History-deterministic Timed Automata are Not Determinizable

Sougata Bose[1], Thomas A. Henzinger[2], Karoliina Lehtinen[3], Sven Schewe[1], and Patrick Totzke[1]

[1] University of Liverpool, UK
[2] IST Austria
[3] CNRS, Aix-Marseille University, University of Toulon, LIS

Abstract. An automaton is history-deterministic (HD) if one can safely resolve its non-deterministic choices on the fly. In a recent paper, Henzinger, Lehtinen and Totzke studied this in the context of Timed Automata [9], where it was conjectured that the class of timed ω-languages recognised by HD-timed automata strictly extends that of deterministic ones. We provide a proof for this fact.

Keywords: Timed Automata · History-determinism.

1 Introduction

History-determinism asks for the existence of a strategy to produce a run on an input word that is given one letter at a time, so that the resulting run is accepting whenever the given word is in the language.

Similar to automata with bounded ambiguity, history-deterministic ones provide a middle ground between determinism and non-determinism. They are typically more succinct, or even more expressive, than their deterministic counterparts while preserving some of their good algorithmic properties. For example, when verifying finite or ω-automata against history-deterministic specifications (i.e. checking inclusion with languages given by a HD automaton), the costly step of complementing the specification automaton can be avoided, as checking language inclusion can be replaced by checking fair simulation [9], which is polynomial for finite, Büchi and co-Büchi automata [8]. For some coBüchi-recognisable languages, history-deterministic automata can be exponentially more succinct than any equivalent deterministic automaton [12], and checking if a Büchi or coBüchi automaton is history-deterministic is decidable in polynomial time [2, 12].

History-determinism has mostly been studied in the ω-regular setting, i.e., for finite-state automata recognising languages of infinite words or trees, where the concept of history-determinism has also been called "good-for-games" [10, 13, 6, 3]. Recently, the notion has been extended to richer automata models, such as pushdown automata [14, 7] and quantitative automata [4, 5], where deterministic and nondeterministic models have different expressivity.

In [9], history-determinism was first studied in the context of *timed* automata with ω-regular acceptance conditions [1]. It is shown that for history-deterministic TA with arbitrary parity acceptance, timed universality, inclusion, and synthesis are ExpTime-complete, contrary to their undecidability for non-deterministic Büchi TA [1]. History-deterministic TA with safety acceptance are effectively determinisable; checking if a given timed automaton is history-deterministic is decidable in ExpTime for safety or reachability acceptance, and open for more general acceptance conditions such as Büchi, coBüchi and Parity.

In terms of expressivity, it was conjectured that history-deterministic timed automata recognise a class of ω-languages strictly between those defined by deterministic and non-deterministic TAs. The following language is proposed as a candidate to separate deterministic and HD timed languages.

Let L be the language of all timed words along which *eventually* events appear at unit distances: from some time t onwards, for every nonnegative integer i, there is an event at time $t + i$.

It is not difficult to see that this language is recognised by a HD coBüchi automaton. One can commit to the fractional time at which the longest chain of events has been observed so far, and can afford to be wrong a finite number of times. It is intuitively clear that L is not deterministic, considering that any DTA has only finitely many clocks and thus "cannot remember unboundedly many past timestamps" for comparisons. It is however notoriously technical to provide formal arguments for showing that timed languages are not deterministic. Herrmann [11] suggests some high-level lemmas based on reset points, but these only apply to the Büchi setting.

The main contribution of this paper is a formal argument that the language L is indeed not recognised by any deterministic timed automaton, even with general Parity acceptance. We present a scheme to recursively produce, for a given DTA, a suitable pair of words so that their runs are region-equivalent (and so either both are accepting or both rejecting) but where only one of them is in L. The main idea is to produce events and observe the resulting run until it closes a loop in the region graph, then force that same loop again twice more. Any resets that occurred in the intermediate loop are lost and overwritten in the final iteration, which allows to move the timing of the intermediate loop arbitrarily.

We also provide an example that separates history-deterministic from non-deterministic timed automata, concluding that indeed, this class of timed languages sits strictly in between deterministic and non-deterministic ones.

2 Notations

Let $\mathbb{N}$ and $\mathbb{R}_{\geq 0}$ denote the nonnegative integers and reals, respectively. For $c \in \mathbb{R}_{\geq 0}$ we write $\lfloor c \rfloor$ for its integer and $\mathit{fract}(c) \stackrel{\text{def}}{=} c - \lfloor c \rfloor$ for its fractional part.

