Refinement Plans for Reasoned Modelling

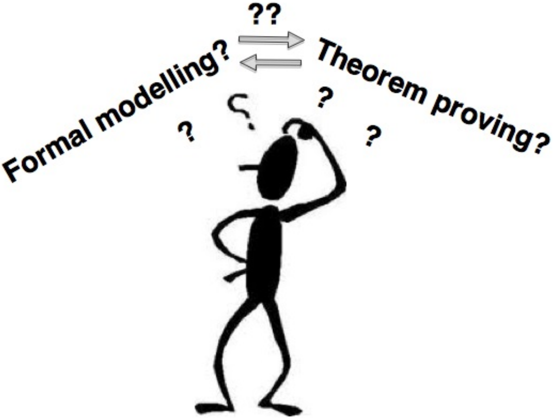
Maria Teresa Llano[1]
Andrew Ireland[1] Gudmund Grov[2]

[1]School of Mathematical and Computer Sciences
Heriot-Watt University

[2]School of Informatics
University of Edinburgh

Rodin Workshop, September, 2010

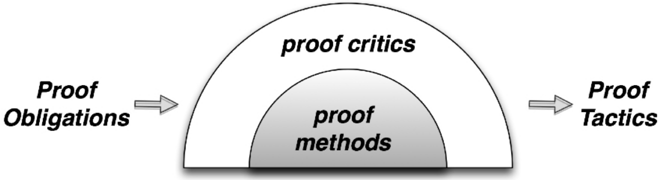
Motivation



Goal: To abstract away from the complexities of proof obligations, providing high-level modelling guidance.

Proof planning

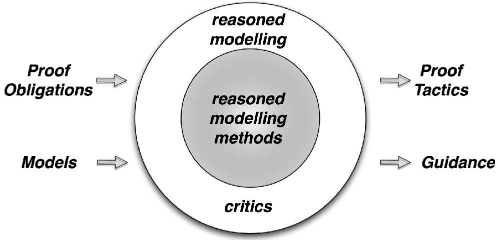
Proof planning uses proof patterns to automate the search for proofs.



- Reusability of proof strategies.
- Automatic proof failure analysis and patching.

Reasoned Modelling

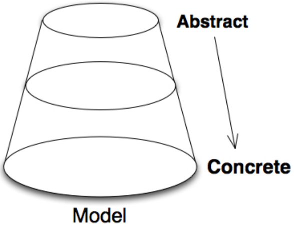
An approach that provides high-level modelling guidance by combining proof and modelling patterns.



- Currently the ideas of reasoned modelling are being developed in Event-B.
- We are working in *Refinement Plans*, a type of reasoned modelling method that focus on refinement.

Refinement in Event-B

Handle the complexity of large systems through the use of abstraction.



Different ways of doing refinement in Event-B:

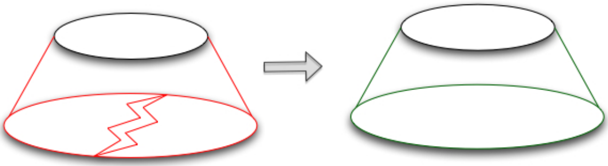
- Variables can be added and/or removed.
- New events can be added.
- Existing events can be split, merged or modified.
- Gluing invariants, etc..
- Plus combinations of all the previous ones.

Refinement Plans

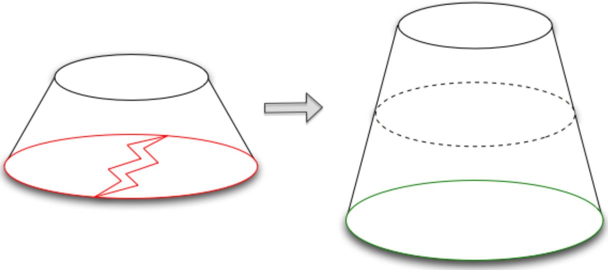
- Classify common patterns of refinement at the modelling level.
- Combine modelling and proof knowledge.
- Detect partial matches of known patterns of refinement in a development.
  - This allows failure analysis of a user's refinement step.
- Provide user guidance in terms of modelling decisions.

