A Proposal for a Rodin Proof Planner & Reasoned Modelling Plug-in

Gudmund Grov & Andrew Ireland – Heriot-Watt University
Michael Butler – University of Southampton
Alan Bundy – University of Edinburgh
Cliff Jones – University of Newcastle

Rodin Workshop 16-17 July, 2009

Outline

- Introduce **proof planning** for Rodin.
- Discuss reasoning & modelling interplay.
- Introduce the idea of **reasoned modelling**:
  - reasoning & modelling synergy;
  - new paradigm based on proof planning.

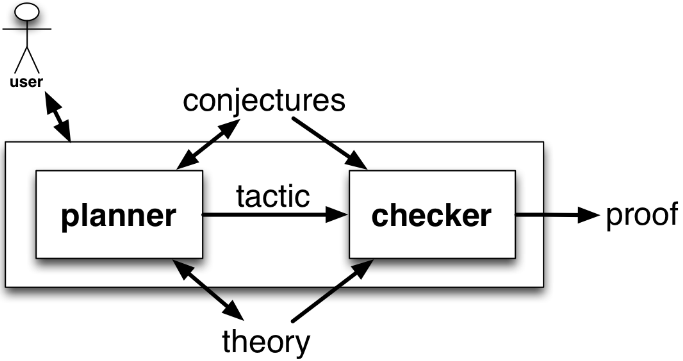
Proof plans

- AI technique for mechanising formal reasoning based upon high level proof patterns

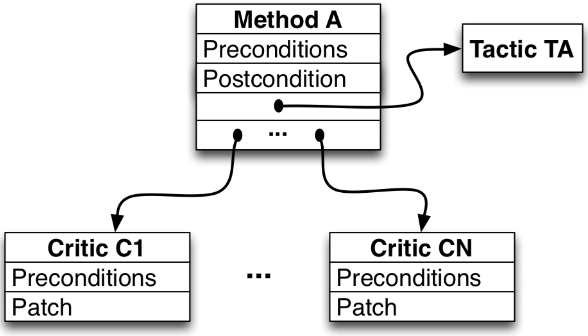
$$\textit{proof plan} = \textit{tactics} + \textit{methods} + \textit{critics}.$$

- *Tactic:* the structure of a proof at the level of primitive inference rules.
- *Method:* a meta-level description of a tactic.
- *Critic:* failure of methods triggers associated critics which suggests proof patches.

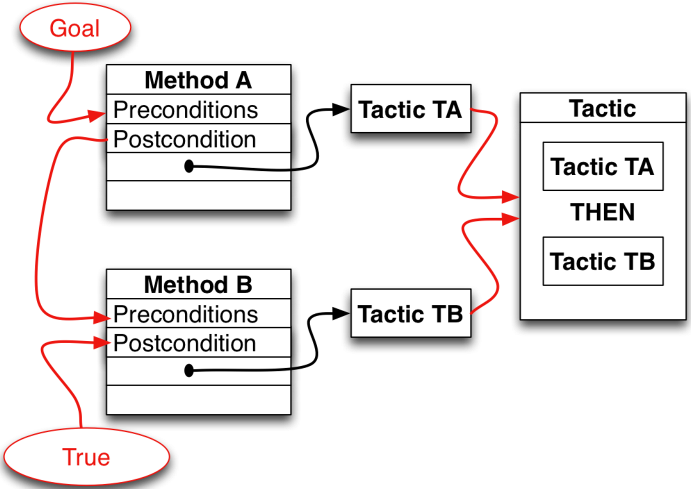
A proof planning system



A proof planning system



A proof planning system



Advantages of proof planning

motivating heuristics & the planning level

- **Reduces** user level **search** by automating the "big steps" within proof development
  - search space at the meta-level smaller than at the object-level.
- Promotes **reuse** of strategies across domains
  - e.g. rippling, recursion to iteration.
- Enables **cooperation** between strategies.
- Greater **flexibility** when using strategies
  - e.g. induction revision, generalisation, lemma speculation;
  - delayed instantiation (middle-out reasoning);
  - productive **use of failure**.

