Alternating Timed Automata over Bounded Time

Mark Jenkins
Oxford University Computing Laboratory, UK
Email: mark.jenkins@comlab.ox.ac.uk

Joël Ouaknine
Oxford University Computing Laboratory, UK
Email: joel@comlab.ox.ac.uk

Alexander Rabinovich
School of Computer Science, Tel Aviv University, Israel
Email: rabinoa@post.tau.ac.il

James Worrell
Oxford University Computing Laboratory, UK
Email: jbw@comlab.ox.ac.uk

Abstract

Alternating timed automata are a powerful extension of classical Alur-Dill timed automata that are closed under all Boolean operations. They have played a key role, among others, in providing verification algorithms for prominent specification formalisms such as Metric Temporal Logic. Unfortunately, when interpreted over an infinite dense time domain (such as the reals), alternating timed automata have an undecidable language emptiness problem.

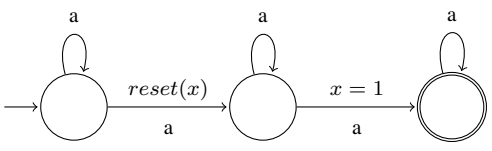
The main result of this paper is that, over bounded time domains, language emptiness for alternating timed automata is decidable (but non-elementary). The proof involves showing decidability of a class of parametric McNaughton games that are played over timed words and that have winning conditions expressed in monadic logic over the signature with order and the $+1$ function.

As a corollary, we establish the decidability of the time-bounded model-checking problem for Alur-Dill timed automata against specifications expressed as alternating timed automata.

1. Introduction

Timed automata were introduced by Alur and Dill in [3] as a natural and versatile model for real-time systems. They have been widely studied ever since, both by practitioners and theoreticians. A celebrated result concerning timed automata, which originally appeared in [2], is the PSPACE decidability of the *language emptiness* (or *reachability*) problem.

Unfortunately, the *language inclusion* problem—given two timed automata $\mathcal{A}$ and $\mathcal{B}$, is every timed word accepted by $\mathcal{A}$ also accepted by $\mathcal{B}$?—is known to be undecidable. A related fact is that timed automata are not closed under complementation. For example, the automaton below accepts every timed word in which there are two a-events separated by one time unit:



The complement automaton would have to accept a timed word precisely when no two a-events are separated by one time unit. Intuitively, this is not expressible by a timed automaton, since such an automaton would need a potentially unbounded number of clocks to keep track of the time delay from each a-event. We refer the reader to [17] for a formal analysis of these considerations.

In one sense, the non-closure under complementation is easy to remedy—one simply generalises the transition mode to allow both conjunctive and disjunctive transitions, an idea borrowed from the theory of untimed automata that dates back 30 years [11]. Such untimed *alternating automata* have played key roles in algorithms for complementing Büchi automata (see, e.g., [20]), temporal logic verification [38], [23], and analysis of parity games [14]. In the timed world, the resulting *alternating timed automata* [21], [27], [22], [29], [13] subsume ordinary timed automata and can be shown to be closed under all Boolean operations. They have been used, among others, to provide model-checking algorithms for various fragments of Metric Temporal Logic (MTL); see, e.g., [27], [28], [8]. Unfortunately, the price to pay for the increase in expressiveness is the undecidability of language emptiness for alternating timed automata!

This undecidability follows immediately from the undecidability of universality for timed automata. The proof of the latter in [3] uses in a crucial way the unboundedness of the time domain. Roughly speaking, this allows one to encode arbitrarily long computations of a Turing machine. On the other hand, many verification questions are naturally phrased over bounded time domains [34], [5], [19]. For example, a run of a communication protocol might normally be expected to have an *a priori* time bound. In fact, most hard real-time problems, which typically involve deadlines, timeouts, and delays, are only pertinent over a finely circumscribed time span. This leads us to consider the *time-bounded language emptiness problem* for alternating timed automata. This problem asks, given an alternating timed automaton $\mathcal{A}$ and a time bound N, whether some finite timed word of duration at most N is accepted by $\mathcal{A}$. (Note that, since we are working with a dense model of time, time-bounded words may still contain arbitrarily many events.) The main result of this paper is that this problem is decidable but non-elementary. Since alternating timed automata are closed under all Boolean operations, an immediate corollary is the decidability of the time-bounded model-checking problem for timed automata against specifications expressed as alternating timed automata.

Our proofs exploit the close correspondence between automata and monadic predicate logic. Büchi [10] and Rabin [30] have respectively proven decidability of monadic second-order logic (MSO) over the naturals and the infinite binary tree using translations of MSO to automata. Conversely, automata can easily be transformed into equivalent MSO formulas. The proof of our main result involves a translation from alternating timed automata to monadic predicate logic over the structure $(\mathbb{T}, <, +1)$, where $\mathbb{T}$ is a bounded interval of reals and $+1$ is the relation defined by $+1(x, y)$ if and only if $x + 1 = y$. The $+1$ relation is used to encode timing constraints in automata.

We translate timed alternating automata into *games* over $(\mathbb{T}, <, +1)$ with winning conditions expressed in monadic predicate logic. The class of games that we obtain is a variation of that introduced by McNaughton [25] in connection with *Church's problem* [12], [37]. Given an $\mathsf{MSO}(<)$-formula $\varphi(X, Y)$, Church's problem asks whether there exists a *causal* operator[1] F on predicates such that $\forall X\, \varphi(X, F(X))$. Thus Church's problem generalises the satisfiability problem for MSO to a *uniformisation* problem. The games that we introduce can be generalised and used to analyse a natural extension of Church's problem for $\mathsf{MSO}(<, +1)$ over bounded intervals of the reals, but we do not pursue this direction here. For our current purpose the key result is a procedure to determine the winner of games from this class. This enables us to establish the decidability of the language emptiness problem for alternating timed automata.

It is worth noting that we reduce the language emptiness problem for alternating timed automata to a *uniformisation* problem for *first-order* logic over $(\mathbb{T}, <, +1)$. By contrast, in the classical translations of (untimed) automata to monadic logic, language emptiness is reduced to a *satisfiability* problem for *second-order* logic over $(\mathbb{N}, <)$ [23]. It does not seem possible to give a uniform reduction of the language emptiness problem for alternating timed automata to the satisfiability problem for monadic second-order logic over $(\mathbb{T}, <, +1)$. We discuss this question further in Section 6.

Related Work. The quest for a decidable class of timed automata with good closure properties has led to a considerable body of work, including the introduction of determinisable subclasses of timed automata [4], restrictions to one-clock automata [29], [22], and bounded-variability semantics [39]. See also Henzinger *et al.*'s paper on *fully decidable* formalisms [16].

The present paper substantially generalises some of the main results of [26], in which we established the decidability of both the time-bounded language inclusion problem for timed automata (2EXPSPACE-complete), and the time-bounded MTL model-checking problem for timed automata (EXPSPACE-complete). Both decidability results now easily follow from our new Theorem 15, since MTL formulas can be encoded as one-clock alternating timed automata of a particular type [27]. Theorem 15, in which no restrictions whatsoever are placed on alternating timed automata, does not seem amenable to the proof techniques of [26] and instead requires novel game-theoretic tools. The increased expressiveness, however, comes at the price of a significant blow-up in complexity: from EXPSPACE or 2EXPSPACE to non-elementary (see Theorem 19).

There is an extensive body of work concerning games on timed automata and related timed-graph formalisms. This originates in [18], [24] and encompasses concurrent games [9], weighted timed automata [1], [7] and tool support [6]. Turn-based games on timed automata can easily be encoded as McNaughton games in the sense of the present paper, thanks to the expressiveness of the logic $\mathsf{MSO}(<, +1)$. However the generality of our class of McNaughton games entails that our decidability results are restricted to bounded time domains, in contrast with [24].

A key aspect of our technical development is the use of *parametric* games. This is related to the works of [15], [33] on Church's problem with parameters.

2. Alternating Timed Automata

Let Σ be a finite alphabet and let $\mathbb{R}_+$ denote the set of non-negative reals. A *timed event* is a pair (a, t), where $t \in \mathbb{R}_+$ is called the *timestamp* of the *event* $a \in \Sigma$. A *timed word* is a finite sequence $w = (a_1, t_1)(a_2, t_2) \ldots (a_n, t_n)$ of timed events whose corresponding sequence of timestamps is strictly increasing.[2]

We denote by $\mathrm{untime}(w)$ the underlying untimed word $a_1 a_2 \ldots a_n$. A set of timed words is called a *timed language*. If $\mathbb{T} \subseteq \mathbb{R}_+$ then $\mathbb{T}\Sigma^*$ denotes the set of timed words $w = (a_1, t_1)(a_2, t_2) \ldots (a_n, t_n)$ such that $t_1, t_2, \ldots, t_n \in \mathbb{T}$. In this paper we are particularly interested in timed languages $L \subseteq \mathbb{T}\Sigma^*$ for bounded sub-intervals of $\mathbb{R}_+$. For uniformity we assume that $\mathbb{T} = [0, N)$ for $N \in \mathbb{N}$, although our results can easily be adapted arbitrary bounded intervals.

