Abstract

Safety-critical electronic systems are becoming increasingly complex and simultaneously ubiquitous. As in other engineering disciplines, computer science needs to offer viable approaches to the correct design of such systems. Especially important within this design process is the phase of initial *specification*, since its results have impact on all subsequent stages of development. The most extensive and reliable analysis of system specifications is offered by using *formal methods* that allow us to obtain mathematical proofs of system correctness by applying automatic verification techniques.

In the area of formal system specification there is, however, not one single general-purpose notation, that would be equally well suited for all system aspects. Instead, *integrated formal methods* are investigated, which combine different specification languages, exploiting their individual strengths, while still maintaining a common semantic foundation for subsequent verification. One such notation is the high-level specification language *CSP-OZ-DC*, combining the process algebra *Communicating Sequential Processes* (CSP) for expressing behavioural aspects, the state-based notation *Object-Z* (OZ) for expressing data aspects, and the real-time logic *Duration Calculus* (DC) for expressing real-time aspects of systems.

The main obstacle for successful application of automatic verification, however, is the problem of *state space explosion*, i.e., the exponential blow-up in the number of system states to be analysed. Many techniques have been proposed to tackle this problem, one of them being the method of *slicing* that has its origins in the area of program analysis where it is used to compute those parts of a program that are relevant with respect to a specific analysis task.

Within this thesis, we develop a slicing approach for integrated formal specifications that is custom-tailored to the rich syntactical structure of CSP-OZ-DC specifications and that is applicable in the context of their verification with respect to real-time requirements. The slicing approach essentially consists of three steps: First, the specification is analysed with respect to dependences between its syntactical elements with several new types of dependence being defined such as synchronisation and timing dependence. Second, these dependences as a whole are used to identify those specification parts that are relevant for the given verification property. Third, the specification slice is computed, i.e., a reduced version of the full specification that does not contain any elements without influence on the verification property.

A *correctness proof* shows that verification can be carried out on the slice instead of the full original specification without changing the verification result. The proof is based on a notion of *projection* between a specification and its slice. The existence of such a projection relation is shown to be guaranteed by the slicing approach. The particular logic used to express verification properties is then shown to be stuttering-invariant, i.e., provided that the projection relation exists, it cannot distinguish between the slice and the original specification such that the verification result will in both cases always be the same.

Furthermore, we present tool support that has been implemented for developing, slicing, and verifying CSP-OZ-DC specifications along with several case studies and experimental results for evaluating the effectiveness of the slicing approach.