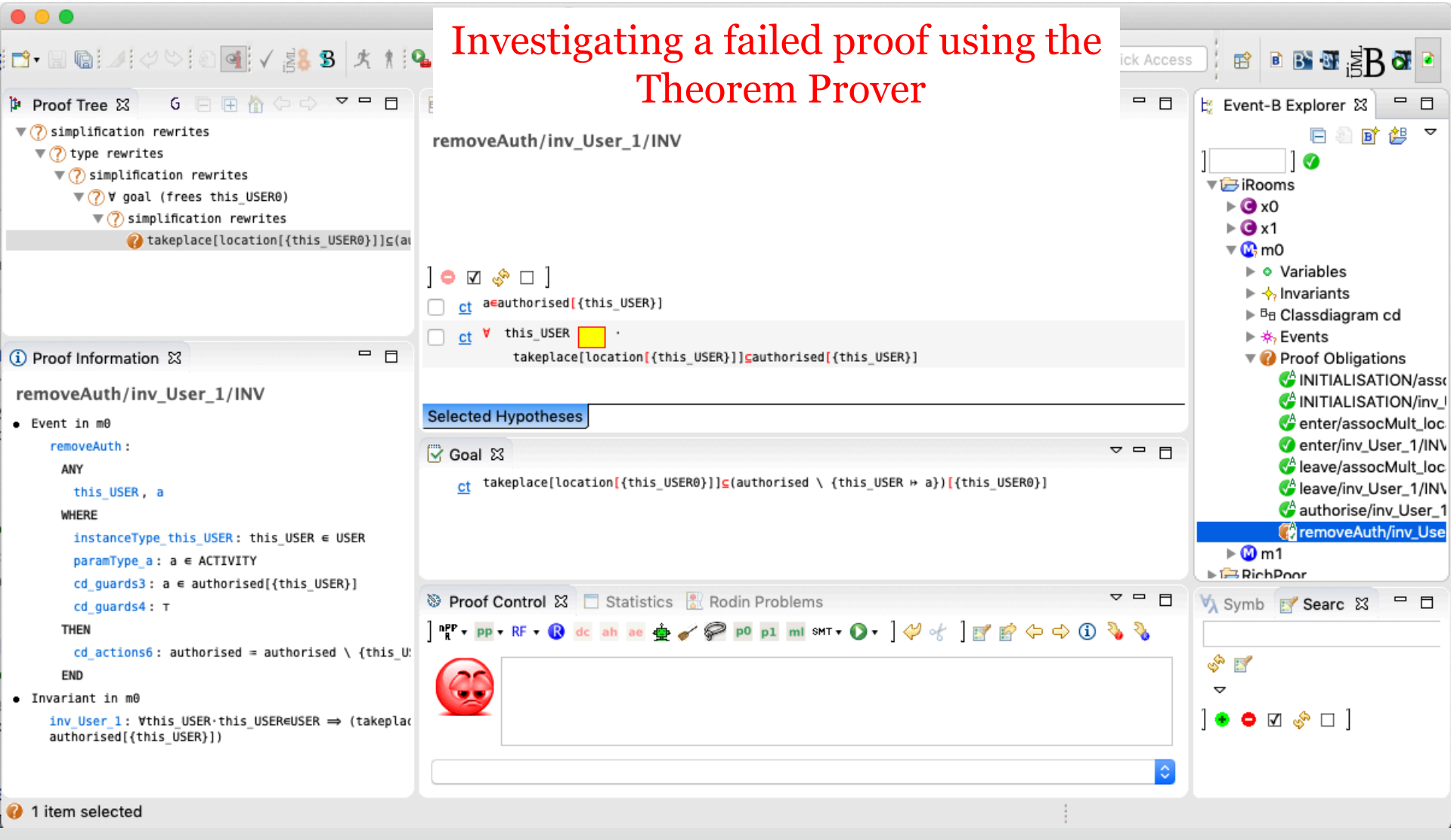
New guard of enter:

$\exists t \cdot t \in \text{token} \wedge \text{holder}(t) = \text{this_USER} \wedge \text{room}(t) = r$