2.1. Automata

Let C be a set of *clock variables*. A *clock valuation* is a function $\nu : C \to \mathbb{R}_+$. If $R \subseteq C$ is a set of clock variables then $\nu[R := 0]$ denotes the valuation that maps each clock $x \in R$ to 0 and agrees with ν on all other clocks. The zero clock valuation $\mathbf{0}$ is defined by $\mathbf{0}(x) = 0$ for all $x \in C$. The set $\Phi(C)$ of *clock constraints* φ is defined by the following grammar:

$$\varphi := \mathbf{true} \mid \mathbf{false} \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid x \sim k\,,$$

where $k \in \mathbb{N}$ and $\sim \in \{<, \leq, =, \geq, >\}$.

[1] F is a causal operator if the truth value of $F(X)(n)$ for a monadic predicate X and individual n only depends on the values of $X(m)$ for $m \leq n$.

[2] Thus our model of time is *strongly monotonic*; in contrast, *weakly monotonic* models allow multiple events to happen 'simultaneously' (or, more precisely, with null-duration delays between them). The main results of this paper remain substantively the same under either approach, although weakly monotonic models cause slight complications.

Note also that we are restricting ourselves to *finite* timed words (with, however, no *a priori* bound on the number of events). This is a natural assumption in the context of bounded time domains: infinite timed words would necessarily be Zeno, a situation usually proscribed in research on real-time systems.

Recall from [3] that the transition relation of a timed automaton can be seen as a partial function

$$\delta : S \times \Sigma \times \Phi(C) \to \mathcal{P}(S \times \mathcal{P}(C))\,,$$

where S is the set of *locations* and Σ the alphabet of the automaton. The meaning of $(s', R) \in \delta(s, a, \varphi)$ is that from state (s, ν), where ν satisfies φ, the automaton can read letter a and transition to state $(s', \nu[R := 0])$.

In the definition of an *alternating timed automaton* the transition function is generalised to a partial function

$$\delta : S \times \Sigma \times \Phi(C) \to \mathcal{B}_+(S \times \mathcal{P}(C))\,.$$

Here $\mathcal{B}_+(P)$ denotes the set of positive Boolean formulas over a set of propositions P, generated by the grammar

$$\psi := \mathbf{true} \mid \mathbf{false} \mid p \mid \psi_1 \vee \psi_2 \mid \psi_1 \wedge \psi_2\,,$$

where $p \in P$. A subset M of P *satisfies* $\varphi \in \mathcal{B}_+(P)$ if the truth assignment that assigns $\mathbf{true}$ to elements of M and $\mathbf{false}$ to elements of $P \setminus M$ satisfies φ.

In state (s, ν), if ν satisfies the clock constraint φ and $\{(s_1, R_1), \ldots, (s_k, R_k)\}$ satisfies $\delta(s, a, \varphi)$, then we think of the automaton as having a *conjunctive transition* $(s, \nu) \xrightarrow{a} \{(s_1, \nu_1), \ldots, (s_k, \nu_k)\}$, where $\nu_i = \nu[R_i := 0]$.

Definition 1: An ***alternating timed automaton*** is a tuple $\mathcal{A} = (\Sigma, S, C, s_0, F, \delta)$, where

- Σ is a finite alphabet
- S is a finite set of locations
- C is a finite set of clock variables
- $s_0 \in S$ is the initial location
- $F \subseteq S$ is a set of accepting locations
- $\delta : S \times \Sigma \times \Phi(C) \to \mathcal{B}_+(S \times \mathcal{P}(C))$ is a partial function with finite domain.

We also adopt the following simplifying assumption without loss of generality:[3]

> **Partition.** For each location s and letter $a \in \Sigma$, the set of constraints φ such that $\delta(s, a, \varphi)$ is defined forms a partition of the set $(\mathbb{R}_+)^C$ of clock valuations.

Before formally defining the language accepted by an alternating timed automaton, we give some examples.

Example 2: We define an automaton $\mathcal{A}$ over alphabet $\Sigma = \{a\}$ that accepts those words such that for every timed event (a, t) with $t < 1$ there is an event $(a, t + 1)$ exactly one time unit later. $\mathcal{A}$ has a single clock x and set of locations $\{s, u\}$, with s initial and accepting, and u non-accepting. The transition function is defined by

$$\delta(s, a, x < 1) = (s, \emptyset) \wedge (u, \{x\}) \qquad \delta(s, a, x \geq 1) = (s, \emptyset)$$
$$\delta(u, a, x \neq 1) = (u, \emptyset) \qquad \delta(u, a, x = 1) = \mathbf{true}$$

[3]This assumption does not affect the decidability of the model. It can affect the complexity by an exponential factor, but in our main result the complexity is non-elementary so this is inconsequential.

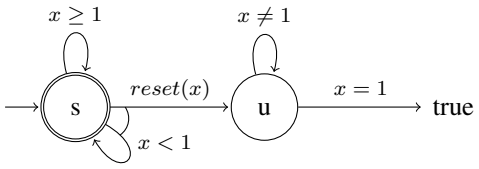


Fig. 1. Automaton $\mathcal{A}$

The automaton is illustrated in Figure 1 in which we represent the conjunctive transition by connecting two arrows with an arc.

A run of $\mathcal{A}$ starts in location s. Every time an a-event occurs in the first time unit, the automaton makes a simultaneous transition to both s and u, thus opening up a new thread of computation equipped with a fresh copy of the clock x. The automaton must eventually leave location u, which is non-accepting, and it can only do so exactly one time unit after first entering the location.

Example 3: We define an automaton $\mathcal{B}$, shown in Figure 2, over alphabet $\{a\}$ that accepts those words such that for all consecutive pairs of events (a, t_i) and (a, t_{i+1}) with $t_i, t_{i+1} < 1$ there is no subsequent event (a, t_j) with $t_i + 1 < t_j < t_{i+1} + 1$. Excepting some corner cases[4], this requirement says that for each event (a, t_j) with $1 < t_j < 2$ there is an event $(a, t_j - 1)$ exactly one time unit earlier. Indeed, if there were no such event then letting (a, t_i) be the latest event with $t_i < t_j - 1$ and (a, t_{i+1}) the earliest event with $t_j - 1 < t_{i+1}$ we see that the input word would be rejected by $\mathcal{B}$.

$\mathcal{B}$ has two clocks x and y, and set of locations $\{s, u, v\}$, with s initial and all locations accepting. The transition function is defined by

$$\begin{aligned}
\delta(s, a, x < 1) &= (s, \emptyset) \wedge (u, \{x\}) \\
\delta(s, a, x \geq 1) &= (s, \emptyset) \\
\delta(u, a, \mathbf{true}) &= (v, \{y\}) \\
\delta(v, a, x < 1 \vee y > 1) &= (v, \emptyset) \\
\delta(v, a, x \geq 1 \wedge y \leq 1) &= \mathbf{false}\,.
\end{aligned}$$

From location s every time an a-event occurs the automaton starts a new thread in location u, resetting clock x. On the next a-event this new thread transitions to location v, resetting clock y. Thereafter the thread only allows a-events if $x < 1$ or $y > 1$; by this mechanism the automaton blocks timed events (a, t_j), $t > 1$, for which there is a consecutive pairs of events (a, t_i) and (a, t_{i+1}) with $t_i + 1 < t_j < t_{i+1} + 1$.

A run of an alternating timed automaton $\mathcal{A} = (\Sigma, S, C, s_0, F, \delta)$ over a timed word $w = (a_1, t_1)(a_2, t_2) \ldots (a_n, t_n)$ is a finite dag satisfying the following conditions: (i) each vertex is a triple (i, s, ν), with $0 \leq i \leq n$, $s \in S$ is a location, and ν is a clock

[4]We should also require that if (a, t_j) is the first or last event with $1 \leq t_j < 2$, then there be an earlier event (a, t_i) with $t_i = t_j - 1$. Based on the same ideas involved in the definition of $\mathcal{B}$, one can easily define an automaton $\mathcal{C}$ that accepts precisely those words satisfying this requirement.

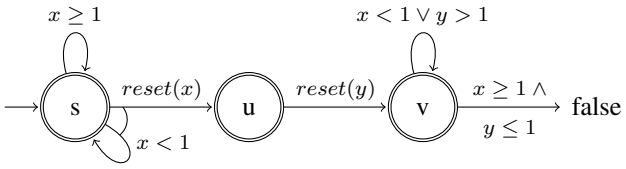


Fig. 2. Automaton $\mathcal{B}$

valuation; (ii) there is a vertex $(0, s_0, \mathbf{0})$; (iii) each vertex (i, s, ν), $i \leq n-1$, has a (possibly empty) set of children of the form $\{(i+1, s_1, \nu_1), \ldots, (i+1, s_k, \nu_k)\}$ where, writing $\nu' = \nu + t_{i+1} - t_i$ (and adopting the convention that $t_0 = 0$), there is a conjunctive transition $(s, \nu') \xrightarrow{a_i} \{(s_1, \nu_1), \ldots, (s_k, \nu_k)\}$.

