Kronos: a model-checking tool for real-time systems
Marius Bozga, Conrado Daws, Oded Maler, Alfredo Olivero, Stavros Tripakis, Sergio Yovine

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

Marius Bozga1, Conrado Daws1, Oded Maler1, Alfredo Olivos2,
Stavros Tripakis3 and Sergio Yovine3

1 VERIMAG, Centre Équation, 2 avenue de Vignate, 38030 Gêres, 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
designed 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 Buchi automata, or

* Kronos is developed at VERIMAG, a joint laboratory of UJF, Ensimag and CNRS.
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 losing 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 StarChip [4] and to the synthesis of real-time schedulers.

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”.

<table>
<thead>
<tr>
<th>Case study</th>
<th>Method</th>
<th>Time</th>
<th>State space</th>
</tr>
</thead>
<tbody>
<tr>
<td>Production plant</td>
<td>Fixpoint</td>
<td>20</td>
<td>not available</td>
</tr>
<tr>
<td>CNET</td>
<td>Forward</td>
<td>3</td>
<td>not available</td>
</tr>
<tr>
<td>Philips</td>
<td>Forward</td>
<td>2</td>
<td>not available</td>
</tr>
</tbody>
</table>
| 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 & Buchi automata | 4813      | 57500 symb. states   |
| FDDI (12 stations)  | Forward      | 1123       | 13000 symb. states   |
| FDDI (50 stations)  | Forward & Abstractions | 3000  | 40000 symb. states   |
| START (17 stages)   | Fixpoint & BDD | 1000000  | 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 Buchi 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:
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