Verification of Communication Protocols using SDL

Calin Jebelean

Department of Computer Science "Politehnica" University of Timişoara
Institute e-Austria Timişoara
calin@cs.utt.ro

Abstract

Software systems will inevitably grow in size and functionality, as recent evolution shows. While there is a certain investment in new, state of the art technologies, one should not forget the other side of the coin, which involves validating the existing technologies. People should be able to build reliable software systems, no matter how complex they are. One way of achieving this goal is by making use of formal methods.

In this report, we approach the field of verifying communication protocols, using SDL as the description language and model checking as the driving force to validate or invalidate them. We also argument the utility of having an equivalent intermediate representation for SDL, which could largely improve the process of validating certain properties of the initial SDL model.

SDL (Specification and Description Language) is an object-oriented, formal language defined by the International Telecommunications Union - Telecommunications Standardization Sector (ITU-T) as recommendation Z.100. The language is intended for the specification of complex, event-driven, real-time systems involving concurrent activities that communicate using signals. This specific feature makes SDL extremely suitable for describing communication protocols.

One of the greatest challenges for SDL is that of addressing specification and validation of systems of increasing size. Tool support is required for that purpose and, by now, there are commercial tools and toolsets (Telelogic's *Object*GEODE being the most important among them) dedicated to analysis, design, simulation, code generation and testing of real-time and distributed applications.

However, the inherent complexity of most protocol specifications led to the idea that verification and validation should involve a combination of techniques such as static analysis, abstraction and model-checking, which have already proven their power. Since these are almost impossible to integrate within commercial tools by third parties, an intermediate representation for SDL called IF has been developed at Verimag [4].

IF (short for *Intermediate Format*) is based on a simple and semantically sound model for distributed timed systems, namely asynchronously communicating timed automata (automata with clocks). The most important advantage of having an intermediate format for SDL has already been presented. Analysis algorithms can be run directly on the IF translation of the SDL specification, instead of waiting for their integration within some commercial tool. Another advantage would be the possibility to interconnect SDL with existing verification tools such as SMV, Spin or CADP, if a proper translator from IF to the input language of these tools is made available. And yet, another advantage lies in the fact that an IF model is generally smaller than the corresponding model generated by a commercial tool such as *Object*GEODE, for example, even though the two models are equivalent. The difference is due to several optimizations implemented by the translator from SDL to IF, such as eliminating certain transient states. These optimizations are presented in [4].

Results of applying different analysis techniques on IF representations have already been reported in [3] and [4]. *"Live variables" analysis* is a technique which reduces the state space of the model by considering only the so-called *"live variables"*. A variable is *live* in a control state if there is a path from this state along which its value can be used before it is redefined. *"Irrelevant variables"*

abstraction is another technique which promises a reduction of the state space by eliminating or simplifying expressions that contain such irrelevant variables. A variable is irrelevant with respect to a property of interest if it has no effect on the respective property. Obviously, they enlarge the state space of the model for no reason.

The use of techniques such as those already presented has been reported to produce models up to 100 times smaller than the initial IF model, both in the number of states and in the number of transitions [4]. Once the reduced model is available, a property of interest could be expressed in a temporal logic, such as CTL or LTL and traditional model checking could easily be applied to validate the property or produce a counter-example, if the property is found not to hold. The process is described in more detail in [2].

Some of the properties one should keep an eye on when dealing with communication protocols are listed below:

- make sure the initiator of the communication will either get connected or will get an error response within finite time
- if, in some state, proper protocol ongoing requires the arrival of some signal, make sure the model provides at least one scenario where that signal will arrive
- if, in some state, the process waits for a signal, make sure it has previously set a timer to prevent blockage
- if, in some state, an "abandon" signal is received, make sure the process deallocates all resources it is supposed to deallocate

One signigicant challenge we face is scale. IF has been applied to large examples so far, but all involved significant manual intervention to make the system amenable to verification. Since systems involve hundreds of thousands of lines of code, with dozens of interacting modules, we will attempt to extract interfaces abstracting the behavior of modules in order to use them in the compositional verification of other modules for which they constitute the environment. Another issue is translating the specifications from natural language into a formalism such as temporal logic. We propose to identify specification patterns that can easily be reused and instantiated by designers with appropriate training. We also expect to expand the capabilities of IF for dealing with external code written in other languages such as C.

As a conclusion, we believe the IF formalism has a lot of advantages over other approaches and is very well suited for verification of communication protocols. We hope this report will help in the process of choosing a solution to such a modern problem.

References

[1] E. Clarke, J. Wing: "Formal Methods: State of the Art and Future Directions" *ACM Computing Surveys*, vol. 28, no. 4, pp. 626-643, 1996.

[2] E. Clarke, O. Grumberg, D. Long: "Verification Tools for Finite-State Concurrent Systems", *A Decade of Concurrency - Reflections and Perspectives, REX School/Symposium*, Noordwijkerhout, Netherlands, Lecture Notes in Computer Science, vol. 803, pp. 124-175, Springer Verlag, 1993.

[3] M. Bozga, J. Fernandez, L. Ghirvu, S. Graf, J. Krimm, L. Mounier: "IF: An Intermediate Representation and Validation Environment for Timed Asynchronous Systems", *Proceedings of FM'99*, Toulouse, France, 1999.

[4] M. Bozga, J. Fernandez, L. Ghirvu, S. Graf, J. Krimm, L. Mounier, J. Sifakis: "IF: An Intermediate Representation for SDL and its Applications", *Proceedings of SDL-Forum'99*, Montreal, Canada, 1999.