The run is *accepting* if for each vertex (n, s, ν), s is an accepting location; in this case we say that the timed word w is *accepted* by $\mathcal{A}$. The *language* $L_{\mathbb{T}}(\mathcal{A})$, is the set of words in $\mathbb{T}\Sigma^*$ that are accepted by $\mathcal{A}$.

2.2. The Acceptance Game

In this paper we exploit the fact that acceptance of a timed word $w = (a_1, t_1)(a_2, t_2)\ldots(a_n, t_n)$ by an alternating timed automaton $\mathcal{A} = (\Sigma, S, C, s_0, F, \delta)$ has a game-theoretic characterisation.

Define the *acceptance game* $\mathbb{G}(\mathcal{A}, w)$ between *Automaton* and *Pathfinder* as follows. A *state* of the game $\mathbb{G}(\mathcal{A}, w)$ is a triple (i, s, ν), with $0 \leq i \leq n$, $s \in S$ is a location, and ν is a clock valuation. Play starts in state $(0, s_0, \mathbf{0})$ and consists of n rounds. Suppose that at the beginning of the $(i+1)$-th round, $0 \leq i \leq n-1$, the state is (i, s_i, ν_i) and write $\nu' = \nu_i + t_{i+1} - t_i$. Then Automaton selects a model M for $\delta(s_i, a_i, \varphi)$, where ν' satisfies φ; Pathfinder responds by choosing an atom $(s, R) \in M$, and the next state is $(i+1, s, \nu'[R := 0])$. Automaton wins if either Pathfinder cannot move on his turn (in case Automaton selects the empty model $M = \emptyset$), or if the game ends in an accepting state after the last round.

A *partial play* is a finite sequence of consecutive game states. A *strategy* for Automaton is a mapping that assigns to each such sequence a next move of Automaton. Such a strategy is *winning* if Automaton wins any play in which the strategy is followed. The following result is straightforward.

Proposition 4: A timed word $w \in \mathbb{T}\Sigma^*$ is accepted by $\mathcal{A}$ if and only if Automaton has a *winning strategy* in $\mathbb{G}(\mathcal{A}, w)$.

One of the motivations for introducing alternating timed automata is that they enjoy better closure properties than ordinary timed automata.

Proposition 5—[22], [29]: For any time domain $\mathbb{T} \subseteq \mathbb{R}_+$ the class of languages $L \subseteq \mathbb{T}\Sigma^*$ accepted by alternating timed automata is effectively closed under union, intersection, and complement.

Closure under union and intersection is straightforward since we allow both disjunction and conjunction in the transition function. Thanks to the Partition Assumption one can complement an automaton by simply interchanging accepting and non-accepting states and exchanging conjunctions and disjunctions in the transition function.

Example 6: Taking the intersection of the automaton $\mathcal{A}$ in Example 2, the automaton $\mathcal{B}$ in Example 3, and the automaton $\mathcal{C}$ mentioned in the footnote to Example 3, all defined over alphabet $\{a\}$, one obtains the automaton $\mathcal{A}_{\text{copy}}$. Over time domain $\mathbb{T} = [0, 2)$, $L_{\mathbb{T}}(\mathcal{A}_{\text{copy}})$ consists of those timed words $w = (a, t_1)(a, t_2)\ldots(a, t_{2n})$ such that $t_{i+n} = t_i + 1$ for $1 \leq i \leq n$, i.e., such that the $+1$ function defines a one-to-one correspondence between the set of events in the first time unit and the set of events in the second time unit.

3. Monadic Second-Order Logic

Throughout this section we assume a fixed time domain $\mathbb{T} = [0, N)$. We consider *monadic second-order logic* (MSO) over the structure $(\mathbb{T}, <, +1)$, where $+1(x, y)$ holds if and only if $x + 1 = y$. The syntax of $\mathsf{MSO}(<, +1)$ has as vocabulary first-order variables $t_1, t_2, \ldots$, monadic predicate variables $X_1, X_2, \ldots$, and the binary relations $+1$ and $<$. Atomic formulas are of the form $X(t)$, $t_1 < t_2$, $+1(t_1, t_2)$, and $t_1 = t_2$. Well-formed formulas are obtained from atomic formulas using Boolean connectives, the first-order quantifiers $\exists t$ and $\forall t$, and the second-order quantifiers $\exists X$ and $\forall X$. If we omit the $+1$ relation then we obtain the sub-logic $\mathsf{MSO}(<)$. We denote sets of monadic predicates in boldface and write $\varphi(\mathbf{X})$ for a formula whose free second-order variables are drawn from the set $\mathbf{X}$. In the sequel we reserve the letters W, X, Y to denote monadic predicate variables, and P, Q, R to denote their interpretations as subsets of $\mathbb{T}$.

Example 7: Fix a finite set $\mathbf{W}$ of monadic predicate variables, and consider the timed word $w = (a_1, t_1)(a_2, t_2)\ldots(a_n, t_n)$ over alphabet $\Sigma = 2^{\mathbf{W}} \setminus \emptyset$. We associate with w a structure M_w that extends $(\mathbb{T}, <, +1)$ with interpretations of the monadic predicate variables $\mathbf{W}$, where $W \in \mathbf{W}$ is interpreted as the set $\{t_i : W \in a_i\}$.

Shelah [35] showed that the satisfiability problem for $\mathsf{MSO}(<)$ over the non-negative reals (and hence also over any non-empty interval of reals) is undecidable. He also proved, however, that decidability can be recovered by restricting second-order quantification to countable sets. Given our interest in modelling finite timed words in the manner of Example 7, the following stronger restriction is natural and is assumed henceforth.

> **Finiteness.** All free predicate variables are interpreted by finite sets, and second-order quantification ranges over finite sets.

Decidability of $\mathsf{MSO}(<)$ under this restriction follows from Shelah's result mentioned above.

Observe that the models arising from timed words in Example 7 obey the finiteness assumption; conversely any model arises from a unique timed word. Thus we may identify the set

of models of a formula $\varphi(\mathbf{W})$ of $\mathsf{MSO}(<,+1)$ with a timed language over alphabet $2^{\mathbf{W}} \setminus \emptyset$.

Let $w = (a_1,t_1)(a_2,t_2)\ldots(a_n,t_n)$ and $w' = (a_1,t'_1)(a_2,t'_2)\ldots(a_n,t'_n)$ be timed words with $\mathrm{untime}(w) = \mathrm{untime}(w')$. Then the associated models are order-isomorphic provided that t_1 and t'_1 are either both zero or both non-zero. Building on this observation, we can represent the set of all models of an $\mathsf{MSO}(<)$-formula as a regular untimed language.

Definition 8: Let $\mathbf{W}$ be a set of predicate variables. Without loss of generality, assume that $\mathbf{W}$ contains a distinguished predicate W_0 with a fixed interpretation as the set $\{0\}$. A collection $\mathcal{M}$ of interpretations of $\mathbf{W}$ in $\mathbb{T}$ is said to be ***regular*** if there exists a regular (untimed) language L on alphabet $\Sigma = 2^{\mathbf{W}} \setminus \emptyset$ such that

$$\mathcal{M} = \{M_w : w \in \mathbb{T}\Sigma^*,\ \mathrm{untime}(w) \in L\}\,.$$

Thanks to the convention on W_0, the first letter of any word in L denotes the set of predicates true at time 0 in the associated model.

The following proposition follows from [31, Theorem 1].

Proposition 9: The set of models of $\varphi(\mathbf{W}) \in \mathsf{MSO}(<)$ is effectively regular.

4. McNaughton Games

McNaughton [25] gave a formulation of Church's problem in terms of two-player games over the structure $(\mathbb{N},<)$ with $\mathsf{MSO}(<)$ winning conditions. In this section we introduce a class of McNaughton-like games over bounded time domains $\mathbb{T} = [0,N)$ with winning conditions in $\mathsf{MSO}(<,+1)$. These games are shown to generalise the class of acceptance games for alternating timed automata as defined in Section 2. Following [15], [32] we consider games with parameters. The presence of parameters is crucial both in setting up the correspondence with the acceptance game and in our inductive proof of decidability for McNaughton games.

