Expressiveness of Updatable Timed Automata

P. Bouyer, C. Dufourd, E. Fleury, A. Petit

LSV, UMR 8643, CNRS & ENS de Cachan,
61 Av. du Président Wilson, 94235 Cachan cedex, France
{bouyer, dufourd, fleury, petit}@lsv.ens-cachan.fr

Abstract. Since their introduction by Alur and Dill, timed automata have been one of the most widely studied models for real-time systems. The syntactic extension of so-called updatable timed automata allows more powerful updates of clocks than the reset operation proposed in the original model.

We prove that any language accepted by an updatable timed automaton (from classes where emptiness is decidable) is also accepted by a "classical" timed automaton. We propose even more precise results on bisimilarity between updatable and classical timed automata.

1 Introduction

Since their introduction by Alur and Dill [2, 3], timed automata have been one of the most studied models for real-time systems (see [4, 1, 16, 8, 12, 17, 13]). In particular numerous works proposed extensions of timed automata [7, 10, 11].

This paper focuses on one of this extension, the so-called updatable timed automata, introduced in order to model the ATM protocol ABR [9]. Updatable timed automata are constructed with updates of the following forms:

$$x :\sim c \mid x :\sim y + c \text{ where } \mathrm{x, y} \text{ are clocks}, \mathrm{c} \in \mathbb{Q}_+ \text{ and } \sim \in \{<, \leq, =, \neq, \geq, >\}$$

In [5], the (un)decidability of emptiness of updatable timed automata has been characterized in a precise way (see Section 2 for detailed results). We address here the open question of the expressive power of updatable timed automata (from decidable classes). We solve completely this problem by proving that any language accepted by an updatable timed automaton is also accepted by a "classical" timed automaton with ε-transitions. In fact, we propose even more precise results by showing that any updatable timed automaton using only deterministic updates is strongly bisimilar to a classical timed automaton and that any updatable timed automaton using arbitrary updates is weakly bisimilar (but not strongly bisimilar) to a classical timed automaton.

The paper is organized as follows. In Section 2, we present updatable timed automata, generalizing classical definitions of Alur and Dill. Several natural equivalences of updatable timed automata are introduced in Section 3. The bisimulation algorithms are presented in Section 4.

For lack of space, this paper contains only some sketchs of proofs. They are available on the technical report [6].

2 Updatable Timed Automata

Timed Words and Clocks

If Z is any set, let Z^* (respectively Z^ω) be the set of *finite* (resp. *infinite*) sequences of elements in Z and let $Z^\infty = Z^* \cup Z^\omega$. We consider as time domain $\mathbb{T}$ the set of non-negative rational $\mathbb{Q}_+$ and Σ as finite set of *actions*. A *time sequence* over $\mathbb{T}$ is a finite or infinite non decreasing sequence $\tau = (t_i)_{i\geq 1} \in \mathbb{T}^\infty$. A *timed word* $\omega = (a_i, t_i)_{i\geq 1}$ is an element of $(\Sigma \times \mathbb{T})^\infty$.
We consider an at most countable set $\mathbb{X}$ of variables, called *clocks*. A clock valuation over $\mathbb{X}$ is a mapping $v : \mathbb{X} \to \mathbb{T}$ that assigns to each clock a time value. Let $t \in \mathbb{T}$, the valuation $v + t$ is defined by $(v + t)(x) = v(x) + t$, $\forall x \in \mathbb{X}$.

Clock Constraints

Given a subset of clocks $X \subseteq \mathbb{X}$, we introduce two sets of clock constraints over X. The most general one, denoted by $\mathcal{C}(X)$, is defined by the following grammar:

$$\varphi ::= x \sim c \mid x - y \sim c \mid \varphi \wedge \varphi \mid \neg\, \varphi \mid true,\ with\ x, y \in X, c \in \mathbb{Q}_+, \sim \in \{<, \leq, =, \neq, \geq, >\}$$

The proper subset $\mathcal{C}_{df}(X)$ of "diagonal-free" constraints in which the comparison between two clocks is not allowed, is defined by the grammar:

$$\varphi ::= x \sim c \mid \varphi \wedge \varphi \mid \neg\, \varphi \mid true,\ \text{with}\ x \in X,\ c \in \mathbb{Q}_+\ \text{and}\ \sim \in \{<, \leq, =, \neq, \geq, >\}$$

We write $v \models \varphi$ when the clock valuation v satisfies the clock constraint φ.

Updates

An *update* is a function which assigns to each valuation a set of valuations. Here, we restrict ourselves to local updates which are defined in the following way. A *simple update* over a clock z is of one of the two following forms:

$$up ::= z :\sim c \mid z :\sim y + d,\ \text{where}\ c, d \in \mathbb{Q}_+,\ y \in \mathbb{X}\ \text{and}\ \sim \in \{<, \leq, =, \neq, \geq, >\}$$

