Folk Theorems on the Determinization and Minimization of Timed Automata

Stavros Tripakis

VERIMAG
Centre Equation
2, avenue de Vignate, 38610 Gières, France
www-verimag.imag.fr

Abstract. Timed automata are known not to be complementable or determinizable. Natural questions are, then, could we check whether a given TA enjoys these properties? These problems are not algorithmically solvable, if we require not just a yes/no answer, but also a witness. Minimizing the "resources" of a TA (number of clocks or size of constants) are also unsolvable problems. Proofs are provided as simple reductions from the universality problem. These proofs are not applicable to the corresponding decision problems, the decidability of which remains open.

1 Introduction

Timed automata [2] (TA) have been established as a convenient model for describing timed systems. This is despite the fact that the model does not enjoy a number of important properties which hold, for instance, in its untimed counter-part, finite-state automata. In particular, timed automata are not *complementable* in general, meaning that, given a TA A, there does not always exist a TA accepting the complement of the language accepted by A. This holds even if we interpret timed automata as accepting finite-length words, which is the framework we follow in this paper. Timed automata are also not *determinizable* in general, meaning that, given a (non-deterministic) TA A, there does not always exist a deterministic TA accepting the same language.

Complementation is important for capturing the negation of logical specification by automata, in so-called automata-theoretic verification. Determinization is crucial for implementability and essential in problems of observation, fault diagnosis and test generation (e.g., [13]). Often, works in such domains (e.g., [11, 10]) are restricted to determinizable sub-classes of TA, for instance, so-called event-clock automata [3].

Given these facts, it is natural to ask: "can it be *checked* whether a given TA A is complementable/determinizable ?". Unfortunately, as we show in this paper, this cannot be done algorithmically, assuming we require not just a "yes/no" answer but also a *witness*, that is, a TA complementing/determinizing A. Interestingly, we do not know if the decision problems (admitting a "yes/no" answer) are decidable. The proofs we provide rely on the construction of a witness and are based on a reduction of the universality problem, known to be undecidable.

K.G. Larsen and P. Niebert (Eds.): FORMATS 2003, LNCS 2791, pp. 182–188, 2004.

Another set of questions concerns TA *minimization*, in the sense of reduction of "resources" of timed automata.[1] While the resources of untimed automata can be seen to be states and transitions, in timed automata, the clocks and the constants used in the guards are also important resources. In fact, these are in some sense more "expensive" resources than states and transitions, since most decidable problems concerning timed automata have worst-case complexity polynomial in the states and transitions and exponential in the clocks and constants [2, 7].

Given these facts, it is natural to ask: "can it be checked whether the number of clocks or the size of the constants of a given TA can be reduced ?". Unfortunately, this cannot be done algorithmically (assuming, as previously, that we require not just a "yes/no" answer but also a witness).

These results are probably folk theorems in the timed automata community. However, to the best of our knowledge, they have not yet been published. An exception is a similar result appearing in [15]. There, it is shown that computing the *clock-degree* of a given timed language (represented as a timed automaton) cannot be done algorithmically. The clock-degree is the minimum number of clocks necessary to recognize the language.

Reduction of clocks by removing *inactive* clocks has been considered in [8]. The idea is that the value of some clocks is irrelevant in certain discrete states, because the clock is not tested and will be reset upon leaving that state. However, the static analysis technique used in [8] to remove inactive clocks is not powerful enough to answer the question we are asking. Indeed, active clocks may still be redundant with respect to language equivalence. Also, *minimizable timed automata* (MTA) are introduced in [12]. In an MTA, clocks have bounded time domains. The MTA is also equipped with a set of *relevance formulas* permitting to identify equivalent states modulo inactive clocks. The authors show how a minimal MTA can be algorithmically obtained from a given MTA, where minimality is with respect to states and bisimulation equivalence.

Preliminaries: We assume the reader is familiar with timed automata. We consider a basic TA model, namely, automata with a finite set of discrete states and transitions, where each transition is labeled with a letter in a finite alphabet Σ, has a so-called *diagonal-free* clock guard [5] (i.e., no constraints of the form $x - y \leq c$) and a set of clocks to reset to zero. We use the following notation. $\mathcal{R}$ is the set of positive reals. $\mathcal{U} = (\Sigma \times \mathcal{R})^*$ is the set of all finite-length timed words over Σ. Given $L \subseteq \mathcal{U}$, $\overline{L}$ is the complement of L, that is, $\overline{L} = \mathcal{U} - L$. Given a timed automaton A over Σ, $L(A) \subseteq \mathcal{U}$ is the set of all finite-length timed words accepted by A. The *universality problem* is to check, given a TA A, whether $L(A) = \mathcal{U}$. The problem is known to be undecidable [2]. The *untimed language* of A, denoted $L_u(A)$, is equal to the set of all finite-length words in Σ^*

[1] We use the term minimization in the sense of reduction of "resources", and not in the sense of computing the quotient with respect to a bisimulation relation, as in [1, 16, 14].

accepted by A if we interpret A as a finite-state automaton, that is, ignoring its clock constraints.

