Pierre-Alain Reynier

Diagonal Constraints handled efficiently in UPPAAL

Research Report LSV-07-02

January 2007

Diagonal constraints handled efficiently in UPPAAL

Pierre-Alain Reynier

LSV, CNRS & ENS Cachan, France
reynier@lsv.ens-cachan.fr

Abstract. Timed automata (TA) are widely used to model real-time systems, and UPPAAL is one of the most popular model-checker for this framework which has been successfully applied over numerous industrial case studies. Diagonal constraints are a natural extension of TA, that does not increase expressive power, but gives conciseness. Unfortunately the classical forward algorithm for reachability analysis cannot be used to deal directly with diagonal constraints. Thus the current method implemented consists in removing them on-the-fly, which implies a complexity blow-up. In [8], a counter-example guided refinement algorithm has been proposed. In this paper, we present its implementation, and give some benchmarks on a variant of Fischer's protocol.

Keywords: Timed automata, reachability analysis, diagonal constraints.

1 Introduction

Timed Automata. The development of reactive, critical or embedded systems requires the use of formal verification methods, such as model checking. It is often necessary to consider quantitative informations on time elapsing in the model description and timed automata (TA) have been proposed by Alur and Dill [1] to model such real-time systems. Many tools have been developed to analyze these systems, among them UPPAAL [10] has become one of the most popular, because of its efficiency, and its numerous extensions which allow an easier building of the model.

Reachability analysis. Reachability properties are the basis of most verification techniques. For checking those properties, a symbolic representation of configurations (called *zones*) is used to perform a forward on-the-fly analysis. To ensure termination, an abstraction operator is used, which leads to the computation of an over-approximation of the set of reachable states. This approximation, correct for diagonal-free TA, is however not enough precise in presence of diagonal constraints: locations of TA with diagonal constraints may be found reachable by the algorithm while they are not!

Diagonal constraints. Classical timed automata [1] consider simple constraints $x \sim c$ and *diagonal constraints* $x - y \sim c$. From [1, 4] we know that diagonal constraints can be removed from TA, with an exponential cost. It has been shown in [7] that this over-cost is unavoidable, since TA with diagonal constraints can be exponentially more concise than TA without. Moreover, scheduling problems are modeled using diagonal constraints (see [9]). Thus, the design of efficient algorithms in presence of diagonal constraints is relevant.

2 Theoretical framework

As explained before, standard forward algorithm uses an abstraction operator for enforcing termination, it thus computes an over-approximation of the set of reachable states. This approximation is sound if there is no diagonal constraints [6]. Unfortunately, it is no more the case when allowing diagonal constraints. Refer to [5, 2] for more details on this topic.

A first solution. Since we know how to build from a TA with diagonal constraints an equivalent TA without diagonal constrains, a first solution consists in removing, in a first preprocessing step, every diagonal constraint appearing in the TA, and then apply the classical algorithm. In order to do it more efficiently, [3] proposed a way to simulate the resulting TA on-the-fly, *i.e.* during the reachability analysis of the automaton. This method is the algorithm currently implemented in UPPAAL. The drawback is again that the cost of the new reachability analysis may be exponentially higher.

A "CEGAR" solution. In order to avoid this over-cost, we proposed in [8] a new method based on the paradigm of counter-example guided abstraction refinement (CEGAR). Essentially, we want to remove as few diagonal constraints as possible. Indeed, since false positives are very seldom, our algorithm will in most cases (when diagonal constraints do not generate false positives) make no more work than the classical algorithm. In the worst case, it will behave roughly as the algorithm proposed in [3], as the experiments will show. The global behavior is depicted on Figure 1.

