Reasoned Modelling Critics: Turning Failed Proofs into Modelling Guidance

Andrew Ireland[1], Gudmund Grov[2], and Michael Butler[3]

[1] School of Mathematical & Computer Sciences, Heriot-Watt University, Edinburgh, EH14 4AS, UK. A.Ireland@hw.ac.uk
[2] School of Informatics, University of Edinburgh, Informatics Forum, Edinburgh, EH8 9AB, UK. ggrov@inf.ed.ac.uk
[3] School of Electronics & Computer Science, University of Southampton, Highfield, Southampton, SO17 1BJ, UK. mjb@ecs.soton.ac.uk

Abstract. The activities of formal modelling and reasoning are closely related. But while the rigour of building formal models brings significant benefits, formal reasoning remains a major barrier to the wider acceptance of formalism within design. Here we propose *reasoned modelling critics* – a technique which aims to abstract away from the complexities of low-level proof obligations, and provide high-level modelling guidance to designers when proofs fail. Inspired by proof planning critics, the technique combines proof-failure analysis with modelling heuristics. Here, we present the details of our proposal and outline future plans.

1 Introduction

The use of mathematical techniques for system-level modelling and analysis brings significant benefits, as well as challenges. While the rigour of mathematical argument can offer early feedback on design decisions, a key challenge centres on *how* feedback derived from the analysis of complex *proof obligations* (POs) can be used to improve design decisions. Such feedback currently requires user intervention, drawing upon skilled knowledge of the subtle interplay that exists between modelling and reasoning. For example, consider a simple cruise control system with the safety requirement that the brakes (*brake*) cannot be on while the cruise control (*cc*) is enabled. This can be formalised as the following invariant:

$$cc = on \Rightarrow brake = off$$

When the driver applies the brakes, i.e. $brake := on$ it must be shown that the safety requirement (invariant) is preserved, creating a PO of the form:

$$brake = off, cc = on \ \vdash \ on = off$$

Note that this PO is false. An (unsafe) solution to this failure would be to restrict the application of the brakes via a guard, i.e. only allow the brakes to be applied when the cruise control is disabled ($cc = off$). Clearly, the desirable solution requires the introduction of an auto disable mechanism, i.e. the cruise control is disabled if the brakes are applied ($cc := \mathit{off}; \mathit{brake} := \mathit{on}$). While the two alternatives do not represent a significant burden to a designer, in general many solutions may arise from proof-failure analysis. In order to make good use of a designer's time, modelling heuristics are needed in order to rank the alternatives.

This need for both proof and modelling heuristics is central to our proposal. We aim to identify common modelling and reasoning patterns, and use these to abstract away from complex POs. This will enable us to automatically provide designers with high-level decision support oriented towards modelling choices. In turn this will increase the productivity of designers as well as the accessibility of formal modelling tools. We plan to implement our proposal through what we call *reasoned modelling critics*.

A key inspiration for our proposal is *proof planning*, a technique for automating the search for proofs through the use of high-level proof outlines, known as *proof plans* [6]. Central to proof planning is its proof-failure analysis and proof patching capabilities [10]. Our proposal aims to build directly upon these features. Specifically by combining the proof-failure analysis capabilities with modelling heuristics we aim to automatically generate modelling guidance. Our longer term aim is to combine the proof plan representation with a complementary notion of *modelling patterns*, which we call *reasoned modelling methods*. This, combined with reasoned modelling critics, are the building blocks of our notion of *reasoned modelling*: the study of the interplay between reasoning and modelling.

While our vision is generic, our starting point is Event-B, which we motivate and introduce together with proof planning in §2. §3 outlines the reasoned modelling critics mechanism and shows its application through examples. We discuss related and future work in §4 and conclude in §5.

2 Background

Event-B and the Rodin toolset Event-B provides a formal framework for modelling discrete complex systems [1], and is mechanized through the Rodin toolset [2]. Event-B promotes an incremental style of formal modelling, where each step of a development is underpinned by formal reasoning. As a result, there is strong interplay between modelling and reasoning and this is partly supported by the Rodin toolset. This interplay requires skilled user interaction, *i.e.* typically a user will analyse failed proofs, and translate the analysis by hand into corrective actions at the level of modelling. This is exemplified in e.g. [1, 8]. Typical corrective actions include strengthening invariants and guards or modifying actions. Our aim is to provide high-level decision support, by automating the generation, filtering and ranking of modelling suggestions. Event-B models and POs are closely aligned [1], whilst Rodin [2] is an extensible framework. Event-B and the Rodin toolset thus represent a unique opportunity for us to investigate reasoned modelling.

