Testing object Interactions

Grüner, A.

Part II

Testing Multi-threaded Components

In the previous part of this thesis we presented a formal framework for testing object-oriented components in a *sequential* setting. That is, the language allowed for a single-threaded flow of control, only. In the following part, we suggest an extension of the framework regarding multi-threaded components. In particular, we will extend the underlying programming language with the notion of *threads*. In languages like *Java* and C♯ objects are passive entities residing in the heap of the program – instantiated from classes that serve as "generators of state"; the active part of the program is represented by threads. Indeed, in a multi-threaded setting, there is also a mechanism for *"generating new activity"*, i.e., for creating new threads. Thus, we extend our previous work by thread instantiation from *thread classes*, meaning that new activities can be dynamically spawned from "templates".

Correspondingly, we have to adapt the test specification language. The underlying idea is that we cope with multi-threading by providing a specification statement for each thread. Hence, only the order of interactions which belong to the same thread is specified.

Finally, we sketch how the code generation algorithm of the single-threaded setting can be modified in order to generate test programs also for multi-threaded components.