

How to Explain Mistakes

Stefan Hallerstede and Michael Leuschel

Universität Düsseldorf

Teaching Formal Methods 2009
6 November 2009

Contents

Event-B

Event-B in General Terms

Event-B by Example

Tool Support

Making Mistakes

Explaining Mistakes

On-going Work and Conclusion

Contents

Event-B

Event-B in General Terms

Event-B by Example

Tool Support

Making Mistakes

Explaining Mistakes

On-going Work and Conclusion

Contents

Event-B

Event-B in General Terms

Event-B by Example

Tool Support

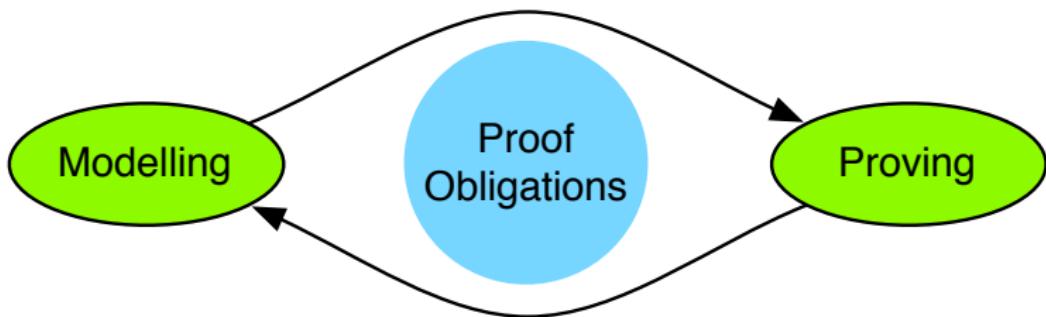
Making Mistakes

Explaining Mistakes

On-going Work and Conclusion

Modelling and Proving in Event-B

- ▶ Main purpose of modelling is **reasoning**
- ▶ Models determine what is to be **formally proved**
- ▶ **Proof obligations** are automatically generated
- ▶ **Tool support** is essential
- ▶ **Refinement** is a proof technique
- ▶ Models and proof obligations **correspond closely**



Contents

Event-B

Event-B in General Terms

Event-B by Example

Tool Support

Making Mistakes

Explaining Mistakes

On-going Work and Conclusion

A Simple Example of an Event-B Model

► Invariants

inv1 : $auth \in Person \leftrightarrow Room$

A person is authorised to be in certain rooms

inv2 : $in \in Person \rightarrow Room$

A person can be at most in one room

inv3 : $in \subseteq auth$

A person can only be in rooms where he is authorised to be

► Events

enter

any

p r

when

grd1 : $p \notin \text{dom}(in)$ Person is not in building

grd2 : $p \mapsto r \in auth$ Person is authorised to enter room

then

act1 : $in := in \cup \{p \mapsto r\}$

end

Proof Obligations of the Event-B Model

- ▶ Preservation of invariant $inv3$ by event $enter$

- ▶ Name of proof obligation

“ $enter/inv3/INV$ ”

- ▶ Sequent

$auth \in Person \leftrightarrow Room$	invariant $inv1$
$in \in Person \rightarrow Room$	invariant $inv2$
$in \subseteq auth$	invariant $inv3$
$p \notin \text{dom}(in)$	guard $grd1$
$p \mapsto r \in auth$	guard $grd2$
$\vdash in \cup \{p \mapsto r\} \subseteq auth$	modified (act1) invariant $inv3$

- ▶ Simple correspondence between proof obligations and model

Contents

Event-B

Event-B in General Terms

Event-B by Example

Tool Support

Making Mistakes

Explaining Mistakes

On-going Work and Conclusion

The Rodin Tool — Modelling

```
event search
  when  $f(i) = v$  then
     $k := i$ 
  end
event inc
  when  $f(i) < v$  then
     $p := i + 1$ 
     $i := (i + 1 + q) \div 2$ 
  end
event dec
  when  $v < f(i)$  then
     $q := i - 1$ 
     $i := (p + i - 1) \div 2$ 
```

