The handle http://hdl.handle.net/1887/20325 holds various files of this Leiden University dissertation.

Author: Grabe, Immo
Title: Static analysis of unbounded structures in object-oriented programs
Issue Date: 2012-12-19
Abstract

Formal methods provide the foundation to reason about systems and their characteristics, e.g. safety or security properties. To be able to reason about systems and to provide valid statements about the system the formal methods must provide means to address key features of the language, e.g. concurrency or object creation. These features introduce complexity to systems and the formal methods to reason about these systems.

Both the development of hardware, e.g. multicore processors or the increase of available memory, and the development of software, e.g. networked and integrated systems and the introduction of high-level programming languages like Java and C♯, lead to an increased usage of the aforementioned features. New concepts to address problems like concurrency and delegation, e.g. futures and promises, are developed and promoted. Formal methods need to include these features to be capable of reasoning about state of the art software systems.

In this thesis, we show how to address the complexity introduced by modern software systems, e.g. structural complexity introduced by object creation and behavioural complexity introduced by multi-threading. We present formalisms to check interesting properties, e.g. deadlock freedom or termination, of such systems. Furthermore we show how to extend a calculus to reason about systems based on active objects with futures and promises.

In the first part of the thesis we address complexity introduced by object creation. Object creation is an abstraction from the underlying representation of objects used for example in object-oriented programming languages like Java or C♯. For practical purposes it is important to be able to specify and verify properties of objects at the abstraction level of the programming language. We give a representation of a weakest precondition calculus for abstract object creation in dynamic logic which allows to both specify and verify properties of objects at the abstraction level of the programming language. We generalize this approach to allow for symbolic execution to integrate our approach to the setting of the KeY theorem prover.
In the second part of this thesis we address complexity introduced by multi-threading. Multi-threaded programs show complex behaviour due to the interleaving of activities of the individual processes and the sharing of state among these processes. One example of such complex behaviour is the so-called deadlock. Deadlock describes a situation in which concurrent processes share resources. Though shared among the processes a single access to a resource is exclusive. Depending on the order the processes are interleaved the deadlock may or may not arise. The number of interleavings of the processes is in general not bound which makes analysis hard. We present a formalism to reason about deadlock in multi-threaded systems. The formalism focuses on the control flow of such systems and abstracts from data.

In the third part of this thesis we extend a calculus to reason about active objects with futures and promises. We present an open semantics for the core of the Creol language including first-class futures and promises. A future acts as a proxy for, or reference to, the delayed result of a computation. As the consumer of the result can proceed its own execution until it actually needs the result, futures provide a natural, lightweight, and transparent mechanism to introduce parallelism into a language. A promise is a generalization of a future as it allows for delegation with respect to which process performs the computation. The formalization is given as a typed, imperative object calculus to facilitate the comparison with the multi-threaded concurrency model of object-oriented languages, e.g. Java.

We close the third part of this thesis by presenting a technique to detect deadlocks in concurrent systems of active objects. Our technique is based on a translation of the system to analyse into a P/T net and the application of a technique to detect termination in such P/T nets. We illustrate our technique by application to an Actor-like subset of the Creol language featuring asynchronous calls using futures as means of communication. The so-called discipline of cooperative multi-tasking within an object as found in Creol can lead to deadlock. Our technique can be applied to detect such deadlocks.