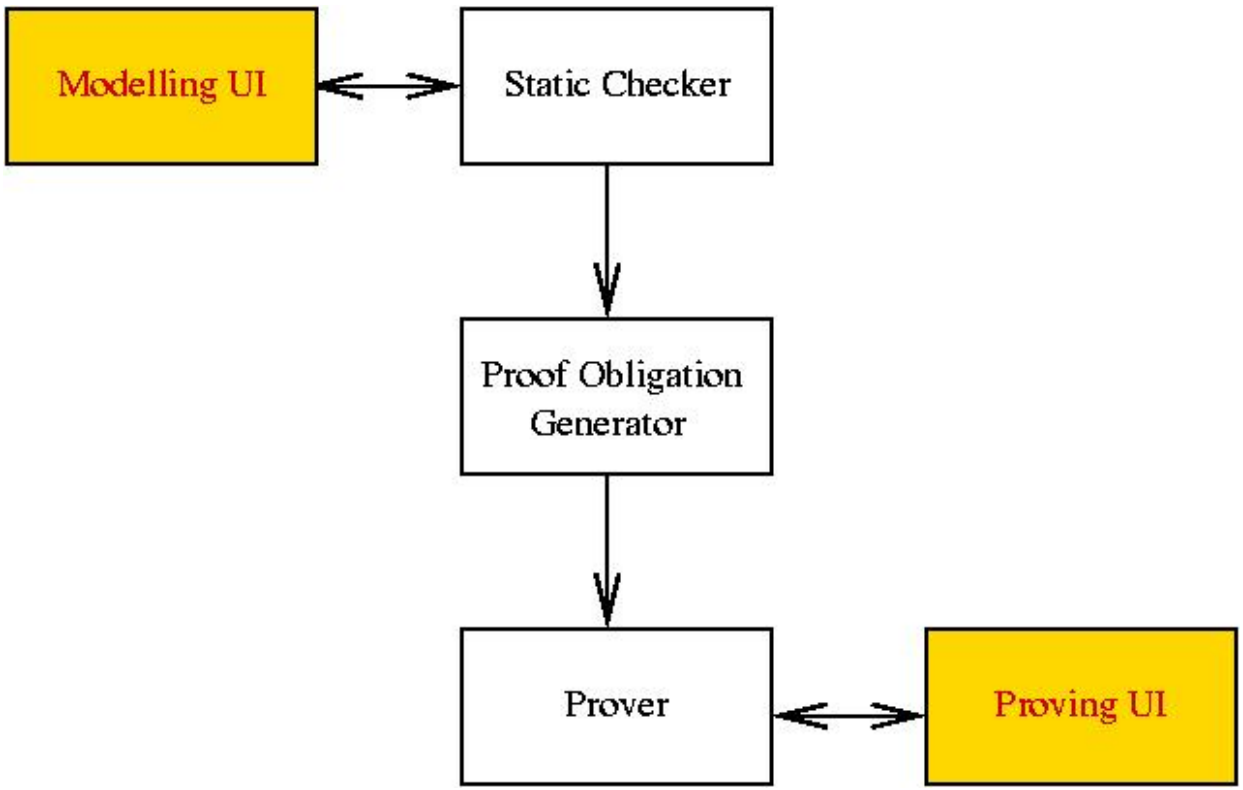
Event-B User Interfaces

Thai Son Hoang

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