Error: 'x' is not a variable

Messages

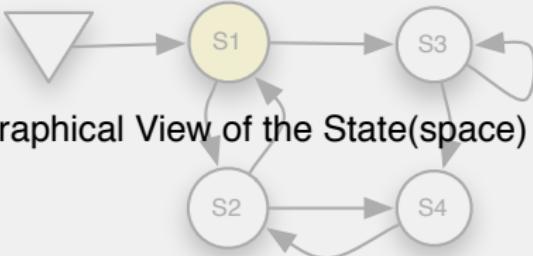
<input checked="" type="checkbox"/> search/i1/INV
<input checked="" type="checkbox"/> search/i2/INV
<input checked="" type="checkbox"/> inc/i1/INV
<input type="checkbox"/> inc/i2/INV
<input type="checkbox"/> dec/i1/INV
<input checked="" type="checkbox"/> dec/i2/INV

Proof
Obligations

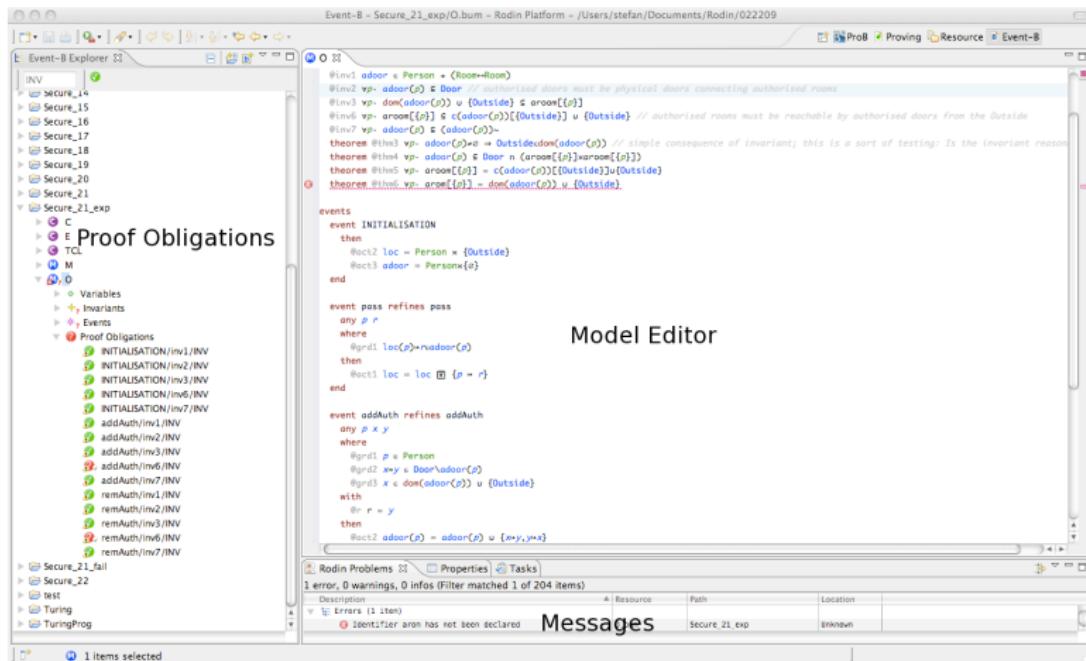
The Rodin Tool — **Proving**

$p \in 1..N$ $i < N$ $f(i) < v$	<input checked="" type="checkbox"/> search/i1/INV <input checked="" type="checkbox"/> search/i2/INV <input checked="" type="checkbox"/> inc/i1/INV <input checked="" type="checkbox"/> inc/i2/INV <input checked="" type="checkbox"/> dec/i1/INV <input checked="" type="checkbox"/> dec/i2/INV
Premises	Proof Obligations
$i + 1 \in 1..N$	
Conclusion	