Example - Access Control

UNIVERSITY OF Southampton
School of Electronics and Computer Science

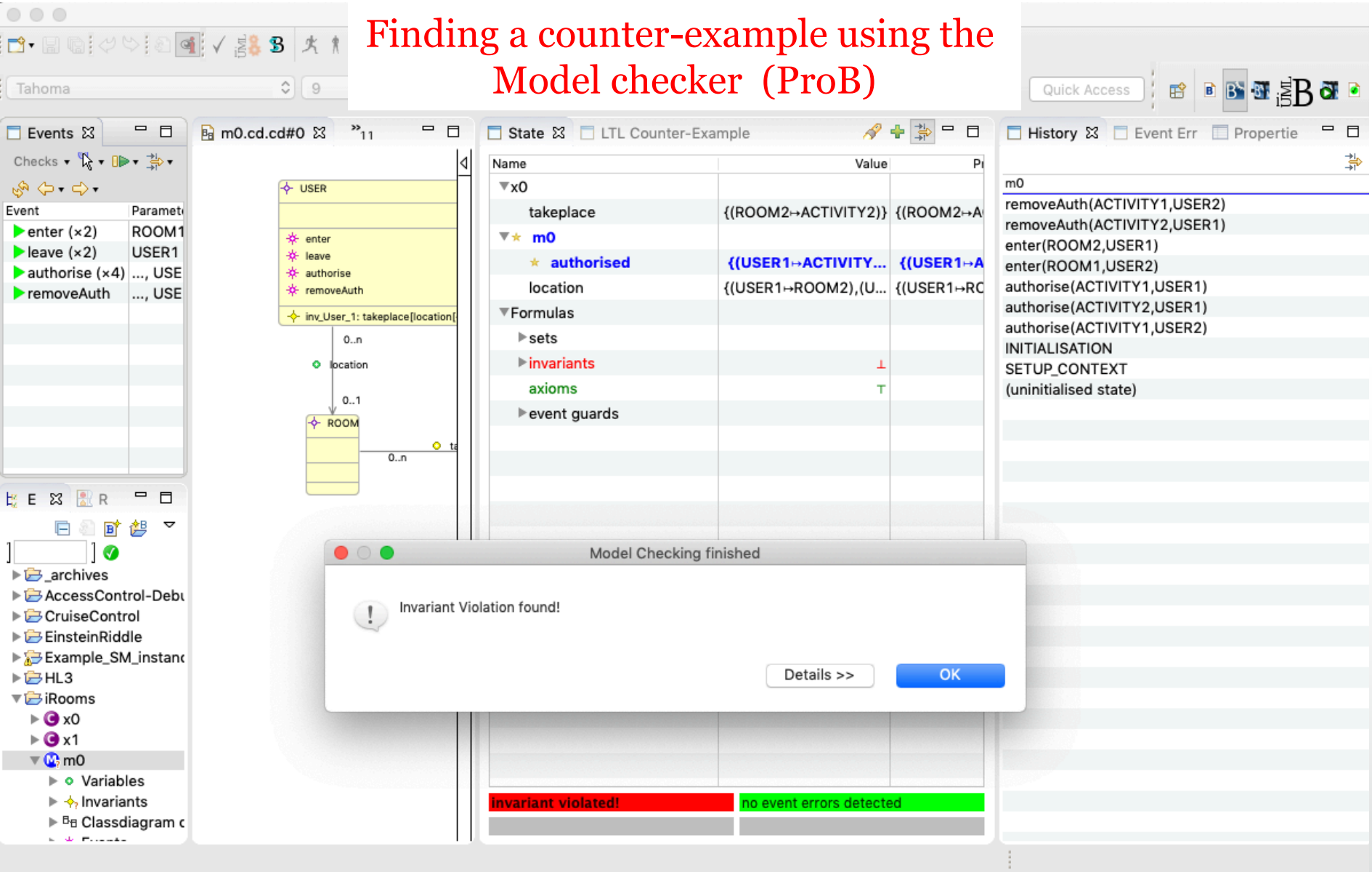
Investigating a failed proof using the Theorem Prover



Example - Access Control

UNIVERSITY OF Southampton
School of Electronics and Computer Science

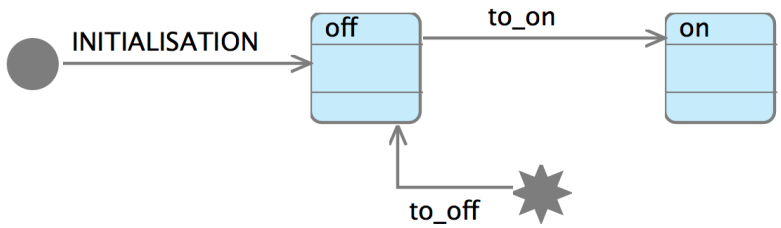
Finding a counter-example using the Model checker (ProB)



