

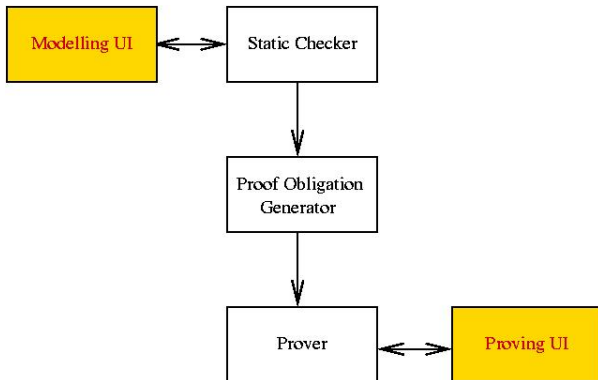
Event-B User Interfaces

Thai Son Hoang

Department of Computer Science
Swiss Federal Institute of Technology Zürich (ETH Zürich)

RODIN Plenary Meeting, 3rd-4th April 2006, Aix-en-Provence

User Interfaces



Outline

- 1 Modelling Interface: Views and Editors
- 2 Proving Interface: Views and Editors
- 3 Extending the User Interface
- 4 Extending the Proving Interface
- 5 To be done next

Modelling Interface

Contains 3 views and an editor.

The screenshot displays the Modelling Interface software. The interface is divided into several panes:

- Project Explorer (Left):** Shows a hierarchical tree of project elements including CANTOR, CLOS, DAGS, Exercise7, FIX, INDUC, Marriage, m0, c0, MINI-SORT, NAT_INDUC, PERSON, SEARCH_MATR, SIMPLE, TERM_DET, TOPO1, TOPO2, TRANS_CLOS, TREE, ULTR, WFD, and WWW.
- Events View (Center):** A table listing events with columns for Element, Name, and Content.

Element	Name	Content
x birth	birth	
+ x	x	
guard1	guard1	$x \in P \wedge p$
• p = p ∨ (x)		$p = p \vee (x)$
x death	death	
x marriage	marriage	
+ x	x	
+ y	y	
guard9	guard9	$x \neq p \wedge \text{don}(s)$
- Invariants View (Center):** Lists invariants:
 - inv0: $p \leq p$
 - inv1: $s \in p \Rightarrow p$
 - inv2: $s = s$
 - inv3: $s \text{ nId}(P) = p$
- Events Editor (Center):** Shows the formal definition of the 'birth' event:


```

ANY x WHERE
  guard1: x ∈ P ∧ p
THEN
  p = p ∨ (x)
END
      
```
- Outline (Right):** A list of elements including c0, p, s, inv0, inv1, inv2, inv3, birth, death, marriage, divorce, and INITIALISATION.
- Bottom Panel:** Shows a 'Problems' tab with 42 errors, 0 warnings, and 0 infos. The error list includes:
 - (event INITIALISATION) Not all variables are initialised. Missing (finish_min, m0.burn, MINI-SORT
 - (axiom axm3) Syntax error: Syntax error at 0:0: invalid SimpleExpr c11.buc WWW
 - (axiom axm4) Syntax error: Syntax error at 0:0: invalid SimpleExpr c11.buc WWW
 - (axiom axm5) Syntax error: Syntax error at 0:0: invalid SimpleExpr c11.buc WWW
 - (axiom axm6) Syntax error: Syntax error at 0:0: invalid SimpleExpr c11.buc WWW

Modelling Views and Editors: Summary

- **Explorer View** Showing a tree structured view of the workspace.
 - Connects to the *Database*.
 - Connects to the *Event-B Editor* for editing components.
- **Event-B Editor** A specific editor for creating and modifying event-B components.
 - Multi-page Editor.
 - Form Editor.
 - Connects with the *Database*.
- **Outline View** Showing the tree structured view of the current editing component.
 - Connects with the current active *Event-B Editor*.
 - Provides navigations for the editing component.
- **Problems View** Showing error/warning messages.
 - Connects with *Event-B Editors* for navigations of error messages.

Modelling Views and Editors: Summary

- **Explorer View** Showing a tree structured view of the workspace.
 - Connects to the *Database*.
 - Connects to the *Event-B Editor* for editing components.
- **Event-B Editor** A specific editor for creating and modifying event-B components.
 - Multi-page Editor.
 - Form Editor.
 - Connects with the *Database*.
- **Outline View** Showing the tree structured view of the current editing component.
 - Connects with the current active *Event-B Editor*.
 - Provides navigations for the editing component.
- **Problems View** Showing error/warning messages.
 - Connects with *Event-B Editors* for navigations of error messages.

Modelling Views and Editors: Summary

- **Explorer View** Showing a tree structured view of the workspace.
 - Connects to the *Database*.
 - Connects to the *Event-B Editor* for editing components.
- **Event-B Editor** A specific editor for creating and modifying event-B components.
 - Multi-page Editor.
 - Form Editor.
 - Connects with the *Database*.
- **Outline View** Showing the tree structured view of the current editing component.
 - Connects with the current active *Event-B Editor*.
 - Provides navigations for the editing component.
- **Problems View** Showing error/warning messages.
 - Connects with *Event-B Editors* for navigations of error messages.