The Rodin Tool — **Animation** (ProB)

$x=2$	Event1	INITIALISATION
Current State of the Model	Event2	History
	Enabled Events	
	 <p>Graphical View of the State(space)</p>	

Screen Shot of the Rodin Tool — **Modelling**



Contents

Event-B

Event-B in General Terms

Event-B by Example

Tool Support

Making Mistakes

Explaining Mistakes

On-going Work and Conclusion

Starting From a Perfect Solutions

- ▶ Usually we present (perfect) **solutions** to selected problems
- ▶ This does not show **how** the solution was obtained
- ▶ It creates the **illusion** there would be a perfect solution
- ▶ This fails to demonstrate a **major strength** of formal methods
 - ▶ Support towards finding a **good solution**
 - ▶ It is not just about correctness

Finding a Good Solution

Problem solving (Pólya, Lakatos)

- ▶ **Think** about how to approach the problem
- ▶ Start with a model that appears reasonable
- ▶ **Make mistakes**
- ▶ Analyse the model
- ▶ **Think** again
- ▶ Improve the model
- ▶ **Make mistakes**

Contents

Event-B

Event-B in General Terms

Event-B by Example

Tool Support

Making Mistakes

Explaining Mistakes

On-going Work and Conclusion

Analysing and Explaining Mistakes

- ▶ Proof is a good tool for **analysing** inconsistent models
- ▶ It points to the place where the **inconsistency** occurs
- ▶ It does not serve well for **explaining** inconsistencies
- ▶ Useful tools for explanation:
 - ▶ **Model checking**: counter examples
 - ▶ **Animation**: see “how it happens”
- ▶ ProB can also be used for this

Example of requirements

- P1 : The system consists of persons and one building.
- P2 : The building consists of rooms and doors.
- P3 : Each person can be at most in one room.
- P4 : Each person is authorised to be in certain rooms (but not others).
- P5 : Each person is authorised to use certain doors (but not others).
- P6 : Each person can only be in a room where the person is authorised to be.
- P7 : Each person must be able to leave the building from any room where the person is authorised to be.
- P8 : Each person can pass from one room to another if there is a door connecting the two rooms and the person has the proper authorisation.
- P9 : Authorisations can be granted and revoked.

- ▶ Example provides room for **misunderstanding**
- ▶ **Unlike** a sequential program, for instance
- ▶ Model is much **simplified**
from “Event Driven System Construction” by Abrial

Getting Started

- ▶ the **abstract machine** models room authorisations
- ▶ the **concrete machine** models room and door authorisations

Abstract invariants

$inv1 : arm \in Person \leftrightarrow Room$ Property P4

$inv2 : Person \times \{\mathbf{0}\} \subseteq arm$

$inv3 : loc \in Person \rightarrow Room$ Property P3

$inv4 : loc \subseteq arm$ Property P6

Revoking an Authorisation

P9 Authorisations can be granted and **revoked**.

revoke

any *p r* when

grd1 : $p \in Person$

grd2 : $p \mapsto r \notin loc$

then

act1 : $arm := arm \setminus \{p \mapsto r\}$

end

How the model looks in Rodin

The screenshot shows the Rodin Platform interface with the following components:

- Event-B Explorer** (left): Lists model elements: C, D, E, TCL, thm, M, and Proof Obligations. Proof Obligations are expanded to show: INITIALISATION/inv1/INV, INITIALISATION/inv2/INV, INITIALISATION/inv3/INV, INITIALISATION/inv4/INV, pass/inv2/INV, pass/inv4/INV, revoke/grd2/WD, revoke/inv1/INV, revoke/inv3/INV, revoke/inv4/INV, grant/inv1/INV, grant/inv3/INV, and grant/inv4/INV.
- Editor Area** (center): Displays the Event-B model text. The text includes:

```
variables aroom loc

invariants
@inv1 aroom ∈ Person ↔ Room
@inv2 loc ∈ Person → Room
@inv3 Person × {Outside} ⊑ aroom
@inv4 loc ⊑ aroom

events
event INITIALISATION
then
@act1 aroom = Person × {Outside}
@act2 loc = Person × {Outside}
end

event pass
any p r
where
@grdl p ↳ r ∈ aroom
then
@act1 loc = loc ∪ {p ↳ r}
end

event revoke
any p r
```
- Rodin Problems** (bottom): Shows 20 errors, 0 warnings, and 0 infos. The errors table lists:

Description	Resource	Path	Location
Errors (20 items)	N.bum	Secure_12	Un
Abstract event addAuth not found			

What have to prove:

Event *revoke* preserves invariant *inv2*:

$Person \times \{\mathbf{0}\} \subseteq arm$	<i>Invariant inv2</i>
$p \in Person$	<i>Guard grd1</i>
$p \mapsto r \notin loc$	<i>Guard grd2</i>
$\vdash Person \times \{\mathbf{0}\} \subseteq arm \setminus \{p \mapsto r\}$	<i>Modified invariant inv2</i>

How the proof obligation looks in Rodin

The screenshot shows the Rodin Platform interface with the following components:

- Proof Tree** (left): Shows a tree structure with a node labeled "simplification rewrites" and a sub-node "Person × {Outside}garoom \ {p + r}".
- Proving** (center): The main workspace for proving. It displays the goal: `Person × {Outside}garoom \ {p + r}`. The proof tree below it shows three sub-goals:
 - Person × {Outside}garoom
 - pePerson
 - ~ loc(p)=r
- Event-B Explore** (right): A tree view of the Event-B model. It shows nodes for Variables, Invariants, Events, and Proof Obligations. The "Proof Obligations" node is expanded, showing a list of obligations:
 - INITIALISATION/inv1
 - INITIALISATION/inv2
 - INITIALISATION/inv3
 - INITIALISATION/inv4
 - pass/inv2/INV
 - pass/inv4/INV
 - revoke/grd2/WD
 - revoke/inv1/INV
 - revoke/inv3/INV
 - revoke/inv4/INV
 - grant/inv1/INV
 - grant/inv3/INV
 - grant/inv4/INV

New current obligation

What have to prove:

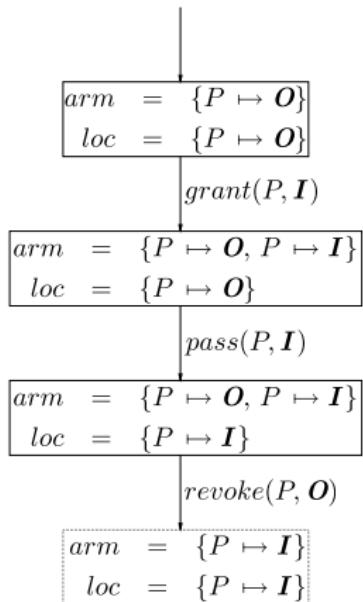
Event *revoke* preserves invariant *inv2*:

$Person \times \{\mathbf{0}\} \subseteq arm$	<i>Invariant inv2</i>
$p \in Person$	<i>Guard grd1</i>
$p \mapsto r \notin loc$	<i>Guard grd2</i>
$\vdash Person \times \{\mathbf{0}\} \subseteq arm \setminus \{p \mapsto r\}$	<i>Modified invariant inv2</i>

- ▶ Is it (*not*) provable?
- ▶ Why?
- ▶ Our aim is to **improve the model**
- ▶ **Not** to make the proof obligation “pass”

Change of perspective

- ▶ Look at a problematic **state trace**
(leading to an inconsistent state)



- ▶ ProB alerts us that it violates invariant *inv2*:
 $Person \times \{O\} \subseteq arm$

