Electronic Notes in Theoretical Computer Science 65 No. 6 (2002)
URL: http://www.elsevier.nl/locate/entcs/volume65.html 8 pages

On Improving Backwards Verification of Timed Automata (Extended Abstract)

V. Braberman [a,1,4], C. López Pombo [a,2,5], A. Olivero [b,3,6]

[a] *Computer Science Department, FCEyN, Universidad de Buenos Aires, Buenos Aires, Argentina*

[b] *Department of Information Technology, FIyCE, Universidad Argentina de la Empresa, Buenos Aires, Argentina*

Abstract

Verification techniques for Timed Automata [2] built in tools like Kronos [7] are based on the fixpoint calculus of an appropriate operator. In this work, we present different alternatives to calculate that fixpoint, which have direct impact in the number of iterations needed to converge.

1 Introduction

Verification techniques for Timed Automata [2] built in tools like Kronos [7] are based on the fixpoint calculus of an appropriate operator. For instance, to verify logics like TCTL [1] over Timed Automata, it is usually required to obtain the set of states from which the system can evolve and reach a set of states satisfying a formula ϕ (the characteristic set of $\texttt{true}\ \exists\mathcal{U}\ \phi$). That set can be characterized as a fixpoint. It is well known that the chaotic iteration results of Cousot [6] suggest the existence of many algorithmic alternatives to do such a calculus. In this work we explore different iterative methods to solve the question "Does the initial state belong to the characteristic set of $\phi_1\ \exists\mathcal{U}_{\mathcal{I}}\ \phi_2$?", where ϕ_1 and ϕ_2 are TCTL formulas, and $\mathcal{I}$ bounds the time elapsed to reach ϕ_2. On the one hand, we present a method that tries to reduce the number of iterations needed to converge. This is achieved by making use of intermediate results obtained in the same iteration instead the intermediate

[1] Research supported by UBACyT grants X156 and TW72.
[2] Research supported by FOMEC 376 Schoolarship.
[3] Research supported by UADE grant ING6-01.
[4] Email: vbraber@dc.uba.ar
[5] Email: clpombo@dc.uba.ar
[6] Email: aolivero@uade.edu.ar

results of previous iteration like the traditional iterative algorithm in Kronos. On the other hand, we present a local convergence method that -like in [11]- calculates fixpoints over graph components thus avoiding the propagation of results till they are locally stabilized. We have implemented a prototype for both strategies based on Kronos tool and we have obtained some preliminary experimental data. Finally, we compare both alternatives and suggest combinations that could outperform previous implementations.

2 Timed Automata

Timed Automata (TA) [2] has become one of the most widely used formalism to model and analyze timed systems and it is supported by several tools (e.g., Kronos [7], Uppaal [3], HyTech [10], Red [17], Treat [12], etc.). They have been successfully applied to automatically check communication protocols, real-time systems and circuits.

TAs are finite automata where time is incorporated by means of clocks. As finite automata, a TA $G = \langle S = \{s_1, \ldots, s_n\}, X, E, I\rangle$ is composed by a finite set of nodes S (called locations in TA literature) and a set of edges E. There is no notion of final locations since executions are infinite. Edges model event occurrences while clocks declared in the set X measure time elapsed. Each edge has associated a timing condition –a guard– and a set of clocks indicating which ones are reset when the edge is traversed. A guard is a constraint on clock values of the form $x \sim n$ where $\sim \in \{\leq, <, =, >, \geq\}$ and $n \in \mathbb{N}$. An edge can be traversed whenever its associated guard is true. Time elapses at locations, and edge traversal is instantaneous and the associated clocks are reset. Also, timing conditions are associated with each location by means of I. These conditions are called invariants, and determine the valid clock values for locations. Hence, it is possible to use an invariant to express that the control can not remain in the location more than a certain amount of time (i.e. a deadline). Semantics is given by means of a labeled transition system, where a state is composed by a control location and the values of clocks (positive real numbers), for more details, see for instance [19]. There are two kinds of transitions between states: temporal (associated with the progress of time) and discrete (related with the crossing of an edge in the automata). Set of states are usually decribed by means of certain timed predicates called regions (in Kronos literature), that we generic denote as ϕ. Given a predicate φ on the states, we denote the set of states satisfying it as $[[\varphi]]$ (the characteristic set of φ). The set of states that have a temporal transition to a state in φ is denoted as $pred_t(\varphi)$, and the set of states that reach a state of φ by traversing a discrete transition associated with an edge e is denoted as $pred_e(\varphi)$. Both operators are closed over regions.

