MTL-Model Checking of One-Clock Parametric Timed Automata is Undecidable

Karin Quaas*
Institut für Informatik
Universität Leipzig
D-04109 Leipzig, Germany

Parametric timed automata extend timed automata (Alur and Dill, 1991) in that they allow the specification of *parametric* bounds on the clock values. Since their introduction in 1993 by Alur, Henzinger, and Vardi, it is known that the emptiness problem for parametric timed automata with one clock is decidable, whereas it is undecidable if the automaton uses three or more parametric clocks. The problem is open for parametric timed automata with two parametric clocks. Metric temporal logic, MTL for short, is a widely used specification language for real-time systems. MTL-model checking of timed automata is decidable, no matter how many clocks are used in the timed automaton. In this paper, we prove that MTL-model checking for parametric timed automata is undecidable, even if the automaton uses only one clock and one parameter and is deterministic.

1 Introduction

An important field of algorithmic verification is the analysis of real-time systems, *i.e.*, systems whose behaviour depend on time-critical aspects. Since the early nineties, numerous formalisms have been investigated to express and verify real-time properties. Two prominent examples of such formalisms are *timed automata* and *metric temporal logic*. Timed automata [3] extend classical finite automata with a finite set of real-valued *clocks* whose values grow with the passage of time. The edges of a timed automaton are labelled with *clock constraints* that compare the value of a clock with some constant. An edge can only be taken if the current values of the clocks satisfy the clock constraint labelling the edge. The central property of timed automata is the decidability of the emptiness problem [3].

Metric temporal logic (MTL, for short) extends classical linear temporal logic by constraining the temporal modalities with intervals of the non-negative reals. For example, the formula $\mathsf{F}_{[0,2]}\varphi$ means that φ will hold within two time units from now. Introduced by Koymans in 1990 [17], the satisfiability problem and the model checking problem for timed automata were assumed to be undecidable for a long time. However, more than 20 years later it was proved by Ouaknine and Worrell [19] that both problems are decidable if MTL is interpreted in the pointwise semantics over *finite* timed words. The decidability of the MTL-model checking problem for timed automata is independent of the number of clocks that the timed automaton uses.

A major drawback of timed automata and MTL is that they only allow the specification of *concrete* constraints on timing properties, *i.e.*, one has to provide the concrete values of all time-related constraints that occur in the real-time system. However, it is often more realistic to provide *symbolic* (or, *parametric*) constraints, in particular, if the real-time system under construction is not known in full details in the early stages of design. With the purpose to overcome the incapability of timed automata to express parametric time constraints, *parametric timed automata* were introduced [6]. Parametric timed automata are timed

*The author is supported by Deutsche Forschungsgemeinschaft (DFG), project QU 316/1-1.

É. André and G. Frehse (Eds.): 1st International Workshop
on Synthesis of Continuous Parameters (SynCoP'14).
EPTCS 145, 2014, pp. 5–17, doi:10.4204/EPTCS.145.3

automata defined over a finite set of parameters, which can be used in clock constraints labelling the edges of the automaton. For an example, consider the parametric timed automaton shown in Fig.1 on page 4. The clock y is concretely constrained by a constant like in ordinary timed automata. In contrast to this, the clock x is parametrically constrained by the parameter p. The value of p is determined by a parameter valuation, *i.e.*, a function mapping each parameter to a value in the non-negative reals.

A crucial verification problem for parametric timed automata is the emptiness problem: given a parametric timed automaton $\mathscr{A}$, does there exist some parameter valuation such that $\mathscr{A}$ has an accepting run? However, it turns out that this problem is undecidable already if $\mathscr{A}$ uses three or more parametric clocks [6]. On the positive side, the problem is decidable if in $\mathscr{A}$ at most one clock is compared to parameters. So far nothing is known about the decidability status for parametric timed automata with two parametric clocks; the problem is closely related to some hard and open problems of logic and automata theory [6].

In this paper, we concern ourselves with the MTL-model checking problem for parametric timed automata: given a parametric timed automaton $\mathscr{A}$ and a specification in form of an MTL formula φ, does there exist some parameter valuation such that all finite runs of $\mathscr{A}$ satisfy φ? For parametric timed automata with three clocks, the undecidability of this problem follows from the undecidability of the emptiness problem. Here, we prove that the problem is undecidable even if $\mathscr{A}$ uses only one clock and one parameter and is deterministic. This negative result is in contrast to the decidability of the emptiness problem for one-clock parametric timed automata, and the decidability of MTL-model checking of timed automata. The result can be regarded as further step towards the precise decidability border for the reachability problem for parametric timed automata with two parametric clocks, which is open for more than 20 years.