Timed Alphabets and words. A timed word is a finite or infinite word $w = (a_0, t_0)(a_1, t_1), \ldots$ over the alphabet $(\Sigma \times \mathbb{R}_{\geq 0})$ where the first components

is a letter from some finite alphabet Σ and the second components are non-decreasing and progressing, that is, for all $n \in \mathbb{N}$, there is some i and a such that $w[i] = (a, t)$ and $t > n$. We sometimes call the (a_i, t_i) an a_i-*event* with timestamp t_i. For convenience, we will confuse timed words as above with words over $(\Sigma \cup \mathbb{R}_{\geq 0})$, interpreting each letter either as discrete event or a delay. The *duration* of a (finite or infinite) timed word is the combined sum of all its delays. More precisely, a timed word w as above gives rise to the word $d_0 a_0 d_1 a_1 \ldots$ over $(\Sigma \cup \mathbb{R}_{\geq 0})$, where $t_0 = d_0$ and $t_{i+1} = t_i + d_{i+1}$. Conversely, the duration and the timed word of a word over $(\Sigma \cup \mathbb{R}_{\geq 0})$ is given inductively as follows. For any $d \in \mathbb{R}_{\geq 0}$, $\tau \in \Sigma$, $\alpha \in (\Sigma \cup \mathbb{R})^*$, and $\beta \in (\Sigma \cup \mathbb{R})^\infty$ let $\mathit{duration}(\tau) \stackrel{\text{def}}{=} 0$; $\mathit{duration}(d) \stackrel{\text{def}}{=} d$; $\mathit{duration}(\alpha\beta) = \mathit{duration}(\alpha) + \mathit{duration}(\beta)$; $\mathit{tword}(\varepsilon) = \mathit{tword}(d) \stackrel{\text{def}}{=} \varepsilon$; $\mathit{tword}(\alpha d) \stackrel{\text{def}}{=} \mathit{tword}(\alpha)$; and $\mathit{tword}(\alpha\tau) \stackrel{\text{def}}{=} \mathit{tword}(\alpha)(\tau, \mathit{duration}(\alpha))$.

Timed Automata are finite-state automata equipped with finitely many real-valued variables called *clocks*, whose transitions are guarded by constraints on clocks. Constraints on clocks $C = \{x, y, \ldots\}$ are (in)equalities $x \triangleleft n$ where $x \in C$, $n \in \mathbb{N}$ and $\triangleleft \in \{\leq, <\}$. Let $\mathcal{B}(C)$ denote the set of Boolean combinations of clock constraints, called *guards*. A clock *valuation* $\nu \in \mathbb{R}^C_{\geq 0}$ assigns a value $\nu(x)$ to each clock $x \in C$. We write $\nu \models g$ if ν satisfies the guard g. A timed automaton (TA) $\mathcal{T} = (Q, \iota, C, \Delta, \Sigma, \mathit{Acc})$ is given by

- Q a finite set of states including an initial state ι;
- Σ an input alphabet;
- C a finite set of clocks;
- $\Delta \subseteq Q \times \mathcal{B}(C) \times \Sigma \times 2^C \times Q$ a set of transitions; each transition is associated with a guard, a letter, and a set of clocks to reset.
- $\mathit{Acc} \subseteq \Delta^\omega$ an acceptance condition.

We assume that for all $(s, \nu, a) \in Q \times \mathbb{R}^C_{\geq 0} \times \Sigma$ there is at least one transition $(s, g, a, r, s') \in \Delta$ so that ν satisfies g. $\mathcal{T}$ is *deterministic* if there is at most one such enabled transition. I.e., for every state s, all transitions from s have mutually exclusive guards.

A *configuration* is a pair consisting of a control state and a clock valuation. For every configuration $(s, \nu) \in Q \times \mathbb{R}^C_{\geq 0}$,

1. there is a *delay* step $(s, \nu) \xrightarrow{d} (s, \nu + d)$ for every $d \geq 0$, which increments all clocks by d.
2. there is a *discrete* step $(s, \nu) \xrightarrow{\tau} (s', \nu')$ if $\tau = (s, g, a, r, s') \in \Delta$ is a transition so that ν satisfies g and $\nu' = \nu[r \to 0]$, that is, it maps r to 0 and agrees with ν on all other values.