Modelling Views and Editors: Summary

- **Explorer View** Showing a tree structured view of the workspace.
 - Connects to the *Database*.
 - Connects to the *Event-B Editor* for editing components.
- **Event-B Editor** A specific editor for creating and modifying event-B components.
 - Multi-page Editor.
 - Form Editor.
 - Connects with the *Database*.
- **Outline View** Showing the tree structured view of the current editing component.
 - Connects with the current active *Event-B Editor*.
 - Provides navigations for the editing component.
- **Problems View** Showing error/warning messages.
 - Connects with *Event-B Editors* for navigations of error messages.

Modelling Views and Editors: Summary

- **Explorer View** Showing a tree structured view of the workspace.
 - Connects to the *Database*.
 - Connects to the *Event-B Editor* for editing components.
- **Event-B Editor** A specific editor for creating and modifying event-B components.
 - Multi-page Editor.
 - Form Editor.
 - Connects with the *Database*.
- **Outline View** Showing the tree structured view of the current editing component.
 - Connects with the current active *Event-B Editor*.
 - Provides navigations for the editing component.
- **Problems View** Showing error/warning messages.
 - Connects with *Event-B Editors* for navigations of error messages.

Prover Interface

Contains 4 views and an editor.

The screenshot displays the Prover Interface software with the following components:

- Project Explorer:** Shows a tree structure of proof goals:
 - add hyp 1 = 1: $x = 1 \wedge 1$
 - add hyp 1 = 1: $1 = 1$
 - assm : \vdash
 - assm : $1 = 1$
 - assm : \vdash
 - contradict goal: $x = 1 \wedge 1$
 - add hyp 2 = 2: \vdash
 - add hyp 3 = 3: $2 = 2$
 - DESELECT: $3 = 3$
- Proof State:**
 - Searched Hypotheses:**
 - $s1$ ds
 - \wedge eh he $2 = 2$
 - \wedge \vdash
 - \wedge xel
 - \wedge yel1
 - Cached Hypotheses:**
 - $s1$ ds
 - \wedge \vdash $x = 1 \wedge 1$
 - Selected Hypotheses:**
 - ds
 - \wedge $x \neq 0$
 - \wedge eh he $1 = 1$
 - Goal:** $3 = 3$
- Proof Editor:**
 - Proof Control: Problems
 - Editor: $3 = 3$
 - Bottom status: Tactic applied successfully
- Right Panel:**
 - Navigation tree: INDUC, Manage, MINO_SCRIPT, NAT_INDUC, PERSON, SEARCH_MATR, SIMPLE, TERM_DET, TOPDI, TOPDI2, TRANS_CLOS, TREE, ULTR, WFD, WWW, c0, has1, has2, c1, m0, decInv1INV, INITIALISATIONInv1INV, INITIALISATIONInv2INV, DLX.
 - Proof Information:
 - decInv1INV
 - Invariant: $inv1: x \neq N$
 - event
 - dec:
 - WHEN
 - grd1: $x \neq 0$
 - THEN
 - $x = x - 1$
 - END

Proving Views and Editors: Summary

- **Obligation Explorer** Showing a tree structured view of the obligations in the workspace.
 - Connects to the *Database*.
 - Connects to the *Prover UI Editor* for proving obligations.
- **Prover UI Editor** Showing the current state of the proof.
 - Showing different set of hypotheses: selected, cached or searched.
 - The current goal.
- **Proof Tree** Showing the tree structured view of the current proof.
 - Connects with the current active *Prover UI Editor*.
 - Provides easy navigations on the proof tree (e.g. travel between different sub-goals).
- **Proof Control** Controlling the proof.
 - A set of buttons.
 - A text field for optional input.
 - Showing proof messages (successful, hint, etc.)
- **Proof Information** Showing related information to the current obligation.

Proving Views and Editors: Summary

- **Obligation Explorer** Showing a tree structured view of the obligations in the workspace.
 - Connects to the *Database*.
 - Connects to the *Prover UI Editor* for proving obligations.
- **Prover UI Editor** Showing the current state of the proof.
 - Showing different set of hypotheses: selected, cached or searched.
 - The current goal.
- **Proof Tree** Showing the tree structured view of the current proof.
 - Connects with the current active *Prover UI Editor*.
 - Provides easy navigations on the proof tree (e.g. travel between different sub-goals).
- **Proof Control** Controlling the proof.
 - A set of buttons.
 - A text field for optional input.
 - Showing proof messages (successful, hint, etc.)
- **Proof Information** Showing related information to the current obligation.