2 Complementability

Problem 1. Given a TA A, does there exist a TA B such that $L(B) = \overline{L(A)}$? If so, construct such a B.

Theorem 1. *Problem 1 is not Turing computable.*[2]

Proof. We can reduce the universality problem to Problem 1, as follows. Given A, solve Problem 1. If B exists, $L(A) = \mathcal{U}$ iff $L(B) = \emptyset$. If B does not exist, then $L(A) \neq \mathcal{U}$, because the empty language can be accepted by a timed automaton with no accepting states.

Note that the proof relies on the fact that we have a witness and can check emptiness on it. We do not know whether the decision problem corresponding to Problem 1 (which only asks whether B exists) is decidable. Also notice that knowing the existence of a witness does not help in finding one. Enumerating all possible witnesses does not help, since checking for a given B whether $L(B) = \overline{L(A)}$ is undecidable.

3 Determinizability

Problem 2. Given a TA A, does there exist a deterministic TA B such that $L(B) = L(A)$? If so, construct such a B.

Theorem 2. *Problem 2 is not Turing computable.*

Proof. We can reduce the universality problem to Problem 2, as follows. Given A, solve Problem 2. If B exists, compute C such that $L(C) = \overline{L(B)}$: since B is deterministic, this can be done simply by turning accepting states into non-accepting states and vice-versa. Then, $L(A) = \mathcal{U}$ iff $L(C) = \emptyset$. If B does not exist, then $L(A) \neq \mathcal{U}$, because the language $\mathcal{U}$ can be accepted by a deterministic timed automaton with a single accepting state, no clocks, and a self-loop for each letter in Σ.

4 Minimization

4.1 Reducing the Number of Clocks

Problem 3. Given a TA A with n clocks, does there exist a TA B with $n-1$ clocks, such that $L(B) = L(A)$? If so, construct such a B.

[2] With a slight language abuse, when we say a problem is computable we mean that the implicitly defined function solving the problem is computable. For instance, in the case of Problem 1, this function takes a TA A and returns, either a TA B such that $L(B) = \overline{L(A)}$, or $\perp$, when such a B does not exist.

We should note that the technique of clock reduction by removing inactive clocks, proposed in [8], does not solve Problem 3. Indeed, consider the timed automaton that performs a and resets $x := 0$, then has two transitions with b, one with guard $x > 1$ and another with guard $x \leq 1$. In this automaton, clock x is redundant: the two transitions labeled with b can be replaced by a single transition without any guard. However, the method of [8] finds that clock x is active, because it is tested after it is reset.

Solving Problem 3 can be used in minimizing the number of clocks of A: just keep trying to remove clocks one by one until no clocks are left or until no more clocks can be removed. In particular, the problem, given A, to find, if it exists, an automaton B without any clocks, such that $L(B) = L(A)$, can be reduced to Problem 3. This observation is used in the proof below.

Theorem 3. *Problem 3 is not Turing computable.*

Proof. We can reduce the universality problem to the minimization problem, as follows. Given A, check whether there exists B with no clocks such that $L(B) = L(A)$ and if so construct such a B. If B exists, check whether the untimed language of B, $L_u(B)$, is equal to Σ^*. If $L_u(B) = \Sigma^*$, then $L(A) = L(B) = \mathcal{U}$. Indeed, B interpreted as a regular automaton accepts all finite words over Σ, and since it has no clocks, it cannot put any time constraints on them. Thus, when interpreted as a timed automaton, B accepts all finite timed words over Σ. If $L_u(B) \neq \Sigma^*$, then there is some $w \in \Sigma^* - L_u(B)$. Again, since B has no timing constraints, $w \notin L(B)$, thus, $L(A) \neq \mathcal{U}$. If B does not exist then $L(A) \neq \mathcal{U}$, since $\mathcal{U}$ can be accepted by an automaton with no clocks.

4.2 Reducing the Size of Constants

Problem 4. Given a TA A where constants are not greater than c, does there exist a TA B where constants are not greater than $c-1$, such that $L(B) = L(A)$? If so, construct such a B.