A path $\rho = (s_0, \nu_0) \xrightarrow{l_1} (s_1, \nu_1) \xrightarrow{l_2} (s_2, \nu_2) \ldots$ is called a *run on* timed word $w \in (\Sigma_T)^\infty$ if $\mathit{tword}(l_1 l_2 \ldots) = w$. It is *accepting* if $\rho \in \mathit{Acc}$. The *language* $L(s, \nu) \subseteq \Sigma_T^\omega$ of a configuration (s, ν) consists of all timed words for which there exists an accepting run from (s, ν). The language of $\mathcal{T}$ is $L(\mathcal{T}) \stackrel{\text{def}}{=} L((\iota, 0))$, the languages if the initial configuration with state ι and all clocks set to zero. We

assume that Acc is determined by a parity condition: $Q \to D$ maps states to a integer priority domain $D = [i..p]$ with minimal priority $i \in \{0, 1\}$ and maximal priority $p \in \mathbb{N}$. A run is accepting if the highest priority seen infinitely often along the run is even. Büchi acceptance corresponds to $D = \{1, 2\}$ and co-Büchi acceptance corresponds to $D = \{0, 1\}$.

Regions. The following is the standard definition of regions (cf. [1], def. 4.3). Let $\mathcal{T} = (Q, \iota, C, \Delta, \Sigma, Acc)$ be a timed automaton and for any clock $x \in C$ let c_x denote the largest constant in any clock constraint involving x. Two valuations $\nu, \nu' \in \mathbb{R}^C_{\geq 0}$ are *(region) equivalent* (write $\nu \sim \nu'$) if all of the following hold.

1. For all $x \in C$ either $\lfloor \nu(x) \rfloor = \lfloor \nu'(x) \rfloor$ or both $\nu(x)$ and $\nu'(x)$ are greater than c_x.
2. For all $x, y \in C$ with $\nu(x) \leq c_x$ and $\nu(y) \leq c_y$, $fract(\nu(x)) \leq fract(\nu(y))$ iff $fract(\nu'(x)) \leq fract(\nu'(y))$.
3. For all $x \in C$ with $\nu(x) \leq c_x$, $fract(\nu(x)) = 0$ iff $fract(\nu'(x)) = 0$.

Two configurations (q, ν) and (q', ν') are (region) equivalent, write $(q, \nu) \sim (q', \nu')$, if $q = q'$ and $\nu \sim \nu'$. Let $maxfrac(\nu) = \max\{fract(\nu(x)) \mid x \in C\}$ denote the maximal fractional value of any clock in configuration ν. We will make use of the following two properties.

Proposition 1.

1. *For any valuation ν and $d \leq 1 - maxfrac(\nu)$ we have $\nu \sim \nu + d$.*
2. *Suppose that $\nu \sim \nu'$ and for all clocks $x \in C$, both $\nu(x) > c_x$ $\nu'(x) > c_x$. Then for any run $\rho \in (\Delta \cup \mathbb{R}_{\geq 0})^*$, $(p, \nu) \xrightarrow{\rho} (q, \mu)$ iff $(p, \nu') \xrightarrow{\rho} (q, \mu)$*
3. *Suppose that $(p, \nu) \sim (p', \nu')$ and let $\rho \in (\Delta \cup \mathbb{R}_{\geq 0})^*$ satisfy $duration(\rho) < 1 - maxfrac(\nu)$, $duration(\rho) < 1 - maxfrac(\nu')$ and $(p, \nu) \xrightarrow{\rho} (q, \mu)$. Then $(p', \nu') \xrightarrow{\rho} (q', \mu')$ for some $(q', \mu') \sim (q, \mu)$.*

Proof (sketch). Part 1 is immediate from the definition of regions.

Parts 2 and 3 can be shown by induction on the length of ρ using the facts that region-equivalent configurations enable the same discrete transitions. For Part 3 one additionally uses the fact that any delay decreases the duration of the remaining path by the same amount it increases clocks. □

History-deterministic TA. A timed automaton is history-deterministic if one can resolve non-deterministic choices on-the-fly.