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

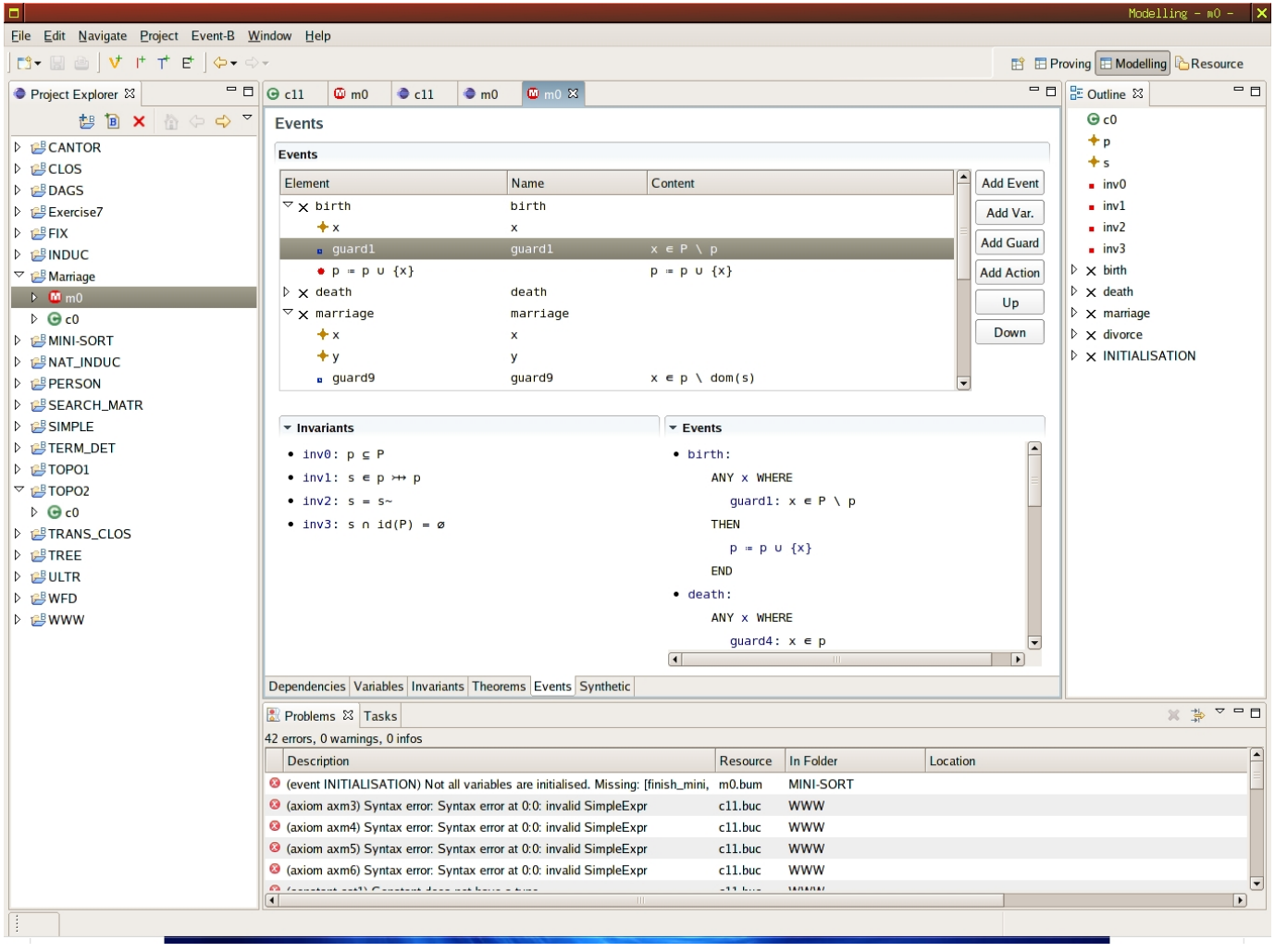
User Interfaces



Outline

Modelling Interface

Contains 3 views and an editor.