UNIVERSITY OF Southampton
School of Electronics and Computer Science

UML-B State-machines

m0.sm.smd#0 m1.sm.smd#0



m0 m1

```
MACHINE
    m0 ›
VARIABLES
○   off ›
○   on ›
INVARIANTS
○   typeof_off: off ∈ BOOL not theorem ›
○   typeof_on:  on ∈ BOOL not theorem ›
○   distinct_states_in_sm:  partition({TRUE}, {off} ∩ {TRUE}, {on} ∩ {TRUE})
EVENTS
○   INITIALISATION:  not extended ordinary ›
    THEN
    ○   init_off:   off ≔ TRUE ›
    ○   init_on:    on ≔ FALSE ›
    END

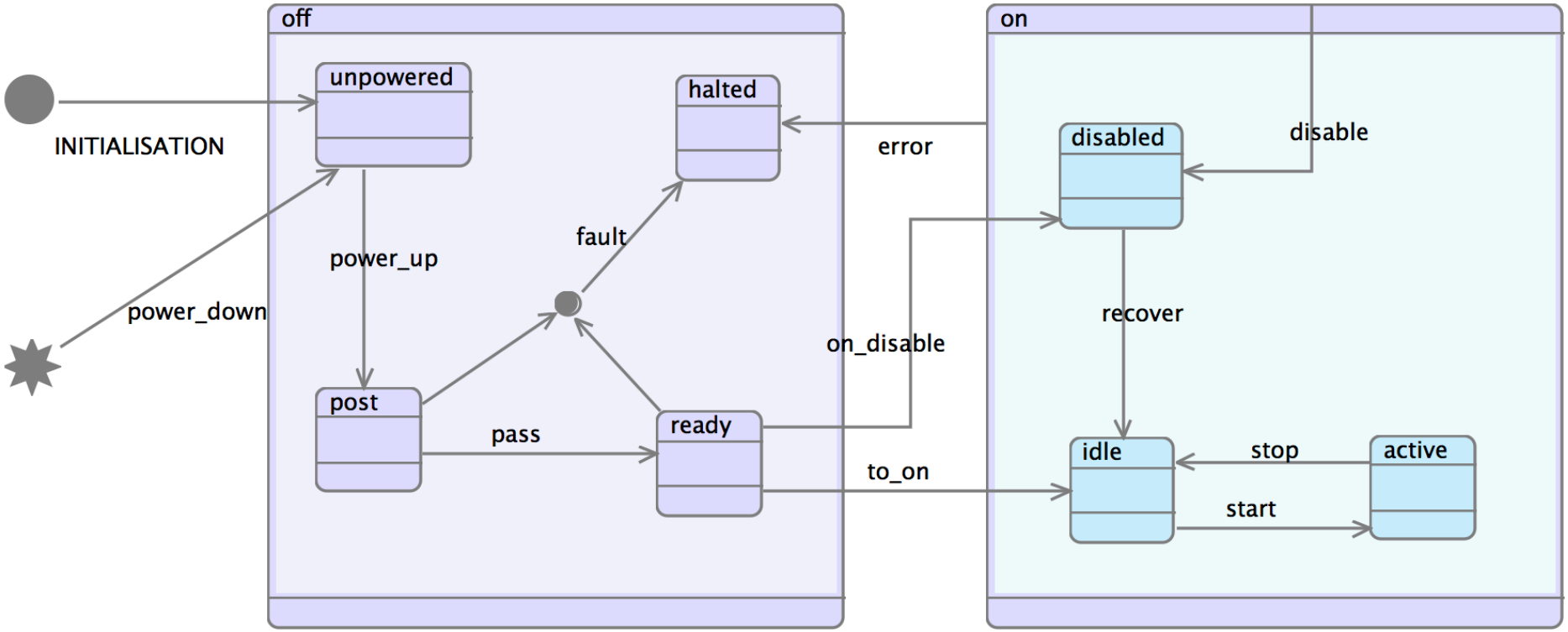
○   to_on:   not extended ordinary ›
    WHERE
    ○   isin_off:   off = TRUE not theorem ›
    THEN
    ○   leave_off:  off ≔ FALSE ›
    ○   enter_on:   on ≔ TRUE ›
    END

○   to_off:  not extended ordinary ›
    THEN
    ○   leave_on:   on ≔ FALSE ›
    ○   enter_off:  off ≔ TRUE ›
    END

END
```

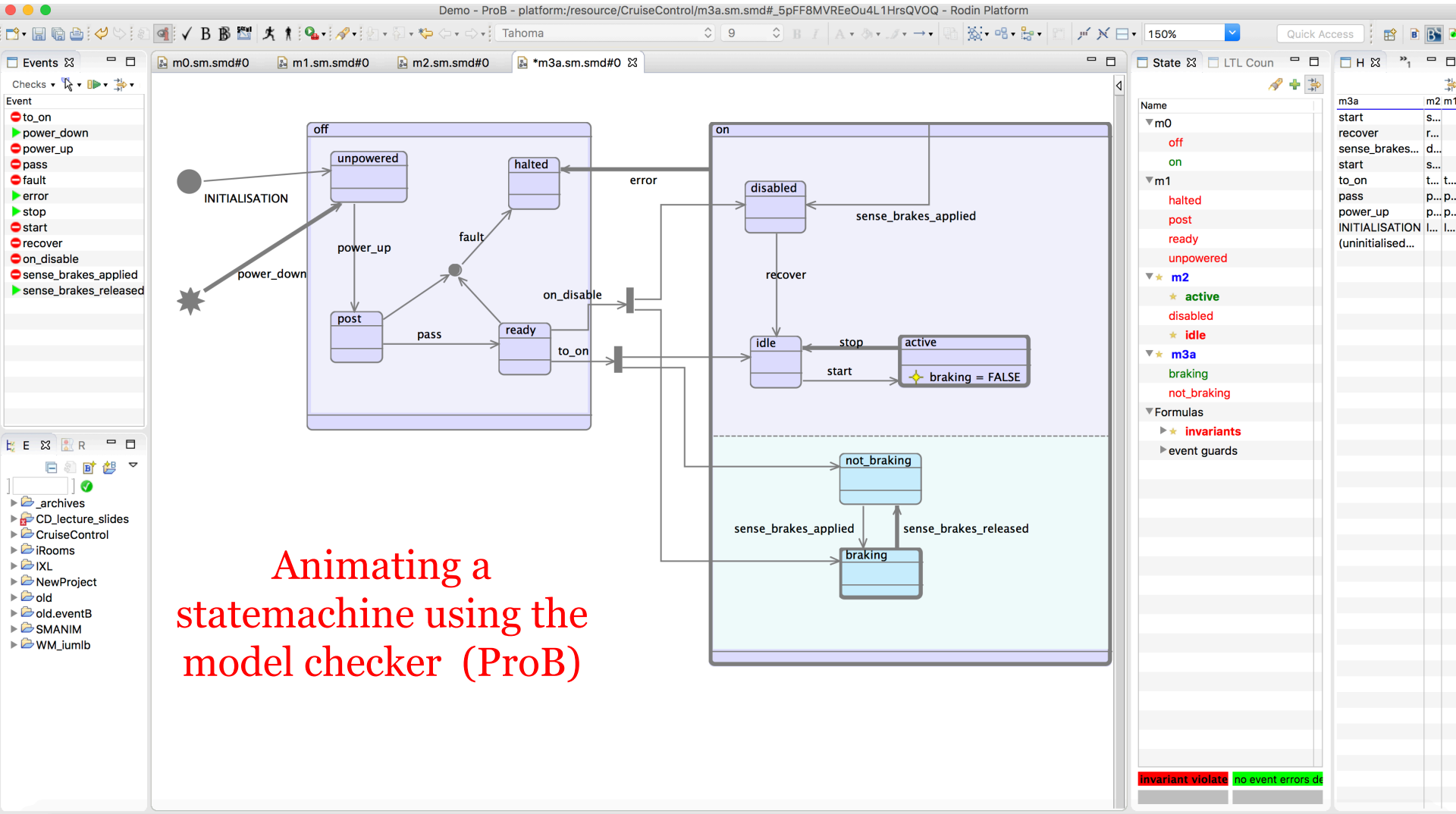
UNIVERSITY OF Southampton
School of Electronics and Computer Science