More formally, consider a function $r : (\Delta \cup \mathbb{R}_{\geq 0})^* \times \Sigma \to \Delta$ that, given a finite run $\rho_i = (s_0, \nu_0) \xrightarrow{a_0} (s_1, \nu_1) \xrightarrow{a_1} (s_2, \nu_2) \dots (s_i, \nu_i)$ and a next letter $a_i \in \Sigma$, returns a transition $r(\rho_i, a_i) = (s_i, g_i, a_1, s_{i+1}) \in \Delta$ such that $\nu_i \models g$. This yields, for every word $w = a_0 a_1 \dots \in (\Sigma \cup \mathbb{R}_{\geq 0})^\omega$ and initial configuration (s_0, ν_0), a unique run in which the ith step $(s_i, \nu_i) \xrightarrow{a_i} (s_{i+1}, \nu_{i+1})$ either results from a delay (if $a_i \in \mathbb{R}_{\geq 0}$ and $(s_{i+1}, \nu_{i+1}) = (s_i, \nu_i + a_i)$) or a discrete step chosen by r (if $a_i \in \Sigma$ and $r(\rho_i, a_i) = (s_i, g_i, a_i, s_{i+1})$).

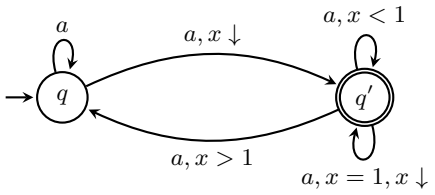

Fig. 1. A history-deterministic timed co-Büchi automaton for L. The state q' has priority 0, i.e. is accepting, while the state q has priority 1.

Such a function is called *resolver* if for any input word $w \in L(\mathcal{T})$ the constructed run ρ from initial configuration $(s_0, \nu_0) = (\iota, 0)$ is accepting. A timed automaton is *history-deterministic* if such a resolver exists.

3 The Construction

The interesting aspect of our claim is to show that HD timed automata are strictly more expressive than deterministic ones. We show that the following language L over the singleton alphabet $\Sigma = \{a\}$ is recognised by a one-clock history-deterministic coBüchi automaton yet not by any deterministic Parity timed automaton. In words, L asks to eventually see events a at unit distance. Formally,

$$L \stackrel{\text{def}}{=} \{(a,\tau_0)(a,\tau_1)... \mid \exists i \in \mathbb{N}. \quad \forall n \in \mathbb{N} \quad \exists j > i. \quad \tau_j - \tau_i = n\}.$$

***L* is HD recognisable.**

We show that L is recognised by the history-deterministic timed ω-automaton, in Fig. 1. This automaton has an initial rejecting state q, from where there is a nondeterministic choice to either remain in this state or transition to an accepting state q', which resets the unique clock. There are two transitions to stay in the accepting state: one enabled when the clock value is smaller than 1, and one enabled at clock value 1, which also resets the clock. If the clock value grows larger than 1, the only enabled transition goes back to the initial state. Since this is a co-Büchi automaton, an accepting run must eventually remain in the accepting state.

First, this automaton recognises L: if $w \in L$ then there is an accepting run that moves to state q' at time t, where it then remains since the clock x is reset at the occurrence of each event $(a, t+n)$ for $n \in \mathbb{N}$, so the clock value never grows larger than 1. Conversely, a word accepted by this automaton has a run that eventually moves to q' at a time t, and then remains in q'. For the run to stay in q', it must reset x at every time-unit after t, so $(a, t+n)$ must occur in the word for all $n \in \mathbb{N}$, that is, the word is in L.

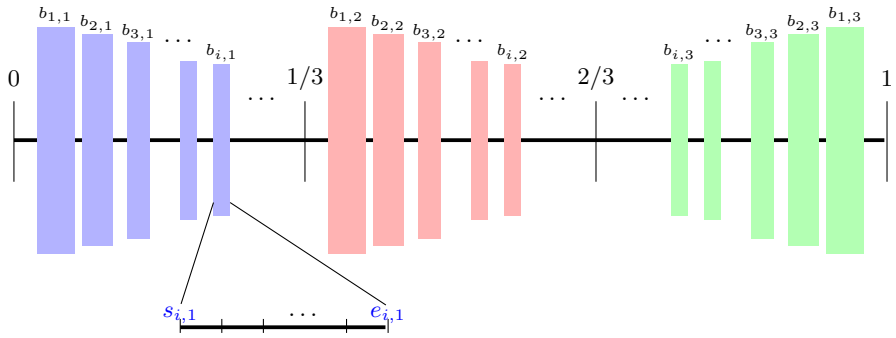

Fig. 2. Blocks within an interval and ticks within a block

We now argue that this automaton is also history-deterministic. Given a finite word read so far and a new letter a at time t_{new}, the resolver identifies the earliest time t_{early} such that a has so far occurred at time $t_{early} + n$ for all integers n such that $t_{early} + n \leq t_{new}$. Let r be the function that maps a run ρ ending in q to q' if $t_{new} = t_{early} + m$ for some integer m, and otherwise to the only other available transition.