Solving Problem 4 is enough for minimizing the size of constants of A: just keep trying to reduce the size of constants by one until it becomes zero or until it can be reduced no more. In particular, the problem, given A, to find, if it exists, an automaton B with constants at most zero, such that $L(B) = L(A)$, can be reduced to Problem 3.

Lemma 1. *Let A be a TA over Σ with constants at most 0. There exists a finite-state automaton A_u over $\Gamma = \Sigma \cup \{\tau\}$, where $\tau \notin \Sigma$, such that $L(A) = \mathcal{U}$ iff $L(A_u) = \Gamma^*$.*

Proof. There are two types of guards in A: $x > 0$ or $x = 0$, where x is a clock. For each clock x of A, A_u will have one variable $b_x \in \{0, 1\}$. A_u will have the same set of discrete states as A. For every discrete transition of A, A_u will have a transition labeled with the same letter. For every reset $x := 0$ of the transition of A, we add a reset $b_x := 0$ to the transition of A_u. For every guard $x = 0$

(resp. $x > 0$) of the transition of A, we add a guard $b_x = 0$ (resp. $b_x = 1$) to the transition of A_u. At every discrete state of A_u we add a *self-loop* transition labeled by τ, which sets each variable b_x to 1. Notice that the language of A_u is closed under "stuttering" of τ, for instance, if $\tau a_0 \tau a_1 \cdots \tau a_n \in L(A_u)$ then $\tau^+ a_0 \tau^+ a_1 \cdots \tau^+ a_n \subseteq L(A_u)$ and vice-versa. This is because taking a τ transition two or more times in a row leaves the state of A_u unchanged.

Define the following equivalence between states of A and states of A_u. Given a state (q, v) of A (q is a discrete state and v is a vector of values for each clock x) and a state (q', u) of A_u (q' is a discrete state and u is a vector of values for each variable b_x), the two states are equivalent, denoted $(q, v) \sim (q', u)$, if $q = q'$ and for all i, $v(i) = 0 \Leftrightarrow u(i) = 0$. We claim that if $s_1 \sim s_2$ then:

1. for each $a \in \Sigma$ and state s_1' of A such that $s_1 \xrightarrow{a} s_1'$, there exists state s_2' of A_u such that $s_2 \xrightarrow{a} s_2'$ and $s_1' \sim s_2'$;
2. for each $a \in \Sigma$ and state s_2' of A_u such that $s_2 \xrightarrow{a} s_2'$, there exists state s_1' of A such that $s_1 \xrightarrow{a} s_1'$ and $s_1' \sim s_2'$;
3. for each $t \in \mathcal{R}$ and state s_1' of A such that $s_1 \xrightarrow{t} s_1'$, there exists state s_2' of A_u such that $s_2 \xrightarrow{\tau} s_2'$ and $s_1' \sim s_2'$;
4. for each state s_2' of A_u such that $s_2 \xrightarrow{\tau} s_2'$, for each $t \in \mathcal{R}$, there exists state s_1' of A such that $s_1 \xrightarrow{t} s_1'$ and $s_1' \sim s_2'$.

The above four properties allow us to prove that $L(A) = \mathcal{U}$ iff $L(A_u) = \Gamma^*$.

Theorem 4. *Problem 4 is not Turing computable.*

Proof. By Lemma 1, checking universality of a TA with constants at most zero is decidable. Since checking universality of a general TA is undecidable, Problem 4 is not computable.

5 Similar Problems with "Bounded Resources"

One might think that the above negative results could be remedied if one bounds the resources of the witness automaton. A similar approach is taken in [9, 4], where it actually results in a decidable version of an otherwise undecidable problem. Unfortunately, this is not the case for the problems defined in this paper.

More precisely, given non-negative integers n and c, let $TA(n, c)$ be the class of timed automata having at most n clocks and where constants are at most c. Then, the bounded-resource versions of Problems 1, 2, 3, and 4 can be stated as follows.

Problem 5. Given a TA A and non-negative integers n, c, does there exist a TA $B \in TA(n, c)$ such that $L(B) = \overline{L(A)}$? If so, construct such a B.

Problem 6. Given a TA A and non-negative integers n, c, does there exist a deterministic TA $B \in TA(n, c)$ such that $L(B) = L(A)$? If so, construct such a B.

Problem 7. Given a TA A with n clocks and non-negative integer c, does there exist a TA $B \in TA(n-1, c)$, such that $L(B) = L(A)$? If so, construct such a B.

Problem 8. Given a TA A with constants not greater than c and non-negative integer n, does there exist a TA $B \in TA(n, c-1)$, such that $L(B) = L(A)$? If so, construct such a B.