We will use the term *models* for both Event-B *contexts* (the static part) and *machines* (the dynamic part). We will only use the basic features of Event-B and use standard notation:

EVENT $\langle$*name*$\rangle \mathrel{\widehat{=}}$ **BEGIN** $\langle$*general substitution*$\rangle$ **END**
EVENT $\langle$*name*$\rangle \mathrel{\widehat{=}}$ **WHEN** $\langle$*guard*$\rangle$ **THEN** $\langle$*general substitution*$\rangle$ **END**

where the event is only executed when the guard holds. Moreover, we use the term *action* for the generalised substitution, while INITIALISATION is a special event without guards defining the initial state.

Proof planning and proof critics mechanism Proof planning builds upon the tactic-based tradition of theorem proving, where primitive proof steps are packaged-up into programs known as *tactics*. Starting with a set of general purpose tactics, plan formation techniques are used to construct a customised tactic for a given conjecture. The search for a customised tactic is constrained by a set of *methods*, collectively known as a *proof plan*. Using preconditions, each method specifies the applicability of a general purpose tactic. Having explicit preconditions provides insights when proof planning fails. The proof *critics* mechanism [10] enables "interesting" failures at the level of precondition evaluation to be represented. For a given method a number of critics will typically exist. Each critic represents an alternative generic proof patch, e.g. conjecture generalisation. The analysis of a specific failure, and subsequent proof planning, provides guidance in the selection and instantiation of a generic proof patch. In Fig 1 we present a method and critic. These are deliberately simple – their role is to illustrate the basic critics mechanism. Proof patching has been applied successfully to the problems of inductive conjecture generalisation and lemma discovery [5], as well as loop invariant discovery [11]. Proof critics have also been developed for *abductive reasoning*, i.e. the speculation of hypotheses in order to explain a consequence, and applied to the problem of patching faulty conjectures [12]. The proof critics mechanism provides the starting point for reasoned modelling critics. Moreover, in terms of exploiting existing critics, we see the work on abduction playing an important role in our reasoned modelling critics.

3 From proof critics to reasoned modelling critics

As described above, our reasoned modelling critics will extend the existing proof critics mechanism. This extension will involve two key components:

1. Firstly, in order to combine proof and modelling, our new critics need access to models as well as POs. In addition, critics will now have the option of providing modelling guidance, as well as proof patches.
2. Secondly, both methods and critics are associated with preconditions. We need to extend the meta-language of critics to allow us to represent preconditions which specify modelling heuristics.

To support access to models and modelling guidance, an extended critic schema will be required. Our proposed extension is described in Fig 2. In order to illustrate the new schema and the kind of meta-language extensions we propose for

method (rewrite)

INPUT: $\Delta \vdash G$

PRECONDITIONS:

1. *exp_at(G, Pos, L)*
2. $\mathit{rewrite\_rule}(C \Rightarrow L :\Rightarrow R)$
3. $\mathit{provable}(\Delta \vdash C)$

EFFECTS:

$\mathit{replace\_at}(Pos, R, G, NewG)$

OUTPUT:

$[\Delta \vdash NewG]$

critic (casesplit)

INPUT: POs

PRECONDITIONS:

1. $\exists\ \mathit{failed\_po} \in \{\langle \mathit{rewrite}, \_, PO\rangle \in POs |$ $\mathit{failed\_proof}(PO)\}$. $\mathit{failed\_po} = \langle \_, Pre, \_\rangle$
2. $\langle\ \mathit{exp\_at}(\_, \_, \_),$ $\mathit{rewrite\_rule}(C \Rightarrow \_ :\Rightarrow \_),$ $\mathit{fail}\ \rangle \in Pre$

PATCH:

$\mathit{insert\_casesplit}(C, PO)$

The meta-predicates used above are defined in Fig 3. Note that Δ denotes a list of hypotheses, while G and $NewG$ denote single goal formula. Note also the use of Prolog meta-variables, in particular the use of "_" for anonymous variables. The scope of meta-variables extends across all slots of the method and critic schemas.

Methods: A method takes as input a PO. The applicability of a method is determined by evaluating its associated preconditions. In the case of our example method, i.e. **rewrite**, there are three preconditions, i.e. i) there exists an expression L at position Pos within the goal G, ii) there exists a rewrite rule such that L matches the left-hand side of the rule, and iii) any condition C attached to the rule is provable within the proof context. If the preconditions of a method succeed, then the output POs (potentially empty) are computed via the effects slot. In the case of **rewrite**, the effects slot applies the selected rule, *i.e.* the expression at position Pos within the goal G is replaced by R (rule right-hand side) giving rise to new goal $NewG$ from which the output PO is constructed.