We claim that this is indeed a resolver. If $w \in L$ then there is an earliest time t such that $(a, t+n)$ occurs in w for all integers n. Since t is minimal, eventually the resolver r will make its choice whether to move to q' over a letter (a, t_{new}) based on whether $t_{new} = t + m$ for some integer m. Since time progresses and $(a, t+n)$ occurs in w for all integers n, the run will eventually transition to q' at a time $t + m$ for some m. From there, since $(a, t+n)$ occurs in w for all integers n, the run over w remains in q' and is therefore accepting.

It remains to be shown that L is not recognised by a deterministic timed automaton.

L is not Deterministic Parity recognisable

Suppose towards a contradiction that L is recognisable by some deterministic Timed Automaton D with Parity acceptance. Let r be the number of its regions.

We will construct two words, one belonging to L and not that does not, so that the run of D on w is region equivalent to the one on w'. The two words can only differ in the timing of events since there is only one letter in the alphabet.

Both words will be constructed on the fly, according to the following schema.

Consider the intervals and fractional values in Fig. 2; There are infinitely many disjoint intervals, $b_{i,j} = [s_{i,j}, e_{i,j}]$ so that all $b_{i,1}$ have start and endpoint strictly between 0 and $\frac{1}{3}$ and are increasing, i.e., $s_{i+1,1} < e_{i,1}$ for all i. Similarly, $b_{i,2} \subseteq [\frac{1}{3}, \frac{2}{3}]$, and $s_{i+1,2} < e_{i,2}$ for all i. The third sequence of intervals $b_{i,3} \subseteq [\frac{2}{3}, 1]$ have start and endpoint strictly between $\frac{1}{3}$ and 1 and are *decreasing*: $e_{i+1,3} < s_{i,3}$ for all i. Each interval $b_{i,j}$ contains equi-distant values $f_{i,j,0}, f_{i,j,1}, \ldots, f_{i,j,r}$ starting at $f_{i,j,0} = s_{i,0}$.

We step-wise construct w (and w') together with the run of D on it. In every integral interval from i to $i+1$, we place events as follows.

- start with a delay of $f_{i,1,1}$, followed by a discrete event a, then delay of $f_{i,1,2} - f_{i,1,1}$ followed by a, and so on. This induces a run of D on the prefix constructed and we continue constructing the prefix until the induced run closes a cycle in the region graph. This implies existence of times $f_{i,1,k}$ and $f_{i,1,k+\ell}$ such that the automaton is in configurations $(s_{i,1,k}, \nu_{i,1,k})$ and $(s_{i,1,k+\ell}, \nu_{i,1,k+\ell})$ and $(s_{i,1,k+\ell}, \nu_{i,1,k+\ell}) \sim (s_{i,1,k}, \nu_{i,1,k})$. We denote by L_i the run between $f_{i,1,k}$ and $f_{i,1,k+\ell}$.
- Now we force the automaton to close the same cycle, but with all events occurring at times in the interval $b_{i,2}$ (respectively $b_{1,2}$) in w (respectively w'). This can be done by adding a time delay by $s_{i,2} - f_{i,1,k+\ell}$ in w followed by an events a at times $f_{i,2,\ell'}$ for all $\ell' \leq \ell$. We prove this formally in Lemma 1.
- Finally we force the automaton to close the same cycle once more, with all times in interval $b_{i,3}$. This can be done by adding a time delay $s_{i,3} - f_{i,2,\ell}$ followed by events at times $f_{i,3,1}, f_{i,3,2}, \ldots f_{i,3,\ell}$. We prove the correctness of the construction in Lemma 1.

Consider the cycle L_i in the region graph obtained in step 1 above in the interval $[i-1, i]$, between $f_{i,1,k}$ and $f_{i,1,k+l}$. Note that the k and ℓ depends on i. However, we write k and ℓ without as we only reason about loops within an integral interval. The duration of the loop, denoted by $\mathit{duration}(L_i)$ is $f_{i,1,k+\ell} - f_{i,1,k}$. An important observation is that $\mathit{duration}(L_i) \leq e_{i,j} - s_{i,j}$ as the loop occurs within the interval between $s_{i,1}$ and $e_{i,1}$.