Proving Views and Editors: Summary

- **Obligation Explorer** Showing a tree structured view of the obligations in the workspace.
 - Connects to the *Database*.
 - Connects to the *Prover UI Editor* for proving obligations.
- **Prover UI Editor** Showing the current state of the proof.
 - Showing different set of hypotheses: selected, cached or searched.
 - The current goal.
- **Proof Tree** Showing the tree structured view of the current proof.
 - Connects with the current active *Prover UI Editor*.
 - Provides easy navigations on the proof tree (e.g. travel between different sub-goals).
- **Proof Control** Controlling the proof.
 - A set of buttons.
 - A text field for optional input.
 - Showing proof messages (successful, hint, etc.)
- **Proof Information** Showing related information to the current obligation.

Proving Views and Editors: Summary

- **Obligation Explorer** Showing a tree structured view of the obligations in the workspace.
 - Connects to the *Database*.
 - Connects to the *Prover UI Editor* for proving obligations.
- **Prover UI Editor** Showing the current state of the proof.
 - Showing different set of hypotheses: selected, cached or searched.
 - The current goal.
- **Proof Tree** Showing the tree structured view of the current proof.
 - Connects with the current active *Prover UI Editor*.
 - Provides easy navigations on the proof tree (e.g. travel between different sub-goals).
- **Proof Control** Controlling the proof.
 - A set of buttons.
 - A text field for optional input.
 - Showing proof messages (successful, hint, etc.)
- **Proof Information** Showing related information to the current obligation.

Proving Views and Editors: Summary

- **Obligation Explorer** Showing a tree structured view of the obligations in the workspace.
 - Connects to the *Database*.
 - Connects to the *Prover UI Editor* for proving obligations.
- **Prover UI Editor** Showing the current state of the proof.
 - Showing different set of hypotheses: selected, cached or searched.
 - The current goal.
- **Proof Tree** Showing the tree structured view of the current proof.
 - Connects with the current active *Prover UI Editor*.
 - Provides easy navigations on the proof tree (e.g. travel between different sub-goals).
- **Proof Control** Controlling the proof.
 - A set of buttons.
 - A text field for optional input.
 - Showing proof messages (successful, hint, etc.)
- **Proof Information** Showing related information to the current obligation.

Extending the Modelling Interface

- Adding a new element type (e.g. probabilistic invariants):
 - Implement a form page;
 - Extend the extension point to add this page to the Event-B Editor.
- Adding an attribute to an element:
 - Adding a new column to the editing page (e.g. when adding probabilities to guards),
or
 - Editing the new attribute in the detail page (if the attribute needs more space, e.g. multi-line comments).

Extending the Modelling Interface

- Adding a new element type (e.g. probabilistic invariants):
 - Implement a form page;
 - Extend the extension point to add this page to the Event-B Editor.
- Adding an attribute to an element:
 - Adding a new column to the editing page (e.g. when adding probabilities to guards),
or
 - Editing the new attribute in the detail page (if the attribute needs more space, e.g. multi-line comments).

Extending the Proving Interface

- Adding a new goal tactic (added to the goal section):
 - Declare a new goal tactic with a new symbol (shown next to the goal) and when the tactic is applicable;
 - Implement the call to new tactic.
- Adding a new hypothesis tactic (added to the hypothesis section):
 - Declare a new hypothesis tactic with a new symbol (shown next to the hypothesis) and when the tactic is applicable;
 - Implement the call to the new tactic.
- Adding a new global tactic (added to the *Proof Control*):
 - Add a new button (or extend the current button) in the Proof Control and when this is applicable;
 - Implement the call to the new tactic.

Extending the Proving Interface

- Adding a new goal tactic (added to the goal section):
 - Declare a new goal tactic with a new symbol (shown next to the goal) and when the tactic is applicable;
 - Implement the call to new tactic.
- Adding a new hypothesis tactic (added to the hypothesis section):
 - Declare a new hypothesis tactic with a new symbol (shown next to the hypothesis) and when the tactic is applicable;
 - Implement the call to the new tactic.
- Adding a new global tactic (added to the *Proof Control*):
 - Add a new button (or extend the current button) in the Proof Control and when this is applicable;
 - Implement the call to the new tactic.

Extending the Proving Interface

- Adding a new goal tactic (added to the goal section):
 - Declare a new goal tactic with a new symbol (shown next to the goal) and when the tactic is applicable;
 - Implement the call to new tactic.
- Adding a new hypothesis tactic (added to the hypothesis section):
 - Declare a new hypothesis tactic with a new symbol (shown next to the hypothesis) and when the tactic is applicable;
 - Implement the call to the new tactic.
- Adding a new global tactic (added to the *Proof Control*):
 - Add a new button (or extend the current button) in the Proof Control and when this is applicable;
 - Implement the call to the new tactic.

Next ...

- Improve the GUI's usability: Adding more buttons, menu, toolbar, to both modelling and proving interfaces, etc.
- Declare extension points.
- Extend the GUI for refinement component.

Next ...

- Improve the GUI's usability: Adding more buttons, menu, toolbar, to both modelling and proving interfaces, etc.
- Declare extension points.
- Extend the GUI for refinement component.

Next ...

- Improve the GUI's usability: Adding more buttons, menu, toolbar, to both modelling and proving interfaces, etc.
- Declare extension points.
- Extend the GUI for refinement component.