Let $\varphi(\mathbf{W},\mathbf{X},\mathbf{Y})$ be an $\mathsf{MSO}(<,+1)$ formula with free variables among $\mathbf{W}$, $\mathbf{X}$ and $\mathbf{Y}$. We think of $\mathbf{X}$ as a set of variables under the control of *Player I*, $\mathbf{Y}$ as a set of variables under the control of *Player II*, and $\mathbf{W}$ as a set of *parameters*: each instantiation of $\mathbf{W}$ yields a different game. Let $\mathbf{P}$ be an interpretation of $\mathbf{W}$ with $t_1 < t_2 < \cdots < t_n$ an enumeration of $\bigcup \mathbf{P}$, i.e., the set of points where some predicate in $\mathbf{P}$ holds. The *McNaughton game* $\mathbb{G}(\varphi,\mathbf{P})$ is a turn-based game in which Player I and Player II run through the sequence of timestamps $t_1,\ldots,t_n$, successively choosing values for their predicates at each timestamp. More formally:

(i) The games consists of n rounds. In the i-th round Player I chooses a bit vector $b_i \in \{0,1\}^{\mathbf{X}}$ and then Player II chooses a bit vector $b'_i \in \{0,1\}^{\mathbf{Y}}$;

(ii) At the conclusion of the game, Player I has constructed an interpretation $\mathbf{Q}$ of $\mathbf{X}$ that assigns each variable $X \in \mathbf{X}$ to the set $\{t_i : 1 \le i \le n, b_i(X) = 1\}$. Likewise, Player II has constructed an interpretation $\mathbf{R}$ of $\mathbf{Y}$ that assigns each variable $Y \in \mathbf{Y}$ to the set $\{t_i : 1 \le i \le n, b'_i(Y) = 1\}$;

(iii) If $\mathbb{T} \models \varphi(\mathbf{P},\mathbf{Q},\mathbf{R})$ then Player I is the winner; otherwise Player II is the winner.

Say that the sequence of moves in the first i rounds of $\mathbb{G}(\varphi,\mathbf{P})$ determines a *partial play* $(b_1,b'_1)\ldots(b_i,b'_i)$, where $b_j \in \{0,1\}^{\mathbf{X}}$ and $b'_j \in \{0,1\}^{\mathbf{Y}}$ for $1 \le j \le i$. A strategy for Player I is a function from the set of partial plays to the set $\{0,1\}^{\mathbf{X}}$. Such a strategy is *winning* if Player I wins all plays in which the strategy is followed.

In the next section we consider the decidability of the following two questions:

Decision Problem. Given a formula $\varphi(\mathbf{W},\mathbf{X},\mathbf{Y})$, does there exist an interpretation $\mathbf{P}$ of the set of parameters $\mathbf{W}$ for which Player I has a winning strategy in the game $\mathbb{G}(\varphi,\mathbf{P})$?

Winning-Region Problem. Compute a representation of the set of parameters $\mathbf{P}$ for which Player I has a winning strategy in the game $\mathbb{G}(\varphi,\mathbf{P})$.

We use the computability of the winning-region problem for $\mathsf{MSO}(<)$ winning conditions to establish decidability of the decision problem for $\mathsf{MSO}(<,+1)$ winning conditions.

4.1. Relating the McNaughton and Acceptance Games

We briefly sketch how the language emptiness problem for an alternating timed automaton $\mathcal{A} = (\Sigma,S,C,s_0,F,\delta)$ over time domain $\mathbb{T}$ can be reduced to the decision problem for McNaughton games over $\mathbb{T}$. To this end, we construct a formula $\varphi_{\mathcal{A}}(\mathbf{W},\mathbf{X},\mathbf{Y})$ of $\mathsf{MSO}(<,+1)$ so as to establish a correspondence between the acceptance game $\mathbb{G}(\mathcal{A},w)$ for a given timed word w (cf. Section 2) and the McNaughton game $\mathbb{G}(\varphi_{\mathcal{A}},\mathbf{P})$ for a given interpretation $\mathbf{P}$ of $\mathbf{W}$ that is determined by w.

The set of parameters $\mathbf{W} = \{W_\sigma : \sigma \in \Sigma\}$ contains a predicate variable for each alphabet symbol. Each timed word $w \in \mathbb{T}\Sigma^*$ naturally determines an interpretation $\mathbf{P}$ of $\mathbf{W}$ over $\mathbb{T}$, cf. Example 7.

We imagine that Player I in $\mathbb{G}(\varphi_{\mathcal{A}},\mathbf{P})$ takes the role of Automaton in $\mathbb{G}(\mathcal{A},w)$, and Player II takes the role of Pathfinder. For each set of atoms $M \subseteq S \times \mathcal{P}(C)$ we postulate a variable X_M whose truth value encodes the choice of Automaton to select model M in a given round. Similarly, for each atom $\alpha \in S \times \mathcal{P}(C)$ we postulate a variable Y_α whose truth value encodes the choice of Pathfinder to select atom α in a given round.

We instrument the formula $\varphi_{\mathcal{A}}$ so that Player I wins the McNaughton game $\mathbb{G}(\varphi_{\mathcal{A}},\mathbf{P})$ if and only if Automaton wins the acceptance game $\mathbb{G}(\mathcal{A},w)$. The key components of $\varphi_{\mathcal{A}}$ are subformulas φ_{aut} and φ_{path} that ensure that Player I and Player II correctly simulate Automaton and Pathfinder respectively.

The formula φ_{path} ensures that in each round Pathfinder can only choose one atom α, which must moreover be selected from

the set M chosen by Automaton. This is expressed as:

$$\forall u \bigwedge_{\alpha} \left(Y_\alpha(u) \rightarrow \bigvee_{M \ni \alpha} X_M(u) \right) \wedge \forall u \bigwedge_{\alpha \neq \beta} \neg(Y_\alpha(u) \wedge Y_\beta(u)) \,.$$

The formula φ_{aut} ensures that in each round Automaton chooses a model M satisfying the transition function δ. It is the conjunction over all locations $s \in S$, inputs $\sigma \in \Sigma$ and guards $\psi \in \Phi(C)$, such that $\delta(s, \sigma, \psi)$ is defined, of the formulas

$$\forall u \forall v \left((\mathit{state}_s(u) \wedge \mathit{next}(u, v) \wedge W_\sigma(v) \wedge \mathit{const}_\psi(v)) \rightarrow \bigvee_{M \models \delta(s,\sigma,\psi)} X_M(v) \right).$$

Here $\mathit{state}_s(u)$ and $\mathit{next}(u, v)$ are easily defined auxiliary formulas, respectively expressing that the automaton is in state s at time u, and that u and v are consecutive timestamps in the input word. Similarly, $\mathit{const}_\psi(v)$ expresses that the clock constraint $\psi \in \Phi(C)$ holds at time u. For example, in case $\psi \equiv x \sim k$ we define $\mathit{const}_\psi(v)$ to be the formula

$$\forall u \forall w \left((\mathit{reset}_x(u) \wedge (u < w < v \rightarrow \neg \mathit{reset}_x(w))) \rightarrow v \sim u + k \right),$$

where $\mathit{reset}_x(u)$ is an auxiliary formula expressing that clock x was reset at time u (which information is available from $Y_\alpha(u)$).

We define $\varphi_{\mathcal{A}} := \varphi_{\mathrm{aut}} \wedge (\varphi_{\mathrm{path}} \wedge \varphi_{\mathrm{init}} \rightarrow \varphi_{\mathrm{fin}})$, where φ_{init} and φ_{fin} are easily defined formulas, respectively expressing that play in the acceptance game must start in an initial location and finish in an accepting location.

It is now straightforward to prove the following:

Proposition 10: Automaton wins the acceptance game $\mathbb{G}(\mathcal{A}, w)$ if and only if Player I wins the McNaughton game $\mathbb{G}(\varphi_{\mathcal{A}}, \mathbf{P})$, where $\mathbf{P}$ is the set of parameters associated with the timed word w.

Remark 11: Note that $\varphi_{\mathcal{A}}$ has no second-order quantifiers, that is, it is a formula in the first-order fragment of $\mathsf{MSO}(<, +1)$.

5. Main Result

In this section we show decidability of the decision problem for McNaughton games over bounded time domains. From this we derive decidability of the time-bounded language emptiness problem for alternating timed automata. The proof has three parts: first we recall from [26] a satisfiability preserving and reflecting translation from $\mathsf{MSO}(<, +1)$ to $\mathsf{MSO}(<)$ over bounded time domains; next we show how to solve the winning-region problem for McNaughton games with $\mathsf{MSO}(<)$ winning conditions; finally we combine the first two contributions to solve the decision problem for games with $\mathsf{MSO}(<, +1)$ winning conditions.

Throughout this section, let $\mathbb{T} = [0, N)$ be a fixed time domain.

5.1. Eliminating the Metric

Given an $\mathsf{MSO}(<, +1)$ formula φ, we define a straightforward syntactic transformation into an $\mathsf{MSO}(<)$ formula $\overline{\varphi}$ such that there is a natural bijection between models of φ over $[0, N)$ and models of $\overline{\varphi}$ over $[0, 1)$.

