

How to Explain Mistakes

Stefan Hallerstedte 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

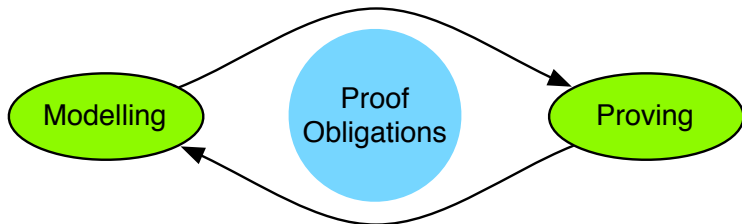
Making Mistakes

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 $inv1$
$in \in Person \leftrightarrow Room$	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

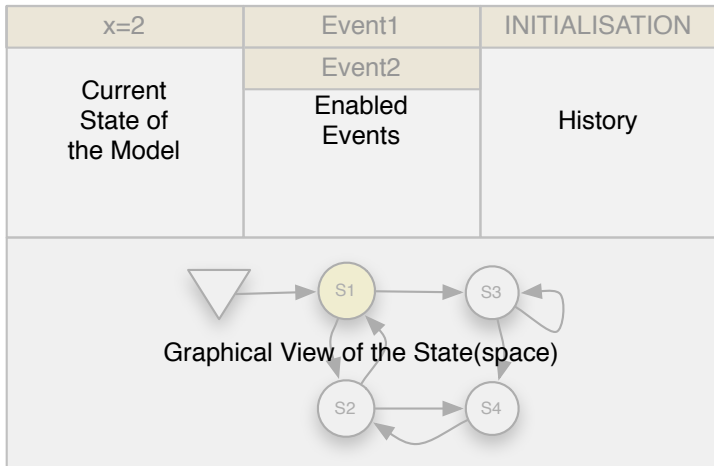
The Rodin Tool — Modelling

<pre> 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 $q := i - 1$ $i := (p + i - 1) \div 2$ </pre>	<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
Error: 'x' is not a variable	<div>Proof Obligations</div>
Messages	

The Rodin Tool — Proving

$p \in 1..N$ $i < N$ $f(i) < v$ Premises	<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
$i + 1 \in 1..N$ Conclusion	Proof Obligations

The Rodin Tool — **Animation** (ProB)



Screen Shot of the Rodin Tool — Modelling

The screenshot displays the Rodin Tool interface with the following components:

- Event-B Explorer (Left Panel):** A tree view showing the project structure. The 'Secure_21_exp' folder is expanded, revealing a list of 'Proof Obligations' including 'INITIALISATION/inv1/INV' through 'remAuth/inv7/INV'.
- Model Editor (Main Window):** Displays the formal model code for 'Event-B'. The code includes invariants, events, and a task, with comments explaining their purpose. For example, it defines a set of doors and rooms, and an event for door opening that updates the location of a person.
- Messages (Bottom Panel):** Shows the status of the model. It reports '1 error, 0 warnings, 0 infos (Filter matched 1 of 204 items)'. The error message is 'Identifier anon has not been declared' at 'Secure_21_exp'.

Labels within the image:

- Proof Obligations:** Points to the list of obligations in the Event-B Explorer.
- Model Editor:** Points to the main code editing area.
- Messages:** Points to the bottom status and error panel.

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 displays the Rodin Platform interface for a model named "Event-B - Secure_12/M.bum". The interface is divided into three main sections:

- Event-B Explorer (Left):** A tree view showing the model's structure. It includes sections for Variables, Invariants, Events, and Proof Obligations. The "Proof Obligations" section is expanded, showing a list of obligations such as "INITIALISATION/inv1/INV", "pass/inv2/INV", "revoke/grd2/WD", and "revoke/inv3/INV".
- Event-B Model (Center):** A text editor displaying the Event-B model code. The code defines variables, invariants, and events. The invariants section includes:

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

The events section includes:

```
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
```
- Rodin Problems (Bottom):** A panel showing the status of the model. It indicates "20 errors, 0 warnings, 0 infos (Filter matched 20 of 210 items)". A table lists the errors, with the first error being "Abstract event addAuth not found" in the "N.bum" resource at the "Secure_12" path.