3 Classical Backwards Verification for $\exists\Diamond\phi$

We are interested in calculating TCTL formulas of the form $\phi_1 \exists\mathcal{U}_{\mathcal{I}} \phi_2$ where ϕ_1, ϕ_2 are TCTL formulas. A state belongs to $[[\phi_1 \exists\mathcal{U}_{\mathcal{I}} \phi_2]]$ whenever the automata can evolve through a path of ϕ_1-states of time-length in $\mathcal{I}$ and reach a ϕ_2-state. In what follows, for the sake of simplicity we restrict ourself to the formulas $\exists\Diamond\phi \stackrel{def}{=} \texttt{true}\exists\mathcal{U}\phi$. In [18], it is shown that the region equivalent to $\exists\Diamond\phi$ set can be calculated as the least fix point: $\mu\mathcal{X}.(\phi \vee \texttt{true} \triangleright \mathcal{X})$ where $\texttt{true} \triangleright \mathcal{X}$ is $pred_t(pred_e(\mathcal{X}) \vee \mathcal{X})$, that is the set of states that are temporal predecessors of a state in the union of $\mathcal{X}$ and its discrete predecessors (i.e. predecessors by a discrete transition). In the following proposition, we show the abstract functional description of the computation of the fixpoint as it is done in Kronos.

Proposition 3.1 (K) *Given $G = \langle S = \{s_1, \ldots, s_n\}, X, E, I\rangle$ a timed automata and $\phi = \bigvee_{i\in\{1,\ldots,n\}} @ = s_i \wedge \phi_i$ a predicate that characterizes simbolically a set of states, then $\mu\mathcal{X}.(\phi \vee \texttt{true} \triangleright \mathcal{X})$ can be computed iterativelly as follows:*

$$\begin{array}{l}\{\mathcal{X}_{ij}\}_{(1\leq i\leq n),(0\leq j)} \\ \mathcal{X}_{i0} = pred_t(\phi_i) \\ \mathcal{X}_{ij} = \mathcal{X}_{ij-1} \vee \bigvee_{e\in E_{ik}} pred_t(pred_e(\mathcal{X}_{kj-1})); \text{ for } 1 \leq j.\end{array}$$

where $\mathcal{X}_{ij}$ is the value of the component i of $\mathcal{X}$ at iteration number j, and E_{ik} is the set of edges from s_i to s_k.

□

4 A Method for Reducing the Number of Iterations $(K^\dagger)$

The first idea is based on a simple observation: to calculate $\mathcal{X}_{i,j}$ is not always necessary to make use of the region calculated for location k during the last iteration (i.e., $\mathcal{X}_{k,j-1}$). We can expect an speed up if the method resorts to the region of location k which has been calculated in the current iteration if location k has been already treated (i.e., $\mathcal{X}_{k,j}$). In [13] it is proven that this computation method is correct using Cousot's results. That is:

Proposition 4.1 $(K^\dagger)$ *Given $G = \langle S = \{s_1, \ldots, s_n\}, X, E, I\rangle$ a timed automata and $\phi = \bigvee_{i\in\{1,\ldots,n\}} @ = s_i \wedge \phi_i$ a predicate that characterizes simbolically a set of states, then $\mu\mathcal{X}.(\phi \vee \texttt{true} \triangleright \mathcal{X})$ can be computed as follows:*