State-machine refinement



State-machine animation

UNIVERSITY OF Southampton
School of Electronics and Computer Science



Animating a statemachine using the model checker (ProB)

Einstein's Riddle

UNIVERSITY OF Southampton
School of Electronics and Computer Science



Constraint Modeling with Albert Einstein

Einstein's Riddle in xtUML

Cortland Starrett - One Fact Inc



With only these conditions there are many possible solutions.

There are 5 houses in five different colors.

In each house lives a person with a different nationality.

These five owners drink a certain type of beverage, smoke a certain brand of cigar and keep a certain pet.

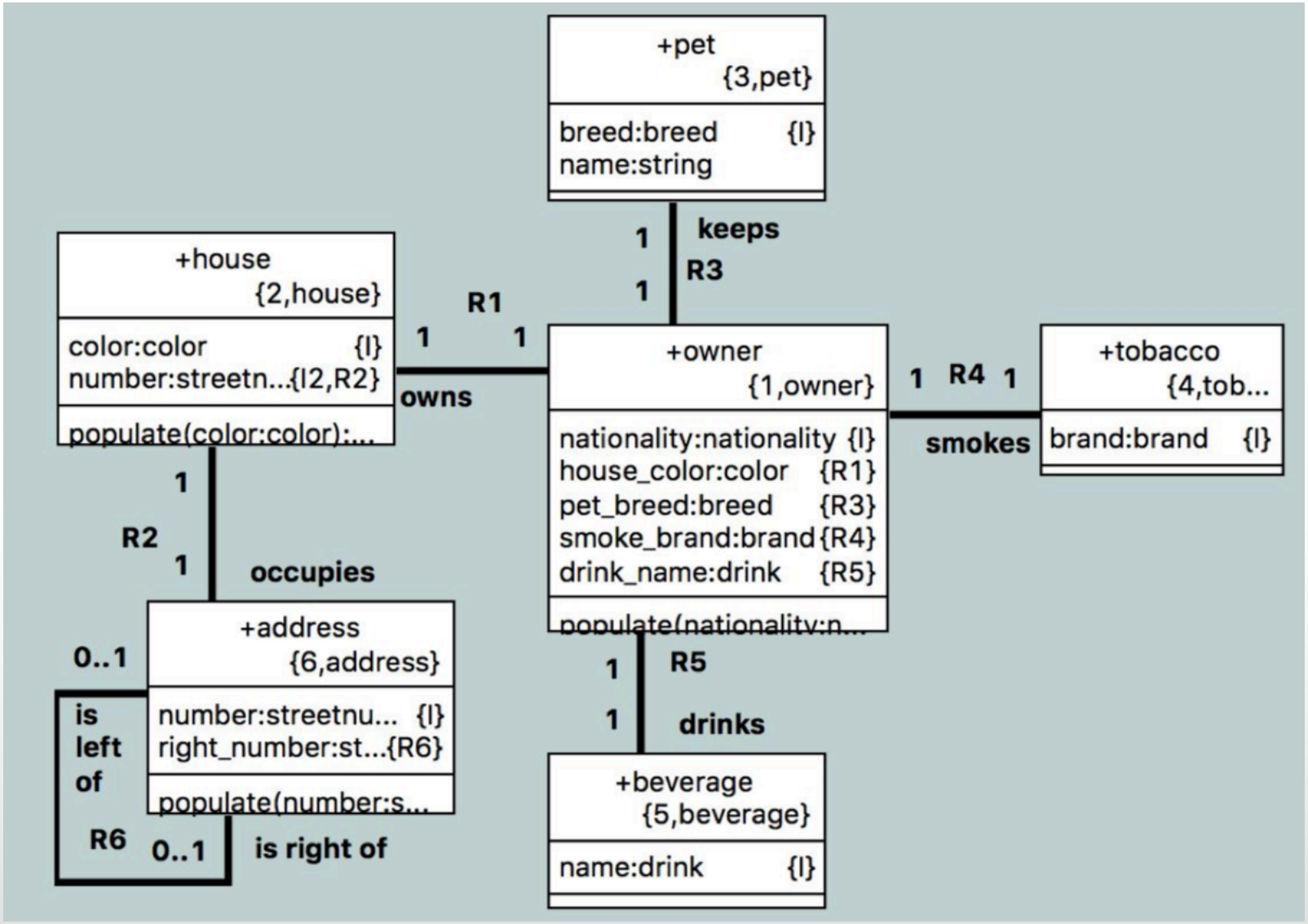
No owners have the same pet, smoke the same brand of cigar or drink the same beverage.