What have to prove:

Event *revoke* preserves invariant *inv2*:

$Person \times \{\mathbf{O}\} \subseteq arm$

$p \in Person$

$p \mapsto r \notin loc$

$\vdash Person \times \{\mathbf{O}\} \subseteq arm \setminus \{p \mapsto r\}$

Invariant inv2

Guard grd1

Guard grd2

Modified invariant inv2

How the proof obligation looks in Rodin

The screenshot displays the Rodin Platform interface for a proof obligation. The title bar indicates the project is 'Proving - Secure_12/M.bps - Rodin Platform' located at '/Users/stefan/Documents/Rodin/022209'. The interface is divided into several panels:

- Proof Tree:** Shows a tree structure with a node labeled 'Person × {Outside}caroom \ {p → r}'. A red question mark icon is next to the node.
- revoke/inv3/INV:** The main panel displays the current proof obligation. It includes a list of hypotheses (ct) and a goal. The hypotheses are:
 - Person × {Outside}caroom
 - pePerson
 - ¬ loc(p)=rThe goal is: Person × {Outside}caroom \ {p → r}.
- Event-B Explore:** A sidebar on the right showing a list of events and proof obligations. The list includes: INITIALISATION/inv1, INITIALISATION/inv2, INITIALISATION/inv3, INITIALISATION/inv4, pass/inv2/INV, pass/inv4/INV, revoke/grd2/WD, revoke/inv1/INV, revoke/inv3/INV (highlighted with a red question mark), revoke/inv4/INV, grant/inv1/INV, grant/inv3/INV, and grant/inv4/INV.
- Proof Control:** A bottom panel with a 'Statistics' tab and a 'Rodin Problems' tab. It features a red sad face icon and a text input field.

The bottom status bar shows 'New current obligation'.

What have to prove:

Event *revoke* preserves invariant *inv2*:

$Person \times \{O\} \subseteq arm$

$p \in Person$

$p \mapsto r \notin loc$

$\vdash Person \times \{O\} \subseteq arm \setminus \{p \mapsto r\}$

Invariant inv2

Guard grd1

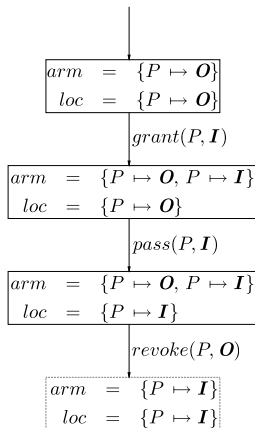
Guard grd2

Modified invariant inv2

- 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*:
 $Person \times \{\mathbf{O}\} \subseteq arm$

How the counter example looks in ProB

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

Eve Rodin M C State History

Secure_11
Secure_12
C
D
E
TCL
thm
M
N

Operations

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

```
variables aroom loc

invariants
  @inv1 aroom  $\subseteq$  Person  $\leftrightarrow$  Room
  @inv2 loc  $\in$  Person  $\rightarrow$  Room
  @inv3 Person $\times$ {Outside}  $\subseteq$  aroom
  @inv4 loc  $\subseteq$  aroom

events
  event INITIALISATION
  then
    @act1 aroom = Person  $\times$  {Outside}
    @act2 loc = Person  $\times$  {Outside}
  end

  event pass
  any p r
  where
    @grd1 p  $\mapsto$  r  $\in$  aroom
  then
    @act1 loc = loc  $\cup$  {p  $\mapsto$  r}
  end

  event revoke
  any p r
  where
    @grd1 p  $\in$  Person
    @grd2 loc(p)  $\neq$  r
    //@grd3 r  $\neq$  Outside
  then
    @act1 aroom = aroom  $\setminus$  {p  $\mapsto$  r}
  end
```

Name	Value	Previous value
Variables		
aroom	{{P \mapsto Inside}}	{{P \mapsto Inside}, {P \mapsto Outside}}
loc	{{P \mapsto Inside}}	

Operations

revoked(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

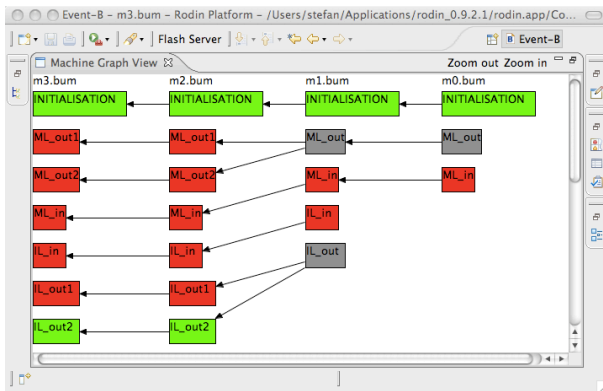
Tool Support

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**