Kronos: a model-checking tool for real-time systems

Marius Bozga, Conrado Daws, Oded Maler, Alfredo Olivero, Stavros Tripakis, Sergio Yovine

Kronos: a model-checking tool for real-time systems *
(Tool-presentation submission for CAV'98)

Marius Bozga[1], Conrado Daws[1], Oded Maler[1], Alfredo Olivero[2], Stavros Tripakis[1] and Sergio Yovine[3]

[1] VERIMAG, Centre Équation, 2 avenue de Vignate, 38610 Gières, France.
e-mail: {bozga, daws, maler, tripakis}@imag.fr
[2] Instituto de Computación, Universidad de la República, Montevideo, Uruguay.
e-mail: yovine@imag.fr, sergio@path.berkeley.edu
[3] VERIMAG; currently visiting California PATH, UC Berkeley.
e-mail: alfredo@ungs.edu.ar

General presentation

KRONOS [8, 10, 7, 11, 20, 16, 4, 3, 9] is a software tool aiming at assisting designers of real-time systems to develop projects meeting the specified requirements.

One major objective of KRONOS is to provide a verification engine to be integrated into design environments for real-time systems in a wide range of application areas. Real-time communication protocols [8, 10], timed asynchronous circuits [16, 4], and hybrid systems [18, 10] are some examples of application domains where KRONOS has already been used.

KRONOS has been also used in analyzing real-time systems modeled in several other process description formalisms, such as ATP [17], AORTA [5], ET-LOTOS [8], and T-ARGOS [15]. On the other direction, the tool itself provides an interface to untimed formalisms such as *labeled-transition systems* (LTS) which has been used to exploit untimed verification techniques [20].

Theoretical background

The system-description language of KRONOS is the model of *timed automata* [2], which are communicating finite-state machines extended with continuous real-valued variables (*clocks*) used to measure time delays. Usually a system is modeled as a *network* of automata. Communication is achieved by label synchronization à la CCS or CSP (binary or n-ary *rendez-vous*), or shared variables (of bounded integer or enumeration type).

System requirements can be specified in KRONOS using a variety of formalisms, such as the real-time logic TCTL [1, 14], timed Büchi automata, or

* KRONOS is developed at VERIMAG, a joint laboratory of UJF, Ensimag and CNRS. http://www.imag.fr/VERIMAG/PEOPLE/Sergio.Yovine/kronos/kronos.html.
C. Daws, A. Olivero and S. Yovine partially supported by European Contract KIT 139 HYBSYS.

untimed LTS. These formalisms are useful for expressing most interesting classes of (timed or untimed) properties about systems, namely, *safety* properties (for example, absence of deadlock, invariant, bounded-delay response, etc), as well as *liveness* properties (for example, time progress, regular occurrence of certain events, etc) [1].

The main verification engine of the tool is based on the *model-checking* approach which comprises both *analysis*: (a) checking whether requirements are satisfied, (b) providing *diagnostic trails* (i.e., execution sequences) demonstrating why a property holds or does not hold; and *synthesis*: adjusting the system (for instance, by computing a restricted sub-system) so that it meets its requirements. Model-checking is done using two methods: (a) the *fixpoint* method, which, given a timed automaton and a TCTL formula, performs a nested fixpoint computation starting from an initial set of states and iterating a *precondition operator* until stabilization (the operator depends on the type of the formula); (b) the *explorative* method, which, given a network of timed automata and a specification (in terms of a TCTL formula or a timed Büchi automaton), generates the reachability graph of the system while checking at the same time whether the property holds. In the case of safety properties a simple (depth-first or breadth-first) search of the reachability graph suffices. In the case of general properties, specified as timed Büchi automata, a double search is performed, refining parts of the graph whenever necessary. Both methods are interesting: the main advantage of the fixpoint method is that it can be implemented in a purely *symbolic* manner, using structures like BDD for efficiency (see below); on the other hand, the explorative method is more suitable for *on-the-fly* verification (see below) and can also provide diagnostic trails.

