IF: A Validation Environment for Timed Asynchronous Systems

(category B: tool presentation)

Marius Bozga[1], Jean-Claude Fernandez[2], Lucian Ghirvu[1*], Susanne Graf[1], Jean-Pierre Krimm[1], and Laurent Mounier[1]

[1] VERIMAG[***] Centre Equation, 2 avenue de Vignate, F-38610 Gières Marius.Bozga@imag.fr
[2] LSR/IMAG, BP 82, F-38402 Saint Martin d'Hères Cedex Jean-Claude.Fernandez@imag.fr

1 Introduction

Formal validation of distributed systems relies on several specification formalisms (such as the international standards LOTOS [16] or SDL [17]), and it requires different kinds of tools to cover the whole development process. Presently, a wide range of tools are available, either commercial or academic ones, but none of them fulfills all the practical needs.

Commercial tools (like *Object*GEODE [21], SDT [1], STATEMATE [15], ...) provide several development facilities, like editing, code generation and testing. However, they are usually restricted to basic verification techniques (exhaustive simulation, deadlock detection, ...) and are "closed" in the sense that there are only limited possibilities to interface them with others. Academic tools (like SMV [20], HYTECH [13], KRONOS [23], UPPAL [19], SPIN [14], INVEST [2], ...) are in general based on ad hoc input formalisms and the gap between them and standard languages may restrict their use at an industrial scale. Even if these tools are in general less closed than commercial ones, they not always provide well-defined interfaces. Nevertheless, they offer a broad spectrum of quite efficient verification facilities (symbolic verification, on-the-fly verification, abstraction techniques, ...).

This situation motivated the development of IF, an intermediate representation for timed asynchronous systems together with an open validation environment. This environment fulfills several requirements. First of all, it is able to support *different validation techniques*, from interactive simulation to automatic property checking, together with test case and executable code generation. Indeed, all these functionalities cannot be embodied in a single tool and only tool integration facilities can provide all of them. For a sake of efficiency, this environment supports *several levels of program representations*. For instance it is well-known that model-checking verification of real life case studies usually needs to combine different optimization techniques to overcome the state explosion problem. In particular, some of these techniques rely on a syntactic level representation (like static analysis and computations of abstractions) whereas others techniques operate on the underlying semantic level. Another important feature is to keep this environment *open* and *evolutive*. Therefore, tool connections are performed by sharing either input/output formats, or libraries of components. For this purpose several well-defined application programming interfaces (APIs) are provided.

[*] Work partially supported by Région Rhône-Alpes, France

[***] Verimag is Research Laboratory of CNRS, UJF and INPG

2 Architecture

The IF validation environment relies on three levels of program representation: the *specification level*, the IF *intermediate level*, and the LTS *semantic model level*. Figure 1 describes the overall architecture and the connections between the toolbox components.

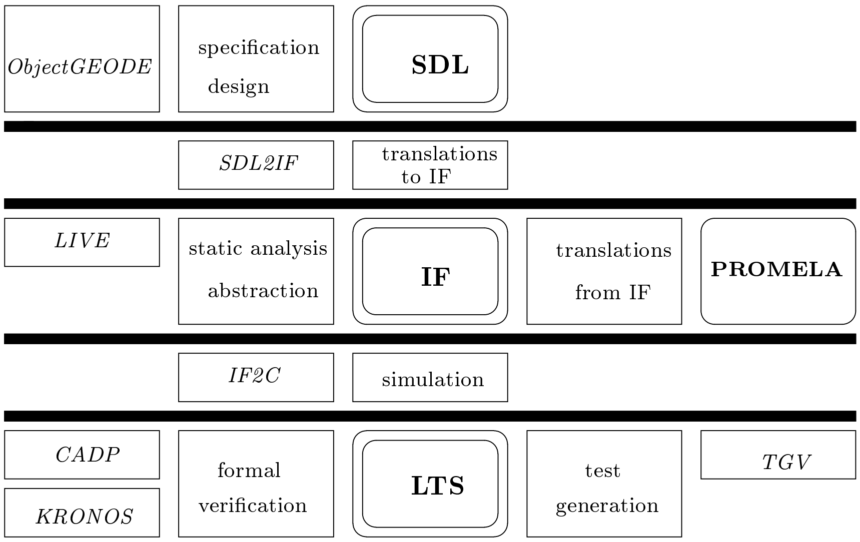

Fig. 1. An open validation environment for IF

The *specification level* is the initial program description, expressed for instance using an existing language. To be processed, this description is (automatically) translated into its IF representation. Currently the main input specification formalism we consider is SDL, but connections with other languages such as LOTOS or PROMELA could be also possible.