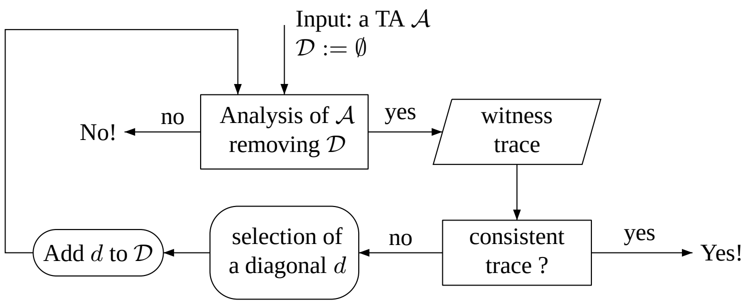
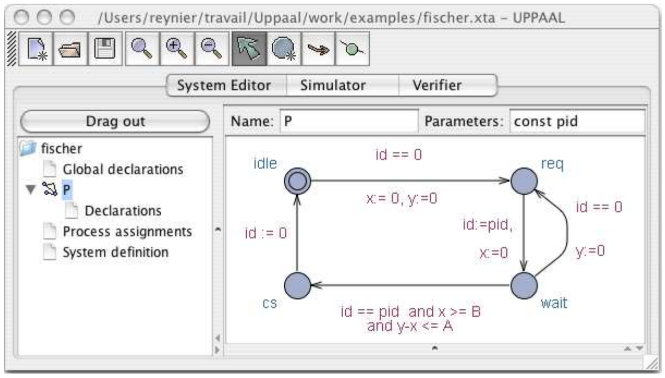

Input: a TA $\mathcal{A}$
$\mathcal{D} := \emptyset$
Analysis of $\mathcal{A}$ removing $\mathcal{D}$
no
No!
yes
witness trace
consistent trace ?
yes
Yes!
no
selection of a diagonal d
Add d to $\mathcal{D}$

Fig. 1. Refinement-based method

3 UPPAAL

UPPAAL is a tool box for validation and verification of real-time systems. It consists of two main parts: a graphical user interface and a model-checker engine. Since its first release in 1995, the tool has acquired a strong maturity and has been applied in many case studies. A complete presentation of the tool can be found on its webpage (http://www.uppaal.com). The work presented in this paper is only related to the model-checker engine. Thus, we present very briefly the GUI here (see Figure 2). It consists of three parts: first, the system editor allows to design easily any complex system. Second, the user can simulate any finite trace with the simulator, such as counter-examples produced by the verification engine. Last, the verifier is devoted to the call to verification functions. The user defines there the queries and launches the verification engine.



Name: P
Parameters: const pid

Fig. 2. Overview of UPPAAL: Fischer's protocol.

4 Experiments

We provide here different benchmarks to establish the efficiency of our new algorithm. There are two kinds of tests, depending of whether our algorithm needs to refine the model or not. Time is measured in seconds, and space in Mbytes.

Fischer's protocol. We first present a modified version of Fischer's protocol, which uses diagonal constraints, depicted on Figure 2. A similar model has been proposed in [11]. It is easy to verify that the protocol ensures mutual exclusion, *i.e.* concurrent access to the critical section (location `cs`) if and only if we have $A < B$. We thus want to verify that, for values $A = 1$ and $B = 2$, there is at most one process in critical section. The results are summarized in first half of Table 1. This model produces no false positives, thus our algorithm avoids the combinatorial explosion due to removing diagonal constraints (it indeed removes no diagonal!).

Fischer	UPPAAL with refinement			Existing UPPAAL		
Nb of processes	$\#\mathcal{D}$	Time	Space	$\#\mathcal{D}$	Time	Space
2	0	0.01	1.4	4	0.01	1.4
3	0	0.02	1.4	6	0.42	38.3
4	0	11.6	40.4	8	560.8	50.2
$\mathcal{A}_1 \vert\vert \mathcal{A}_2 \vert\vert \mathcal{A}_3$	UPPAAL with refinement			Existing UPPAAL		
Query	$\#\mathcal{D}$	Time	Space	$\#\mathcal{D}$	Time	Space
$\varphi_{\{1\}}$	1	67.8	45.4	3	165.3	48.2
$\varphi_{\{1,2\}}$	2	115.5	46.6	3	164.8	47.8
$\varphi_{\{1,2,3\}}$	3	176.5	49.3	3	165.6	48.3

Table 1. Results for the two models.

