Expressing Realtime Properties in VSE

Andreas Nonnengart Georg Rock Werner Stephan
Deutsches Forschungszentrum für
Künstliche Intelligenz (DFKI GmbH)
Email: stephan,rock,nonnenga@dfki.de

February 26, 2001

It is well known that in many cases severe errors occur in the early stages of the software development process. Following this insight, considerable emphasis was put on requirements engineering. Formal methods provide means for the abstract yet precise description of required system properties. However, specialized description techniques are needed to treat certain aspects adequately. Moreover, complex systems often make it necessary to cover several of these views.

Among the most critical and most difficult (to describe) aspects of systems behaviors are realtime properties. A major problem in this area lies in the complexity of timing constraints and the variety of situations that can appear in such systems. People tend to draw diagrams that describe particular behaviors and they try to find critical situations in this way. It is clear that these diagrams might help to understand the system, but they certainly are not suitable to express an entire temporal behavior in a comprehensive way. A more adequate way is to exploit timed automata or (linear) hybrid automata in this context.

The actual design starts with an abstract description of the technical measures that are claimed to satisfy the formal requirements resulting from the analysis phase. With respect to realtime requirements, one might take a global view on the entire scenario, while the specification that forms the starting point of the design (and implementation) process has to be structured into several components. First of all the system to be developed has to be separated from its environment. In most cases the system will consist of several components that are able to communicate with each other. For this kind of specification different techniques have to be used as they are offered from a system like VSE. Verification Support Environment (VSE) supports abstract system specifications, the formulation and proof of safety/security properties, and correct refinements. The formal development method of VSE is based on Abstract Data Types and Temporal Logic. The aim of the work presented here is to integrate hybrid automata as a means for comprehensively describing realtime properties into the VSE method where an abstract system specification serves as the starting point of a formal refinement process. Hybrid automata (translated into VSE) provide a realtime view on these systems. Additional requirements specifications representing different views of the behavior can be "linked" to the same specification.

The integration of hybrid automata into the overall VSE development method proceeds in two steps: First hybrid automata are translated into VSE's temporal logic. Then system specifications in VSE are considered as (potential) models of these descriptions thereby providing a "satisfies-relation" between system specifications and realtime requirements.

Translating hybrid automata into VSE is a challenging task. Conceptually, hybrid automata perform continuous (time) steps as well as discrete steps. The time steps are executed while control resides within a so-called location whereas with discrete steps the location is to be changed (a transition might lead back to the source location, though). The integration of such automata into VSE requires several new concepts to be introduced thoroughly. Since we use natural numbers in our model to represent time, we have to discretize the continuous (time) steps of hybrid automata when transferring them into VSE. Discretization is a technique that is quite common in the relevant literature. However, it often results in a mere approximation of the original behavior. The technique we use differs from these standard techniques as it performs an exact discretization which means that we do not lose any (safety) properties of the original hybrid automaton. In addition, we introduce a method to arbitrarily change the base unit granularity of a hybrid automaton. Both techniques are realized by our translation function which not only translates the hybrid automaton but also the properties the automaton is supposed to guarantee.

The starting point of our methodology is the specification of the behavior of a realtime system using a hybrid automaton Hybrid-AutSpec. Properties that should be satisfied by this automaton are specified in Hybrid-PropSpec which is described by a formula in PSL (which is a special property specification language). The underlying idea is to translate the hybrid automaton specification into a VSE-SL specification. From this translated specification, we want to prove that the properties described in VSE-PropSpec hold. Some of these are created by translating the properties in Hybrid-PropSpec into properties formulated in VSE. The proof of these properties can be performed within the VSE tool. The soundness and completeness of the above transformation methods guarantee that we do not lose any properties compared to tools supporting hybrid automata.

So far we have translated hybrid automata into a VSE requirements specification. The starting point of the formal development is a VSE specification (VSE-SystemSpec) which describes the technical solutions in an abstract way. To be able to interpret this specification as a (potential) model of the hybrid automaton (that underlies the requirements specification) it has to stick to a certain format (system components, environment, clock). This style of specification (of reactive systems) was successfully used for the emergency closing system of a storm surge barrier in the Netherlands.

To formalize the intended interpretation within VSE the user has to relate states of the hybrid automaton to situations of the specified scenario. After that a "satisfies-link" between VSE-SystemSpec and VSE-AutSpec gives rise to proof obligations (generated by the system). If we are able to prove these formulae, we have established a mapping under which the VSE system specification satisfies the realtime behavior given by the hybrid automaton.

The approach we are proposing offers several advantages. The VSE reqirements specifications (resulting from the translation) can be modelchecked whenever possible using tools like HyTech. If this turns out to be not possible (for instance because of non-termination of fixpoint computations) proofs can be carried out deductively in VSE. According to our main result both ways will be equivalent (in case the model checker terminates at all). Moreover, having a translation of hybrid automata into VSE at hand we are no longer restricted to the narrow bounds of (efficiently) decidable theories. Without changing the basic framework the description technique can be generalized. For example this allows us to extend hybrid automata with abstract data types, theory of natural numbers, etc. Finally, realtime modelling can be integrated into the general software development process as one of possibly several views.