The *intermediate level* corresponds to the IF representation [7]. In IF, a system is expressed by a set of parallel processes communicating either asynchronously through a set of buffers, or synchronously through a set of gates. Processes are based on timed automata with deadlines [3], extended with discrete variables. Process transitions are guarded commands consisting of synchronous/asynchronous inputs and outputs, variable assignments, and clock settings. Buffers have various queuing policies (fifo, stack, bag, ...), can be bounded or unbounded, and reliable or lossy.

A well-defined API allows to consult and modify the abstract tree of the IF representation. Since all the variables, clocks, buffers and the communication structure are still explicit, high-level transformations based on static analysis (such as *live variables* computation) or program abstraction can be applied. Moreover, this API is also well suited to implement translators from IF to other specification formalisms.

The *semantic model level* gives access to the LTS representing the behaviour of the IF program. Depending on the application considered, three kinds of API are proposed:

- The **implicit enumerative representation** is based on the OPEN-CAESAR [12] philosophy. It consists in a set of C functions and data structures allowing to compute on demand the successors of a given state. This piece of C code is generated by the IF2C compiler, and it can be linked with a "generic" exploration program performing *on-the-fly* analysis.
- In the **symbolic representation** sets of states and transitions of the LTS are expressed by their characteristic predicates over a set of finite variables. These predicates are implemented using decision diagrams (BDDs). Existing applications based on this API are symbolic model-checking and minimal model generation.
- Finally, the **explicit enumerative representation** simply consists in an LTS file with an associated access library. Although such an explicit representation is not suitable for handling large systems globally, it is still useful in practice to minimize some of its abstractions with respect to bisimulation based relations.

3 Components description

We briefly present here the main components of the environment, together with some external tools for which a strong connection exists.

3.1 The specification level components

*Object*GEODE [21] is a commercial toolset developed by VERILOG supporting SDL, MSC and OMT. It includes graphical editors and compilers for each of these formalisms. It also provides a simulator to interactively debug an SDL specification and a test case generator.

*Object*GEODE provides an API to access the abstract tree generated from an SDL specification. We have used this API to implement the SDL2IF translator, which generates operationally equivalent IF specifications from SDL ones.

3.2 The intermediate level components

Two kinds of tools are offered at this level: static analysis and translations from IF to other formalisms.

LIVE [5] implements several algorithms based on static analysis to transform an IF specification. A first transformation concerns *dead variable resetting* (a variable is dead at some control point if its value is not used before being redefined). This optimisation can be also applied to buffer contents (a message parameter is dead if its value is not used when the message is consumed). Although very simple, such optimisation is particularly efficient for state space generation (reductions up to a factor 100 were frequently observed), while preserving the exact behaviour of the original specification. A second transformation is based on the *slicing* technique [22]. It allows to automatically abstract a given specification by eliminating some irrelevant parts w.r.t. a given property or test purpose [6].

IF2PROMELA [4] is a tool developed at Eindhoven TU to translate IF specifications into PROMELA.

3.3 The semantic model level components

CADP [10, 9] is a toolset for the verification of LOTOS specifications. It is developed by VERIMAG and the VASY team of INRIA Rhône-Alpes. Two of its model-checkers are connected to the IF environment: ALDEBARAN compares and minimizes finite LTSs with respect to various *simulation* or *bisimulation* relations. EVALUATOR is an "on-the-fly" model-checker for formulas of the alternating-free μ-calculus. For both tools, diagnostic sequences are computed on the LTS level and they can be translated back into MSC to be observed at the specification level.

KRONOS [23] is a model-checker for symbolic verification of TCTL formulae on communicating timed automata. The current connection with the IF environment is as follows: control states and discrete variables are expressed using the implicit enumerative representation, whereas clocks are expressed using a symbolic representation (particular polyhedra).

TGV [11] is a test sequence generator built upon CADP jointly by VERIMAG and the PAMPA project of IRISA. TGV aims to automatically generate test cases for conformance testing of distributed systems. Test cases are computed during the exploration of the model and they are selected by means of *test purposes*.

4 Results and perspectives

The IF environment has already been used to analyze some representative SDL specifications, like SSCOP, an ATM signalisation layer protocol [8], and MASCARA, an ATM wireless transport protocol. It is currently used in several on going industrial case-studies, including the real-time multicast protocol PGM, and the control part of the ARIANE 5 launcher flight sequencer.

Several directions can be investigated to improve this environment.

First of all, other formalisms than SDL could be connected to IF. In particular, the translation from a subset of UML is envisaged. However, even if the IF primitives provide the main features to model asynchronous timed systems, some new one will be added to handle *dynamic process creation* and *parametrized network specifications*.

From the verification point of view, the results obtained using the currently implemented static analysis techniques on large size examples are very encouraging. We now plan to connect the INVEST tool [2] which provides more sophisticated analysis, such as *structural invariant generation* and general *abstraction computation techniques* for infinite space systems.