Productive use of failure

- Proof planners carry across extra information of the proof
- A **proof critic** explores this information to patch a faulty conjecture.
- This reduces required user-interaction
- ... and user expertise.

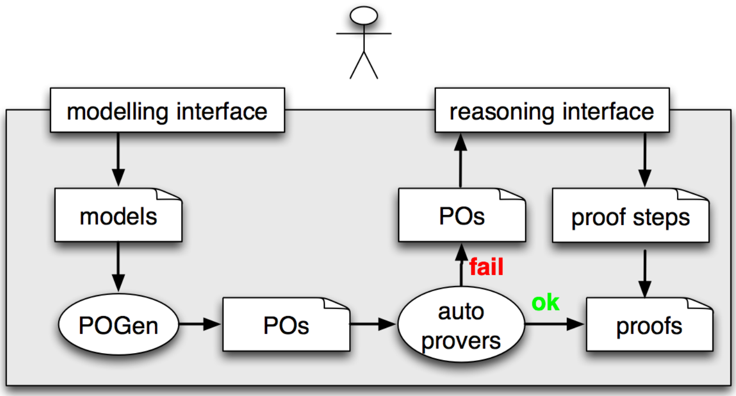
Application of proof planning

- *Mathematical induction:* program verification, synthesis, optimisation, hardware verification, correction of faulty specification.
- *Non-inductive proofs:* summing series, limit theorems, non-standard analysis.
- *Automatic proof patching:* conjecture generalisation, lemma discovery, induction revision, case splitting, loop invariant discovery.

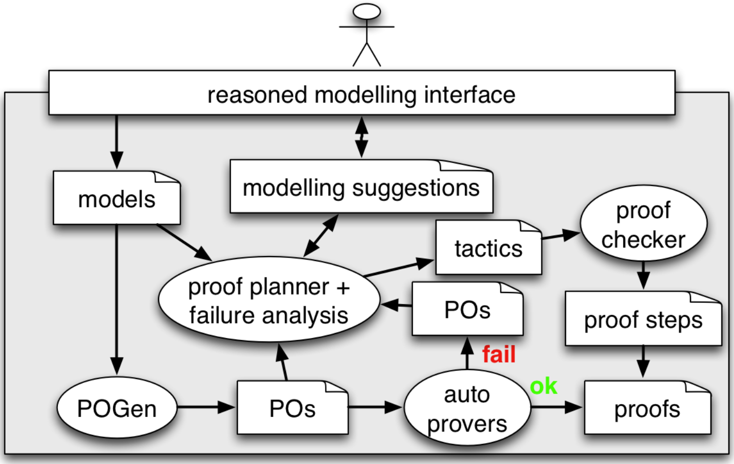
Proof planning in Rodin

- Proof planning has been successful in many domains.
- We believe it can increase proof automation in Rodin.
- Challenge is finding (generic) proof patterns (including critics).

Current Rodin architecture



Reasoned modelling overview



Modelling critics

- Generalise proof critics with modelling suggestions.
- May work as a proof critic (proof patching).
- May also suggest changes to models
  - based on variants of *abduction*
- Examples
  - guard modification (e.g. Abrial's "cars on a bridge")
  - action modification (e.g. Abrial's "cars on a bridge")
  - invariant discovery (e.g. Abrial's "cars on a bridge")
  - gluing invariant discovery (e.g. Butler's Mondex case-study)

A simple cruise-control system

switching the cruise control on

Invariant:

$$\mathtt{inv1} : \mathtt{cc} = \mathtt{on} \Rightarrow \mathtt{brake} = \mathtt{off}$$

Event:

begin
  $\mathtt{act1} : \mathtt{cc} := \mathtt{on}$
end

Proof obligation:

$$cc = on \Rightarrow brake = off \ \vdash \ \{cc \mapsto on\}(cc = on \Rightarrow brake = off)$$
$$cc = on \Rightarrow brake = off \ \vdash \ brake = off$$

A simple cruise-control system

switching the cruise control on

Invariant:

$$\texttt{inv1} : \texttt{cc} = \texttt{on} \Rightarrow \texttt{brake} = \texttt{off}$$