How the counter example looks in ProB

ProB - Secure_12/M.bum - Rodin Platform - /Users/stefan/Documents/Rodin/022209

Operations

Operation	Parameter(s)
revoke	P, Outside
pass	P, Inside
grant	P, Inside

M M C

```
variables aroom loc

invariants
  @inv1 aroom = Person ↔ Room
  @inv2 loc ∈ Person → Room
  @inv3 Person × {Outside} ⊑ aroom
  @inv4 loc ⊑ aroom

events
  event INITIALISATION
    then
      @act1 aroom = Person × {Outside}
      @act2 loc = Person × {Outside}
    end

  event pass
    any p r
    where
      @grd1 p ↳ r ∈ aroom
    then
      @act1 loc = loc + {p ↳ r}
    end

  event revoke
    any p r
    where
      @grd1 p ∈ Person
      @grd2 loc(p) ≠ r
      // @grd3 r ≠ Outside
    then
      @act1 aroom = aroom \ {p ↳ r}
    end
```

State

Name	Value	Previous value
Variables		
aroom	{(P)→Inside}	{(P)→Inside}, {(P)→Outside}
loc	{(P)→Inside}	

History

Operations	Loops
revoke(P,Outside)	
pass(P,Inside)	
grant(P,Inside)	
revoke(P,Inside)	
(root)	

invariant violated!

Contents

Event-B

Event-B in General Terms

Event-B by Example

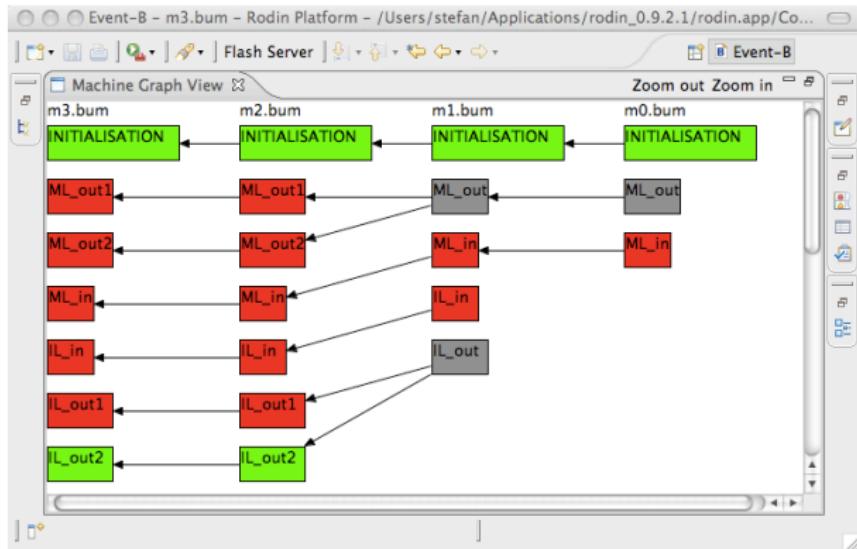
Tool Support

Making Mistakes

Explaining Mistakes

On-going Work and Conclusion

Refinement Animation



We work on refinement animation similar to Brama

Refinement Animation

<i>CoffeeR2</i>	<i>Inv</i>	<i>CoffeeR1</i>	<i>Inv</i>	<i>CoffeeM</i>
<i>INITIALSATION</i>		<i>INITIALSATION</i>		<i>INITIALSATION</i>
<i>fill_mug</i> ↪		<i>fill_mug</i> ↪		<i>fill_mug</i> ↪
<i>drink</i> ↓		<i>drink</i> ↓		<i>drink</i> ↓
<i>insert_coin</i> ↓		<i>insert_coin</i> →		

Sketch of non-graphical display of state of refinement animation

Conclusion

- ▶ Teach formal modelling **how** it is done
- ▶ Teach **incremental** modelling
- ▶ Teach how to **improve** a model in small increments
- ▶ Teach making mistakes (how to **profit** from making mistakes)
- ▶ Teach how to **explain** mistakes and to **justify** improvements
- ▶ Use a **software tool** like Rodin/ProB in class and in exercises
- ▶ Getting a model right is **not easy**