A Rodin plug-in for constructing reusable schematic lemmas

Alexei Iliasov, Paulius Stankaitis,
David Adjepon-Yamoah, Alexander Romanovsky

Newcastle University, UK

Abstract. In the paper we present an approach and tool for making proofs more generic and thus less fragile and more reusable. The crux of the technique is offering an engineer an opportunity to complete a proof by positing and proving a generic lemma that may be reused in the same or even another project.

1 Introduction

There was a concerted effort, funded by a succession of EU research projects, to make Event-B [1] and its toolkit, Rodin Platform [2], appealing and competitive in an industrial setting. One of the lessons of this mainly positive exercise is the general aversion of industrial users to interactive proof. It is possible, in principle, to learn, through experience and determination, the ways of underlying verification tools and master refinement and decomposition to minimise proof effort. The methodological implications are far more serious: building a good model is necessarily a trial and error process; one often has to start from a scratch or do considerable refactoring to produce an adequate model. This, obviously, necessitates redoing proofs and makes time spent proving dead-end efforts seem pointlessly wasted. Hence, proof-shy engineers too often do not make a good use of formal specification stage as they tend to hold on to the very first, often incoherent design. We want to change the way proofs are done, at least in an industrial setting. In place of an interactive proof - something that is inherently a one-off effort in Event-B - we want to invite modellers to write and prove a non-model specific condition called a schematic lemma that would, once added to hypothesis set, discharge an open proof obligation. Such a lemma may not refer to any model variables or types and is, in essence, a property supporting the definition of the Event-B mathematical language. If such a lemma cannot be found or seems to be to difficult to prove, the model must be changed. Since every modelling project is likely to have a fairly distinctive usage of data types and mathematical constructs, we might also expect a distinctive set of traits in supporting lemmas. We hypothesise that such distinctness is pronounced and dictated by the modellers experience and background as well as the model subject domain. We have also observed that the style of informal requirements - structured text, hierarchical diagram, structural diagram - has an impact on a modelling style. A schematic lemma is a tangible and persistent outcome of any

modelling effort, even an abortive one. Being generic, a schematic lemma is likely to be useful in a next iteration and, as we might hope, there is a point when all relevant lemmas are collected and modelling, in a given domain and for a given engineer, is nearly completely free of interactive proofs.

2 Generic lemmas plug-in

We have built a prototype implementation of the schematic lemma mechanism as a plug-in to the Rodin Platform. It integrates into the prover perspective and offers an alternative way to conduct an interactive proof either at a root node level or indeed for any open sub-branch of a proof obligation. At the moment, the notation employed is the native notation of Why3 but the first release will support entering schematic lemma in the Event-B mathematical notation.

The plug-in automatically constructs the first attempt at a schematic lemma through a simple syntactic transformation of a context proof obligation. All the identifiers occurring in either hypotheses or goal of the proof obligation are mapped into schematic lemma identifiers and then this mapping is used to translate hypotheses and the goal.

From this starting point it is up to the modeller to construct a sensible lemma by changing identifier, hypotheses and goal definitions. A prepared lemma is committed where the Why3 plug-in is used to prove that the lemma holds,and also that adding to the proof obligation in context discharges the proof obligation. If either fails, a user gets an indication of what has happened and it is not until both generic and concrete proofs are carried out that the schematic lemma may be used in the local library and assigned a binding level (machine, project or global). In the case of a success, the current open goal is closed.

To aid in the construction of a schematic lemma, the plug-in provides some simple productivity mechanisms. A hypotheses can be deselected without removing it to check whether both the lemma goal and the context proof obligation are still provable. An identifier may also be deselected and this automatically deselects all the hypotheses mentioning the identifier.

In this work we have tried to weave the process of constructing generalised proofs into the very process of model construction and address two long standing challenges of model-based design: turning proofs into tangible artefacts that can survive deep model refactoring, and making interactive proof on organic part of model construction rather than an unfortunate side activity.

References

1. J.-R. Abrial. *Modelling in Event-B.* Cambridge University Press, 2010.
2. The RODIN platform. Online at http://rodin-b-sharp.sourceforge.net/.