Critics: During proof planning, both the success and failure (*fail*) of a method's preconditions are recorded. In particular, when a method fails a set of partial instantiations of the preconditions are associated with the PO. In the **casesplit** critic above, this set is denoted by *Pre*. The critics mechanism uses *Pre* to search for patchable exceptions to the method. If the pattern represented by a given critic's preconditions matches with a recorded failure pattern then the associated patch is applied. This is illustrated above where the **casesplit** critic associates the failure (*fail*) of precondition 3 of the **rewrite** method with a casesplit patch. In general, a critic may require multiple failures in order to trigger a patch. In this way, critics can apply both local and global proof-failure analysis.

Fig. 1. An example proof method and critic

critic (*name*)
  INPUTS:
    PO_SET *POs*
    MODEL_SET *Ms*
  PRECONDITIONS:
    precondition 1
    ...
    precondition N
  OUTPUTS:
    PATCH *proof patch (optional)*
    GUIDE *modelling guidance (optional)*

The above schema represents our proposed refinement to the proof critic schema:

– The INPUT slot has been extended to include models, as well as proof obligations. Note that the schema allows for a critic to access multiple models. As a consequence, critics can be developed that consider the internal consistency of individual models as well as refinements and decompositions.
– The declarative PRECONDITIONS determine the applicability of the critic. Preconditions have access to proof obligations as well as models.
– The OUTPUT slot has been extended to include modelling guidance (GUIDE), as well as proof patches (PATCH). While proof patches are automatically applied, it is envisaged that modelling guidance will be communicated to the designer so as to inform their decision making.

As with the original proof critics mechanism, the output slot becomes instantiated as a side-effect of the instantiation of the input and precondition slots.

Fig. 2. Reasoned modelling critic schema

reasoned modelling, we develop below some critics and show their application to simple control system problems.

3.1 Guidance for invariant proof failures via local analysis

Typically, a proof failure arising from an inconsistency in a model may be overcome in a number of ways. For example, consider an invariant which takes the form of an implication, i.e. $X \Rightarrow Y$. If a corresponding invariant proof fails, then the failure may be overcome by revising the model so that either,

1. X is false, or
2. Y is true.

At the level of modelling these effects can be achieved by changing the guards or actions associated with the invariant proof, or the invariant itself. Here we focus on *local* changes to an event, i.e. changes to guards and actions, and delay the discussion of global analysis and invariant changes until §3.3. In general,

$\mathit{disjoint\_sub}(S_1, S_2)$	$\widehat{=}\ \mathit{dom}(S_1) \cap \mathit{dom}(S_2) = \varnothing$
$\mathit{failed\_proof}(P)$	$\widehat{=}$ *"P is a PO which is provably false or has failed to be proven"*
$\mathit{provable}(P)$	$\widehat{=}$ *"P is a PO that is provable"*
$\mathit{max}(A)$	$\widehat{=}\ \epsilon\ x.\ (x \in A \wedge \forall y \in A.\ y \leq x)$
$\mathit{exp\_at}(X, Y, Z)$	$\widehat{=}$ *"Z is the subexpression at position Y within expression X"*
$\mathit{rewrite\_rule}(X \Rightarrow Y :\Rightarrow Z)$	$\widehat{=}$ *"A conditional rewrite rule"*
$\mathit{pri\_var}(V, M)$	$\widehat{=}$ *"Priority of variable V within model M"*
$\mathit{priority}(S, M)$	$\widehat{=}\ \mathit{max}(\{\mathit{pri\_var}(v, M) \mid v \in \mathit{dom}(S)\ \})$
$\mathit{replace\_at}(W, X, Y, Z)$	$\widehat{=}$ *"Z is obtained by replacing the subexpression at position W within Y by X"*
$\mathit{sub2act}(\{V \mapsto D\})$	$\widehat{=}\ V := D$
$\mathit{sub2act}(\{V_1 \mapsto D_1, \cdots, V_n \mapsto D_n\})$	$\widehat{=}\ \mathit{sub2act}(\{V_1 \mapsto D_1\}) \mid\mid \cdots \mid\mid \mathit{sub2act}(\{V_n \mapsto D_n\})$
$\mathit{sub2grd}(\{V \mapsto D\})$	$\widehat{=}\ V = D$
$\mathit{sub2grd}(\{V_1 \mapsto D_1, \cdots, V_n \mapsto D_n\})$	$\widehat{=}\ \mathit{sub2grd}(\{V_1 \mapsto D_1\}) \wedge \cdots \wedge \mathit{sub2grd}(\{V_n \mapsto D_n\})$
$\mathit{guards}(E)$	$\widehat{=}$ *"Conjunction of guards of event E"*
$\mathit{sat}(P)$	$\widehat{=}$ *"The predicate P is satisfiable"*
$\mathit{generalisable}(H)$	$\widehat{=}\ \exists r, e_1, e_2, g_1, g_2.\ \langle e_1, g_1\rangle \in H\ \wedge \langle e_2, g_2\rangle \in H \wedge e_1 \neq e_2 \wedge r \Rightarrow g_1 \wedge r \Rightarrow g_2 \wedge \mathit{sat}(r)$
$\mathit{generalise}(H)$	$\widehat{=}$ *"The weakest r such that sat(r) and for the most* $\langle e, g\rangle \in H,\ r \Rightarrow p$ *(for at least 2 events)"*
$\mathit{add\_guard}(G, E, M)$	$\widehat{=}$ *"Adds guard G to event E of model M"*
$\mathit{add\_action}(A, E, M)$	$\widehat{=}$ *"Adds action A to event E of model M"*
$\mathit{add\_invariant}(I, M)$	$\widehat{=}$ *"Adds invariant I to model M"*
$\mathit{insert\_casesplit}(C, P)$	$\widehat{=}$ *"Progress the proof of P by a casesplit on X"*