Roles of refinement plans (1)

- Correcting refinements:

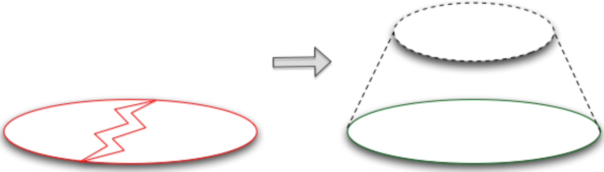


- Layering refinements:

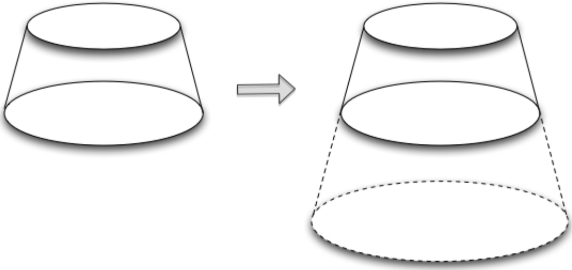


Roles of refinement plans (2)

- Abstracting refinements:



- Suggesting refinements:



Structure of refinement plans

$$
\begin{aligned}
\textbf{refinement plan} &= \textbf{refinement method} \\
&+ \textbf{proof methods} \\
&+ \textbf{critics}
\end{aligned}
$$

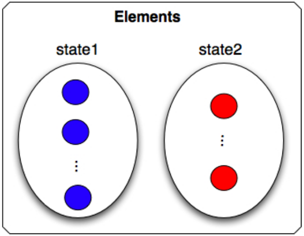
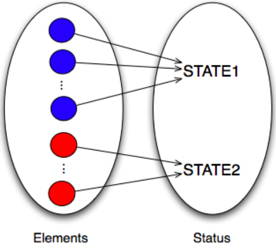
Refinement method

- Describes a common pattern of refinement, i.e.:
  - abstract model
  - concrete model
  - gluing invariant
- Applicability of the pattern is determined by declarative preconditions.
- Through the verification of these preconditions partial or complete instances of the pattern are identified within a user's refinement.
- Partial matches are then handled by the critics mechanism (more on this later).

Refinement method example: ***sets-to-function***

An instance of this pattern requires the:

- abstract model to contain a partition of sets (many variables),
- concrete model to replace the partition with a function (one variable)

Abstract model	**Concrete model**
$state1 \subseteq Elements$ $state2 \subseteq Elements$ $state1 \cap state2 = \varnothing$ $state1 \cup state2 = Elements$	$Status = \{STATE1, STATE2\}$ $fStatus \in Elements \rightarrow Status$ $\underline{state1 = fStatus^{-1}[\{STATE1\}]}$ $\underline{state2 = fStatus^{-1}[\{STATE2\}]}$
	

Proof methods

- Knowing the pattern/structure of the refinement provides us with a better idea about the pattern/structure of the associated proofs, then:

  Proof Methods = Reasoning patterns associated to a refinement method

- Proof methods are used when a user's refinement fully matches with a pattern but it has a set of unproven POs.
- The use of proof methods and the analysis of partial success at the level of proof planning represents future work.

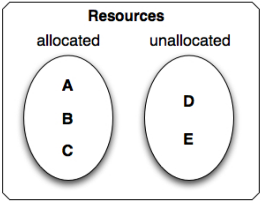
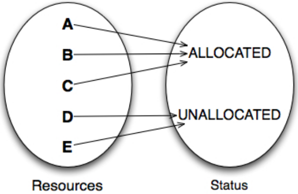
Critics

- Exception handling mechanism.
- If a partial instance of a refinement method is found, critics are applied.
- Applicability is determined by declarative preconditions.
- All preconditions must succeed for a critic to be applicable.
- Modelling guidance to overcome the failure is automatically generated, instances of possible guidance are:
  - Modify/add guards.
  - Modify/add actions.
  - Modify/add (gluing) invariants, etc.
- The selection and decision to apply automatically generated guidance is left to the user.