Event:

```
begin
   act1 : cc := on
end
```

Modified event:

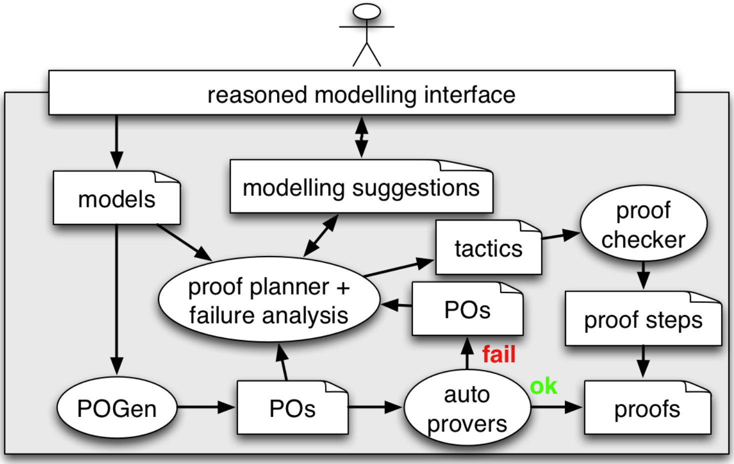
```
when
   grd1 : brake = off
then
   act1 : cc := on
end
```

Proof obligation:

$$\frac{\ldots}{cc = on \Rightarrow brake = off, brake = off \;\vdash\; brake = off}$$

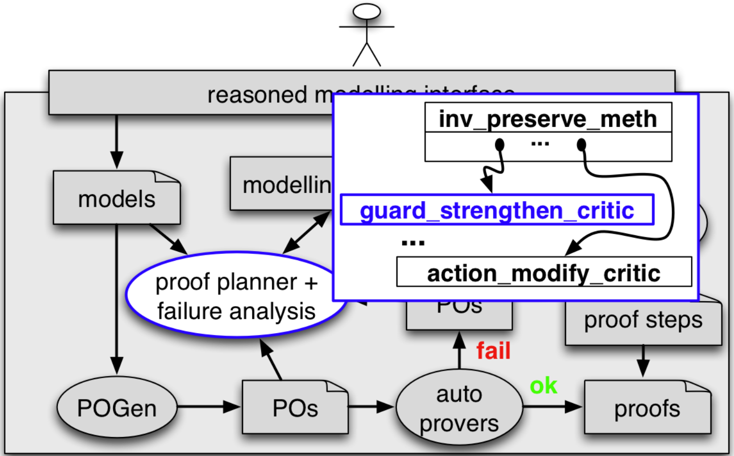
A simple cruise-control system

switching the cruise control on – what happened?



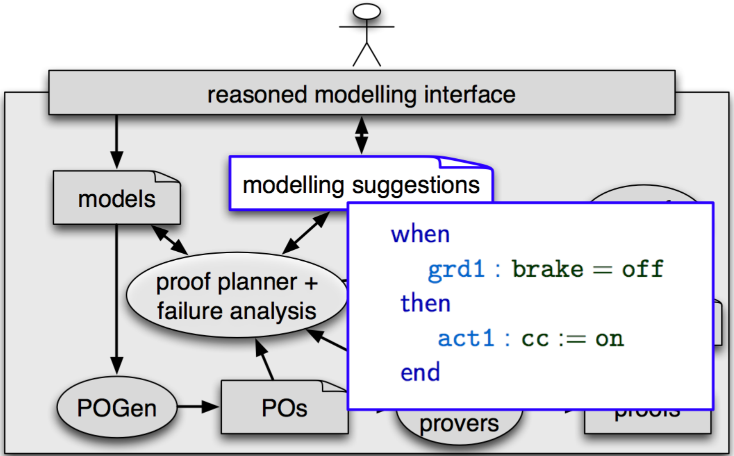
A simple cruise-control system

switching the cruise control on – what happened?



A simple cruise-control system

switching the cruise control on – what happened?



