Folk theorems on the determinization and minimization of timed automata [⋆]

Stavros Tripakis

CNRS, Verimag Laboratory [1]

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. Minimizing the "resources" of a TA (number of clocks or size of constants) are also unsolvable problems. In this paper we provide simple undecidability proofs using a "constructive" version of the problems where we require not just a yes/no answer, but also a "witness". Proofs are then simple reductions from the universality problem. Recent work of Finkel shows that the corresponding decision problems are also undecidable [1].

Key words: formal methods, specification languages, timed automata, determinization, decidability

1 Introduction

Timed automata [3] (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

⋆ A previous version of this paper first appeared in [2]. Work partially supported by CNRS STIC project "CORTOS" and by IST Network of Excellence "ARTIST2".
Email address: `tripakis@imag.fr` (Stavros Tripakis).
URL: `http://www-verimag.imag.fr/ tripakis/` (Stavros Tripakis).

[1] Centre Equation, 2, avenue de Vignate, 38610 Gières, France.

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 useful in problems of observation and testing.

Given these facts, it is natural to ask: "can it be *checked* whether a given TA A is complementable ?" or "can it be *checked* whether a given TA A is determinizable ?". Unfortunately, this cannot be done algorithmically. In this paper we provide simple undecidability proofs of "constructive" versions of the above problems, where we require not just a "yes/no" answer but also a *witness*, that is, a TA complementing/determinizing A. Recent work of Finkel shows that the corresponding decision problems are also undecidable [1]. Our proofs rely on having a witness and are based on a reduction of the universality problem, known to be undecidable.

Another set of questions concerns TA *minimization*, in the sense of reduction of "resources" of timed automata. [2] 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 [3,7].

Given this, 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. As with the previous problems, we provide simple reductions of the universality problem. Again, we use the versions requiring a witness.

Related work

These results are probably folk theorems in the timed automata community. However, to the best of our knowledge, they had not been published before [2]. An exception is the result of Wilke [8] which shows 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. Also, recent work of Finkel shows that the

[2] 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 [4–6].

decision problems corresponding to the problems we study in this paper are also undecidable [1]. His proofs also reduce the universality problem, however, the constructions are more elaborate.

Reduction of clocks by removing *inactive* clocks has been considered in [9]. A clock is inactive in a given discrete state when the clock will be reset before it is tested. The static analysis technique used in [9] to find 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 (see Section 4.1 for an example).

Minimizable timed automata (MTA) are introduced in [10]. 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, a clock guard of one of the forms $x < k$, $x \leq k$, $x - y < k$, $x - y \leq k$ or boolean combinations of these, 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 [3]. The *untimed language* of A, denoted $L_u(A)$, is equal to the *projection* of $L(A)$ onto Σ, that is, the set of all finite-length words in Σ^* obtained by removing from a timed word of A all timestamps and leaving only the discrete letters in Σ. For example, the projection of the timed word $(a, 1)(b, 2)$ is the word ab.

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 undecidable.*[3]

[3] Problem 1 is not a decision problem thus, strictly speaking, the term "undecid-

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. $\square$

The above proof relies on the fact that we have a witness and can check emptiness on it. Using a more elaborate reduction of universality, Finkel recently showed that the decision problem corresponding to Problem 1 (which only asks whether B exists) is also undecidable [1]. 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 undecidable.*

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. $\square$

able" is not appropriate. We use this term throughout the paper, however, for the sake of brevity. What we mean is that the implicitly-defined function solving the problem is not Turing-computable. 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.

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.*

The above problem is related to the problem of computing the *clock degree* of a TA A, considered in [8]. The clock degree is the minimum number of clocks necessary to recognize $L(A)$, that is, the minimum m for which there exists a TA B with m clocks such that $L(B) = L(A)$. Wilke shows that computing the clock degree is undecidable. Let us recall his result here.