Example: sets-to-function refinement method applied to a logging system

Precondition: There must exists a set of gluing invariants with the pattern:

$$\textbf{stateVariable} = \textbf{concreteFunction}^{-1}[\{\textbf{stateConstant}\}]$$

Abstract model	**Concrete model**
$allocated \subseteq Resources$	$Status = \{ALLOCATED, UNALLOCATED\}$
$unallocated \subseteq Resources$	$rStatus \in Resources \rightarrow Status$
$allocated \cap unallocated = \varnothing$	
$allocated \cup unallocated = Resources$	*Missing gluing invariants*
	

There exists a partial instance of the refinement method!

Gluing_Invariant_Speculation critic – Key ideas

1. There exists a partial instance of the sets-to-function refinement method for which the gluing invariant precondition fails.
2. There exists a failed guard strengthening PO in the concrete model with the form:

$$\exists failed\_po \in \{\langle \_, \_, \_, PO\rangle \in POs \mid failed\_proof(PO)\}.$$
$$failed\_po = \langle M, E, \_/GRD, (\Delta, \underbrace{stateFunction(x) = Y}_{concrete\ guard} \vdash \underbrace{x \in \{z\}}_{abstract\ guard})\rangle$$

3. There exists an instance of the gluing invariant pattern that makes the failed PO provable:

$$provable\ (\Delta, stateFunction(x) = Y, \underline{\{z\} = stateFunction^{-1}[\{Y\}]} \vdash x \in \{z\})$$

Instantiation of the critic: logging example

1. There exists a partial instance of the sets-to-function refinement method for which the gluing invariant precondition fails. ✔
2. Failed POs with the form "$\Delta, stateFunction(x) = Y \vdash x \in \{z\}$": ✔

$$\ldots, rStatus(r) = ALLOCATED \vdash r \in \{allocated\}$$
$$\ldots, rStatus(r) = UNALLOCATED \vdash r \in \{unallocated\}$$

3. The exists an instance of the gluing invariant pattern that makes the failed PO provable: ✔

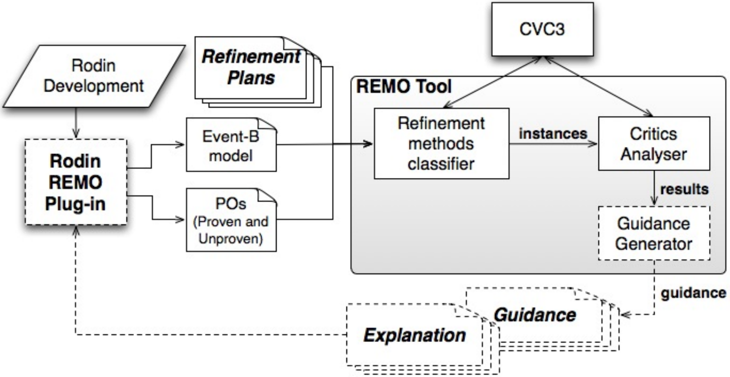
$$provable\ (\ldots, rStatus(r) = ALLOCATED, \underline{\{allocated\} = rStatus^{-1}[\{ALLOCATED\}]} \vdash r \in \{allocated\})$$

(A similar instantiation is given for state unallocated)

All preconditions succeed, then the guidance is the addition of the gluing invariants to the concrete model.

The REMO[1] tool

- Prototype tool developed in OCaml (Objective Caml Language).
- Uses the CVC3 SMT solver to verify preconditions.
- Currently REMO is partially integrated with Rodin.



[1] The REMO acronym follows from REasoned MOdelling

Ongoing and future work

- Further development of refinement plans and their evaluation through case studies.
- Explore the role of refinement plans for:
  - Guiding users in their initial choice of refinement.
  - Suggesting intermediate refinement steps.
- Tool implementation: *REMO*.
- Analysis of faulty user-given invariants through term-synthesis and automated theory formation techniques.
- Discover proof patterns associated to our refinement methods in order to enhance proof automation (through proof methods).

Summary

- We have introduced refinement plans, and outline several areas we believe they can help modellers.
- We have shown an example of the role of refinement plans when correcting a broken refinement step.
- Aim: to provide modelling guidance by automatically analysing specifications that lie just outside a known pattern of refinement.
- Analysis of failure and generation of guidance is automatic; however, final decision is left to the user.
- We believe that this approach will enable us to turn low-level proof-failures into high-level modelling guidance.

Thanks!