When the operator $\sim$ is the equality ($=$), the update is said to be *deterministic*, *non deterministic* otherwise. Let v be a valuation and up be a simple update over z. A valuation v' is in $up(v)$ if $v'(y) = v(y)$ for any clock $y \neq z$ and if $v'(z) \sim c$ ($v'(z) \sim v(y) + d$ resp.) if $up = z :\sim c$ ($up = z :\sim y + d$ resp.)
The set $lu(\mathcal{U})$ of local updates generated by a set of simple updates $\mathcal{U}$ is defined as follows. A collection $up = (up_i)_{1\leq i\leq k}$ is in $lu(\mathcal{U})$ if, for each i, up_i is a simple update of $\mathcal{U}$ over some clock $x_i \in X$ (note that it could happen that $x_i = x_j$ for some $i \neq j$). Let v, $v' \in \mathbb{T}^n$ be two clock valuations. We have $v' \in up(v)$ if and only if, for any i, the clock valuation v'' defined by $v''(x_i) = v'(x_i)$ and $v''(y) = v(y)$ for any $y \neq x_i$ verifies $v'' \in up_i(v)$.
Note that $up(v)$ may be empty. For instance, the local update $(x :< 1, x :> 1)$ leads to an empty set. But if we take the local update $(x :> y, x :< 7)$, the value $v'(x)$ has to satisfy : $v'(x) > v(y) \wedge v'(x) < 7$.
For any subset X of $\mathbb{X}$, $\mathcal{U}(X)$ is the set of local updates which are collections of simple updates over clocks of X. In the following, $\mathcal{U}_0(X)$ denotes the set of reset updates. A reset update is an update up such that for every clock valuation v, v' with $v' \in up(v)$ and any clock $x \in X$, either $v'(x) = v(x)$ or $v'(x) = 0$. It is precisely this set of updates which was used in "classical" timed automata [3].

Updatable Timed Automata

An *updatable timed automaton* over $\mathbb{T}$ is a tuple $\mathcal{A} = (\Sigma, Q, X, T, I, F, R)$, where Σ is a finite alphabet of actions, Q a finite set of states, $X \subseteq \mathbb{X}$ a finite set of clocks, $T \subseteq Q \times [\mathcal{C}(X) \times \Sigma \cup \{\varepsilon\} \times \mathcal{U}(X)] \times Q$ a finite set of transitions, $I \subseteq Q$ ($F \subseteq Q$, $R \subseteq Q$ resp.) the subset of initial (final, repeated resp.) states.
Let $\mathcal{C} \subset \mathcal{C}(\mathbb{X})$ be a subset of clock constraints and $\mathcal{U} \subset \mathcal{U}(\mathbb{X})$ be a subset of updates, the class $Aut_\varepsilon(\mathcal{C},\mathcal{U})$ is the set of all timed automata whose transitions only use clock constraints of $\mathcal{C}$ and updates of $\mathcal{U}$. The usual class of timed automata, defined in [2], is the family $Aut_\varepsilon(\mathcal{C}_{df}(\mathbb{X}),\mathcal{U}_0(\mathbb{X}))$.
A *path* in $\mathcal{A}$ is a finite or an infinite sequence of consecutive transitions:

$$P = q_0 \xrightarrow{\varphi_1, a_1, up_1} q_1 \xrightarrow{\varphi_2, a_2, up_2} q_2 \ldots, \text{ where } (q_{i-1}, \varphi_i, a_i, up_i, q_i) \in T, \ \forall i > 0$$

The path is said *accepting* if $q_0 \in I$ and *either* it is finite and it ends in an final state, *or* it is infinite and passes infinitely often through a repeated state. A *run* of the automaton through the path P is a sequence of the form:

$$\langle q_0, v_0 \rangle \xrightarrow[t_1]{\varphi_1, a_1, up_1} \langle q_1, v_1 \rangle \xrightarrow[t_2]{\varphi_2, a_2, up_2} \langle q_2, v_2 \rangle \ldots$$

where $\tau = (t_i)_{i \geq 1}$ is a time sequence and $(v_i)_{i \geq 0}$ are clock valuations such that $\forall x \in \mathbb{X}, v_0(x) = 0$ and $\forall i \geq 1, v_{i-1} + (t_i - t_{i-1}) \models \varphi_i$ and $v_i \in up_i(v_{i-1} + (t_i - t_{i-1}))$. Remark that any set $up_i(v_{i-1} + (t_i - t_{i-1}))$ of a run is non empty.
The label of the run is the sequence $(a_1, t_1)(a_2, t_2) \cdots \in ((\Sigma \cup \{\varepsilon\}) \times \mathbb{T})^\infty$. The timed word associated with this sequence is $w = (a_{i_1}, t_{i_1})(a_{i_2}, t_{i_2}) \ldots$ where $a_{i_1} a_{i_2} \ldots$ is the sequence of actions which are in Σ (i.e. distinct from ε). If the path P is accepting then the timed word w is accepted by the timed automaton.