...Who owns the fish?

15 constraints (hints) restrict the solution to one possibility
Removing a hint results in more than one possible solution
Except for one hint which is unnecessary

The Brit lives in the red house.
The Swede keeps dogs as pets.
The Dane drinks tea.
The green house is on the immediate left of the white house.
The green house's owner drinks coffee.
The owner who smokes Pall Mall rears birds.
The owner of the yellow house smokes Dunhill.
The owner living in the center house drinks milk.
The Norwegian lives in the first house.
The owner who smokes Blends lives next to the one who keeps cats.
The owner who keeps the horse lives next to the one who smokes Dunhill.
The owner who smokes Bluemasters drinks beer.
The German smokes Prince.
The Norwegian lives next to the blue house.
The owner who smokes Blends lives next to the one who drinks water.

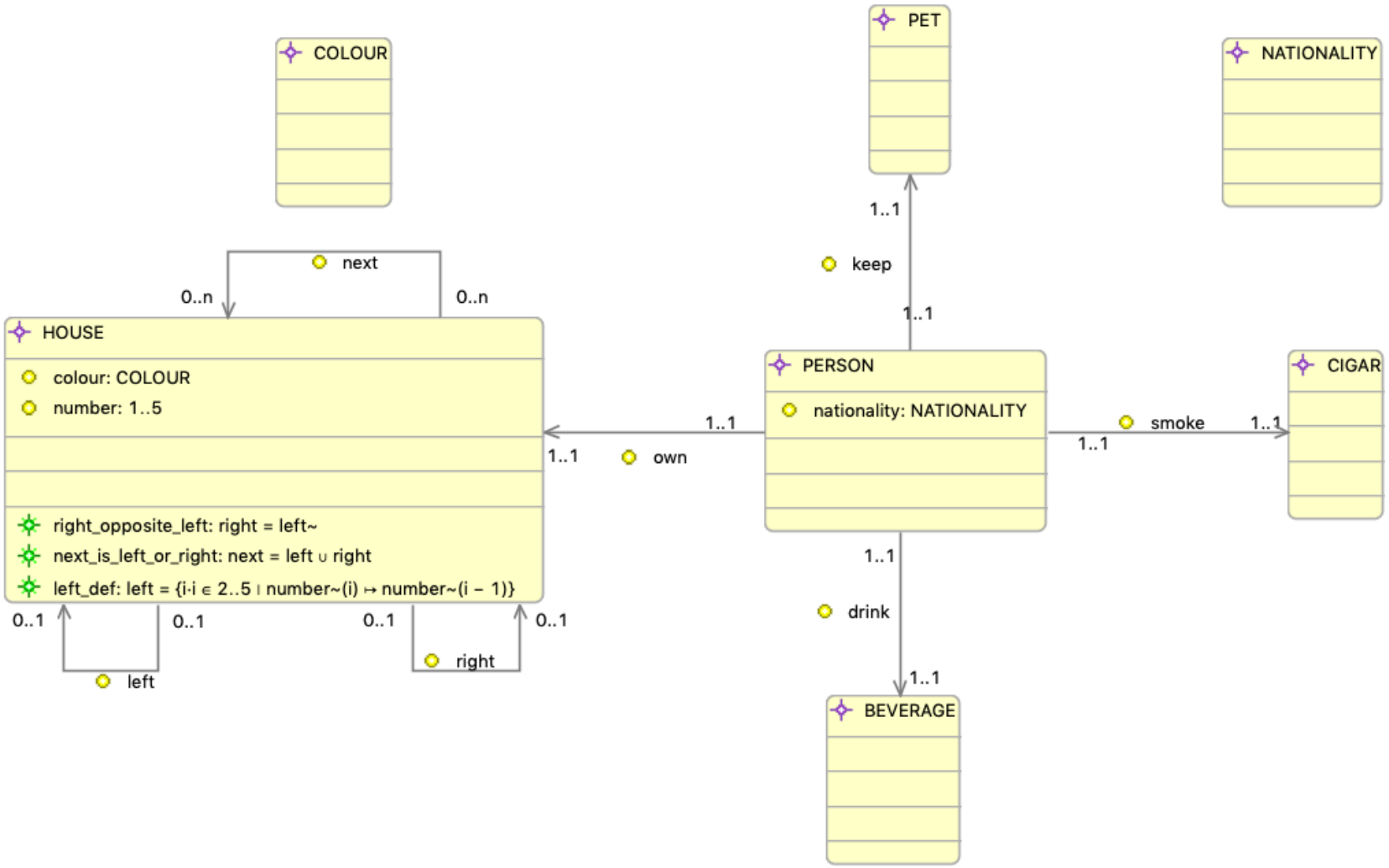