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

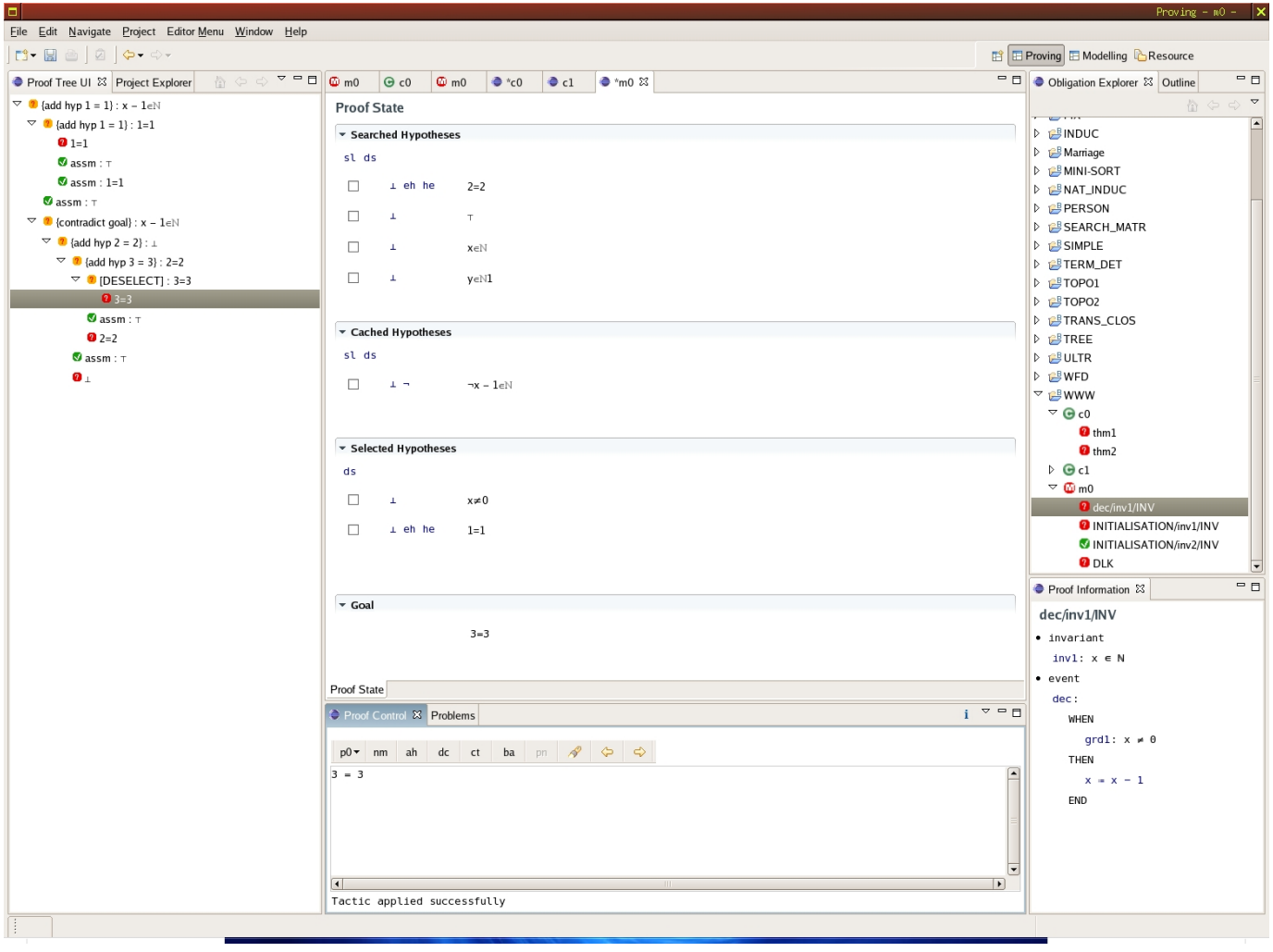
- Explorer View Showing a tree structured view of the workspace.
  - Connects to the *Database*.
  - Connects to the *Event-B Editor* for editing components.
- Event-B Editor A specific editor for creating and modifying event-B components.
  - Multi-page Editor.
  - Form Editor.
  - Connects with the *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.



Rodin

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

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.