Theorem 3 ([8]) *Computing the clock degree of a TA is undecidable.*

Proof Suppose we can compute the clock degree i of a given TA A. If $i > 0$ then $L(A) \neq \mathcal{U}$, since there is obviously a timed automaton without clocks accepting $\mathcal{U}$. If $i = 0$ then $L(A) = \mathcal{U}$ iff $L_u(A) = \Sigma^*$, that is, the untimed language of A contains all finite untimed words. Indeed, if $L(A) = \mathcal{U}$ then obviously $L_u(A) = \Sigma^*$. Conversely, if $L(A) \neq \mathcal{U}$ then let $\rho \in \mathcal{U} - L(A)$. Also, let B be the automaton without clocks such that $L(B) = L(A)$ (we will only use B in the proof, thus we do not need the witness B). We claim that no ρ' that has the same untimed projection as ρ can be in $L(B)$ (thus, neither in $L(A)$). This is because B has no clocks therefore it cannot distinguish between ρ and ρ'. Consequently, the untimed projection of ρ is not in $L_u(A)$, thus, $L_u(A) \neq \Sigma^*$. $\Box$

The undecidability of Problem 3 follows from the above result.

Theorem 4 *Problem 3 is undecidable.*

Proof We can reduce the problem of computing the clock degree of a TA A to Problem 3 as follows. We keep trying to remove clocks from A one by one until no clocks are left or until no more clocks can be removed. We do this by repeatedly solving Problem 3: if a witness B with one clock less is found then A is replaced by B and we continue; otherwise we stop. The clock degree of A is the number of clocks remaining when we stop. $\Box$

We should note that the technique of clock reduction by removing inactive clocks, proposed in [9], 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 [9] finds that clock x is active, because it is tested after it is reset.

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 4.

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 We can assume that A is *diagonal-free*, that is, contains only two types of clock guards: $x > 0$ or $x = 0$. This is because any TA can be transformed to a diagonal-free TA accepting the same language [3,11]. This transformation only adds discrete states and does not modify the constants used in the guards.

We next construct A_u. For each clock x of A, A_u will have one variable $b_x \in \{0,1\}$: $b_x = 0$ corresponds to $x = 0$ and $b_x = 1$ corresponds to $x > 0$. 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 5 *Problem 4 is undecidable.*

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 [12,13], 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 difficulties of the timed automata model regarding complementation, determinization and minimization of clocks or constants. We presented some simple undecidability proofs of "constructive" versions of these problems, where a witness is required. Recent work of Finkel shows that the corresponding decision problems are also undecidable [1].

An interesting open problem is minimization of the number of discrete states (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 [11].

References

[1] O. Finkel, On decision problems for timed automata, Bulletin of the European Association for Theoretical Computer Science 87 (2005) 185–190.

[2] S. Tripakis, Folk theorems on the determinization and minimization of timed automata, in: Formal Modeling and Analysis of Timed Systems (FORMATS'03), Vol. 2791 of LNCS, Springer, 2004.

[3] R. Alur, D. Dill, A theory of timed automata, Theoretical Computer Science 126 (1994) 183–235.

[4] R. Alur, C. Courcoubetis, N. Halbwachs, D. Dill, H. Wong-Toi, Minimization of timed transition systems, in: 3rd Conference on Concurrency Theory CONCUR '92, Vol. 630 of LNCS, Springer-Verlag, 1992, pp. 340–354.

[5] M. Yannakakis, D. Lee, An efficient algorithm for minimizing real-time transition systems, Formal Methods in System Design 11 (2).

[6] S. Tripakis, S. Yovine, Analysis of timed systems using time-abstracting bisimulations, Formal Methods in System Design 18 (1) (2001) 25–68.

[7] C. Courcoubetis, M. Yannakakis, Minimum and maximum delay problems in real-time systems, in: CAV'91, LNCS 575, Springer, 1991.

[8] T. Wilke, Automaten und logiken zur beschreibung zeitabhängiger systeme, Ph.D. thesis, Institut Für Informatik und Praktische Mathematik, Christian-Albrechts Universität, Kiel, in German (1994).

[9] C. Daws, S. Yovine, Reducing the number of clock variables of timed automata, in: Proc. 17th IEEE Real-Time Systems Symposium, RTSS'96, 1996.

[10] J. Springintveld, F. Vaandrager, Minimizable timed automata, in: FTRTFT'96, Vol. 1135 of LNCS, Springer, 1996, pp. 130–147.

[11] B. Berard, A. Petit, V. Diekert, P. Gastin, Characterization of the expressive power of silent transitions in timed automata, Fundamenta Informaticae 36 (2-3) (1998) 145–182.

[12] D. D'Souza, P. Madhusudan, Timed control synthesis for external specifications, in: S. LNCS (Ed.), STACS'02, Vol. 2285, 2002.

[13] P. Bouyer, D. D'Souza, P. Madhusudan, A. Petit, Timed control with partial observability, in: CAV'03, 2003.