Note that S denotes a substitution, while H denotes a set of event-guard pairs.

Fig. 3. Meta-terms for reasoned modelling critics

proof-failure analysis will generate a large number of proof patches. But not all proof patches will make sense in terms of modelling. To address this problem we wish to exploit modelling knowledge in order to rank the proof patches that are offered as guidance. To achieve this we envisage designers providing meta-data in addition to their models. The preconditions of our new critics can then exploit modelling heuristics as well as proof-failure heuristics.

To illustrate this idea, we develop two critics below. Each critic targets a different pattern of invariant proof failure, as outlined above. In addition, the critics exploit a notion of variable priority – which is based upon the observation that not all variables within a model may carry equal importance. Crucially, such "meta-data" must be supplied by the designer. For example, being able to brake is clearly more important than driving with the cruise control enabled. This notion of *priority* can be expressed as a heuristic, and used to rank or even filter modelling suggestions. Informally, where proof-failure analysis suggests that the

```
critic (priority action speculation)
  INPUTS:
      PO_SET $POs$
      MODEL_SET $\{M\}$
  PRECONDITIONS:
      1. $\exists failed\_po \in \{\langle \_, \_, \_, PO\rangle \in POs \mid failed\_proof(PO)\}.$
             $failed\_po = \langle M, E, \_/INV, (\Delta, X \Rightarrow Y \vdash \sigma(X \Rightarrow Y))\rangle$
      2. $\exists \tau \in sub.\ disjoint\_sub(\tau, \sigma) \ \wedge\ provable(\Delta \vdash (\tau \cup \sigma)X \Rightarrow false)$
      3. $priority(\tau, M) < priority(\sigma, M)$
  OUTPUTS:
      GUIDE $add\_action(sub2act(\tau), E, M)$
```

Note that *sub* denotes the set of all substitutions, and elements of PO_SET are quadruples, i.e. model identifier, event identifier, PO label/type, and PO. Note also that the scope of the quantification extends across the critic slots.

Fig. 4. A reasoned modelling critic based on a priority heuristic to modify an action

value of a variable should be changed within the context of a given event, we adopt the following heuristics:

H1: If the priority of the candidate variable is lower than the priorities of all the variables updated by the event, then it is strongly suggestive that the change should be achieved via a new action.

H2: If the priority of the candidate variable is higher than the priorities of all the variables updated by the event, then it is strongly suggestive that the change should be achieved via a new guard.

Where priorities are the same, no ranking of the alternatives will be provided. We represent heuristics H1 and H2 as critics in Fig 4 and Fig 5 respectively. The meta-logical terms that appear within the preconditions of these critics are defined in Fig 3. Note that meta-logical predicates, such as *priority*, require input from designers, as mentioned above. Both critics are applicable where an invariant proof has failed, and specifically where the invariant takes the form of an implication. We focus on solutions that involve the addition of either guards or actions, where guards are restricted to equalities. Later we discuss how these basic critics could be generalised.

