A Framework for Developing Distributed Protocols with Event-B/Rodin

Paulius Stankaitis[1], Alexei Iliasov[1], Tsutomu Kobayashi[2], Alexander Romanovsky[1], Fuyuki Ishikawa[2]

[1] Newcastle University, Newcastle upon Tyne, United Kingdom
{p.stankaitis|alexei.iliasov|alexander.romanovsky}@ncl.ac.uk

[2] National Institute of Informatics, Tokyo, Japan
{t-kobayashi|f-ishikawa}@nii.ac.jp

1 Introduction

Formal notations such as Event-B [1] are well suited for development and verification of various protocols. The stepwise and proof driven development provided by such methods is attractive to developers and can notably reduce modelling effort. Yet effectively utilizing the refinement of the Event-B method can be challenging and often result in a poor readability/maintainability of the model as well as reduced verification automation. This is particularly significant for modelling complex distributed systems because of concurrency and communication. Furthermore from our experience in modelling various distributed protocols the model often needs a complete remake which makes the modelling process rather challenging and tedious.

In this short paper we present an ongoing work on developing a toolkit and technique for developing distributed protocols. Such a toolkit would take a semi-formal graphical definition of a protocol and automatically generate Event-B refinement chain. In the following section we summarise the main ongoing work directions and technical challenges.

2 Framework for Developing Distributed Protocols

First of all we propose to start developing a distributed protocol by providing its semi-formal graphical definition. A graphical model would then be automatically translated and proved in Event-B. Not only this would provide a better protocol overview for the developer but should also reduce the effort in case user needs to remake the model. At this moment we are in the process of developing the prototype of such graphical language as existing approaches were not adequate.

The literature review also showed that similar proposed approaches for modelling and reasoning about distributed protocols did not utilize the stepwise development of the Event-B method. The design of a distributed protocol in Event-B generally follows a number of principles in order to be considered an adequate rendering of a protocol[1]. This requires a faithful model of communication, a convincing argument of localisation and finally a formal decomposition of a model into independent communicating parts. In order to transform a graphical definition of a distributed protocol to the Event-B model we propose to develop translation patterns for classes of distributed protocols. Translation patterns would allow automatically generating Event-B refinement chains which could make models more readable and potentially reduce the verification effort. In addition to syntactic model translation patterns, these patterns could contain invariants for correctness verification of distributed protocols.

As another direction we would like to derive generic verification conditions for classes of distributed consensus or resource allocation protocols. To reason about additional distributed protocol properties (e.g. liveness) we also would like to take advantage of other existing Rodin Platform plug-ins such as the Flow plug-in [4]. We are still in early stages of this work and there are many important questions to address. For instance how one should specify the objective of the distributed protocol in a semi-formal graphical language. Since it is common to use Dijkstra backward elicitation style where an abstract model summarises protocol effect and following refinement steps gradually introduce preceding steps. Inferring the objective is particularly important in order to make to make an efficient use of refinement as abstraction and proof structuring technique.

References

1. J.-R. Abrial. *Modeling in Event-B: System and Software Engineering.* Cambridge University Press, New York, NY, USA, 2013.
2. D. Cansell and D. Méry. Formal and incremental construction of distributed algorithms: On the distributed reference counting algorithm. *Theor. Comput. Sci.*, 364(3):318–337, November 2006.
3. T.S. Hoang, H. Kuruma, D. Basin, and J.-R. Abrial. Developing topology discovery in event-b. In *Proceedings of the 7th International Conference on Integrated Formal Methods*, IFM '09, pages 1–19, Berlin, Heidelberg, 2009. Springer-Verlag.
4. Alexei Iliasov. Use case scenarios as verification conditions: Event-b/flow approach. In Elena A. Troubitsyna, editor, *Software Engineering for Resilient Systems*, pages 9–23, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.
5. D. Yadav and M. Butler. Methods, models and tools for fault tolerance. chapter Formal Development of a Total Order Broadcast for Distributed Transactions Using Event-B, pages 152–176. Springer-Verlag, Berlin, Heidelberg, 2009.

[1] These principles can be noticed from existing literature on modelling distributed systems in Event-B (e.g. [2] [3] [5]).