An example with false positives. The only known example which produces false positives, and thus requires to "remove" some diagonal constraints is the counter-example found in [5]. In order to obtain larger models, we add a parameter to this automaton, and synchronize the resulting models. Such a model is depicted on Figure 3. For these

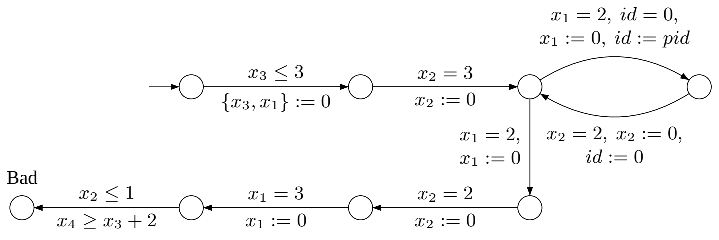


Fig. 3. Automaton $\mathcal{A}_{pid}$ of [5] with an additional parameter pid.

experiments, we consider the synchronization $\mathcal{A}_1||\mathcal{A}_2||\mathcal{A}_3$ of three automata. Depending on the query, we have to remove some diagonal constraints (see column $\#\mathcal{D}$). We consider queries $\varphi_{\mathcal{S}}$, where $\mathcal{S}$ is a set of processes, which ask whether some process in $\mathcal{S}$ can reach location Bad. In the worst case (last line), we remove every diagonal constraints. It is worth noticing that even in this case, our algorithm behaves as the algorithm proposed in [3].

Acknowledgments. We wish to thank Kim Larsen, for his invitation in CISS, in Aalborg, to realize the implementation presented here, and Gerd Behrmann, Alexandre David, and Emmanuel Fleury for helpful discussions on this subject.

References

1. R. Alur and D. Dill. A theory of timed automata. *Theor. Comp. Sci.*, 126(2):183–235, 1994.
2. J. Bengtsson and W. Yi. On clock difference constraints and termination in reachability analysis of timed automata. In *Proc. 5th Int. Conf. on Formal Engineering Methods, LNCS* 2885, 491–503. Springer, 2003.
3. J. Bengtsson and W. Yi. Timed automata: Semantics, algorithms and tools. In *Proc. 4th Advanced Course on Petri Nets, LNCS* 3098, 87–124. Springer, 2004.
4. B. Bérard, V. Diekert, P. Gastin, and A. Petit. Characterization of the expressive power of silent transitions in timed automata. *Fund. Informaticae*, 36(2–3):145–182, 1998.
5. P. Bouyer. Untameable timed automata! In *Proc. 20th Annual Symp. on Theor. Aspects of Comp. Sci., LNCS* 2607, 620–631. Springer, 2003.
6. P. Bouyer. Forward analysis of updatable timed automata. *Formal Methods in System Design*, 24(3):281–320, 2004.
7. P. Bouyer and F. Chevalier. On conciseness of extensions of timed automata. *Journal of Automata, Languages and Combinatorics*, 2006. To appear.
8. P. Bouyer, F. Laroussinie, and P.-A. Reynier. Diagonal constraints in timed automata: Forward analysis of timed systems. In *Proc. 3rd Int. Conf. on Formal Modelling and Analysis of Timed Systems, LNCS* 3829, 112–126. Springer, 2005.
9. E. Fersman, P. Petterson, and W. Yi. Timed automata with asynchrounous processes: Schedulability and decidability. In *Proc. 8th Int. Conf. on Tools and Algo. for the Construction and Analysis of Systems, LNCS* 2280, 67–82. Springer, 2002.
10. K. G. Larsen, P. Pettersson, and W. Yi. UPPAAL in a nutshell. *Journal of Software Tools for Technology Transfer*, 1(1–2):134–152, 1997.
11. A. Zbrzezny. Sat-based reachability checking for timed automata with diagonal constraints. *Fund. Informaticae*, 67(1–3):303–322, 2005.