About Decidability of Updatable Timed Automata

For verification purposes, a fundamental question is to know if the emptiness of (the language accepted by) an updatable timed automaton is decidable or not. The paper [5] proposes a precise characterization which is summarized in the picture below. Note that decidability can depend on the set of clock constraints that are used – diagonal-free or not – which makes an important difference with "classical" timed automata for which it is well known that these two kinds of constraints are equivalent. The technique proposed in [5] shows that all the decidability cases are PSPACE-complete.

		diagonal-free clock constraints	general clock constraints
Deterministic updates	$x := c$; $x := y$	DECIDABLE	DECIDABLE
	$x := y + c,\ c \in \mathbb{Q}^+$	DECIDABLE	UNDECIDABLE
	$x := y + c,\ c \in \mathbb{Q}^-$	UNDECIDABLE	UNDECIDABLE
Non deterministic updates	$x :< c,\ c \in \mathbb{Q}^+$	DECIDABLE	DECIDABLE
	$x :> c,\ c \in \mathbb{Q}^+$	DECIDABLE	UNDECIDABLE
	$x :< y + c,\ c \in \mathbb{Q}^+$	DECIDABLE	UNDECIDABLE
	$x :> y + c,\ c \in \mathbb{Q}^+$	DECIDABLE	UNDECIDABLE

The present paper adresses the natural question of the exact expressive power of the decidable classes. To solve this problem, we first introduce natural and classical equivalences between updatable timed automata.

3 Some Equivalences of Updatable Timed Automata

Language Equivalence

Two updatable timed automata are *language-equivalent* if they accept the same timed language. By extension, two families Aut_1 and Aut_2 are said to be equivalent if any automaton of one of the families is equivalent to one automaton of the other. We write $\equiv_\ell$ in both cases. For instance, $Aut_\varepsilon(\mathcal{C}_{df}(\mathbb{X}),\mathcal{U}_0(\mathbb{X})) \equiv_\ell Aut_\varepsilon(\mathcal{C}(\mathbb{X}),\mathcal{U}_0(\mathbb{X}))$, (see e.g. [7]).

Bisimilarity

Bisimilarity [15, 14] is stronger than language equivalence. It defines a step by step correspondence between two transition systems. Two labelled transition systems $\mathcal{T} = (S, S_0, E, (\xrightarrow{e})_{e\in E})$ and $\mathcal{T}' = (S', S'_0, E, (\xrightarrow{e})_{e\in E})$ are *bisimilar* whenever there exists a relation $\mathcal{R} \subseteq S \times S'$ which meets the following conditions:

$$\text{INITIALIZATION}: \begin{cases} \forall s_0 \in S_0,\ \exists s'_0 \in S'_0 \text{ such that } s_0 \mathcal{R} s'_0 \\ \forall s'_0 \in S'_0,\ \exists s_0 \in S_0 \text{ such that } s_0 \mathcal{R} s'_0 \end{cases}$$

$$\text{PROPAGATION}: \begin{cases} \text{if } s_1 \mathcal{R} s'_1 \text{ and } s_1 \xrightarrow{e} s_2 \text{ then there exists } s'_2 \in S' \\ \qquad \text{such that } s'_1 \xrightarrow{e} s'_2 \text{ and } s_2 \mathcal{R} s'_2 \\ \text{if } s_1 \mathcal{R} s'_1 \text{ and } s'_1 \xrightarrow{e} s'_2 \text{ then there exists } s_2 \in S \\ \qquad \text{such that } s_1 \xrightarrow{e} s_2 \text{ and } s_2 \mathcal{R} s'_2 \end{cases}$$

Strong and Weak Bisimilarity

Timed transition systems - Each updatable timed automaton $\mathcal{A} = (\Sigma, Q, X, T, I, F, R)$ in $Aut_\varepsilon(\mathcal{C}(\mathbb{X}),\mathcal{U}(\mathbb{X}))$ defines a timed transition system $\mathcal{T}_\mathcal{A} = (S, S_0, E, (\xrightarrow{e})_{e\in E})$ as follows :

- $S = Q \times \mathbb{T}^X$, $S_0 = \{\langle q, v\rangle \mid q \in I \text{ and } \forall x \in X,\ v(x) = 0\}$, $E = \Sigma \cup \{\varepsilon\} \cup \mathbb{Q}_+$
- $\forall a \in \Sigma \cup \{\varepsilon\}$, $\langle q, v\rangle \xrightarrow{a} \langle q', v'\rangle$ iff $\exists (q, \varphi, a, up, q') \in T$ s.t. $v \models \varphi$ and $v' \in up(v)$
- $\forall d \in \mathbb{Q}_+$, $\langle q, v\rangle \xrightarrow{d} \langle q', v'\rangle$ iff $q = q'$ and $v' = v + d$

When ε is considered as an invisible action, each updatable timed automaton $\mathcal{A}$ in $Aut_\varepsilon(\mathcal{C}(\mathbb{X}),\mathcal{U}(\mathbb{X}))$ defines another transition system $\mathcal{T}'_\mathcal{A} = (S, S_0, E', (\stackrel{e}{\Rightarrow})_{e\in E})$ as follows:

- $S = Q \times \mathbb{T}^X$, $S_0 = \{\langle q, v\rangle \mid q \in I \text{ and } \forall x \in X,\ v(x) = 0\}$, $E' = \Sigma \cup \mathbb{Q}_+$
- $\forall a \in \Sigma$, $\langle q, v\rangle \stackrel{a}{\Rightarrow} \langle q', v'\rangle$ iff $\langle q, v\rangle \xrightarrow{\varepsilon}^* \xrightarrow{a} \xrightarrow{\varepsilon}^* \langle q', v'\rangle$
- $\forall d \in \mathbb{Q}_+$, $\langle q, v\rangle \stackrel{d}{\Rightarrow} \langle q', v'\rangle$ iff $\langle q, v\rangle \xrightarrow{\varepsilon}^* \xrightarrow{d_1} \xrightarrow{\varepsilon}^* \ldots \xrightarrow{d_k} \xrightarrow{\varepsilon}^* \langle q', v'\rangle$ and $d = \sum_{i=1}^k d_i$