Let $\mathbf{X}$ be the set of monadic predicates appearing in φ. With each predicate $X \in \mathbf{X}$, we associate a collection $X_0, \ldots, X_{N-1}$ of N fresh monadic predicates. We then write $\overline{\mathbf{X}} = \{X_i \mid X \in \mathbf{X}, 0 \leq i \leq N-1\}$. Intuitively, each X_i is a predicate on $[0, 1)$ that represents X over the subinterval $[i, i+1)$. Formally, an interpretation of $X \in \mathbf{X}$ over $[0, N)$ yields an interpretation of X_i over $[0, 1)$ by defining $X_i(t)$ if and only if $X(i + t)$. Note that this correspondence yields a bijection between interpretations of $\mathbf{X}$ on $[0, N)$ and interpretations of $\overline{\mathbf{X}}$ on $[0, 1)$.

We can assume that φ does not contain any (first- or second-order) existential quantifiers, by replacing them with combinations of universal quantifiers and negations if need be. It is also convenient to rewrite φ into a formula that makes use of a unary function $+1$ instead of the $+1$ relation. To this end, replace every occurrence of $+1(x, y)$ in φ by $(x < N - 1 \wedge x + 1 = y)$.

Next, replace every instance of $\forall x\, \psi$ in φ by the formula

$$\forall x \left(\psi[x/x] \wedge \psi[x + 1/x] \wedge \ldots \wedge \psi[x + (N-1)/x] \right),$$

where $\psi[t/x]$ denotes the formula resulting from substituting every free occurrence of the variable x in ψ by the term t. Intuitively, this transformation is legitimate since first-order variables in our target formula will range over $[0, 1)$ rather than $[0, N)$.

Having carried out these substitutions, use simple arithmetic to rewrite every term in φ as $x + k$, where x is a variable and $k \in \mathbb{N}$ is a non-negative integer constant.

Every inequality occurring in φ is now of the form $x + k < N - 1$ or $x + k_1 < y + k_2$. Replace every inequality of the first kind by $\mathbf{true}$ if $k + 2 \leq N$ and by $\mathbf{false}$ otherwise, and replace every inequality of the second kind by (i) $x < y$, if $k_1 = k_2$; (ii) $\mathbf{true}$, if $k_1 < k_2$; and (iii) $\mathbf{false}$ otherwise.

Every equality occurring in φ is now of the form $x + k_1 = y + k_2$. Replace every such equality by $x = y$ if $k_1 = k_2$, and by $\mathbf{false}$ otherwise.

Every use of monadic predicates in φ now has the form $X(x + k)$, for $k \leq N - 1$. Replace every such predicate by $X_k(x)$.

Finally, replace every occurrence of $\forall X\, \psi$ in φ by $\forall X_0 \forall X_1 \ldots \forall X_{N-1}\, \psi$. The resulting formula is the desired $\overline{\varphi}$. Note that $\overline{\varphi}$ does not mention the $+1$ function, and is therefore indeed a non-metric (i.e., purely order-theoretic) sentence in $\mathsf{MSO}(<)$.

The association between φ and $\overline{\varphi}$ is formalised in the following proposition.

Proposition 12: $\varphi(\mathbf{X})$ holds in the structure $([0, N), <, +1)$ if and only if the transformed formula $\overline{\varphi}(\overline{\mathbf{X}})$ holds in the structure $([0, 1), <)$.

5.2. The Regularity Lemma

Recall our assumption that free and bound predicate variables are interpreted by finite sets. Recall the notion of a regular set of predicates from Definition 8.

Lemma 13: [Regularity Lemma] Let $\varphi(\mathbf{W}, \mathbf{X}, \mathbf{Y})$ be a formula of $\mathsf{MSO}(<)$. Then the set $\{\mathbf{P} : \text{Player I wins } \mathbb{G}(\varphi, \mathbf{P})\}$ is effectively regular.

Proof: According to Proposition 9 we can compute a deterministic finite automaton $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ over alphabet $\Sigma = 2^{\mathbf{W} \cup \mathbf{X} \cup \mathbf{Y}} \setminus \emptyset$ whose language represents the set of models of φ. We seek an automaton over alphabet $\Gamma = 2^{\mathbf{W}} \setminus \emptyset$ representing the set of interpretations $\mathbf{P}$ of $\mathbf{W}$ such that Player I wins $\mathbb{G}(\varphi, \mathbf{P})$.

Let $w = w_1 w_2 \dots w_n \in \Gamma^n$ represent an interpretation of $\mathbf{W}$ (in the sense of Definition 8). By distinguishing the contribution of input bits between two protagonists, respectively called Player I and Player II, we define a *graph game*[5] $\mathbb{G}(\mathcal{A}, w)$, which can be seen as a discrete analogue of $\mathbb{G}(\varphi, \mathbf{P})$.

There are two kinds of vertices in $\mathbb{G}(\mathcal{A}, w)$, Player-I vertices and Player-II vertices respectively. For each automaton state $q \in Q$ and position $0 \leq i \leq n$, we include a Player-I vertex (q, i); if moreover $0 \leq i \leq n-1$ then we also include a Player-II vertex (q, i, b) for each bit vector $b \in \{0,1\}^{\mathbf{X}}$. If $0 \leq i \leq n-1$ then we include an edge from (q, i) to (q, i, b), corresponding to a choice of bit vector b by Player I; for $b' \in \{0,1\}^{\mathbf{Y}}$ we also include an edge from (q, i, b) to $(q', i+1)$, where $q' = \delta(q, (w_{i+1}, b, b'))$, corresponding to a choice of bit vector b' by Player II.

The rules of the game $\mathbb{G}(\mathcal{A}, w)$ are as follows. Player I chooses moves at Player-I vertices and Player II chooses moves at Player-II vertices. Play starts in the vertex $(q_0, 0)$ and Player I wins if play reaches a vertex (q, n) with $q \in F$.

The game $\mathbb{G}(\mathcal{A}, w)$ essentially represents an untiming of $\mathbb{G}(\varphi, \mathbf{P})$: the moves in each games are the same once one elides the timestamp associated with each round in the latter game. In particular, Player I wins $\mathbb{G}(\mathcal{A}, w)$ if and only if Player I wins $\mathbb{G}(\varphi, \mathbf{P})$.

For $E \subseteq Q$ a set of automaton states and $w \in \Gamma^*$, we define the set $\mathrm{Pred}_w(E)$ of states from which Player I can force play into E on input w:

$$\begin{array}{rcll} \mathrm{Pred}_\varepsilon(E) & = & E & \\ \mathrm{Pred}_u(E) & = & \{q : \exists b_1 \, \forall b_2 \, \delta(q, (u, b_1, b_2)) \in E\} & u \in \Gamma \\ \mathrm{Pred}_{uw}(E) & = & \mathrm{Pred}_u(\mathrm{Pred}_w(E)) & u \in \Gamma, w \in \Gamma^* \end{array}$$

Given $w \in \Gamma^*$, it is straightforward that Player I wins $\mathbb{G}(\mathcal{A}, w)$ if and only if $q_0 \in \mathrm{Pred}_w(F)$. From this observation one can build an automaton $\mathcal{B}$ on alphabet Γ that accepts those words w such that Player I wins $\mathbb{G}(\mathcal{A}, w)$. The set of states of $\mathcal{B}$ is $\mathcal{P}(Q)$, with F the unique final state, and $\{S \subseteq Q : q_0 \in S\}$ the set of initial states. We include a transition $S \xrightarrow{b} T$ on input $b \in \Gamma$ if and only if $S = \mathrm{Pred}_b(T)$. ∎

[5] Graph games are a simple and classical notion [37]. We treat them informally to avoid overburdening the reader with yet more game-theoretic formalism.

A result similar to Lemma 13 was proven in [15] for parametric games over $(\mathbb{N}, <)$.

5.3. The Decision Procedure

Theorem 14: Let $\mathbb{T} = [0, N)$ be a fixed time domain. Given an $\mathsf{MSO}(<, +1)$-formula $\varphi(\mathbf{W}, \mathbf{X}, \mathbf{Y})$, it is decidable whether there exists an interpretation $\mathbf{P}$ of $\mathbf{W}$ over $\mathbb{T}$ such that Player I wins $\mathbb{G}(\varphi, \mathbf{P})$.

Proof: Applying the transformation described in Section 5.1 to $\varphi(\mathbf{W}, \mathbf{X}, \mathbf{Y})$ yields an $\mathsf{MSO}(<)$ formula $\overline{\varphi}(\overline{\mathbf{W}}, \overline{\mathbf{X}}, \overline{\mathbf{Y}})$, where $\overline{\mathbf{W}} = \{W_i : W \in \mathbf{W}, 0 \leq i < N\}$, $\overline{\mathbf{X}} = \{X_i : X \in \mathbf{X}, 0 \leq i < N\}$, and $\overline{\mathbf{Y}} = \{Y_i : Y \in \mathbf{Y}, 0 \leq i < N\}$. Then interpretations $\mathbf{P}, \mathbf{Q}, \mathbf{R}$ of $\mathbf{W}, \mathbf{X}, \mathbf{Y}$ as predicates on $[0, N)$ naturally yield interpretations $\overline{\mathbf{P}}, \overline{\mathbf{Q}}, \overline{\mathbf{R}}$ of $\overline{\mathbf{W}}, \overline{\mathbf{X}}, \overline{\mathbf{Y}}$ as predicates on $[0, 1)$ where, e.g., $P_i(t)$ holds if and only if $P(i + t)$ holds, $0 \leq t < 1$ and $0 \leq i < N$. By Proposition 12 we have that $\varphi(\mathbf{P}, \mathbf{Q}, \mathbf{R})$ holds if and only if $\overline{\varphi}(\overline{\mathbf{P}}, \overline{\mathbf{Q}}, \overline{\mathbf{R}})$ holds.