$$\begin{array}{l}\{\mathcal{X}_{ij}\}_{(1\leq i\leq n),(0\leq j)} \\ \mathcal{X}_{i0} = pred_t(\phi_i) \\ \mathcal{X}_{ij} = \mathcal{X}_{ij-1} \vee \bigvee_{e\in E_{ik}} pred_t(pred_e(\mathcal{X}_{kj-\phi_\sigma(i,k)})); \text{ for } 1 \leq j.\end{array}$$

where $\phi_\sigma : \{1,\ldots,n\} \times \{1,\ldots,n\} \longrightarrow \{0,1\}$ *is defined as:*

$$\phi_\sigma(i,j) = \begin{cases} 0 \ ; \ \textit{if } \sigma(i) > \sigma(j) \\ 1 \ ; \ \textit{if } \sigma(i) \leq \sigma(j) \end{cases}$$

such that $\sigma \in S_n$ *(i.e.* σ *is a permutation that models the order in which locations are traversed. That is, the method is insensitive to the ordering), and* E_{ik} *is the set of edges from* s_i *to* s_k*.*

□

In many applications, it is not necessary to calculate the entire characteristic set of a formula like $\exists\Diamond\phi$. This is particularly true when we are only interested in knowing whether or not the system can evolve from the initial state to a ϕ-state (i.e., init $\Rightarrow$ $\texttt{true}\exists\mathcal{U}\phi$). Then, we introduce a simple mechanism that stops fixpoint calculation as soon as the initial state belongs to an intermediate set [7]. In fact we introduce that mechanism to the original Kronos implementation ($K - RI$ columns in tables) and our modified version ($K^\dagger$).

We have run in a AMD K7 1333Mhz 256Mbytes LINUX 7.2 platform several case studies: the communication protocol CSMA-CD [15], rail crossing system (RCS) [12] for 5 trains, a freshness problem [5] for a version of the Active Structural Control System [9], a bounded response property on a Mine Drainage design [4] and the FDDI protocol [16] for 9 stations. Experimental data show that in all cases the number of iterations is reduced but that reduction is not always translated into a speed up, specially when ϕ is not reachable ($K^\dagger$ columns in Table 3). Our conjecture is the following: data structures to represent symbolic states (essentially a set of difference bound matrices, DBM [8] or zones) tend to "mature" rapidly in our version and thus iterations are heavier than classical ones, compensating the reduction achieved in the number of iterations. Table 1 shows for the RCS example, the number of zones needed to represent the calculated symbolic region after each iteration of K and $K^\dagger$.

Experiments suggests that our algorithm is better suited to early detection ($K^\dagger$ columns). This may also be explained using the same conjecture about maturation of regions.

5 Local Convergence (K^{part})

Topology of graphs could provide useful information when a fixpoint is calculated over them. In [11] SCCs (strongly connected components) are calculated to perform a two level iterative fixpoint method for untimed systems. Local

[7] the operator $\triangleright$ is monotonous (see for instance [19]).

Iteration number	1	2	3	4	5	6	7	8	9
# Zones for K	165	166	2137	3946	5026	6034	6450	6514	6514
# Zones for $K^{\dagger}$	930	2099	5744	7427	7711	7711	-	-	-

Table 1
Comparison of number of zones per iteration over RCS example

convergence of SCCs are invoked from a main iteration that seeks for global stabilization. The idea behind this strategy is to avoid propagation of intermediate results till the results are locally stabilized. We also show in [13] that theoretical results of Cousot's thesis guarantee that no matter how the graph is partitioned, nesting local iterations into a global one leads to the same result.

Proposition 5.1 (K^{part}) *Given $G = \langle S = \{s_1, \ldots, s_n\}, X, E, I\rangle$ a timed automata*