A simple cruise-control system

hitting the brakes

Invariant:

$$\texttt{inv1} : \texttt{cc} = \texttt{on} \Rightarrow \texttt{brake} = \texttt{off}$$

Event:

begin
  $\texttt{act1} : \texttt{brake} := \texttt{on}$
end

Proof obligation:

$$cc = on \Rightarrow brake = off \vdash \{brake \mapsto on\}(cc = on \Rightarrow brake = off)$$
$$brake = off, cc = on \vdash on = off$$

A simple cruise-control system

hitting the brakes

Invariant:

$\texttt{inv1} : \texttt{cc} = \texttt{on} \Rightarrow \texttt{brake} = \texttt{off}$

Event:

begin
$\texttt{act1} : \texttt{brake} := \texttt{on}$
end

Modified event:

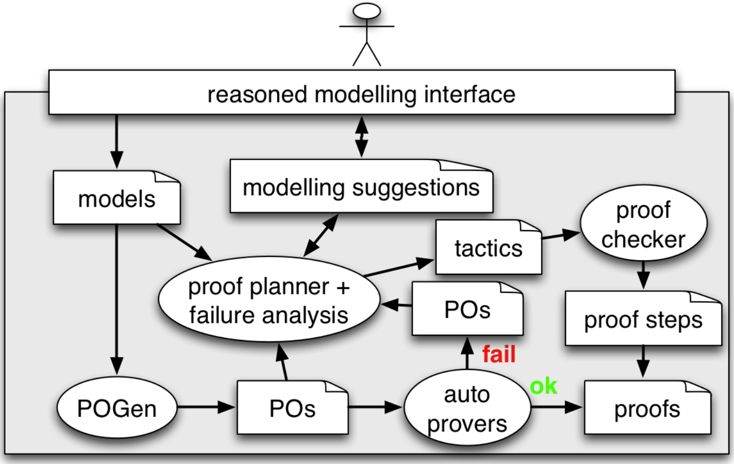
begin
$\texttt{act1} : \texttt{brake} := \texttt{on}$
$\texttt{act2} : \texttt{cc} := \texttt{off}$
end

Proof obligation:

$$\frac{\ldots}{cc = on \Rightarrow brake = off, off = on \quad \vdash \quad on = off}$$

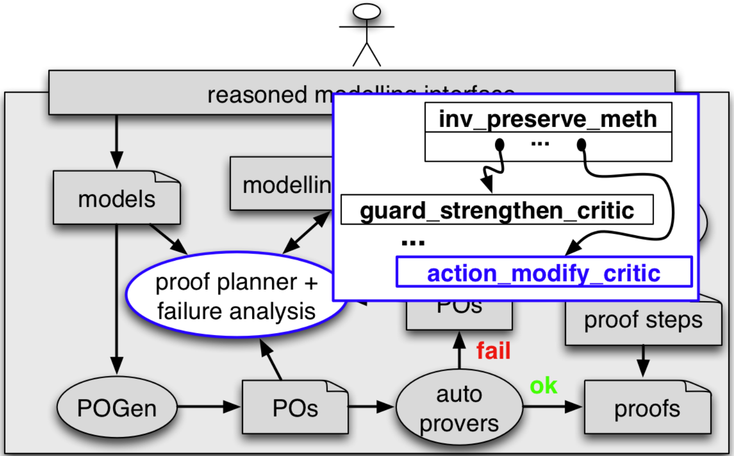
A simple cruise-control system

hitting the brakes – what happened?



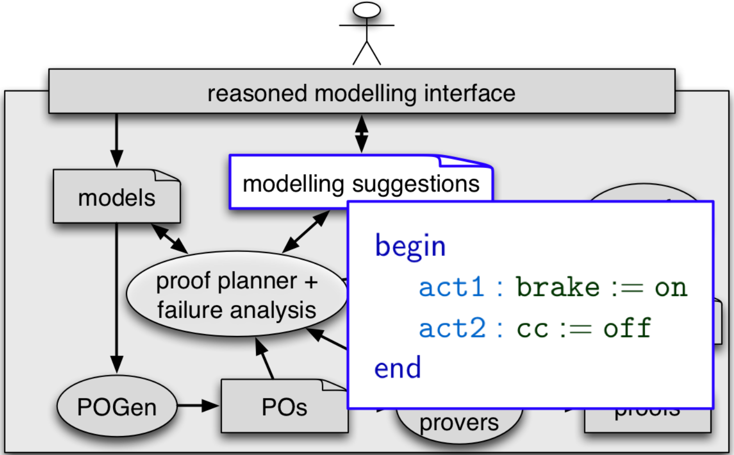
A simple cruise-control system

hitting the brakes – what happened?



