Co-simulation of Event-B and Continuous Models in Rodin

Vitaly Savicks, Michael Butler, John Colley
University of Southampton

Jens Bendisposto
Heinrich-Heine-Universität Düsseldorf

While Event-B formal method, reinforced with the rich and extensible Rodin platform and a great community around it, provides a powerful and effective framework for the formal modelling and verification of safety-critical systems, its discrete nature does not always map well to the heterogeneous complex systems with components from the different domains, including the intrinsically continuous environment. To improve the capability of Rodin and enable it to verify components from the wide range of domains along with formally specified Event-B models we are introducing an extension to Rodin, which is based on the ProB tool and the industry-designed Functional Mock-up Interface (FMI) standard, and allows to co-simulate formal models with any imported models that support FMI standard. The ongoing work consists of the design of a diagrammatic co-simulation surface and generic simulation master algorithm.

The FMI standard will allow us to integrate artefacts called Functional Mockup Units (FMU) into Event-B simulations. These FMUs can be generated by any tool that supports this standard. Many simulation tools[1] such as MATLAB/Simulink or even Microsoft Excel support the FMI standard, as it ensures flexibility and cross-platform execution.

FMUs are coordinated by an FMI master. An individual FMU is a zip archive that contains at least a shared library, which implements the Functional Mock-up Interface, and an XML document describing the communication ports of the model and the capabilities of the simulator.

[1] A list is available at `https://www.fmi-standard.org/tools`.

Our approach is to use a generic implementation that loads a given FMU and provides an API that is more idiomatic for the usage inside the Java-based Rodin application. As FMI does not impose many restrictions on the master algorithm, the master has to be rewritten from scratch for each system. Using the Groovy scripting language this can be done already in the current implementation. We want to provide the user with a more pleasant experience by supporting a generic master that can be configured from the UI and customised using the scripting language. Such a master needs to know about all the units involved in co-simulation and connections between them. In the FMI parlance this information is called component connection graph and is used to validate assembled units, derive a suitable co-simulation algorithm depending on the topology and other properties of the graph and coordinate co-simulation process.

We are planning to develop a component diagram editor and co-simulation driver as part of our Rodin tool that would enable users to import and instantiate existing FMUs and Event-B machines as components and visually construct a graph model that can be used by the master. Each component on the diagram will have a configurable number of input and output ports, compatible with FMI standard types, which can be connected to the corresponding ports of other components via connectors. The diagram will provide controls for running the master and display the simulation state with time, including exchanged signal values between ports and internal state variables of components.

The metamodel of connection graph and the editor will be based on the existing open-source frameworks provided by Eclipse and Rodin and will make use of the Event-B metamodel.

The component diagram tool will support both FMI components, bundled as `.fmu` archives, and Event-B components. An Event-B component may represent a single Event-B machine or a hierarchy of machines with a composed machine as a root element and multiple nested machines that constitute the composition. Each nested machine can be a composed machine itself. We are aiming at supporting such complex compositional structures at the diagram level, so the users will be able to import composed machines and their constituent elements and link them with other components. The interaction between the elements of composed machines may also be part of the visual notation and displayed during the simulation.