Another promising approach to verify large systems consists in generating their underlying model in a *compositional* way: each sub-system is generated in isolation, and the resulting LTSs are minimized before being composed each other. This technique have been already successfully applied to synchronous communicating systems, and the IF environment offers all the required components to experiment it in an asynchronous framework [18].

Finally, the IF package can be freely download at `http://www-verimag.imag.fr/DIST_SYS/IF`.

References

1. Telelogic AB. *SDT Reference Manual*. `http://www.telelogic.se`.

2. S. Bensalem, Y. Lakhnech, and S. Owre. Computing Abstractions of Infinite State Systems Compositionally and Automatically. In *Proceedings of CAV'98, Vancouver, Canada*, volume 1427 of *LNCS*, June 1998.
3. S. Bornot, J. Sifakis, and S. Tripakis. Modeling Urgency in Timed Systems. In *International Symposium: Compositionality - The Significant Difference, Holstein, Germany*, volume 1536 of *LNCS*, 1998.
4. D. Bošnački, D. Dams, L. Holenderski, and N. Sidorova. Model Checking SDL with Spin. In *Proceedings of TACAS'2000, Berlin, Germany*, LNCS, 2000. to appear.
5. M. Bozga, J.-C. Fernandez, and L. Ghirvu. State Space Reduction based on Live Variables Analysis. In *Proceedings of SAS'99, Venezia, Italy*, volume 1694 of *LNCS*, September 1999.
6. M. Bozga, J-C. Fernandez, and L. Ghirvu. Using Static Analysis to Improve Automatic Test Generation. In *Proceedings of TACAS'2000, Berlin, Germany*, LNCS, 2000. to appear.
7. M. Bozga, J.-C. Fernandez, L. Ghirvu, S. Graf, J.P. Krimm, and L. Mounier. IF: An Intermediate Representation and Validation Environment for Timed Asynchronous Systems. In *Proceedings of FM'99, Toulouse, France*, volume 1708 of *LNCS*, 1999.
8. M. Bozga, J.-C. Fernandez, L. Ghirvu, C. Jard, T. Jéron, A. Kerbrat, P. Morel, and L. Mounier. Verification and Test Generation for the SSCOP Protocol. *SCP*, 1998. to appear.
9. M. Bozga, J.-C. Fernandez, A. Kerbrat, and L. Mounier. Protocol Verification with the Aldebaran Toolset. *STTT*, 1(1+2):166–183, December 1997.
10. J.-C. Fernandez, H. Garavel, A. Kerbrat, R. Mateescu, L. Mounier, and M. Sighireanu. CADP: A Protocol Validation and Verification Toolbox. In *Proceedings of CAV'96, New Brunswick, USA*, volume 1102 of *LNCS*, July 1996.
11. J.-C. Fernandez, C. Jard, T. Jéron, and C. Viho. An Experiment in Automatic Generation of Test Suites for Protocols with Verification Technology. *SCP*, 29, 1997.
12. H. Garavel. OPEN/CÆSAR: An Open Software Architecture for Verification, Simulation, and Testing. In *Proceedings of TACAS'98, Lisbon, Portugal*, volume 1384 of *LNCS*, 1998.
13. T.H. Henzinger, P.-H. Ho, and H. Wong-Toi. HyTech : A Model Checker for Hybrid Systems. In *Proceedings of CAV'97, Haifa, Israel*, volume 1254 of *LNCS*, June 1997.
14. Gerard J. Holzmann. *Design and Validation of Computer Protocols*. Prentice Hall Software Series, 1991.
15. I-Logix. *StateMate*. `http://www.ilogix.com/`.
16. ISO/IEC. LOTOS — A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour. Technical Report 8807, International Organization for Standardization — Information Processing Systems — Open Systems Interconnection, 1988.
17. ITU-T. *Recommendation Z.100. Specification and Description Language (SDL)*. 1994.
18. J.P. Krimm and L. Mounier. Compositional State Space Generation with Partial Order Reductions for Asynchronous Communicating Systems. In *Proceedings of TACAS'2000, Berlin, Germany*, LNCS, 2000. to appear.
19. K.G. Larsen, P. Petterson, and W. Yi. UPPAAL: Status & Developments. In *Proceedings of CAV'97, Haifa, Israel*, volume 1254 of *LNCS*, 1997.
20. K.L. McMillan. *Symbolic Model Checking: an Approach to the State Explosion Problem*. Kluwer Academic Publisher, 1993.
21. Verilog. Object*GEODE SDL Reference Manual*. `http://www.verilogusa.com/`.
22. M. Weiser. Program Slicing. *IEEE Transactions on Software Engineering*, SE-10(4), July 1984.
23. S. Yovine. KRONOS: A Verification Tool for Real-Time Systems. *STTT*, 1(1-2), Dec 1997.