It turns out that all four problems above are not computable. The proofs are almost identical to the ones for the unbounded-resource versions, with the addition that we set n and/or c to zero when reducing the universality problem to the problem in question. For example, in the case of Problem 5, if there exists no B in $TA(0,0)$ such that $L(B) = \overline{L(A)}$ then $L(A) \neq \mathcal{U}$, since there is a TA with no clocks accepting the empty language. If $B \in TA(0,0)$ exists then $L(A) \neq \mathcal{U}$ iff $L(B) = \emptyset$.

6 Conclusions and Open Questions

The folk theorems presented in this paper confirm some inherent undecidability properties of the timed automata model. A number of open questions remain. We do not know whether the decision problems corresponding to the problem defined in this paper are decidable.

Another interesting problem is minimization of the number of discrete states of a TA (while possibly increasing the number of clocks or size of constants). The interesting cases are when diagonal guards or resets to constants other than zero are not allowed. Otherwise, a discrete state can be encoded as the ordering $x_1 < x_2 < \cdots < x_m$ of a sufficient number of extra clocks $x_1, ..., x_m$ and moving to this state can be encoded with an appropriate reset, such as $x_1 := 0, x_2 := 1, ..., x_m := m$. Note that, although these features do not add to the expressiveness of the model, removing them can only be done at the expense of adding discrete states [5, 6].

Acknowledgements

The author would like to thank Kim Larsen for his invitation to BRICS, Aalborg, where part of this work was conducted. Thanks to Yassine Lakhnech for pointing out Wilke's Ph.D. thesis. Thanks to Rajeev Alur, Eugene Asarin and Joel Ouaknine for valuable discussions.

This work has been partially supported by European IST project "Next TTA" under project No IST-2001-32111.

References

[1] R. Alur, C.Courcoubetis, N. Halbwachs, D. L. Dill, and H. Wong-Toi. Minimization of timed transition systems. In *3rd Conference on Concurrency Theory CONCUR '92*, volume 630 of *Lecture Notes in Computer Science*, pages 340–354. Springer-Verlag, 1992. 183

[2] R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994. 182, 183
[3] R. Alur, L. Fix, and T. Henzinger. A determinizable class of timed automata. In *CAV'94*, volume 818. LNCS, Springer, 1994. 182
[4] P. Bouyer, D. D'Souza, P. Madhusudan, and A. Petit. Timed control with partial observability. In *CAV'03*, 2003. 186
[5] P. Bouyer, C. Dufourd, E. Fleury, and A. Petit. Are timed automata updatable? In *CAV'00*. LNCS 1855, 2000. 183, 187
[6] P. Bouyer, C. Dufourd, E. Fleury, and A. Petit. Expressiveness of updatable timed automata. In *MFCS'00*. LNCS 1893, 2000. 187
[7] C. Courcoubetis and M. Yannakakis. Minimum and maximum delay problems in real-time systems. In *Computer-Aided Verification*, LNCS 575. Springer-Verlag, 1991. 183
[8] C. Daws and S. Yovine. Reducing the number of clock variables of timed automata. In *Proc. 17th IEEE Real-Time Systems Symposium, RTSS'96*, 1996. 183, 185
[9] D. D'Souza and P. Madhusudan. Controller synthesis for timed specifications. In Springer LNCS, editor, *STACS'02*, volume 2285, 2002. 186
[10] B. Nielsen and A. Skou. Automated test generation from timed automata. In *TACAS'01*. LNCS 2031, Springer, 2001. 182
[11] J. Springintveld, F. Vaandrager, and P. D'Argenio. Testing timed automata. *Theoretical Computer Science*, 254, 2001. 182
[12] J. G. Springintveld and F. W. Vaandrager. Minimizable timed automata. In *FTRTFT'96*, pages 130–147. LNCS 1135, 1996. 183
[13] S. Tripakis. Fault diagnosis for timed automata. In *FTRTFT'02*. LNCS 2469, Springer, 2002. 182
[14] S. Tripakis and S. Yovine. Analysis of timed systems using time-abstracting bisimulations. *Formal Methods in System Design*, 18(1):25–68, January 2001. 183
[15] Thomas Wilke. *Automaten und Logiken zur Beschreibung zeitabhängiger Systeme*. PhD thesis, Institut Für Informatik und Praktische Mathematik, Christian-Albrechts Universität, Kiel, 1994. In German. 183
[16] M. Yannakakis and D. Lee. An efficient algorithm for minimizing real-time transition systems. *Formal Methods in System Design*, 11(2), 1997. 183