The critic representing H1 (Fig 4) has three preconditions. The first precondition holds if there exists a PO of the required form that has failed to be proved, i.e. an invariant PO of the form $X \Rightarrow Y$, which is associated with the event E where substitution σ represents the actions corresponding to E. The second precondition holds if there exists a substitution (τ) which falsifies the antecedent, where τ and σ are disjoint. The third precondition expresses a modelling constraint, i.e. the variable(s) introduced by the new substitution have lower priority than the variable(s) updated by the existing actions. If the preconditions hold, then the guidance provided takes the form of a guard – constructed from τ.

```
critic (priority guard speculation)
  INPUTS:
      PO_SET $POs$
      MODEL_SET $\{M\}$
  PRECONDITIONS:
      1. $\exists failed\_po \in \{\langle \_, \_, \_, PO\rangle \in POs \mid failed\_proof(PO)\}.$
              $failed\_po = \langle M, E, \_/INV, (\Delta, X \Rightarrow Y \vdash \sigma(X \Rightarrow Y))\rangle$
      2. $\exists \tau \in sub.\ disjoint\_sub(\tau, \sigma) \ \wedge\ provable(\Delta \vdash (\tau \cup \sigma)Y)$
      3. $priority(\sigma, M) < priority(\tau, M)$
  OUTPUTS:
      GUIDE $add\_guard(sub2grd(\tau), E, M)$
```

Fig. 5. A reasoned modelling critic based on a priority heuristic to modify a guard

The critic representing H2 (Fig 5) again has three preconditions, but illustrates a slightly different failure analysis pattern. That is, instead of making the antecedent false, the analysis tries to make the consequent true. Note that the third precondition ensures that the variable(s) introduced by the new substitution are of a higher priority than the variable(s) updated by the existing actions. Here the guidance provided takes the form of an action – constructed from τ.

3.2 The cruise control system in Event-B

To illustrate the application of critics introduced above, we will first formalise in Event-B the cruise-control system that was outlined in §1. Recall that the status of the brakes is represented by a variable *brake*, while the status of the cruise control is represented by the variable *cc*. Both variables are of type $Val = \{on, off\}$, where $on \neq off$. The variable *brake* is *on* if the user currently presses the brake (*pressbrake*), and *off* otherwise. While the variable *cc* is *on* if the cruise control is enabled (*enable_cc*), and *off* otherwise. The system invariant is represented as follows:

$$inv1:\ cc = on \Rightarrow brake = off$$

and initially both variables are *off*:

$$\text{INITIALISATION} \mathrel{\widehat{=}} \textbf{BEGIN}\ brake := off \mid\mid cc := off\ \textbf{END}$$

We assume that the following priority relationship between the variables has been specified by the designer:

$$pri\_var(cc, cruise\text{-}control) < pri\_var(brake, cruise\text{-}control)$$

```
critic (priority guard speculation)
  INPUT
      PO_SET $pos$
      MODEL_SET { $cruise\text{-}control$ }
  PRECONDITIONS:
      1. $\langle cruise\text{-}control, enable\_cc, inv1/INV, po\rangle \in pos \wedge$
         $po = \Big(cc = on \Rightarrow brake = off \vdash \{cc \mapsto on\}(cc = on \Rightarrow brake = off)\Big) \wedge$
         $failed\_po\Big((cc = on \Rightarrow brake = off \vdash brake = off\Big)$
      2. $disjoint\_sub(\{brake \mapsto off\}, \{cc \mapsto on\}) \wedge$
         $provable\Big(cc = on \Rightarrow brake = off \vdash \{brake \mapsto off, cc \mapsto on\}(brake = off)\Big)$
      3. $priority(\{cc \mapsto on\}, cruise\text{-}control) < priority(\{brake \mapsto off\}, cruise\text{-}control)$
  OUTPUTS:
      GUIDE $add\_guard(brake = off, enable\_cc, cruise\text{-}control)$
```

Fig. 6. An instantiation of the 'priority guard speculation' critic.

Failure and modelling suggestion 1 The model contains events that control the brakes and cruise control. The consistency proofs of invariant *inv1* over these events fails in two cases. Firstly, the event to enable the cruise control is defined as follows:

$$\textbf{EVENT}\ \text{enable\_cc} \mathrel{\widehat{=}} \textbf{BEGIN}\ cc := on\ \textbf{END}$$

This gives rise to the following proof obligation:

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

which can be reduced to the unprovable goal:

$$cc = on \Rightarrow brake = off \vdash brake = off.$$

However, the preconditions of the 'priority guard speculation' critic given in Fig 5 hold, and an instantiation of this critic for this example is shown in Fig 6 (after some simplification and unfolding). This instantiation suggests the addition of a new guard of the form $brake = off$. If the suggestion is accepted by the designer, then the updated event takes the form:

$$\textbf{EVENT}\ \text{enable\_cc} \mathrel{\widehat{=}} \textbf{WHEN}\ brake = off\ \textbf{THEN}\ cc := on\ \textbf{END}$$