Lemma 1. *Let ν_i and ν'_i be the configurations reached by the run of D at times $i - 1 + f_{i,1,k}$ and $i - 1 + f_{i,1,k+\ell}$. Then $1 - \mathit{maxfrac}(\nu_i + d_{ij}) \geq \mathit{duration}(L_i)$, where $d_{ij} = s_{i,j} - f_{i,1,k}$ for $j \in \{2, 3\}$.*

Furthermore, let ν_{ij} be the configuration reached by the run of D at time $i - 1 + f_{i,j,1}$, where $j = \{2, 3\}$. The cycle L_i is executable from ν_{ij}.

Proof. We prove this lemma by induction on i. The case $i = 1$ is easy to see since $\mathit{maxfrac}(\nu_1 + d_{1j}) \leq s_{1,j}$ and therefore $1 - \mathit{maxfrac}(\nu_1 + d_{1j}) \geq 1 - s_{1,j} \geq e_{1,j} - s_{1,j} \geq \mathit{duration}(L_1)$.

Furthermore, $\nu_{12} = \nu'_1 + d$, where $d = s_{1,2} - f_{1,1,k+\ell} \leq 1 - f_{1,1,k+\ell} \leq 1 - \mathit{maxfrac}(\nu'_1)$. Therefore, by Proposition 1.1, $\nu_{12} \sim \nu'_1 \sim \nu_1$. For $\nu = \nu_1$ and $\nu = \nu_{12}$, $1 - \mathit{maxfrac}(\nu) > e_{1,3} - s_{1,3} > \mathit{duration}(L_1)$ as $1 > e_{1,3}$ and $\mathit{maxfrac}(\nu) < s_{1,3}$. By applying Proposition 1.3, L_1 is executable from ν_{12} and ends in a configuration $\nu'_{12} \sim \nu_{12}$.

The configuration ν_{13} equals $\nu'_{12} + d'$, where $d' = s_{1,3} - f_{1,2,\ell} < 1 - \mathit{maxfrac}(\nu'_{12})$ as $\mathit{maxfrac}(\nu'_{12}) \leq f_{1,2,\ell}$. Proposition 1.1 gives $\nu_{13} \sim \nu_{12}$, and $1 - \mathit{maxfrac}(\nu_{13}) \geq e_{1,3} - s_{1,3} \geq \mathit{duration}(L_1)$. By Proposition1.3, we can conclude that L_1 is executable from ν_{13}.

To prove the inductive case, we bound the value of $\mathit{maxfrac}(\nu_i + d_{ij})$ for $j \in \{2, 3\}$. Consider a clock $x \in C$ and the last time when it was reset. Either it was

never reset or the reset occurred at time $f_{i',j',k'}$. For a clock that is never reset, the fractional part of its value at ν_i will be $f_{i,1,k}$. If the clock was last reset within some blue block, i.e, at time $i' - 1 + f_{i',1,k'}$, then either $i' < i$ (corresponds to previous blue blocks), or $k' < k$ (corresponds to previous ticks within the current blue block). In both cases, the $\mathit{fract}(x) = \mathit{fract}(f_{i,1,k} - (i' - 1 + f_{i',1,k'})) \leq f_{i,1,k}$.

Note that any reset to clock x in a previous red block must also be reset again in the corresponding green block as the runs in the red and green block are the same by construction. For a clock x last reset in some previous green block, i.e, at time $i' - 1 + f_{i',3,k'}$, $\mathit{fract}(x) = \mathit{fract}((i - 1 + f_{i,1,k}) - (i' - 1 + f_{i',3,k'})) = f_{i,1,k} + (1 - f_{i',3,k'})$. Furthermore, $f_{i',3,k'} > s_{i-1,3}$ as $i' \leq i$. Therefore, $\mathit{fract}(x) \leq 1 + f_{i,1,k} - s_{i-1,3} + 1$ which bounds $1 - \mathit{fract}(x) \geq s_{i-1,3} - f_{i,1,k}$. Combining all the posibilities for clock resets, we obtain $1 - \mathit{maxfrac}(\nu_i) \geq s_{i-1,3} - f_{i,1,k}$.