Cortland solved it in xtUML



Einstein's riddle in UML-B

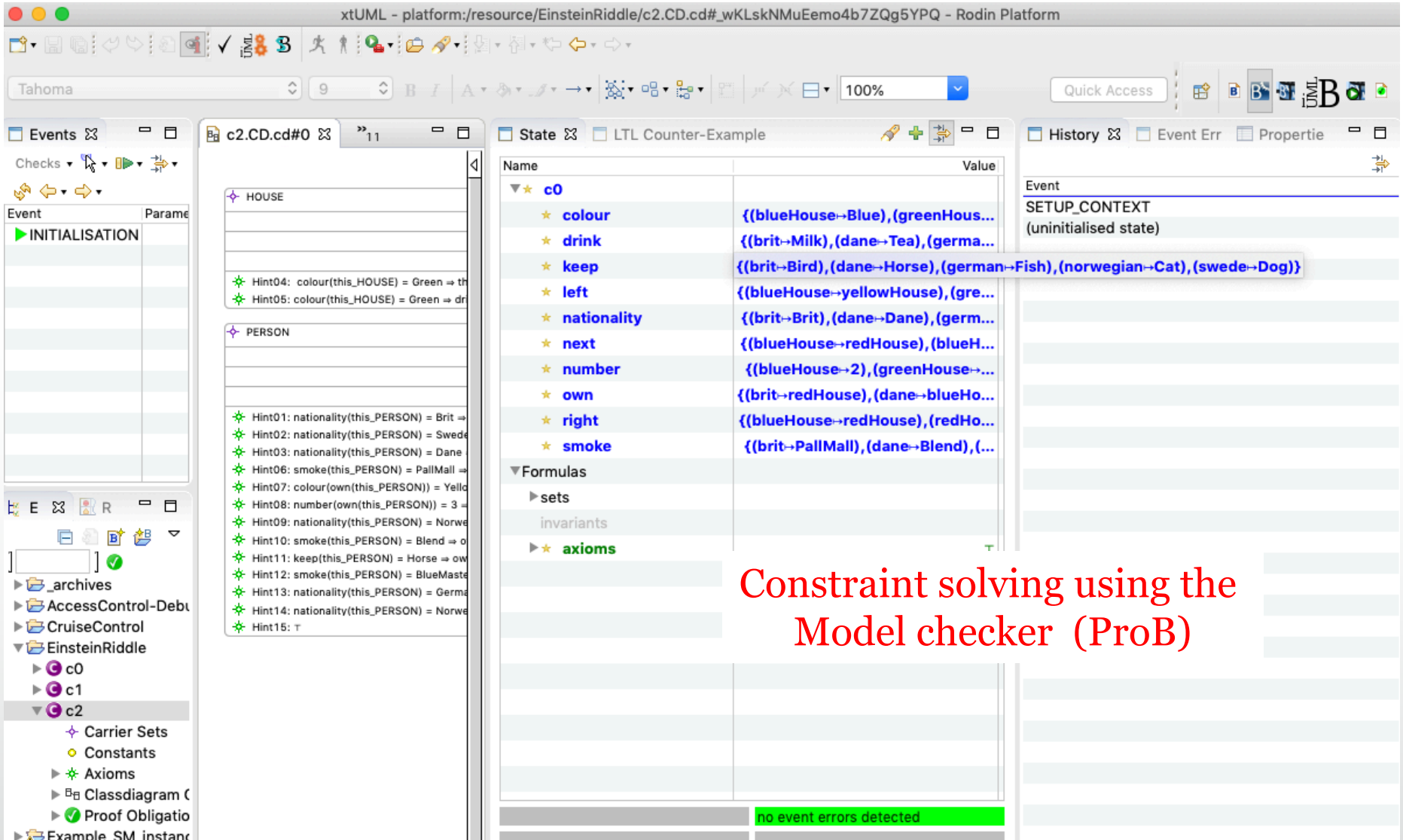
UNIVERSITY OF Southampton
School of Electronics and Computer Science

Modelled by Thai Son Hoang



Einstein's riddle in UML-B

UNIVERSITY OF Southampton
School of Electronics and Computer Science



Constraint solving using the Model checker (ProB)



ENABLE-S3
ECSEL EU
PROJECT

ENABLE-S3

Enable-S3 - Horizon 2020 project

Objective: establish cost-efficient cross-domain virtual and semi-virtual V&V platforms and methods for ACPS

This project has received funding from the ECSEL Joint Undertaking under grant agreement No 692455. This Joint Undertaking receives support from the European Union's Horizon 2020 research and innovation programme and Austria, Denmark, Germany, Finland, Czech Republic, Italy, Spain, Portugal, Poland, Ireland, Belgium, France, Netherlands, United Kingdom, Slovakia, Norway.