Observe, however, that there is a significant difference between the game $\mathbb{G}(\varphi, \mathbf{P})$, which is played over the interval $[0, N)$ and the game $\mathbb{G}(\overline{\varphi}, \overline{\mathbf{P}})$, played over $[0, 1)$. For example, in the former, for $X \in \mathbf{X}$ and $0 \leq t < 1$, Player I chooses the value of $X(t)$ before $X(t+1)$. However, in the latter, these values, respectively represented as $X_0(t)$ and $X_1(t)$, are chosen at the same time.

Instead we associate with $\mathbb{G}(\varphi, \mathbf{P})$ a sequence of McNaughton games $G_0, \dots, G_{N-1}$, each over the interval $[0, 1)$, and each with an $\mathsf{MSO}(<)$ winning condition. Intuitively the i-th game G_i corresponds to the restriction of $\mathbb{G}(\varphi, \mathbf{P})$ to the time interval $[i, i+1)$. Accordingly we have Player I choose the value of X_i, $X \in \mathbf{X}$ and Player II choose the value of Y_i, $Y \in \mathbf{Y}$ in G_i. The key insight in defining G_i is to treat the variables X_j, $X \in \mathbf{X}$ and Y_j, $Y \in \mathbf{Y}$ as additional parameters for each $j < i$. That is, the respective choices of Player I and Player II in the preceding games G_j, $j < i$ become parameters in G_i. (Strictly speaking each G_i is a *family* of games, one game for each instantiation of the extra parameters in φ_i.)

To be precise, the winning condition of G_i is an $\mathsf{MSO}(<)$-formula φ_i with free variables $\overline{\mathbf{W}}$, $\{X_j : X \in \mathbf{X}, 0 \leq j \leq i\}$ and $\{Y_j : Y \in \mathbf{Y}, 0 \leq j \leq i\}$. Of these, Player I controls X_i, $X \in \mathbf{X}$, Player II controls Y_i, $Y \in \mathbf{Y}$, and the remaining variables are parameters. The definition of the φ_i proceeds backwards, from φ_{N-1} down to φ_0, and is such that Player I wins $G(\varphi, \mathbf{P})$ if and only if he wins $G_0(\varphi_0, \overline{\mathbf{P}})$, which is an instantiation of G_0. This equivalence allows us to decide the winner of $G(\varphi, \mathbf{P})$.

To start with we define $\varphi_{N-1} := \overline{\varphi} \wedge \chi_{N-1}$, where χ_{N-1}, which constrains Player I and Player II to move only when one of the predicates W_{N-1}, $W \in \mathbf{W}$ is true, is defined by

$$\left(\bigwedge_{X \in \mathbf{X}} X_{N-1} \subseteq \bigcup_{W \in \mathbf{W}} W_{N-1} \right) \vee \left(\bigvee_{Y \in \mathbf{Y}} Y_{N-1} \not\subseteq \bigcup_{W \in \mathbf{W}} W_{N-1} \right).$$

Suppose we have defined the game G_{i+1} involving parameters $\overline{\mathbf{W}}$, $\{X_j : 0 \leq j \leq i, X \in \mathbf{X}\}$ and $\{Y_j : 0 \leq j \leq i, Y \in$

$\mathbf{Y}\}$. By the Regularity Lemma the set of interpretations of these parameters such that Player I wins G_{i+1} is effectively regular and is expressible by an $\mathsf{MSO}(<)$-formula ψ_i with the above parameters as its set of free variables. The winning condition for the game G_i is then defined to be $\varphi_i := \psi_i \wedge \chi_i$, where χ_i, which constrains Player I and Player II to move only when one of the predicates W_i, $W \in \mathbf{W}$ is true, is defined by

$$\left(\bigwedge_{X \in \mathbf{X}} X_i \subseteq \bigcup_{W \in \mathbf{W}} W_i\right) \vee \left(\bigvee_{Y \in \mathbf{Y}} Y_i \not\subseteq \bigcup_{W \in \mathbf{W}} W_i\right).$$

This completes the definition of the games $G_0, \ldots, G_{N-1}$. There is a natural bijective correspondence between the set of positions of $G(\varphi, \mathbf{P})$ and the set of positions of (the various instantiations of) the G_i, where a position of $G(\varphi, \mathbf{P})$ with timestamp t corresponds to a position of $G_{\lfloor t \rfloor}$ with timestamp $t - \lfloor t \rfloor$. Moreover this association preserves the identity of the winning player. In particular, Player I wins $G(\varphi, \mathbf{P})$ if and only if he wins $\mathbb{G}(\varphi_0, \overline{\mathbf{P}})$. We omit the details. ■

Recalling from Section 4.1 the reduction of the language emptiness problem for alternating timed automata to the game decision problem for $\mathsf{MSO}(<, +1)$, we obtain the following theorem, which is the central result of our paper:

Theorem 15: The time-bounded language emptiness problem for alternating timed automata is decidable.

Note, as an immediate corollary, that Theorem 15 entails the decidability of the time-bounded model-checking problem of timed automata against alternating timed automata specifications.

6. Complexity and Expressiveness

Define a family of functions $\exp_k : \mathbb{N} \to \mathbb{N}$ by $\exp_0(n) = n$ and $\exp_{k+1}(n) = 2^{\exp_k(n)}$. A function $f : \mathbb{N} \to \mathbb{N}$ is *non-elementary* if it grows faster than any $\exp_k$.

Our procedure for determining language emptiness for alternating timed automata has non-elementary complexity. This blow-up does not arise from the translation of $\mathsf{MSO}(<)$ formulas to automata in the proof of Proposition 9, since the quantifier-alternation depth of the relevant formulas is bounded independently of the size of the automata. Rather, the culprit is the exponential blow-up that occurs with each application of the Regularity Lemma in the proof of Theorem 14.

In this section we give a non-elementary lower bound for the language emptiness problem for alternating timed automata. We prove this by reduction from the emptiness problem for star-free regular expressions. We also discuss the related question of translating from alternating timed automata to equivalent $\mathsf{MSO}(<, +1)$ formulas.

6.1. Complexity Lower Bound

A *star-free regular expression* over alphabet Σ is built from the symbols $\emptyset$ and σ, for any $\sigma \in \Sigma$, using the operations of union ($+$), concatenation ($\cdot$), and complementation ($\neg$). Such an expression E denotes a language $L(E) \subseteq \Sigma^*$ which is defined as follows:

$L(\emptyset) = \emptyset$ and $L(\sigma) = \{\sigma\}$;
$L(E + E') = L(E) \cup L(E')$;
$L(E \cdot E') = L(E) \cdot L(E')$;
$L(\neg E) = \Sigma^* \setminus L(E)$.

The following result was shown in [36].

Theorem 16: The language emptiness problem for star-free regular expressions is non-elementary.

We give a polynomial-time reduction of the language emptiness problem for star-free regular expressions to the time-bounded language emptiness problem for alternating timed automata. Note that since language emptiness for (untimed) alternating automata is PSPACE-complete [11], such a reduction would not be possible in the untimed setting. Before describing the reduction we need some auxiliary notions concerning regular expressions.

The *operator depth* $\mathrm{odp}(E)$ of a regular expression E is defined as follows:

$\mathrm{odp}(\emptyset) = \mathrm{odp}(\sigma) = 1$;
$\mathrm{odp}(E + E') = \max\{\mathrm{odp}(E), \mathrm{odp}(E')\} + 1$;
$\mathrm{odp}(E \cdot E') = \max\{\mathrm{odp}(E), \mathrm{odp}(E')\} + 1$;
$\mathrm{odp}(\neg E) = \mathrm{odp}(E)$.

Note that negation does not count toward the operator depth.