Two bisimilarities for timed automata - Two updatable timed automata $\mathcal{A}$ and $\mathcal{B}$ are *strongly bisimilar*, denoted $\mathcal{A} \equiv_s \mathcal{B}$, if $\mathcal{T}_\mathcal{A}$ and $\mathcal{T}_\mathcal{B}$ are bisimilar. They are *weakly bisimilar*, denoted $\mathcal{A} \equiv_w \mathcal{B}$, if $\mathcal{T}'_\mathcal{A}$ and $\mathcal{T}'_\mathcal{B}$ are bisimilar.

Remark 1. Two timed strongly bisimilar automata are obviously weakly bisimilar. If the bisimulation $\mathcal{R}$ preserves the final and repeated states, weakly or strongly bisimilar updatable timed automata recognize the same language.

Let $\mathcal{A}$ a timed automaton and λ be a constant. We denote by $\lambda\mathcal{A}$ the timed automaton in which all the constants which appear are multiplied by the constant λ. The proof of the following lemma is immediate and similar to the one of Lemma 4.1 in [3]. This lemma allows us to treat only updatable timed automata where all constants appearing in the clock constraints and in the updates are integer (and not arbitrary rationals).

Lemma 1. *Let $\mathcal{A}$ and $\mathcal{B}$ be two timed automata and $\lambda \in \mathbb{Q}^+$ be a constant. Then $\mathcal{A} \equiv_w \mathcal{B} \iff \lambda\mathcal{A} \equiv_w \lambda\mathcal{B}$ and $\mathcal{A} \equiv_s \mathcal{B} \iff \lambda\mathcal{A} \equiv_s \lambda\mathcal{B}$*

4 Expressive Power of Deterministic Updates

We first deal with updatable timed automata where only deterministic updates are used. The following theorem is often considered as a "folklore" result.

Theorem 1. *Let $\mathcal{C} \subseteq \mathcal{C}(X)$ be a set of clock constraints and let $\mathcal{U} \subseteq lu(\{x := d \,|\, x \in X \text{ and } d \in \mathbb{Q}^+\} \cup \{x := y \,|\, x, y \in X\})$. Let $\mathcal{A}$ be in $Aut_\varepsilon(\mathcal{C},\mathcal{U})$. Then there exists $\mathcal{B}$ in $Aut_\varepsilon(\mathcal{C}(X),\mathcal{U}_0(X))$ such that $\mathcal{A} \equiv_s \mathcal{B}$.*

The next theorem is close to the previous one. Note nevertheless that this theorem becomes false if we consider arbitrary clock constraints, since as we recalled in section 2, the corresponding class is undecidable.

Theorem 2. *Let $\mathcal{C} \subseteq \mathcal{C}_{df}(X)$ be a set of diagonal-free clock constraints. Let $\mathcal{U} \subseteq lu(\{x := d \,|\, x \in X \text{ and } d \in \mathbb{Q}^+\} \cup \{x := y + d \,|\, x, y \in X \text{ and } d \in \mathbb{Q}^+\})$. Let $\mathcal{A}$ be in $Aut_\varepsilon(\mathcal{C},\mathcal{U})$. Then there exists $\mathcal{B}$ in $Aut_\varepsilon(\mathcal{C}_{df}(X),\mathcal{U}_0(X))$ such that $\mathcal{A} \equiv_s \mathcal{B}$.*

5 Expressive Power of Non Deterministic Updates

In the case of non deterministic updates, we first show that it is hopeless to obtain strong bisimulation with classical timed automata. To this purpose, let us consider the automaton $\mathcal{C}$ of Figure 1. It has been proved in [7] that there is no classical timed automaton without $\varepsilon-$transitions that recognize the same language than $\mathcal{C}$.
Now, it is not difficult to prove that the automaton $\mathcal{C}$ recognizes the same language than the automaton $\mathcal{B}$ and that $\mathcal{B}$ recognizes itself the same language than $\mathcal{A}$. If $\mathcal{A}$ was strongly bisimilar to some automaton $\mathcal{D}$ of $Aut_\varepsilon(\mathcal{C}(X),\mathcal{U}_0(X))$, this automaton $\mathcal{D}$ would not contain any $\varepsilon-$transition (since $\mathcal{A}$ does not contain such transition). Hence $L(\mathcal{D})$ would be equal to $L(\mathcal{A}) = L(\mathcal{C})$, in contradiction with the result of [7] recalled above. Since $\mathcal{A}$ belongs to the class $Aut_\varepsilon(\mathcal{C}(X),\mathcal{U}_1(X))$ (where $\mathcal{U}_1(X)$ denotes the set of updates corresponding to the cells labelled "decidable" in the "general clock constraints" column in tabular of Section 2), we thus have proved:

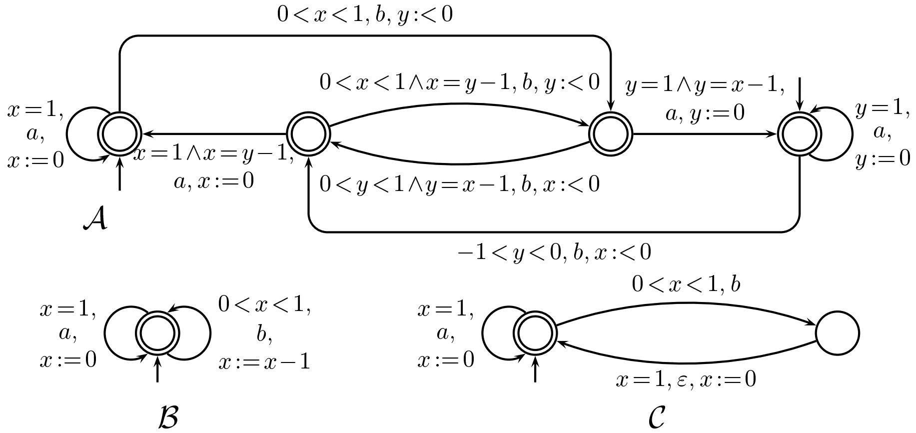

Fig. 1. *Timed automata $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{C}$*

Proposition 1. $Aut_\varepsilon(\mathcal{C}(\mathbb{X}), \mathcal{U}_1(\mathbb{X})) \not\equiv_s Aut_\varepsilon(\mathcal{C}(\mathbb{X}), \mathcal{U}_0(\mathbb{X}))$

We now focus on weak bisimilarity. As it will appear, the construction of an automaton of $Aut(\mathcal{C}(\mathbb{X}), \mathcal{U}_0(\mathbb{X}))$ weakly bisimilar to a given automaton of $Aut(\mathcal{C}(\mathbb{X}), \mathcal{U}_1(\mathbb{X}))$ is rather technical. As we recalled in Section 2, the decidable classes of updatable timed automata depend on the set of clock constraints that are used. We consider first the case of diagonal-free clock constraints.

We first propose a normal form for diagonal-free updatable automata. Let $(c_x)_{x \in X}$ be a family of constants of $\mathbb{N}$. In what follows we will restrict ourselves to the clock constraints $x \sim c$ where $c \leq c_x$. We define: $\mathcal{I}_x = \{]d; d+1[\mid 0 \leq d < c_x\} \cup \{[d] \mid 0 \leq d \leq c_x\} \cup \{]c_x, \infty[\}$
A clock constraint φ is said to be *total* if φ is a conjunction $\bigwedge_{x \in X} I_x$ where for each clock x, I_x is an element of $\mathcal{I}_x$. Any diagonal free clock constraint bounded by the constants $(c_x)_{x \in X}$ is equivalent to a disjunction of total clock constraints.

We define $\mathcal{I}'_x = \{]d; d+1[\mid 0 \leq d < c_x\} \cup \{]c_x, \infty[\}$. An update up_x is *elementary* if it is of one of the two following forms:

- $x := c$ or $x \in I'_x$ with $I'_x \in \mathcal{I}'_x$,
- $\bigwedge_{y \in H} x :\sim y + c \wedge x \in I'_x$ with $\sim \in \{=, <, >\}$, $I'_x \in \mathcal{I}'_x$ and $\forall y \in H$, $c_x \leq c_y + c$.

An elementary update $((\bigwedge_{y \in H} x :\sim y + c) \wedge x \in I'_x)$ is *compatible* with a total guard $\bigwedge_{x \in X} I_x$ if, for any $y \in H$, $I_y + c \subseteq I'_x$. By applying classical rules of propositional calculus and splitting the transitions, we obtain the following normal form for diagonal-free updatable timed automata.

Proposition 2. *Any diagonal-free updatable timed automaton from $Aut_\varepsilon(\mathcal{C}_{df}(X), \mathcal{U}(X))$ is strongly bisimilar to a diagonal-free updatable timed automaton from $Aut_\varepsilon(\mathcal{C}_{df}(X), \mathcal{U}(X))$ in which for any transition (p, φ, a, up, q) it holds:*

- *φ is a total guard*
- *$up = \bigwedge_{x \in X} up_x$ with for any x, up_x is an elementary update compatible with φ*

Construction for Diagonal-Free Updatable Timed Automata

We can now state our main result concerning updatable diagonal-free timed automata:

Theorem 3. *Let $\mathcal{C} \subseteq \mathcal{C}_{df}(X)$ be a set of diagonal-free clock constraints. Let $\mathcal{U} \subseteq \mathcal{U}(X)$ be a set of updates. Let $\mathcal{A}$ be in $Aut(\mathcal{C},\mathcal{U})$. Then there exists $\mathcal{B}$ in $Aut_{\varepsilon}(\mathcal{C}_{df}(X),\mathcal{U}_0(\mathbb{X}))$ such that $\mathcal{A} \equiv_w \mathcal{B}$. In particular $\mathcal{A}$ and $\mathcal{B}$ accept the same timed language.*

Proof (Sketch of proof). Thanks to Lemma 1 and Proposition 2, we assume that all the constants appearing in $\mathcal{A}$ are integers and that $\mathcal{A}$ is in normal form for some constants $(c_x)_{x \in X}$. For each clock x, we denote by $\mathcal{I}''_x$ the set of intervals $\{]c; c+1[\,|\, 0 \leq c < c_x\}$.
A clock x is said *fixed* if the last update of x was of the form either $x := c$ or $(\bigwedge_{y \in H} x := y + c) \wedge x :\in I'_x$ where all the clocks of H were fixed themselves. A clock which is not fixed is said *floating*.
The transformation algorithm consists in constructing (a lot of) copies of the original automaton $\mathcal{A}$, adding suitable clocks, transforming the existing transitions and finally adding ε−transitions going from one copy to another.

