Decomposition for Compositional Verification

Abstract

Within the domain of safety-critical systems, software engineering becomes a major challenge, as failures of a system may have life-threatening ramifications. In order to ensure the reliability of software, its correctness is essential. For the correctness proof of a model, integrated formalisms with an underlying formal semantics can be used.

Several obstacles complicate a successful application of model checking software models. The main challenge is to cope with the *state explosion problem*, that is, the exponential growth of the system's state space in the size of the model. Several approaches deal with this well-known problem. One of them is *compositional verification*. The basic idea of compositional verification is that the check of correctness of a complex system can be divided into smaller verification tasks. The technique avoids to build up the entire state space of the model, as it solely needs to deal with the individual state spaces of the single components of a system.

In order to facilitate an application of this technique, two problems need to be addressed: the model itself must be *assembled from several components* which is, in general, not the case. Furthermore, an application of compositional reasoning must provide an *efficiency advantage* over monolithic model checking.

Within this thesis, we develop a technique on how to *decompose* software models specified in the integrated formalism *CSP-OZ*. Such a decomposition results in two components suitable for the application of compositional reasoning. A first challenge is posed by a *proof of correctness*, showing the equivalence of the original specification and a decomposition in our semantic domain. In order to achieve this, we carry out a *dependence analysis* by means of a specification's dependence graph. The analysis leads to a set of *correctness criteria*, based on which the graph is fragmented into two parts. The fragmentation then results in the decomposition of the specification. In addition, we introduce several techniques and algorithms to restore the specification's original control flow and its data flow. As a second challenge, we address the *practicability* of compositional reasoning: we identify *heuristics* for measuring the quality of a valid decomposition. Here, we *neglect* inefficient decompositions. This allows us to consider only those, which most likely result in an *effective* compositional verification.

Overall, our approach facilitates a general application of compositional reasoning, as it does not rely on systems composed of several components. Moreover, valid decompositions, which are assessed as good by our heuristics, are beneficial for a compositional verification. The whole approach is tool-supported due to an integration into a graphical modelling environment, allowing for the *modelling*, *analysis*, *decomposition* and *(compositional) verification* of integrated specifications. Model checking itself is performed within an *assume-guarantee-based* verification framework. Here, we use two proof rules, which are shown to be valid in our semantic domain. Along with this, we provide several case studies and experimental results.