Failure and modelling suggestion 2 The second failure with respect to *inv1* occurs when a driver presses the brakes, as defined by the event:

$$\textbf{EVENT}\ \text{pressbrake} \mathrel{\widehat{=}} \textbf{BEGIN}\ brake := on\ \textbf{END}$$

```
critic (priority action speculation)
  INPUT
    PO_SET $pos$
    MODEL_SET { $cruise\text{-}control$ }
  PRECONDITIONS:
    1. $\langle cruise\text{-}control, pressbrake, inv1/INV, po\rangle \in pos \wedge$
       $po = \Big(cc = on \Rightarrow brake = off \vdash \{brake \mapsto on\}(cc = on \Rightarrow brake = off)\Big) \wedge$
       $failed\_po((cc = on, brake = off \vdash false)$
    2. $disjoint\_sub(\{cc \mapsto off\}, \{brake \mapsto on\}) \wedge$
       $provable\Big(cc = on \Rightarrow brake = off \vdash \{brake \mapsto on, cc \mapsto off\}((cc = on) \Rightarrow false)\Big)$
    3. $priority(\{cc \mapsto off, cruise\text{-}control) < priority(\{brake \mapsto on\}, cruise\text{-}control)$
  OUTPUTS:
    GUIDE $add\_action(cc := off, pressbrake, cruise\text{-}control)$
```

Fig. 7. An instantiation of the 'priority action speculation' critic.

This creates the following proof obligation:

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

which, when simplified, becomes false:

$$cc = on, brake = off \vdash false.$$

In this case the 'priority action speculation' critic given in Fig 4 triggers, and Fig 7 shows the corresponding instantiation. As a result, the following updated *pressbrake* event is suggested to the designer:

$$\textbf{EVENT}\ \text{pressbrake} \mathrel{\widehat{=}} \textbf{BEGIN}\ brake := on\ ||\ cc := off\ \textbf{END}$$

Generalisations of the critics An alternative definition of the invariant is

$$inv2:\ \ brake = on \Rightarrow cc = off$$

which is the contrapositive of *inv1*. To keep our presentation as simple as possible, our critics will not trigger on POs arising from *inv2*. However, this limitation could be overcome by generalising the failure analysis, i.e. allowing the critics to consider the alternatives of falsifying an antecedent and validating a consequent. This could be achieved by changing precondition 2 as follows:

$$provable\big(\varDelta \vdash (\tau \cup \sigma)X \Rightarrow false\big) \vee provable\big(\varDelta \vdash (\tau \cup \sigma)Y\big)$$

Another limitation is that currently we only consider the addition of guards that take the form of equalities. The 'priority guard speculation' critic could be extended to capture a more general notion of a guard. However, this would require a new notion of priority, which is currently not defined for arbitrary terms (only variables and substitutions). Finally, global consistency issues have been ignored. When an action is modified, the validity of other invariants may change, or a deadlock may arise. The critics should be updated with such consistency checks. Depending on how global analysis is dealt with – as discussed below – the 'priority action speculation' critic of Fig 4 may have to also include such a *consistency* predicate.

3.3 From local to global modelling suggestions

As mentioned earlier, we envisage that our critics will support global analysis, as well as the local analysis illustrated via the cruise control example. For instance, when attempting to prove the internal coherence of a machine, local analysis may suggest the need to add guards across multiple events. Taking a more global perspective, these local suggestions may reveal a more general modelling suggestion. For instance, if a formula can be found that is logically weaker than all the suggested guards, then the formula could be suggested as a new invariant.

A concrete example of this kind of global analysis is found in Abrial's 'Cars on a Bridge' example [1][1]. In this example a single-laned bridge between an island and the mainland is modelled. Since it is single-laned, cars can only travel in a given direction at a given time – and this is controlled by two traffic-lights: *ml_tl* is the traffic light on the mainland; and *il_tl* is the traffic light on the island. The lights can have two colours – *green* and *red* – and it is assumed that $green \neq red$. Our discussion will focus on two events: *IL_out*, where cars enter the bridge from the island; and *ML_out*, where cars enter the bridge from the mainland. Without giving all the details[2], two different invariant proofs fail for these events. The corresponding POs have the following form:

$$\ldots, il\_tl = green \vdash \sigma_1(ml\_tl = green \Rightarrow \ldots) \quad where\ ml\_tl \notin dom(\sigma_1) \tag{1}$$

$$\ldots, ml\_tl = green \vdash \sigma_2(il\_tl = green \Rightarrow \ldots) \quad where\ il\_tl \notin dom(\sigma_2) \tag{2}$$