Duplication of the initial automaton - For each subset $Y \subseteq X$, for each tuple $(I_y)_{y \in Y}$ with $I_y \in \mathcal{I}''_y$, for each partial preorder $\prec$ defined on Y and for each subset $Z \subseteq Y$, we consider a copy of $\mathcal{A}$, denoted by $\mathcal{A}_{(I_y)_{y \in Y},\prec,Z}$. Intuitively, each clock $y \in Y$ will be floating and with I_y as set of possible values. The preorder $\prec$ corresponds to the partial order between the fractional parts of the clocks of Y. The role of Z will be explained below.

Keeping in mind the fractional part of the clocks - We associate with each clock x an other clock z_x representing the fractional part of x. In an automaton $\mathcal{A}_{(I_y)_{y \in Y},\prec,Z}$, we need to force the fractional part of any clock x to stay in $[0; 1[$. If a fractional part reachs the value 1, then either the clock is in Y and we will change of automaton (see below) or the clock is not in Y and the fractional part has to be reset to 0. To this purpose, we add to this automaton:

- on each transition, the clock constraint $\bigwedge_{x \in X}(z_x < 1)$
- on each state r, for each clock $x \in X \setminus Y$, a loop $(r, z_x = 1, \varepsilon, z_x := 0, r)$

Erasing some transitions - Since in an automaton $\mathcal{A}_{(I_y)_{y \in Y},\prec,Z}$, a clock $y \in Y$ will always verify $y \in I_y$, a total clock constraint $\varphi \wedge \bigwedge_x x \in I'_x$ can be satisfied only if $I'_y = I_y$ for all $y \in Y$. Therefore, we erase all the transitions with clock constraints which do not have this property.

Modification of the updates - We consider a copy $\mathcal{A}_{(I_y)_{y \in Y},\prec,Z}$ and a transition (q, φ, a, up, q') inside this copy. This transition will be replaced by another transition $(q, \varphi, a, \widehat{up}, \widehat{q'})$ from $\mathcal{A}_{(I_y)_{y \in Y},\prec,Z}$ to another automaton $\mathcal{A}_{(\widehat{I_y})_{y \in \widehat{Y}},\widehat{\prec},\widehat{Z}}$ (which can be possibly $\mathcal{A}_{(I_y)_{y \in Y},\prec,Z}$ itself) and where $\widehat{q'}$ is the copy of q' in the new automaton. The elements $\widehat{Y}$, $(I_y)_{y \in Y}$, $\widehat{\prec}$ and $\widehat{up}$ are constructed inductively by considering one after the other the updates up_x involved in up (the order in which the updates are treated is irrelevant). The new update $\widehat{up}$ will be only

constituted of updates of the form $x := c$ or $x := y + c$. Initially, we set $\widehat{Y} = Y$, $\widehat{I_y} = I_y$ for all $y \in Y$, $\widehat{\prec} = \prec$, $\widehat{up} = true$ and $\widehat{Z} = Z$.
Before listing the different updates, let us explain the role of the set Z. Assume that a clock x is updated by an instruction $x :< y + c$ where y is floating. Then the clock x is added to the set of floating clocks. Since we do not want to use anymore non deterministic updates, we update the fractional part z_x to 0, $z_x := 0$. But we need to keep the current value of z_y in order to ensure that z_x, which has to be smaller than z_y, will not reach 1 before z_y. Of course, it can be checked easily if y is not updated but if it is the case, we do not have any way to verify this fact. Therefore, in such a case, we add the clock x to the set Z and we use a new clock w_x to keep in mind the current value of z_y: $w_x := z_y$. The required property is then verified by the condition $w_x \geq 1$.

