Supporting Proof in a Reactive Development Environment [*]

Farhad Mehta
ETH Zurich
Claussiusstr. 49, 8092 Zurich, Switzerland
`fmehta@inf.ethz.ch`

1 Introduction

One of the major pains of software engineering is the inevitability of change. Change can occur as the result of a change in requirements, the need for new functionality, the discovery of bugs, etc. But these are not the only sources of change. Smaller incremental changes are unavoidable when building large, real world systems. The construction of such systems is an incremental activity. One typically starts small, and incrementally adds or removes features until the desired functionality is obtained. Modern integrated development environments for programming such as the Eclipse IDE for Java support incremental construction by properly managing change and giving the engineer quick feedback on the consequences of the last modification he just made.

One of the major criticisms of using formal methods is the lack of adequate tool support, especially for proofs. The RODIN platform addresses this by providing such a reactive development environment for the *correct construction* of complex systems using refinement and proof. It is based on the Event-B (Abrial and Hallerstede, 2006; Abrial, 1996) formal method. The development consists of *modelling* the desired system and *proving* proof obligations arising from it. The proving process can be seen as an act of *debugging* the formal model since a failed proof attempt provides important insights into how a model should be modified. This analogue between programing and debugging, and modeling and proving is illustrated in figure 1.

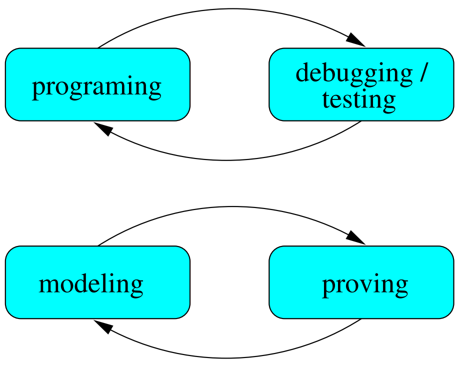

Figure 1: Software Development vs Model Development

To get the maximum benefit from this modeling-proving relationship in practice, the tools present in the RODIN platform are run in a reactive manner. By this we mean that when a model is modified, it is automatically :

- statically checked for syntactic and type errors,
- its proof obligations are generated, and
- the status of its proofs are updated, possibly by calling automated provers, or reusing old proof attempts.

A detailed description of this approach, and the benefits it provides for formal system development can be found in Abrial et al. (2006). Supporting such a reactive development environment opens up new challenges for the tools involved, particularly the prover.

[*]This research was carried out as part of the EU research project IST 511599 RODIN (Rigorous Open Development Environment for Complex Systems) http://rodin.cs.ncl.ac.uk.

2 Challenges

In a reactive development environment, a modeling change is immediately reflected in a change in proof obligations. The proof obligations are therefore in a constant state of change. The main challenge for a proof tool in such an environment is to work *incrementally* to make the best use of previous proof attempts in the midst of changing proof obligations.

In such a setting, it is not only important to reuse past proof attempts to come up with new ones, but to represent them in a form that is *guaranteed* resilient to simple changes in proof obligations, such as addition or removal of unneeded hypotheses.

3 Contributions

We contribute a solution to represent proof attempts such that:

- They are guaranteed to be recoverable for the most common changes that a proof obligation goes through when using a reactive development environment.
- The check to see if a proof attempt is resilient to change is efficient.
- For irrecoverable changes, proof attempts can be reused incrementally to construct new proofs attempts.

These results have been used to implement the proof infrastructure of the Eclipse based RODIN platform for Event-B. This has resulted in a marked improvement in the usability of this tool over the previous generation of tools for the B method. The results are independent of Event-B and the logic it uses and can be reused in many settings where computer aided proof is done in a reactive development environment.

References

Jean-Raymond Abrial. *The B-Book: Assigning programs to meanings*. Cambridge, 1996. ISBN 0-521-49619-5.

Jean-Raymond Abrial and Stefan Hallerstede. Refinement, decomposition and instantiation of discrete models. *Fundamentae Informatica*, 2006. To appear.

Jean-Raymond Abrial, Michael Butler, Stefan Hallerstede, and Laurent Voisin. An open extensible tool environment for Event-B. In Z. Liu and J. He, editors, *ICFEM 2006*, volume 4260, pages 588–605. Springer, 2006.

Eclipse. Eclipse - an open development platform, official website. URL http://www.eclipse.org. http://www.eclipse.org.

RODIN. Rigorous Open Development Environment for Complex Systems (RODIN) official website. URL http://rodin.cs.ncl.ac.uk/index. http://rodin.cs.ncl.ac.uk/index.