Modelling ETCS Hybrid Level 3

UNIVERSITY OF Southampton
School of Electronics and Computer Science

- European Train Control System
- Hybrid Level 3 specification – virtual fixed block
  - but not moving block
- Train gsm position reporting and integrity monitoring
- Retain existing trackside train detectors (TTD)
- Hybrid – copes with existing trains (using TTD)
- New trains supervised more densely using virtual blocks

Modelling ETCS Hybrid Level 3

UNIVERSITY OF Southampton
School of Electronics and Computer Science

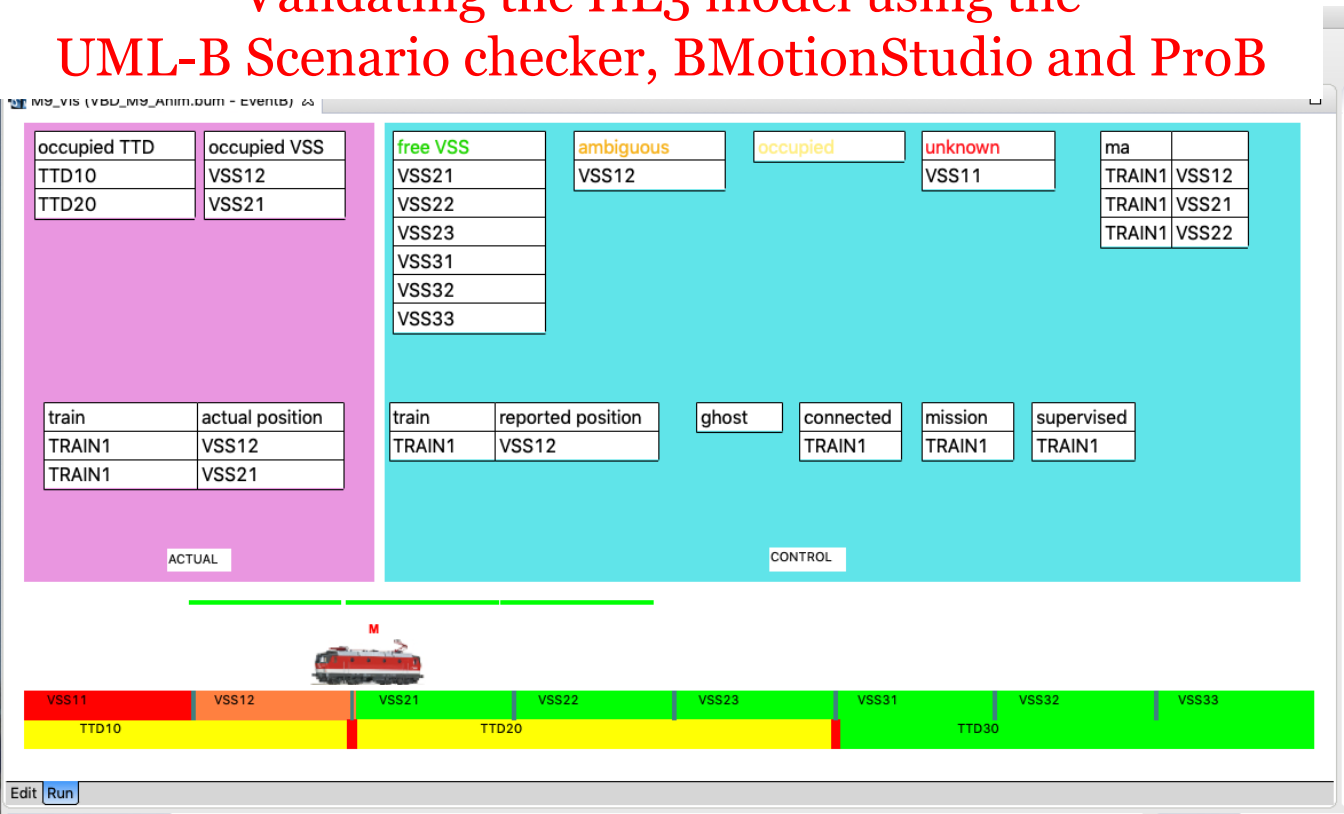
- ETCS HL3 modelled in UML-B
- Could not prove safe in some scenarios
  - if trains roll backwards,
  - when movement authority revoked
- As a result – ETCS spec revised
- Validation of model essential
  - B Motion Studio/ProB,
  - driven by UML-B Scenario checker tool