Apart from model-checking, KRONOS offers the possibility to (a) generate the system's reachable state space (to check, for instance, whether an error state can be reached), and (b) compute the coarsest partition of the state space with respect to the *time-abstracting bisimulation*, an equivalence relating states which lead to the same untimed behavior regardless the exact time delays. This method provides an interface to LTS and verification by bisimulation or simulation equivalences [20] using the ALDEBARAN tool suite [13].

Supported verification techniques

The main obstacle in the applicability of model-checking is the so-called *state-explosion problem* reflecting the fact that the size of the system's state space is often huge. In order to tackle this, KRONOS offers a number of efficient verification techniques, each of which is best suited for different applications.

- *Symbolic* representation of states means dealing with *predicates* representing sets of states rather than individual states. This results into a much more compact representation and storage. In the current KRONOS implementation,

[1] To our knowledge, KRONOS is the only real-time verification tool which can handle liveness properties.

sets of clock values are represented using the *difference bounds matrix* (DBM) structure introduced in [12], whereas discrete variables are encoded as *binary decision diagrams* (BDD) [6].

- *On-the-fly* model-checking means dynamically building the state space during the model-checking process, as directed by the model-checking goal (for instance, the property to be verified); this results in saving up space and time, as well as in giving diagnostics as soon as possible.
- *Abstractions* are used for the exploration of a coarser state space than the "real" (*concrete*) one; they result into space and time savings, at the cost of loosing information, so that sometimes definite conclusions cannot be made.
- *Syntactic optimizations* are used to reduce the number of clocks in the model to only the strict necessary; they allow for space and time savings at almost no cost since they are inexpensive to compute.
- *Forward* or *backward* techniques: in the former (typically used in the explorative method) the exploration starts from initial states and tries to reach some target, while in the latter (typically used in the fixpoint method) it is the inverse that happens. Combined with various *search algorithms* (such as depth-first or breadth-first) implemented in the model-checking engine of the tool, these alternative techniques result in a large flexibility with respect to the different application needs.
- *Minimization*: it is used to generate the time-abstracting *minimal model* of the system, which can then be visualized as an untimed graph, compared or further reduced with respect to untimed equivalences, or checked using untimed temporal logics.

Case studies

Kronos has been used to verify various industrial communication protocols, such as an audio-transmission protocol by Philips [10] (where errors have been found to the previously hand-made proofs) or an ATM protocol by CNET [19] (where a bug was also found relative to the consistency of the network components). Other communication protocols modeled and verified by Kronos include the carrier-sense multiple-access with collision detection (CSMA-CD) protocol [8] and the fiber-optic data-interface (FDDI) protocol [9]. Well-known benchmark case studies verified by Kronos include Fischer's real-time mutual-exclusion protocol [9] and a production-plant case study [10]. Finally, the tool has been also applied to the verification of the Stari chip [4] and to the synthesis of real-time schedulers [2].

The most recent enhancements of Kronos include the implementation of different abstraction mechanisms [9], the implementation of a symbolic on-the-fly algorithm for checking timed Büchi automata emptiness [3] and a BDD-based implementation oriented towards the timing analysis of circuits [4]. Table 1 presents some typical experimental results extracted from the cited papers. The measurements were taken on a Sparc Ultra-1 with 128 Mbytes of main memory. Time is

[2] Unpublished work.

given in seconds. The size of the state space (when available) is given in symbolic states (i.e., control location plus DBM), BDD nodes, or states and transitions. "OTF" stands for "on-the-fly".

Case study	Method	Time	State space
Production plant	Fixpoint	26	not available
CNET	Forward	3	not available
Philips	Forward	2	not available
Fischer (5 processes)	Minimization	32	3000 states & trans.
Fischer (6 processes)	OTF	2783	164935 symb. states
Fischer (9 processes)	OTF & Abstractions	17098	1096194 symb. states
FDDI (7 stations)	OTF & Büchi aut.	4813	57500 symb. states
FDDI (12 stations)	Forward	1123	13000 symb. states
FDDI (50 stations)	Forward & Abstractions	3900	4000 symb. states
STARI (17 stages)	Fixpoint & BDD	100000	1000000 BDD nodes

Table 1. Some performance results.