- *$C = \{c_1, \ldots, c_t\}$ is a partition of the set S.*
- *$\{c^i : c^i \in C\}_{0 \leq i}$ an infinite sequence of elements of C such that $(\forall i)(1 \leq i \leq t \Rightarrow (\forall j)(j \in \mathbb{N} \Rightarrow (\exists k)(j \leq k \wedge c_i = c^k)))$*

and $\phi = \bigvee_{i \in \{1,\ldots,n\}} @ = s_i \wedge \phi_i$ a predicate that characterizes simbolically a set of states, then $\mu\mathcal{X}.(\phi \vee \texttt{true} \triangleright \mathcal{X})$ can be computed as:

$$\{\mathcal{X}_{ij}\}_{(1 \leq i \leq n),(0 \leq j)}$$
$$\mathcal{X}_{i0} = \phi_i$$
$$\mathcal{X}_{ij} = \begin{cases} \mathcal{X}_{ij-1} & ; \text{ if } s_i \notin c^j. \\ pred_t(\mathcal{X}_{ij-1}) \vee \bigvee_{e \in E_{ik}} pred_t(pred_e(\mathcal{X}_{kj-1})) & ; \text{ if } s_i \in c^j. \end{cases}$$

where E_{ik} is the set of edges from s_i to s_k.

□

It remains the problem of efficiently obtaining a reasonable partition of the set of nodes C. One alternative is to calculate SCCs as in [11]. However, the following observation that led us to a simpler and in many cases practical solution. Safety and liveness requirements are commonly modeled by means of virtual components (Observers) which are composed in parallel with the system under analysis (*SUA*). Thus, if we divide the set of nodes of the parallel composition according to the local position of the observer, we would get a reasonable partition conformed by sets of maximal SCCs. Moreover, since in many cases observers can be acyclic [4], if components are topologically ordered, then just one global iteration is really necessary. Preliminary experiments show that, in case the set characterized by ϕ is not reachable, the strategy may lead to important improvements in the verification times (K^{part}

REACHABILITY: True								
Method →	K		$K - RI$		$K^{\dagger} - RI$		K^{part}	
Example ↓	sec.	iter	sec.	iter	sec.	iter	sec.	iter
BOUNDED	195.30	25	18.20	10	17.10	3	86.60	67
RCS	$\perp$	-	0.91	10	0.21	1	$\perp$	-
CSMA/CD	215.72	85	0.17	3	0.19	1	218.20	100
FRESH	$\perp$	-	1,644	16	39.68	3	$\perp$	-
FDDI	26.69	28	27.39	28	2.61	2	171.76	70

Table 2
Comparison of verification time and number of iterations of the different methods

REACHABILITY: False								
Method →	K		$K - RI$		$K^{\dagger} - RI$		K^{part}	
Example ↓	sec.	iter	sec.	iter	sec.	iter	sec.	iter
BOUNDED	91.13	22	89.58	22	91.52	9	29.70	65
RCS	3.10	9	3.00	9	23.30	6	1.33	19
CSMA/CD	0.37	8	0.36	8	0.35	5	0.37	18
FRESH	11,880	22	11,918	22	$\perp$	-	1,490	43
FDDI	$\perp$	-	8.72	21	2.59	1	$\perp$	-

Table 3
Comparison of verification time and number of iterations of the different methods

columns). Note that the number of iterations is increased but these iterations do not involve all locations. There is ongoing work to use a tool for graph partitioning like MeTiS [14] to calculate suitable partitions.

6 Future Work

We are currently working on a promissing combination of both iterative methods to treat TCTL formulas like $\texttt{init} \Rightarrow \phi_1 \exists \mathcal{U}_{\mathcal{I}} \phi_2$ (examples of that fragment are non-zeno formulas [19]). Local convergence is applied to calculate the sets ϕ_1 and ϕ_2 using some partition of the control-graph generated by MeTiS tool. Then, the early detection algorithm explained in Sect. 4 is applied to discover as soon as possible if the initial state is in $[[\phi_1 \exists \mathcal{U}_{\mathcal{I}} \phi_2]]$. We hope to get some speed ups and gain more insight on how data structures affects the abstract improvements on the fixpoint calculation.

References

[1] Alur, R., C. Courcoubetis, and D. Dill, *Model-Checking for Real-Time Systems*, Information and Computation, vol. 104, no. 1, 2-34, 1993.

[2] Alur, R., and D. Dill, *A Theory of Timed Automata*, Theoretical Computer Science, vol. 126, 1994, 183-235.