Given a star-free regular expression E over alphabet Σ and a word $u \in \Sigma^*$ we define the *membership game* $\mathbb{G}(u, E)$. This is a two-player game with N rounds, where N is the operator depth of E. The two players are *Prover*, who is trying to show $u \in E$, and *Refuter*, who is trying to show $u \notin E$. The positions of the game are pairs (v, F) where v is a sub-word of u and F has the form G or $\neg G$ for G a sub-expression of E. The initial position is (u, E). Suppose the position at the start of a given round is (v, F), where $v = v_1 \ldots v_n$; then the round proceeds as follows:

- If $F \equiv F_1 \cdot F_2$ then Prover moves first by choosing an index i in v. Refuter responds by selecting either $(v_1 \ldots v_{i-1}, F_1)$ or $(v_i \ldots v_n, F_2)$ as the position in the next round;
- If $F \equiv \neg(F_1 \cdot F_2)$ then Refuter moves first by choosing an index i in v. Prover responds by selecting either $(v_1 \ldots v_{i-1}, \neg F_1)$ or $(v_i \ldots v_n, \neg F_2)$ as the position in the next round;
- If $F \equiv F_1 + F_2$ then Prover selects either (v, F_1) or (v, F_2) as the position in the next round;
- If $F \equiv \neg(F_1 + F_2)$ then Refuter selects either $(v, \neg F_1)$ or $(v, \neg F_2)$ as the position in the next round.

The positions (v, σ), $(v, \neg\sigma)$, $(v, \emptyset)$, and $(v, \neg\emptyset)$ are terminal. In these cases Prover wins if v is a member of the corresponding expression and Refuter wins otherwise.

It is clear that Prover has a winning strategy in $\mathbb{G}(u, E)$ if and only if $u \in L(E)$.

Definition 17: Suppose that E is a star-free regular expression over alphabet Σ. Given a time domain $\mathbb{T} = [0, N)$, we associate with E a timed language $L_{\mathbb{T}}(E) \subseteq \mathbb{T}\Sigma^*$ containing

those timed words $w = (a_1, t_1) \dots (a_n, t_n)$ satisfying the following two properties:

(i) There exists a word $u \in L(E)$ such that $\mathrm{untime}(w) = u^N$ consists of N copies of u;

(ii) Each successive copy of u in w is separated by one time unit, that is, $t_{i+|u|} = t_i + 1$ for $1 \leq i \leq Nn - |u|$, where $|u|$ denotes the length of u.

Example 18: In case $\mathbb{T} = [0, 2)$ and $\Sigma = \{a\}$ then $L_{\mathbb{T}}(\Sigma^*) = L_{\mathbb{T}}(\mathcal{A}_{\mathrm{copy}})$, where $\mathcal{A}_{\mathrm{copy}}$ is the automaton defined in Example 6.

Let E be a star-free regular expression of operator depth N and write $\mathbb{T} = [0, N)$. It holds by construction that $L_{\mathbb{T}}(E)$ is non-empty if and only if $L(E)$ is non-empty. Next we define an alternating timed automaton $\mathcal{A}_E$ such that $L_{\mathbb{T}}(\mathcal{A}_E) = L_{\mathbb{T}}(E)$.

There are two ideas behind the definition of $\mathcal{A}_E$. The first, following Example 18, is that $L_{\mathbb{T}}(\Sigma^*)$ is the language of a simple variant of $\mathcal{A}_{\mathrm{copy}}$—call it $\mathcal{B}$—that accepts its input if and only if the sub-word occurring in the first time unit is repeated in all subsequent time units. The second idea is that for an arbitrary expression E we can define $\mathcal{A}_E$ as the intersection of $\mathcal{B}$ with another automaton. Note that if w is accepted by $\mathcal{B}$ then $\mathrm{untime}(w) = u^N$ for some $u \in \Sigma^*$. The definition of $\mathcal{A}_E$ is such that $\mathcal{A}_E$ simulates the membership game $\mathbb{G}(u, E)$ by playing one round in each time unit. Intuitively, $\mathcal{A}_E$ simulates moves of Prover by disjunctive transitions and moves of Refuter by conjunctive transitions.

Recall that a position of the membership game $\mathbb{G}(u, E)$ is a pair (v, F), where v is a sub-word of u and F has the form G or $\neg G$ for a sub-expression G of E. In order to remember the game position (v, F) between successive time units, $\mathcal{A}_E$ stores F in its finite control, while it records v by resetting a clock x as it reads the first letter of v and resetting a clock y as it reads the last letter of v. Automaton $\mathcal{A}_E$ consists of N gadgets; one time unit passes between control entering and exiting each gadget.

There is an initialisation gadget that resets clock x on the first event of the timed word and resets clock y on the last event in the first time unit.

The gadget for $F \equiv F_1 \cdot F_2$ is illustrated in Figure 3 (we omit the labels on transitions). Referring to the appropriate clause in the membership game, the choice of when to take the transition exiting location s simulates the move of Prover to select an index of the input word; the two conjunctive branches of this transition simulate the choice of Refuter either to choose the left sub-word or the right sub-word.[6]

The gadget for $F \equiv \neg(F_1 \cdot F_2)$ is illustrated in Figure 4. It operates along similar lines to the gadget for $F_1 \cdot F_2$ and we omit detailed explanation.

Theorem 19: The time-bounded language emptiness problem for alternating timed automata is non-elementary.

It can be seen from the proof of Theorem 19 that the non-elementary lower bound applies for automata with only two

[6] Strictly speaking, to handle the case in which v is the empty word, we should include transitions in Figures 3 and 4 that allow clocks x and y to be reset simultaneously. But these are omitted for readability.

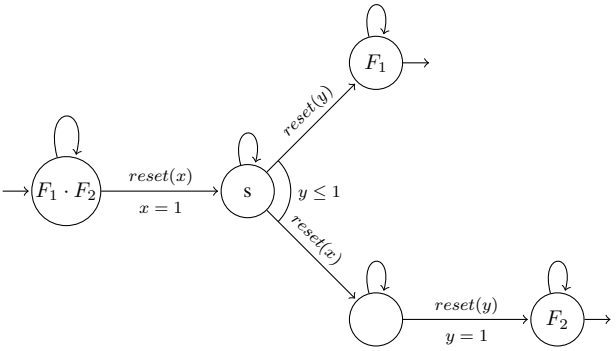


Fig. 3. Gadget for $F_1 \cdot F_2$

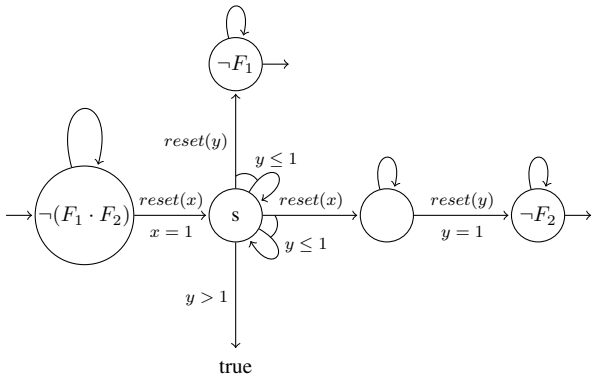


Fig. 4. Gadget for $\neg(F_1 \cdot F_2)$

clocks. On the other hand, for a fixed time bound the decision procedure for language emptiness presented in Theorem 15 is elementary. The translation of the language emptiness problem to the decision problem for McNaughton games detailed in Proposition 10 involves a winning condition $\varphi_{\mathcal{A}}$ whose quantifier depth is absolutely bounded. Furthermore the decision procedure for McNaughton games presented in Theorem 14 is elementary if the quantifier depth of the winning condition and the time bound are fixed.

6.2. Expressiveness

Nondeterministic automata on words and trees can be straightforwardly translated into equivalent $\mathsf{MSO}(<)$ formulas whose size is polynomial in the size of the automata and whose quantifier-alternation depth (both first- and second-order) is bounded independently of the automata. In these translations the formulas encode runs of the automata.

The paper [23] shows how to encode run dags of alternating automata over (untimed) words in $\mathsf{MSO}(<)$; however such a direct encoding fails for alternating timed automata since their run dags can have unbounded width. Indeed, the non-elementary lower bound for the time-bounded language emptiness problem seems to preclude a 'simple' uniform translation of alternating timed automata to $\mathsf{MSO}(<, +1)$. More pre-

cisely, there cannot be an elementary procedure that computes from a given automaton an $\mathsf{MSO}(<,+1)$ formula that defines the same language over all time domains $\mathbb{T}$ and such that the quantifier-alternation depth of the formula is bounded independently of the automaton. The existence of such a procedure would contradict Theorem 19 given that the satisfiability problem for $\mathsf{MSO}(<,+1)$ over bounded time intervals is elementary for formulas of a fixed quantifier-alternation depth [26]. However, for each fixed time domain $\mathbb{T} = [0, N)$, we can extract a translation from automata to equivalent $\mathsf{MSO}(<,+1)$-formulas from the proof of Theorem 14:

Theorem 20: Let $\mathbb{T} = [0, N)$ be a bounded time domain. Given a set of predicate symbols $\mathbf{W}$ and an alternating timed automaton $\mathcal{A}$ over alphabet $2^{\mathbf{W}} \setminus \emptyset$ one can compute a formula $\psi(\mathbf{W})$ of $\mathsf{MSO}(<,+1)$ such that $L(\mathcal{A}) = L(\varphi)$.