A local analysis of each of these failures suggests making the antecedent false, by finding a substitution τ such that:

$$disjoint\_sub(\tau, \sigma_{\{1,2\}}) \ \wedge \ provable(\Delta \vdash (\tau \cup \sigma_{\{1,2\}})X \Rightarrow false)$$

In (1), τ will be instantiated to $\{ml\_tl \mapsto red\}$ while in (2), τ will be instantiated to $\{il\_tl \mapsto red\}$. However, there is a common solution to these failures. Firstly, we assume that the substitutions are turned into guards (by *sub2grd*). Now, if an event is constrained by the guards $G_1, \cdots, G_n$, then adding an additional guard G_{n+1} to patch a failure is equivalent to adding the weaker guard $G_1 \wedge \cdots \wedge G_n \Rightarrow G_{n+1}$ (via modus ponens). Thus the suggested additional guards

[1] The example is modified slightly with respect to types.

[2] These details can be found in [1] and `http://www.event-b.org`.

can be “weakened” in this way with the existing guards of the events. For this particular example, we ignore all guards except those shown in (1,2) – the other guards are irrelevant for this failure. For example, from (1) the guard $ml\_tl = red$ is derived and “weakened” to give $il\_tl = green \Rightarrow ml\_tl = red$. We then try to find a common solution for (1,2), i.e. an r such that:

$$\begin{gathered} r \Rightarrow il\_tl = green \Rightarrow ml\_tl = red \\ r \Rightarrow ml\_tl = green \Rightarrow il\_tl = red. \end{gathered}$$

Moreover, we require that r is satisfiable, i.e. the trivial solution $r = false$ is not considered. The only valid such generalisation r (modulo equivalences) is:

$$il\_tl = red \vee ml\_tl = red$$

which is suggested as an invariant. This represents a key safety requirement – at all times at least one traffic-light is red.

```
critic (multiple failure invariant speculation)
  INPUTS:
     PO_SET $POs$
     MODEL_SET $\{M\}$
  PRECONDITIONS:
     1. $\exists h \subseteq hs.\ h = \{\langle E, G\rangle \mid \exists po \in pos.\ \langle \_, E, \_/INV, po\rangle \in POs \ \wedge\ failed\_proof(po)$
                              $\wedge\ po = (\Delta, X \Rightarrow Y \vdash \sigma(X \Rightarrow Y))$
                              $\wedge\ \exists \tau \in sub.\ disjoint\_sub(\tau, \sigma) \ \wedge\ provable(\Delta \vdash (\tau \cup \sigma)X \Rightarrow false)$
                              $\wedge\ G = guards(E) \Rightarrow sub2grd(\tau)\ \}$
     2. $generalisable(h) \wedge \exists i \in forms.\ i = generalise(h)$
  OUTPUTS:
     GUIDE $add\_invariant(i, M)$
```

Note that hs denotes the set of all event-guard pairs, while pos and $forms$ denote the sets of all proof obligations (sequents) and formulae respectively.

Precondition 1 holds if a local analysis suggests that proof failures can be overcome by the introduction of additional guards.
Precondition 2 holds if the suggested guards can be generalised to give an invariant.

Fig. 8. A global reasoned modelling critic suggesting an invariant from multiple failures

With regards to realizing such global reasoning, we are considering two distinct alternatives via our proposed critic schema:

1. Combine both local and global analysis of the available POs within a single critic, as illustrated in Fig 8.

2. Firstly use critics to perform local analysis of the available POs, recording any modelling suggestions. Secondly, use separate critics to perform a global analysis, exploiting the results of the local analysis.

From a scientific point of view, the approaches are not very different – and should be seen more as an implementation issue. Experimentation will be required in order to determine the most practical approach.

4 Related and future work

Our notion of *reasoned modelling* represents a new paradigm for exploring the interplay between reasoning and modelling. As evident from the above discussion, our general ideas on reasoned modelling are strongly influenced by the proof planning paradigm [6], while the particular work discussed here follows from the notion of *proof critics* [10]. Currently, Event-B users have to manually analyse failed proof attempts and patch their models, e.g. [1, 8]. This form of user interaction represents a significant barrier to the accessibility of the Event-B toolset.