[3] Bengtsson, J., K.G. Larsen, F. Larsson, P. Pettersson, and W. Yi, *UPPAAL- A Tool Suite for the Automatic Verification of Real-Time Systems*, Proc. Hybrid Systems III, Lecture Notes on Computer Science 1066, Springer Verlag, 1996, 232-243.

[4] Braberman, V., "Modeling and Checking Real-Time System Designs", Ph.D. Thesis, Departamento de Computación, Universidad de Buenos Aires, 2000.

[5] Braberman, V., D. Garbervetsky, and A. Olivero, *Improving the Verification of Timed Systems using Influence Information*. To appear in TACAS 2002. Also available as Technical Report TR2001-003. CS Department, FCEyN, Universidad de Buenos Aires.
URL: http://www.dc.uba.ar/people/exclusivos/vbraber/tr01-003.ps.

[6] Cousot, P., "Methodes Iteratives de Construction et D'Aproximation de Points Fixes D'Operateurs Monotones sur un Treillis, Analyse Semantique des Programmes". Ph.D. Thesis Université Scientifique es Médicale de Grenoble, Institut National Polytechnique de Grenoble, 1978.

[7] Daws, C., A. Olivero, S. Tripakis and S. Yovine, *The Tool KRONOS,* In Proc. of Hybrid Systems III, LNCS 1066, Springer Verlag, 208-219, 1996.

[8] Dill, D., *Timing Assumptions and Verification of Finite-State Concurrent Systems*, In Proceedings of the Workshop on Automatic Verification Methods for Finite-State Systems, Lecture Notes in Computer Science 407, Springer Verlag, 197-212, 1989.

[9] Elseaidy, W., R. Cleaveland, and J. Baugh Jr., *Modeling and Verifying Active Structural Control Systems*, Science of Computer Programming, 29(1-2):99-122, July 1997.

[10] Henzinger, T. A., P-H. Ho, and H. Wong-Toi. *HyTech: a model checker for hybrid systems*, Software Tools for Technology Transfer 1:110-122, 1997.

[11] Kerbrat, A., *Reachable State Space Analysis of Lotos Programs*, In 7th international Conference on Formal Description Techniques for Distributed Systems and Communication Protocols, Bern, Switzerland, October 1994.

[12] Kang, I., I. Lee, and Y.S. Kim, *An Efficient Space Generation for the Analysis of Real-Time Systems*, In Proceedings of the International Symposium on Software Testing and Analysis, 1996. Also in Trans. on Software Engineering, 1999.

[13] López Pombo, C. G., "Mejoras a un algoritmo de model checking simbólico, basadas en la topología de los sistemas de tiempo real". Bachelor in Computer Science Thesis. Departamento de Computación, Facultad de ciencias exactas y naturales, Universidad de Buenos Aires, 2001.

[14] Karypis, G., and V. Kumar *Multilevel Algorithms for Multi-Constraint Graph Partitioning* Proceedings of 10th Supercomputing Conference, 1998.

[15] Nicollin, X., J. Sifakis, and S. Yovine, *Compiling Real-Time Specification into Extended Automata*, IEEE Trans. on Soft. Eng., Special Issue on Real-Time Systems, Vol. 18, 9, pp. 794-804, September 1992.

[16] S. Tripakis "L'Analyse Formelle des Systemès Temporisés en Practique", Phd. Thesis, Univesité Joseph Fourier, December 1998.

[17] Wang, F., *RED: Model-Checker for Timed Automata with Clock-Restriction Diagram*, 2nd Workshop on Models for Timed Critical Systems 2001. To be published in Electronic Notes in Theoretical Computer Science.

[18] Yovine, S., "Méthodes et Outils pour la Vérification Symbolique de Systemès Temporisés," Ph.D Thesis, Institut National Polytechnique de Grenoble, 1993.

[19] Yovine, S., *Model-Checking Timed Automata,* Embedded Systems, G. Rozemberg and F. Vaandrager eds., Lecture Notes in Computer Science, Springer Verlag, Vol. 1494, October 1998.