Proof: Consider the McNaughton game $\mathbb{G}(\varphi_{\mathcal{A}}, \mathbf{P})$ derived from the automaton $\mathcal{A}$ in the manner described in Section 4.1. We seek a formula $\psi(\mathbf{W})$ representing the set of predicates $\mathbf{P}$ for which Player I wins $\mathbb{G}(\varphi_{\mathcal{A}}, \mathbf{P})$.

Let $\overline{\mathbf{W}} = \{W_i : W \in \mathbf{W}, 0 \leq i \leq N-1\}$ be a fresh set of monadic predicates. Recall from Section 5.1 the natural bijection between interpretations of $\mathbf{W}$ over $[0, N)$ and interpretations of $\overline{\mathbf{W}}$ over $[0, 1)$. The decision procedure for McNaughton games given in the proof of Theorem 14 yields an $\mathsf{MSO}(<)$ formula $\theta(\overline{\mathbf{W}})$ that represents the image under the above-mentioned bijection of the set of predicates $\mathbf{P}$ for which Player I wins $\mathbb{G}(\varphi_{\mathcal{A}}, \mathbf{P})$. It remains to apply the inverse of the transformation described in Section 5.1, and obtain from $\theta(\overline{\mathbf{W}})$ a corresponding $\mathsf{MSO}(<,+1)$ formula $\psi(\mathbf{W})$.

The transformation from $\theta(\overline{\mathbf{W}})$ to $\psi(\mathbf{W})$ is straightforward. One replaces each atomic formula $W_i(t)$ in $\theta(\overline{\mathbf{W}})$ with $W(t+i)$ and one relativises all first-order quantifiers to the interval $[0, 1)$: thus $\forall x\, \varphi$ becomes $\forall x\, (0 \leq x < 1 \rightarrow \varphi)$ and $\exists x\, \varphi$ becomes $\exists x\, (0 \leq x < 1 \wedge \varphi)$. ∎

References

[1] R. Alur, M. Bernadsky, and P. Madhusudan. Optimal reachability for weighted timed games. In *Proc. ICALP'04*, volume 3142 of *LNCS*. Springer, 2004.

[2] R. Alur, C. Courcoubetis, and D. Dill. Model-checking for real-time systems. In *Proc. LICS'90*. IEEE Computer Society Press, 1990.

[3] R. Alur and D. Dill. A theory of timed automata. *Theoret. Comp. Sci.*, 126:183–235, 1994.

[4] R. Alur, L. Fix, and T. A. Henzinger. Event-clock automata: A determinizable class of timed automata. *Theoret. Comp. Sci.*, 211:253–273, 1999.

[5] C. Baier, H. Hermanns, J.-P. Katoen, and B. R. Haverkort. Efficient computation of time-bounded reachability probabilities in uniform continuous-time Markov decision processes. *Theor. Comput. Sci.*, 345(1), 2005.

[6] G. Behrmann, A. Cougnard, A. David, E. Fleury, K. G. Larsen, and D. Lime. Uppaal-tiga: Time for playing games! In *Proc. CAV'07*, volume 4590 of *LNCS*. Springer, 2007.

[7] P. Bouyer, F. Cassez, E. Fleury, and K. G. Larsen. Optimal strategies in priced timed game automata. In *Proc. FSTTCS'04*, volume 3328 of *LNCS*. Springer, 2004.

[8] P. Bouyer, N. Markey, J. Ouaknine, and J. Worrell. The cost of punctuality. In *LICS'07*. IEEE Computer Society Press, 2007.

[9] T. Brihaye, T. Henzinger, V. Prabhu, and J.-F. Raskin. Minimum-time reachability in timed games. In *Proceedings of ICALP'07*, volume 4596 of *LNCS*. Springer, 2007.

[10] J. R. Büchi. On a decision method in restricted second-order arithmetic. In *Proc. 1960 International Congress on Logic, Methodology and Philosophy of Science*, pages 1–11. Stanford University Press, 1962.

[11] A. K. Chandra, D. Kozen, and L. J. Stockmeyer. Alternation. *J. ACM*, 28(1):114–133, 1981.

[12] A. Church. Applications of recursive arithmetic to the problem of circuit synthesis. In *Summaries of the Summer Institute of Symbolic Logic*, volume 1, pages 3–50. Cornel Univ, Ithaca, 1957.

[13] M. Dickhöfer and T. Wilke. Timed alternating tree automata: The automata-theoretic solution to the TCTL model checking problem. In *Proc. ICALP'99*, volume 1644. Springer, 1999.

[14] E. A. Emerson and C. S. Jutla. Tree automata, mu-calculus and determinacy (extended abstract). In *Proc. FOCS'91*. IEEE, 1991.

[15] P. Hänsch, M. Slaats, and W. Thomas. Parametrized regular infinite games and higher-order pushdown strategies. In *Proc. FCT'09*, volume 5699 of *LNCS*. Springer, 2009.

[16] T. A. Henzinger, J.-F. Raskin, and P.-Y. Schobbens. The regular real-time languages. In *Proc. ICALP'98*, volume 1443 of *LNCS*. Springer, 1998.

[17] P. Herrmann. Timed automata and recognizability. *Information Processing Letters*, 65:313–318, 1998.

[18] G. Hoffmann and H. Wong-Toi. The input-output control of real-time discrete event systems. In *IEEE Real-Time Systems Symposium*, 1992.

[19] J.-P. Katoen and I. S. Zapreev. Safe on-the-fly steady-state detection for time-bounded reachability. In *Proc. QEST'06*. IEEE Computer Society Press, 2006.

[20] O. Kupferman and M. Y. Vardi. Weak alternating automata are not that weak. *ACM Trans. Comput. Log.*, 2(3):408–429, 2001.

[21] S. Lasota and I. Walukiewicz. Alternating timed automata. In *Proc. of FoSSaCS'05*, volume 3441 of *LNCS*. Springer LNCS, 2005.

[22] S. Lasota and I. Walukiewicz. Alternating timed automata. *ACM Trans. on Computational Logic*, 9(2):1–27, 2008.

[23] C. Löding and W. Thomas. Alternating automata and logics over infinite words. In *Proc. IFIP TCS'00*, volume 1872 of *LNCS*. Springer, 2000.

[24] O. Maler, A. Pnueli, and J. Sifakis. On the synthesis of discrete controllers for timed systems (an extended abstract). In *Proc. STACS'95*, volume 900 of *LNCS*. Springer, 1995.

[25] R. McNaughton. Testing and generating infinite sequences by a finite automaton. *Inf. Contr.*, 9:521–530, 1966.

[26] J. Ouaknine, A. Rabinovich, and J. Worrell. Time-bounded verification. In *Proceedings of CONCUR'09*, volume 5710 of *LNCS*. Springer, 2009.

[27] J. Ouaknine and J. Worrell. On the decidability of metric temporal logic. In *Proc. LICS'05*. IEEE Computer Society Press, 2005.

[28] J. Ouaknine and J. Worrell. Safety metric temporal logic is fully decidable. In *Proc. TACAS'06*, volume 3920 of *LNCS*. Springer, 2006.

[29] Joël Ouaknine and James Worrell. On the decidability and complexity of metric temporal logic over finite words. *Logicical Methods in Computer Science*, 3(1):1–27, March 2007.

[30] M. Rabin. Decidability of second order theories and automata on infinite trees. *Transaction of the AMS*, 141:1–35, 1969.

[31] A. Rabinovich. Finite variability interpretation of monadic logic of order. *Theor. Comput. Sci.*, 275(1-2):111–125, 2002.

[32] A. Rabinovich. Church synthesis problem with parameters. In *Proc. CSL'06*, volume 4207 of *LNCS*, pages 546–561. Springer, 2006.

[33] Alexander Rabinovich. The church synthesis problem with parameters. *Logical Methods in Computer Science*, 3(4), 2007.

[34] O. Roux and V. Rusu. Verifying time-bounded properties for ELECTRE reactive programs with stopwatch automata. In *Hybrid Systems*, volume 999 of *LNCS*. Springer, 1994.

[35] S. Shelah. The monadic theory of order. *Annals of Math.*, 102:349–419, 1975.

[36] L. J. Stockmeyer and A. R. Meyer. Word problems requiring exponential time. In *Proc. STOC'73*, pages 1–9, New York, NY, USA, 1973. ACM.

[37] W. Thomas. Church's problem and a tour through automata theory. In *Pillars of Computer Science, Essays Dedicated to Boris (Boaz) Trakhtenbrot on the Occasion of His 85th Birthday*, volume 4800 of *LNCS*, pages 635–655. Springer, 2008.

[38] M. Y. Vardi. Alternating automata and program verification. In *Computer Science Today*, volume 1000 of *LNCS*. Springer, 1995.

[39] T. Wilke. Specifying timed state sequences in powerful decidable logics and timed automata. In *Proc. FTRTFT'94*, volume 863 of *LNCS*. Springer, 1994.