It is worth noting that the entire machinery of KRONOS has been useful for handling the above examples. In particular, the fixpoint method has been used in earlier versions of the tool for liveness properties, as well as for synthesis (see, for instance, [10], where initial constraints have been tightened so that the system behaves correctly). Forward model-checking using timed Büchi automata has been recently used for checking liveness on the FDDI protocol for up to 7 processes, as well as to provide diagnostics in the real-time scheduling problem. Minimization has been used for visualizing the behavior of timed automata. On-the-fly techniques have been used whenever syntactic parallel composition could not be applied due to state explosion. Abstractions and clock-reduction techniques have been essential to the verification of the FDDI example for up to 50 processes, and Fischer's protocol for up to 9 processes [9].

Availability

KRONOS is freely available for universities or any other non-profit organisms. It can be obtained through the web at:
`http://www.imag.fr/VERIMAG/PEOPLE/Sergio.Yovine/kronos/kronos.html`
or by anonymous ftp at:

host: `ftp.imag.fr`, directory: `VERIMAG/KRONOS/tool/`.

The distribution package includes executables for various architectures (Sun5, Linux, Windows NT), documentation and examples.

References

1. R. Alur, C. Courcoubetis, and D.L. Dill. Model checking in dense real time. *Information and Computation*, 104(1):2–34, 1993.

2. R. Alur and D.L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.
3. A. Bouajjani, S. Tripakis, and S. Yovine. On-the-fly symbolic model checking for real-time systems. In *Proc. of the 18th IEEE Real-Time Systems Symposium*, 1997.
4. M. Bozga, O. Maler, A. Pnueli, and S. Yovine. Some progress in the symbolic verification of timed automata. In *CAV'97*, 1997.
5. S. Bradley, W. Henderson, D. Kendall, and A. Robson. Validation, verification and implementation of timed protocols using AORTA. In *Proc. 15th PSTV*, 1995.
6. R.E. Bryant. Symbolic boolean manipulation with ordered binary decision diagrams. Technical report, Carnegie Mellon University, 1992.
7. C. Daws, A. Olivero, S. Tripakis, and S. Yovine. The tool KRONOS. In *Hybrid Systems III*, 1996.
8. C. Daws, A. Olivero, and S. Yovine. Verifying ET-LOTOS programs with KRONOS. In *FORTE'94*, 1994.
9. C. Daws and S. Tripakis. Model checking of real-time reachability properties using abstractions. In *TACAS'98*, 1998. To appear.
10. C. Daws and S. Yovine. Two examples of verification of multirate timed automata with KRONOS. In *RTSS'95*, 1995.
11. C. Daws and S. Yovine. Reducing the number of clock variables of timed automata. In *RTSS'96*, 1996.
12. D. Dill. Timing assumptions and verification of finite-state concurrent systems. In *CAV'89*, 1989.
13. J.Cl. Fernandez, H. Garavel, L. Mounier, A. Rasse, C. Rodriguez, and J. Sifakis. A tool box for the verification of lotos programs. In *14th International Conference on Software Engineering*, 1992.
14. T.A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic model checking for real-time systems. *Information and Computation*, 111(2):193–244, 1994.
15. M. Jourdan, F. Maraninchi, and A. Olivero. Verifying quantitative real-time properties of synchronous programs. In *CAV'93*, 1993.
16. O. Maler and S. Yovine. Hardware timing verification using KRONOS. In *Proc. 7th Israeli Conference on Computer Systems and Software Engineering*, 1996.
17. X. Nicollin, J. Sifakis, and S. Yovine. Compiling real-time specifications into extended automata. *IEEE TSE Special Issue on Real-Time Systems*, 18(9):794–804, September 1992.
18. A. Olivero, J. Sifakis, and S. Yovine. Using abstractions for the verification of linear hybrid systems. In *CAV'94*, 1994.
19. S. Tripakis and S .Yovine. Verification of the fast-reservation protocol with delayed transmission using Kronos. Technical Report 95-23, Verimag, 1995.
20. S. Tripakis and S. Yovine. Analysis of timed systems based on time–abstracting bisimulations. In *CAV'96*, 1996.