Verification of UML 2 State Machines by Automated Model Transformation to the SAL Model Checker

Áron Sisak

Nowadays, Model Driven Software Development aims at offering a framework to automate the synthesis and verification of software. This paper aims at presenting the transformation of UML 2.0 state machine diagrams to a model checking language (i.e., a mathematical model), which is suitable for formal verification. The approach allows to perform early investigation of software models defined in a very widespread modeling language with rich capabilities.

UML 2.0 State Machines provides the de facto standard for modeling reactive, state-based system behavior. It offers both a graphical representation for intuitive, visual system modeling and a semantics that aims to serve as a base for both code generation and verification and validation activities. However, besides the powerful concepts included in the State Machine language, the semantics contained by the standard contains several ambiguities as well. The Precise Statechart Semantics introduced in [1] offers remedy for these problems. The semantics is defined in a *declarative* manner, which suits the declarative model checking paradigm very well. The transformation is based upon Precise Statecharts (PSC) .

Formal verification aims to prove that a system model satisfies certain properties. Model checking is a widespread formal verification approach. It performs the exhaustive exploration of the model to verify the desired properties. The SAL model checking framework [2] is widely used to perform model checking, mostly because its capability of coping with large model state spaces. The PSC semantics is based on the Kripke Transition System formalism, which is semantically very close to the foundations of the SAL module language, which makes possible to find a well-established mapping to transform PSC constructs into the SAL language.

My prototype transformation tool is implemented on the top of the Java API provided for the Precise Statechart Semantics, which makes possible to process UML2 statechart models produced by EMF-based UML modeling tools, with access to the special constructs defined by the PSC semantics (sets and relations). The tool implements a template-based solution, using the Velocity template engine. Metamodel-level constructs are transformed directly without any knowledge about the actual model being processed; e.g., initializing a statechart or firing a transition are mapped into a generic SAL module. States, regions, triggers and other elements are mapped to enumerations based on the actual model being transformed. Guards and effects are implemented as parameterized generic SAL code. Arbitrary SAL code can be used in guard and effect implementations. The actual run-time data consist of a set of boolean arrays representing the active states in the statechart structure, and a set of SAL variables used in the guards and effects. One of the most difficult aspect of implementing the transformation engine is to decide on proper SAL constructs that both capture the semantics properly and avoid state space explosion as much as possible, e.g., using simple boolean arrays instead of records can reduce the state space with multiple orders of magnitude.

Besides support for basic state machine constructions discussed above, verification of interacting state machines is also possible, as event queue models are supported. Basic support to map standard UML variable types to SAL types is also present in the current implementation. After implementing the full semantics, various abstraction possibilities are to be investigated in order to cope with even bigger state spaces.

References

[1] G. Pintér. Model Based Program Synthesis and Runtime Error Detection for Dependable Embedded Systems, *PhD Thesis*, BME MIT, 2007

[2] L. de Moura et al. SAL 2, in *R. Alur and D. Peled, editors, CAV 2004*, LNCS 3114, pp. 496–500