Meta-Predicates for Rodin

Sebastian Krings

Institut für Informatik, Universität Düsseldorf
Universitätsstr. 1, D-40225 Düsseldorf
krings@cs.uni-duesseldorf.de

1 Introduction and Motivation

Event-B [1] provides a concise mathematical language for specifying invariants and guards. While both Event-B and Rodin [2] matured, certain patterns for specifying properties like deadlock freedom emerged and are in use. These patterns are often realized by copy and paste of guards into other guards or invariants, leading to duplicated code and incomprehensible specifications. Furthermore, it may lead to errors if predicates are not kept in sync. We observed errors like these during the case studies of ABZ 2014 [3] and started developing an extension to Event-B that allows accessing guards by corresponding event names. Since, it has been implemented as a plugin for the Rodin platform.

2 Meta-Predicates for Event-B

We added the following meta-predicates based on the syntax accepted by the LTL model checker of ProB [5,4]. To avoid loops, generated predicates are not copied again. With $guards(e)$ containing the guards of e that have been given directly by the user, the added predicates are

- *deadlock* enforces that all given elements are disabled, i.e., $deadlock(Events) = \bigwedge_{e \in Events} \bigvee_{g \in guards(e)} \neg g$.
- *enabled* enforces that all given events are enabled, i.e., $enabled(Events) = \bigwedge_{e \in Events} \bigwedge_{g \in guards(e)} g$.
- *controller* enforces that exactly one event is enabled, i.e., $controller(Events) = \bigvee_{e \in Events}(enabled(\{e\}) \wedge deadlock(Events \setminus \{e\}))$.
- *deterministic* enforces that the order of execution is deterministic, i.e., $deterministic(Events) = controller(Events) \vee deadlock(Events)$.

Embedding the predicates into Event-B involves more work than macro replacement as the following example will show: Let's take events `evt1`, featuring parameter `x1`, and `evt2` with parameter `x2`. Obviously, we can not just copy guards of `evt1` to `evt2`, as referencing `x1` inside `evt2` will result in an undeclared variable. For free variables the plugin thus has to decide between

- Adding a parameter to the surrounding event,
- Introducing them using existential quantification, or

– Skipping, removing or renaming them as far as possible.

We did not want to change existing Event-B code in order to keep semantics as simple as possible. Only new invariants and guards should be added. Thus, we decided to introduce free variables by existential quantification. A preference allows to decide if all variables are quantified or only those not yet defined.

3 Integration into Rodin

Our extended language has been integrated into Rodin as a plugin. Sources are available from `https://github.com/wysiib/RodinMetaPredicatesPlugin`, `https://www3.hhu.de/stups/rodin/meta_predicates/nightly/` contains an update site.

The plugin works as an additional static checker. It runs after the machine itself has been checked, relying only on information Rodin has not discarded.

4 Comparison to Related Plugins

In [6], the authors present `DFT-generator`, a tool for the automatic generation of proof obligations for deadlock freedom. Using `DFT-generator` is comparable to adding $deadlock(\{\ldots\})$ to the invariants. However, rather than extending the predicate syntax, information is stored externally. Our approach is more flexible, e.g., one can use predicates like $state = ERROR \Rightarrow deadlock(\{\ldots\})$.

5 Discussion and Conclusion

Our plugin adds a lightweight language extension to Event-B. Together with tool support, common errors leading to erroneous specifications can be avoided. Furthermore, expressiveness is increased as the intention of a predicate becomes more obvious. In future, we would like to implement a direct export to ProB's LTL model checker to be able to use different model checking algorithms.

References

1. J.-R. Abrial. *Modeling in Event-B: System and Software Engineering.* Cambridge University Press, 2010.
2. J.-R. Abrial, M. Butler, S. Hallerstede, T. S. Hoang, F. Mehta, and L. Voisin. Rodin: An open toolset for modelling and reasoning in Event-B. *Int. J. Softw. Tools Technol. Transf.*, 12(6):447–466, 2010.
3. F. Boniol, V. Wiels, Y. Ameur, and K. Schewe, editors. *ABZ 2014: The Landing Gear Case Study*, CCIS 433. Springer, 2014.
4. M. Leuschel and M. Butler. ProB: A model checker for B. In *Proceedings FME'03*, LNCS 2805, pages 855–874. Springer, 2003.
5. M. Leuschel and M. Butler. ProB: an automated analysis toolset for the B method. *Int. J. Softw. Tools Technol. Transf.*, 10(2):185–203, 2008.
6. F. Yang and J.-P. Jacquot. An Event-B plug-in for creating deadlock-freeness theorems. In *Proceedings SBMF'11*. Brazilian Computer Society, 2011.