It is easy to see that for $j \in \{2, 3\}$, $\mathit{maxfrac}(\nu_i + d_{ij}) \leq \mathit{maxfrac}(\nu_i) + d_{ij}$. Therefore, $1 - \mathit{maxfrac}(\nu_i + d_{ij}) \geq 1 - \mathit{maxfrac}(\nu_i) - d_{ij} \geq 1 - (f_{i,1,k} - s_{i-1,3} + 1) - (s_{i,j} - f_{i,1,k}) \geq s_{i-1,3} - s_{i,j} \geq e_{i,j} - s_{i,2}$. The last step follows from the fact that $e_{i,j} \leq s_{i-1,3}$ for $j \in \{2, 3\}$. Note that the duration of the loop L_i is less than $e_{i,j} - s_{i,j}$ and thus completes the proof for fist part of the lemma.

We now show that L_i is executable from ν_{i2} and ν_{i3}. First, $\nu_{i2} \sim \nu_i$ and $\nu_{i3} \sim \nu_{i2}$ by repeated application of Proposition 1.1. This is similar to the argument in the base case. We just showed that $1 - \mathit{maxfrac}(\nu_{i2}) > s_{i-1,3} - s_{i,2} > e_{i,2} - s_{i,2} = \mathit{duration}(L_i)$. The same argument holds for ν_{i3} as well. Also, $\mathit{maxfrac}(\nu_i) \leq 1 - s_{i-1,3} + f_{i,1,k}$ and hence $1 - \max \nu_i \geq s_{i-1,3} - f_{i,1,k} < e_{i,3} - s_{i,3} < \mathit{duration}(L_i)$. Therefore, by Proposition 1.3, L_i is executable from both ν_{i2} and ν_{i3}.

Notice that the so-constructed word w is not in L because all $b_{i,j}$ are disjoint. The word w' will be constructed almost the same way, with the only exception that the first repetition of the cycle is move not to $b_{i,2}$ but always the same interval, $b_{1,2}$. Its easy to see that Lemma 1 can be modified where $b_{i,2}$ is replaced everywhere by $b_{1,2}$. In particular this means that w contains an event at time $n + s_{1,2}$ for any $n \in \mathbb{N}$, and thus must be contained in L. Therefore, D has an accepting run on w' but the run on w' is visits the same sequence of states as the run of D on w. Therefore, D must accept w as well, which is a contradiction proving that L is not accepted by any deterministic Timed Automaton with Parity acceptance.

Example which is ND recognisable but not HD recognisable

We now show that non-deterministic TA are more expressive than history-deterministic TA. In particular, we show that the following language L' over the singleton alphabet $\Sigma = \{a, b\}$ is recognised by a one-clock non-deterministic TA with reachability acceptance but not by any history-deterministic Parity TA. In words, L' asks to see two events a at unit distance. Formally,

$$L' \stackrel{\text{def}}{=} \{(\sigma_0, \tau_0)(\sigma_1, \tau_1)... \mid \exists i, j \in \mathbb{N}. \quad \tau_j - \tau_i = 1 \text{ and } \sigma_i = a \text{ and } \sigma_j = a\}.$$

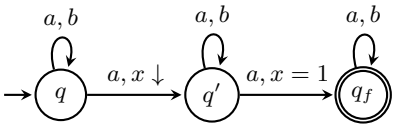

Fig. 3. A non-deterministic timed reachability automaton for L'.

It is easy to see that the non-deterministic TA shown in Figure 3 accepts the language L' by guessing positions i by reading an a, resetting a clock x and checking that it sees an a at distance 1.

The interesting claim is to show that no HD TA accepts the language. To show this, we assume that there exists a HD TA H with k clocks and maximum constant in guards c_x, accepting the language L'.

Consider the word $w = (a, \frac{1}{3})(a, 2\frac{1}{3}) \dots (a, (k+1)\frac{1}{3})(a, 1 + i\frac{1}{3})(b, \tau_1)(b, \tau_2) \dots$ such that $\tau_i > 2 + c_x$. Since this word is in L', the resolver gives a run on this word. Consider the configuration ν reached by this run at time 1 (see Figure 4). Since H has k clocks and $k+1$ events a, there exists an i such that $\nu(x) \neq 1 - i\frac{1}{3}$. As there are more than k a, there exists an i such that either no clock is reset while reading the ith a, or, any clock reset while reading the ith a is also reset later. It follows from the definition of regions that $\nu + i\frac{1}{3} \sim \nu + i\frac{1}{3} + (\frac{1}{3}/2)$.