Previously, the application of *design patterns* [9] has been suggested for Event-B [3]. The ability to reuse design patterns, and their associated proofs, has obvious benefits over the conventional notion of design patterns. Our work with failure-analysis is more closely aligned with *anti-patterns* [4], i.e. common patterns of bad design, coupled with solutions. Bad design, however, does not necessarily mean incorrect design – i.e. the POs may still be provable. For this reason the failure-driven nature of the critics presented here differ from the conventional notion of an anti-pattern. Another related area is ontology repair plans [7] – proof failure is explored using proof planning techniques to repair ontologies – i.e. proof planning is used to describe evolving models.

In terms of future work, our short-term aim is to prototype and further develop the proposal presented here – in particular, investigate modelling heuristics in addition to variable priorities. This will involve developing a plug-in for the Rodin toolset. The core of this plug-in will most likely be implemented in Prolog or Ocaml, both of which are well suited to the development of automated reasoning techniques. In terms of proof, we will initially rely on the existing Rodin theorem provers. However, in the longer term we aim to incorporate a proof planner. This will enable us to also develop critics that analyse successful proofs and provide guidance along the lines of conventional anti-patterns.

5 Conclusion

We have motivated the need to abstract away from the complexities of proof obligations and proof-failure analysis. To address this need, we have proposed a technique of proof management that attempts to turn proof-failure analysis into modelling guidance via *reasoned modelling critics.* In doing so we aim to increase the productivity of designers as well as the accessibility of formal modelling.

Our technique builds upon proof planning and, in particular, proof critics. The proposal differs from previous work in that it provides a uniform framework in which proof-failure heuristics can be combined with modelling heuristics. It is this combined approach that will enable us to abstract away from the complexity of proof obligations. While our initial critics have focused on variable priorities, our framework will be extensible, allowing designers to record meta-data relating to other kinds of modelling decisions. A prototype in the form of a Rodin plug-in is currently under development, this will enable us to empirically test and further develop our technique.

Acknowledgements This research is supported by EPSRC grants EP/F037058 and EP/E005713. Butler's involvement is part of the EU research project ICT 214158 DEPLOY (`www.deploy-project.eu`). Our thanks go to Alan Bundy, who motivated the use of variable priorities. Thanks also go to Jean-Raymond Abrial, Cliff Jones, Ewen Maclean and Maria Teresa Llano Rodriguez for their feedback and encouragement with this work. Finally we thank the three anonymous ABZ 2010 reviewers for their constructive feedback.

References

1. J-R. Abrial. *Modelling in Event-B: System and Software Engineering.* Cambridge University Press, 2009. To be published.
2. J-R. Abrial, M. Butler, S. Hallerstede, and L. Voisin. An Open Extensible Tool Environment for Event-B. In *ICFEM'06*, volume 4260 of *LNCS*, pages 588–605. Springer, 2006.
3. J-R. Abrial and T.S. Hoang. Using Design Patterns in Formal Methods: An Event-B Approach. In *ICTAC'08 1-3, 2008. Proceedings*, volume 5160 of *LNCS*, pages 1–2. Springer, 2008.
4. W. Brown, R. Malveau, H.W.S. McCormick, and T. Mowbray. *AntiPatterns: Refactoring Software, Architectures, and Projects in Crisis.* John Wiley & Sons, 1998.
5. A. Bundy, D. Basin, D. Hutter, and A. Ireland. *Rippling: Meta-level Guidance for Mathematical Reasoning.* Cambridge University Press, 2005.
6. A. Bundy. The Use of Explicit Plans to Guide Inductive Proofs. In R. Lusk and R. Overbeek, editors, *CADE'09*, pages 111–120. Springer-Verlag, 1988.
7. A. Bundy and M. Chan. Towards ontology evolution in physics. In W. Hodges, editor, *Proceedings of Wollic 2008*, LNCS. Springer-Verlag, July 2008.
8. M. Butler and D. Yadav. An Incremental Development of the Mondex System in Event-B. *Formal Aspects of Computing*, 20(1):61–77, 2008.
9. E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison-Wesley, 1995.
10. A. Ireland. The Use of Planning Critics in Mechanizing Inductive Proofs. In A. Voronkov, editor, *LPAR'92*, number 624 in LNCS, pages 178–189. Springer-Verlag, 1992.
11. A. Ireland and J. Stark. Proof Planning for Strategy Development. *Annals of Mathematics and Artificial Intelligence*, 29(1-4):65–97, 2001.
12. R. Monroy, A. Bundy, and A. Ireland. Proof Plans for the Correction of False Conjectures. In F. Pfenning, editor, *LPAR'94*, number 822 in LNAI, pages 54–68. Springer-Verlag, 1994.