New Model Checking Techniques for Software Systems Modeled with Graphs and Graph Transformations

Sebastian Gabmeyer
Reviewer: Prof. Dr. Gerti Kappel, Prof. Dr. Martin Gogolla

June 15, 2015

Abstract. In today’s software, no matter how security and safety critical it may be, defects and failures are common. With the rising complexity of software and our growing dependency on its correct functioning as it permeates our every day life the software development process requires new approaches to integrate formal verification techniques. This thesis presents approaches on efficiently verifying software systems described by model-driven software development artifacts. These artifacts comprise the implementation of the system and consist of both structural and behavioral models. We present two model checking approaches, MocOCL and Gryphon, to verify the temporal specification of a system against its model-based implementation. Central to our approach is the twofold use of graphs to describe the system under verification. First, we describe the admissible static structure of an instance of the system by means of attributed type graphs with inheritance and containment relations, often referred to as metamodel. Second, we represent a state of the system as an object graph that enumerates a system’s active objects, the references among them, and their attribute values. A change in the system, e.g., the deletion of an object or the modification of an attribute value, triggers a state change. The behavior of the system is thus described by actions that modify the state of the system. In this thesis we employ graph transformations to model such state-changing actions as they provide suitable means to formally describe modifications on graphs. The specification of the system, on the other hand, is written in our temporal extension of the Object Constraint Language (OCL) that is based on Computation Tree Logic (CTL). A specification written in our CTL extension for OCL, called cOCL, can be verified against a model-based implementation of the system with our explicit-state model checker MocOCL. Gryphon aims to increase the efficiency and scalability of the verification process and implements a symbolic model checking approach, that focuses the verification on safety specifications.

The work presented in this thesis also encompasses a survey and a feature-based classification of verification approaches that can be used to verify artifacts
of model-driven software development. Methodologically, it provides the motivation for our work on MOCOCL and GYPHON. Both our approaches are novel in their own respect; MOCOCL for its capability to verify CTL-extended OCL expressions and GYPHON for its use of relational logic to build a symbolic representation of the system that can be verified with any model checker participating in the Hardware Model Checking Competition. Finally, MOCOCL and GYPHON are evaluated performance-wise on a set of three representative benchmarks that demonstrate the model checkers’ preferred fields of application and its limitations.