- if up_x is equal to $x := c$ then we just have to consider x as fixed:
  - $\widehat{Y} \leftarrow \widehat{Y} \setminus \{x\}$, $\widehat{Z} \leftarrow \widehat{Z} \setminus \{x\}$, $\widehat{up} \leftarrow \widehat{up} \wedge x := c \wedge z_x := 0$
- if up_x is equal to $\bigwedge_{y \in H} x := y + c \wedge x :\in I'_x$ then :
  1. if I'_x is bounded, then we write H as the disjoint union of $H_1 = H \cap Y$ and $H_2 = H \setminus Y$. We distinguish two cases:
     (a) if $H_1 = \emptyset$, then:
       - $\widehat{Y} \leftarrow \widehat{Y} \setminus \{x\}$, $\widehat{Z} \leftarrow \widehat{Z} \setminus \{x\}$
       - $\widehat{up} \leftarrow \widehat{up} \wedge \bigwedge_{y \in H}(x := y + c \wedge z_x := z_y)$
     (b) if $H_1 \neq \emptyset$, then:
       - $\widehat{Y} \leftarrow \widehat{Y} \cup \{x\}$, $\widehat{I_x} \leftarrow I'_x$, $\widehat{Z} \leftarrow \widehat{Z} \cup \{x\}$
       - for each $y \in H_1$, $x \widehat{\prec} y$ and $y \widehat{\prec} x$
       - $\widehat{up} \leftarrow \widehat{up} \wedge \bigwedge_{y \in H_2}(x := y + c \wedge z_x := z_y) \wedge \bigwedge_{y \in H_2}(w_x := z_y)$ if $H_2 \neq \emptyset$; $\widehat{up} \leftarrow \widehat{up} \wedge (z_x := 0)$ if $H_2 = \emptyset$
  2. if I'_x is non bounded, then we write H as the disjoint union of $H_1 = H \cap Y$ and $H_2 = H \setminus Y$. We distinguish two cases:
     (a) if $H_1 = \emptyset$, then:
       - $\widehat{Y} \leftarrow \widehat{Y} \setminus \{x\}$, $\widehat{Z} \leftarrow \widehat{Z} \setminus \{x\}$
       - $\widehat{up} \leftarrow \widehat{up} \wedge \bigwedge_{y \in H}(x := y + c \wedge z_x := z_y)$
     (b) if $H_1 \neq \emptyset$, then:
       - $\widehat{Y} \leftarrow \widehat{Y} \cup \{t\} \setminus \{x\}$ where t is some clock of H_2
       - $\widehat{I_t}$ is some tested interval (we test whether the value of t is in some interval $]c; c+1[$ and the clock t becomes a floating clock in this interval)
       - $\widehat{Z} \leftarrow \widehat{Z} \cup \{t\} \setminus \{x\}$, for each clock $y \in H_1$, $t \widehat{\prec} y$ and $y \widehat{\prec} t$
       - $\widehat{up} \leftarrow \widehat{up} \wedge \bigwedge_{y \in H_2}(x' := y + c) \wedge (x := c_x + 1 \wedge z_x := 0) \wedge (w_t := z_t)$
- if up_x is equal to $\bigwedge_{y \in H} x :< y + c \wedge x :\in I'_x$ then there are two cases:
  1. if I'_x is bounded, then:
     - $\widehat{Y} \leftarrow \widehat{Y} \cup \{x\}$, $\widehat{I_x} \leftarrow I'_x$
     - for all $y \in Y \cap H$, $x \widehat{\prec} y$ (but not $y \widehat{\prec} x$) is added to $\widehat{\prec}$
     - $\widehat{Z} \leftarrow \widehat{Z} \setminus \{x\}$ if $H \subseteq Y$; $\widehat{Z} \leftarrow \widehat{Z} \cup \{x\}$ if $H \not\subseteq Y$
     - $\widehat{up} \leftarrow \widehat{up} \wedge (z_x := 0) \wedge \bigwedge_{y \in H \setminus Y}(w_x := z_y)$
  2. if I'_x is non bounded, then:
     - $\widehat{Y} \leftarrow \widehat{Y} \setminus \{x\}$, $\widehat{up} \leftarrow \widehat{up} \wedge (x := c_x + 1) \wedge (z_x := 0)$

– if up_x is equal to $\bigwedge_y x :> y + c \wedge x :\in I'_x$ then:

1. if I'_x is non bounded, then:
   - $\widehat{Y} \leftarrow \widehat{Y} \cup \{x\}$, $\widehat{I_x} \leftarrow I'_x$, $\widehat{Z} \leftarrow \widehat{Z} \setminus \{x\}$
   - $y \widehat{\prec} x$ (but not $x \widehat{\prec} y$) is added to $\widehat{\prec}$, $\widehat{up} \leftarrow \widehat{up} \wedge z_x := z_y$
2. if I'_x is bounded, then:
   - $\widehat{Y} \leftarrow \widehat{Y} \setminus \{x\}$, $\widehat{up} \leftarrow \widehat{up} \wedge (x := c_x + 1) \wedge (z_x := 0)$

Adding deterministic updates to go from one copy to another- We consider a particular copy $\mathcal{A}_{(I_y)_{y\in Y},\prec,Z}$. We add new $\varepsilon-$transitions in order to leave this automaton as soon as some clock y leaves the interval I_y. By definition, the clocks which will first leave I_y belong to the maximal elements of the preorder $\prec$. Therefore, for any subset M of Y such that the elements of M are maximal elements of the preorder $\prec$ and if $y \in M$, $x \prec y$ and $y \prec x$ then $x \in M$, we add a transition from any copy of a state q in $\mathcal{A}_{(I_y)_{y\in Y},\prec,Z}$ to the copy of q in the automaton $\mathcal{A}_{(I_y)_{y\in Y'},\prec',Z'}$ with $Y' = Y \setminus M$, $\prec' = \prec \cap (Y' \times Y')$, $Z' = Z \setminus M$. This transition is labelled by

$$\bigwedge_{x\in X\cap Z}(z_x \leq 1) \wedge \bigwedge_{x\in X\setminus Z}(z_x < 1) \wedge \bigwedge_{x\in Z\cap M}(w_x \geq 1),\ \varepsilon,\ \forall x \in M\ x := sup(I_x)$$

where $sup(I_x)$ is the least upper bound of the interval I_x.
Intuitively it means that the values of some maximal elements have reached the upper bound of their floating interval and thus become fixed.
Now, we just need to define a weak bisimulation $\mathcal{R}$. Roughly, a state of the original timed automaton will be in relation with all the copies of this state (with appropriate valuations). This concludes the proof of Theorem 3.