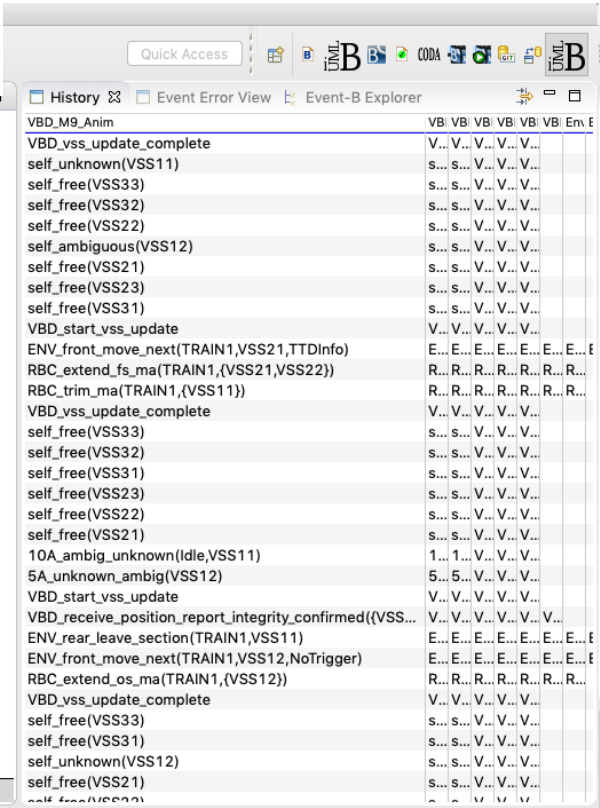
Dghaym, Poppleton, Snook, (2018) Diagram-Led Formal Modelling Using iUML-B for Hybrid ERTMS Level 3. In *International Conference on Abstract State Machines, Alloy, B, TLA, VDM and Z (ABZ2018)*. LNCS Volume 10817 pp338-352.

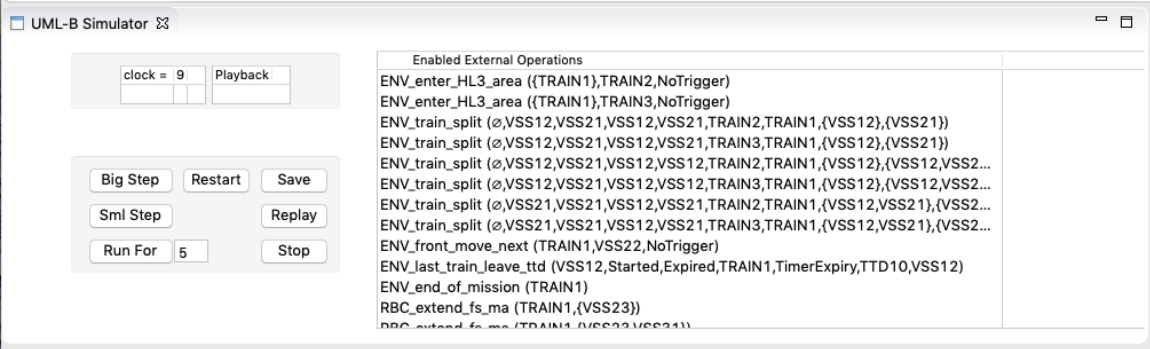
Modelling ETCS Hybrid Level 3

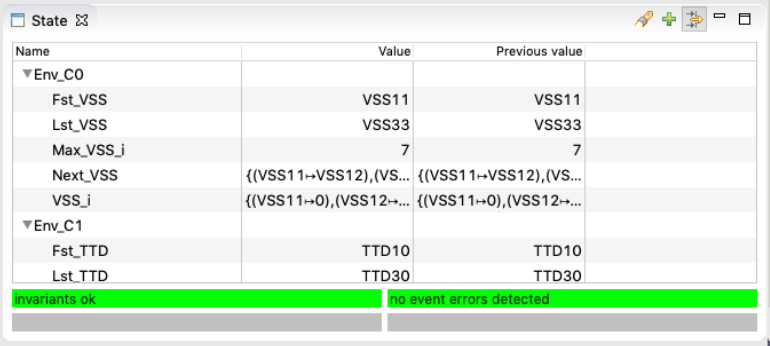
UNIVERSITY OF Southampton
School of Electronics and Computer Science

Validating the HL3 model using the UML-B Scenario checker, BMotionStudio and ProB









UNIVERSITY OF Southampton
School of Electronics and Computer Science

Other collaborations

- HiClass project - (Aerospace Technology Institute)
  - High-Integrity, Complex, Large, Software and Electronic Systems
  - Model driven development for avionics
  - Rolls-Royce, Adacore, Altran, Bae, Callen-Lenz, Cocotec, D-Risq, Cobham (Flight Refuelling), GE, General Dynamics, Rapita, Thales, Ultra,
  - Universities: Oxford, Southampton, York
- Soton work:
  - Text-based notations derived from UML-B
  - Systems modelling into components,
  - M2M generation of design starting point (Scade, others.. xtUML?)

UNIVERSITY OF Southampton
School of Electronics and Computer Science

Other collaborations (cont)

- AWE
  - Component diagrams --> VHDL
- Sandia
  - ‘Run to completion’ Statecharts with refinement
- Deutsche Bahn
  - UML-B for specifying Railway systems equipment – interoperability

UNIVERSITY OF Southampton
School of Electronics and Computer Science

Summary

- UML-B
  - Formal diagrammatic notation
  - Not UML – but somewhat familiar diagram types
  - Refinement – formal : proof obligations generated
- Proof – rigorous, complete (cf testing)
  - Generic (irrespective of instantiation)
  - Mostly automatic (depending on model)
  - Model checker - finds counter examples
    - Also has a constraint solver
- Validation is also important
  - Visualisation of state (B Motion Studio)
  - Scenario checker – save and replay, simulates run to completion

UNIVERSITY OF
Southampton
School of Electronics
and Computer Science

Thank you

Questions?