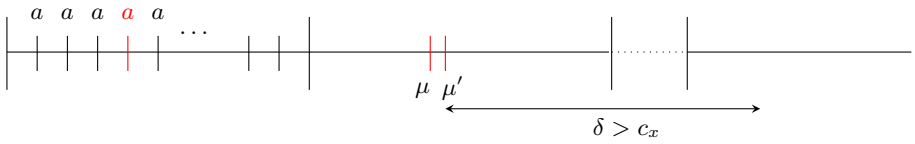

Fig. 4. Construction of $w \in L'$ and $w' \notin L'$

Consider the word $w' = (a, \frac{1}{3})(a, 2\frac{1}{3}) \dots (a, (k+1)\frac{1}{3})(a, 1 + i\frac{1}{3} + (\frac{1}{3}/2))(b, \tau_1)$ $(b, \tau_2) \dots$. Clearly w' is not in L'. However, H has a run on w' which follows the accepting run of H on w until time 1. Since $\nu + i\frac{1}{3} \sim \nu + i\frac{1}{3} + (\frac{1}{3}/2)$, the same transition can be take on reading a at time $1 + i\frac{1}{3}$ and $1 + i\frac{1}{3} + (\frac{1}{3}/2)$ leading to configurations μ and μ' on w and w' respectively. By Proposition 1.2, after a delay of $\delta > c_x$, the same run can be followed from both configurations $\mu + \delta$ and $\mu' + \delta$. Therefore H has an accepting run on w' which is a contradiction. This concludes the proof that L' is not accepted by any parity HD timed automata.

We thus conclude that the classes of languages accepted by deterministic, history-deterministic and non-deterministic TAs are all different.

References

1. Alur, R., Dill, D.L.: A theory of timed automata. Theoretical Computer Science **126**(2), 183 – 235 (1994). https://doi.org/10.1016/0304-3975(94)90010-8

2. Bagnol, M., Kuperberg, D.: Büchi Good-for-Games Automata Are Efficiently Recognizable. In: IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS). Leibniz International Proceedings in Informatics (LIPIcs), vol. 122, pp. 16:1–16:14. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik (2018). https://doi.org/10.4230/LIPIcs.FSTTCS.2018.16
3. Boker, U., Lehtinen, K.: Good for games automata: From nondeterminism to alternation. In: International Conference on Concurrency Theory (CONCUR). LIPIcs, vol. 140, pp. 19:1–19:16 (2019)
4. Boker, U., Lehtinen, K.: History Determinism vs. Good for Gameness in Quantitative Automata. In: IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS). Leibniz International Proceedings in Informatics (LIPIcs), vol. 213, pp. 38:1–38:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik (2021). https://doi.org/10.4230/LIPIcs.FSTTCS.2021.38
5. Boker, U., Lehtinen, K.: Token games and history-deterministic quantitative automata. In: IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS). pp. 120–139. Springer International Publishing (2022)
6. Colcombet, T.: The theory of stabilisation monoids and regular cost functions. In: International Colloquium on Automata, Languages and Programming (ICALP). pp. 139–150 (2009)
7. Guha, S., Jecker, I., Lehtinen, K., Zimmermann, M.: A Bit of Nondeterminism Makes Pushdown Automata Expressive and Succinct. In: International Symposium on Mathematical Foundations of Computer Science (MFCS). Leibniz International Proceedings in Informatics (LIPIcs), vol. 202, pp. 53:1–53:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik (2021). https://doi.org/10.4230/LIPIcs.MFCS.2021.53
8. Henzinger, T.A., Kupferman, O., Rajamani, S.K.: Fair simulation. Information and Computation **173**(1), 64–81 (2002)
9. Henzinger, T.A., Lehtinen, K., Totzke, P.: History-deterministic timed automata. In: International Conference on Concurrency Theory (CONCUR) (2022)
10. Henzinger, T.A., Piterman, N.: Solving games without determinization. In: Computer Science Logic (CSL). pp. 395–410. Springer Berlin Heidelberg (2006)
11. Herrmann, P.: Timed automata and recognizability. Information Processing Letters **65**(6), 313–318 (1998). https://doi.org/10.1016/S0020-0190(97)00217-2
12. Kuperberg, D., Skrzypczak, M.: On determinisation of good-for-games automata. In: International Colloquium on Automata, Languages and Programming (ICALP). pp. 299–310 (2015)
13. Kupferman, O., Safra, S., Vardi, M.Y.: Relating word and tree automata. Ann. Pure Appl. Logic **138**(1-3), 126–146 (2006)
14. Lehtinen, K., Zimmermann, M.: Good-for-games ω-pushdown automata. Logical Methods in Computer Science **18** (2022)