When we deal with the general updatable timed automata, as we recalled in Section 2, we need to restrict deeply the updates that are used in order to have decidable classes. As states the next theorem, these classes are once again weakly bisimilar to classical timed automata with $\varepsilon-$transitions.

Theorem 4. *Let $\mathcal{C} \subseteq \mathcal{C}(X)$ be a set of general clock constraints and $\mathcal{U} \subseteq lu(\{x := y \,|\, x, y \in X\} \cup \{x :\sim c \,|\, x \in X, c \in \mathbb{Q}^+$ and $\sim \in \{<, \leq, =\}\})$ be a set of updates. Let $\mathcal{A}$ be in $Aut(\mathcal{C},\mathcal{U})$. Then there exists $\mathcal{B}$ in $Aut_\varepsilon(\mathcal{C}(X),\mathcal{U}_0(X))$ such that $\mathcal{A} \equiv_w \mathcal{B}$. In particular $\mathcal{A}$ and $\mathcal{B}$ accept the same timed language.*

The proof is quite similar to the one of Theorem 3, and even simpler because there is no non deterministic update allowed and involving two clocks.

6 Conclusion

Our results are summarized in the following tabular (a $\times$ denotes an undecidable case). A cell labelled "(Strongly/Weakly) bisimilar" means that any updatable timed automaton of the class represented by the cell is (strongly/weakly) bisimilar to a "classical" timed automaton with $\varepsilon-$transitions:

		Diagonal-free constraints	General constraints
Determ. updates	$x := c$; $x := y$	Strongly bisimilar	Strongly bisimilar
	$x := y + c$, $c \in \mathbb{Q}^+$	Strongly bisimilar	$\times$
Non Determ. updates	$x :< c$, $c \in \mathbb{Q}^+$	Weakly bisimilar	Weakly bisimilar
	$x :> c$, $c \in \mathbb{Q}^+$	Weakly bisimilar	$\times$
	$x :\sim y + c$, $c \in \mathbb{Q}^+$, $\sim \in \{<, >\}$	Weakly bisimilar	$\times$

References

1. R. Alur, C. Courcoubetis, and T.A. Henzinger. The observational power of clocks. In *Proc. of CONCUR'94*, LNCS 836, pages 162–177, 1994.
2. R. Alur and D. Dill. Automata for modeling real-time systems. In *Proc. of ICALP'90*, LNCS 443, pages 322–335, 1990.
3. R. Alur and D. Dill. A theory of timed automata. *TCS'94*, pages 183–235, 1994.
4. R. Alur, T.A. Henzinger, and M. Vardi. Parametric real-time reasoning. In *Proc. of the 25th ACM STOC*, pages 592–601, 1993.
5. P. Bouyer, C. Dufourd, E. Fleury, and A. Petit. Are timed automata updatable ? In *Proc. of CAV'2000*, LNCS, 2000. To appear.
6. P. Bouyer, C. Dufourd, E. Fleury, and A. Petit. Expressiveness of updatable timed automata. Research report, ENS de Cachan, 2000.
7. B. Bérard, V. Diekert, P. Gastin, and A. Petit. Characterization of the expressive power of silent transitions in timed automata. *Fundamenta Informaticae*, pages 145–182, 1998.
8. B. Bérard and C. Dufourd. Timed automata and additive clock constraints. Research report LSV-00-4, LSV, ENS de Cachan, 2000.
9. B. Bérard and L. Fribourg. Automatic verification of a parametric real-time program : the ABR conformance protocol. In *Proc. of CAV'99*, LNCS 1633, pages 96–107, 1999.
10. C. Choffrut and M. Goldwurm. Timed automata with periodic clock constraints. Technical Report 99/28, LIAFA, Université Paris VII, 1999.
11. F. Demichelis and W. Zielonka. Controlled timed automata. In *Proc. of CONCUR'98*, LNCS 1466, pages 455–469, 1998.
12. T.A. Henzinger, P. Ho, and H. Wong-Toi. Hytech: A model checker for hybrid systems. In *Software Tools for Technology Transfer*, pages 110–122, 1997. (special issue on Timed and Hybrid Systems).
13. K.G. Larsen, P. Pettersson, and W. Yi. Uppaal in a Nutshell. *Int. Journal on Software Tools for Technology Transfer*, 1:134–152, 1997.
14. R. Milner. *Communication and Concurrency.* Prentice Hall Int., 1989.
15. D. M. Park. Concurrency on automata and infinite sequences. In *CTCS'81*, LNCS 104, pages 167–183, 1981.
16. T. Wilke. Specifying timed state sequences in powerful decidable logics and timed automata. In *Proc. of FTRT-FTS*, LNCS 863, pages 694–715, 1994.
17. S. Yovine. A verification tool for real-time systems. *Springer International Journal of Software Tools for Technology Transfer*, 1, 1997.