A simple cruise-control system

hitting the brakes – what happened?



Beyond failure analysis

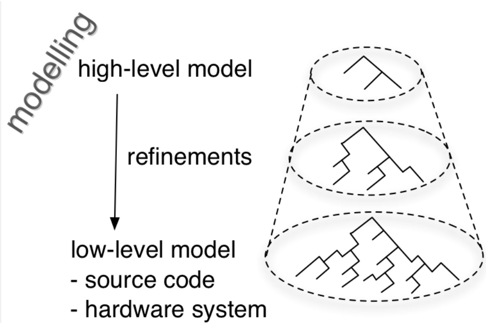
- Event-B
  - strong interplay between modelling & reasoning;
  - modelling guided by reasoning (failures).
- Design patterns
  - high-level modelling patterns;
  - *anti-patterns* – modelling failures and patches.
- Proof plans
  - high-level reasoning patterns;
  - *proof critics* – reasoning failures and patches.
- **Modelling plans**
  - captures interplay between reasoning & modelling;
  - combines reasoning & modelling patterns
  - **modelling critics** – reasoning failures

    .. and reasoning patches & modelling suggestions

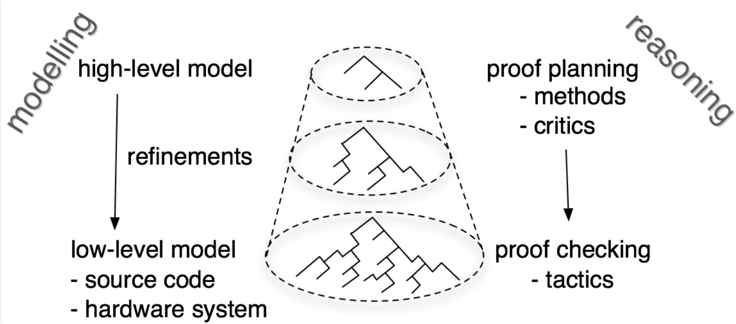
Modelling critics & modelling plans

- Modelling critics
  - subsumes proof critics;
  - contains patches to model;
  - contains heuristics to order patches
    - e.g. priority of variables: $\texttt{cc} < \texttt{brake}$.
- Modelling plans
  - subsumes proof plans;
  - contains modelling patterns;
  - ... with associated proof patterns
  - ... and associated modelling critics

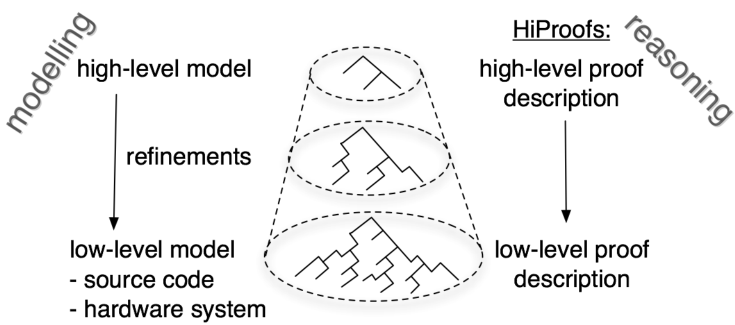
Modelling abstraction & reasoning abstraction



Modelling abstraction & reasoning abstraction



Modelling abstraction & reasoning abstraction



Conclusion

- Introduced proof planning for Rodin.
- Introduced the idea of **reasoned modelling**:
  - a **new paradigm** incorporating:
    - modelling;
    - reasoning;
  - more abstract user interaction:
    - in form of high-level modelling changes;
    - ... and not low-level proof obligations;
  - mechanises features already used in Event-B;
  - **goal:** remove the need of proof expertise.