Related work The reader might wonder why we consider model checking for *parametric* timed automata and *standard* MTL, *i.e.*, a non-parametric extension of MTL. It is well known that if we extend classical LTL with formulae of the form $\varphi_1 \mathsf{U}_{=p} \varphi_2$, meaning that φ_2 has to hold in exactly p steps from now on for some parameter p, then the satisfiability problem ("Given a formula φ, is there some parameter valuation such that φ is satisfiable?") is undecidable: LTL with parameterized *equality modalities* of the form $\mathsf{U}_{=p}$ can be used to encode halting computations of two-counter machines [4]. Undecidablity of the satisfiability problem implies undecidability of the model checking problem for all systems that are capable to recognize the universal language over a given alphabet (as it is the case for, *eg.*, timed automata). In [4] it is also noted that the undecidability proof for LTL with parameterized equality modalities can be adapted to prove the undecidability of the satisfiability problem for LTL extended with parameterized *upper bound modalities* of the form $\mathsf{U}_{\leq p}$ and *lower bound modalities* of the form $\mathsf{U}_{>p}$ unless we restrict every parameter to occur in *either* lower bound modalities *or* upper bound modalities, but not in both.

The restriction on the parameters of a parametric timed automaton to occur either as a lower bound or as an upper bound also forms an important subclass of parametric timed automata, called *lower bound/upper bound (L/U) automata* [15]. For this subclass the emptiness problem is decidable independent of the number of parametric clocks, and for both finite [15] and infinite runs [8]. Model checking L/U automata with parametric extensions of MITL [5] in the *interval-based* semantics is decidable [8, 13]. Recall that constraints occurring at modalities of MITL formulae are not allowed to be of the form $= n$ (not even if the constraint is *concrete, i.e.*, $n \in \mathbb{N}$); in fact, the satisfiability and model checking problems for (non-parametric) MTL in the interval-based semantics are undecidable [14].

A crucial aspect of our undecidability proof is the fact that MTL formulae can be used to encode

computations of *channel machines with insertion errors* [18]: For every channel machine $\mathscr{C}$, there is an MTL formula $\varphi_{\mathscr{C}}$ that is satisfiable if, and only if, $\mathscr{C}$ has a halting computation that may contain insertion errors. This fact was used in [18] to prove the lower complexity bound of the satisfiability problem for MTL over finite timed words. In our proof, we use the parameterized timed automaton to *exclude* insertion errors in the timed words encoding computations of $\mathscr{C}$. We remark that the idea for this proof is similar to the proof of the undecidability for the model checking problem for one-counter machines and Freeze LTL with one register ($\mathrm{LTL}_1^{\downarrow}$, for short) [12]: In [11], it is proved that $\mathrm{LTL}_1^{\downarrow}$ formulae can be used to encode halting computations of *counter automata with incrementing errors*. Like MTL, $\mathrm{LTL}_1^{\downarrow}$ is not capable to exclude such errors. In [12], it is shown that this incapability can be repaired by combining the formula with a non-deterministic one-counter machine. Let us, however, note that there are substantial technical differences between the formalisms MTL and parametric timed automata on the one side, and $\mathrm{LTL}_1^{\downarrow}$ and one-counter machines on the other side.

2 Parametric Timed Automata

We use $\mathbb{N}$, $\mathbb{Q}_{\geq 0}$, and $\mathbb{R}_{\geq 0}$ to denote the non-negative integers, non-negative rationals, and the non-negative reals, respectively. In this section, we fix a finite alphabet Σ, a finite set $\mathscr{P} = \{p_1, \ldots, p_m\}$ of *parameters*, and a finite set $\mathscr{X} = \{x_1, \ldots, x_n\}$ of *clocks*.

We define *clock constraints* ϕ over $\mathscr{X}$ and $\mathscr{P}$ to be conjunctions of formulae of the form $x \sim c$, where $x \in \mathscr{X}$, $c \in \mathbb{N} \cup \mathscr{P}$, and $\sim \in \{<, \leq, =, \geq, >\}$. We use $\Phi(\mathscr{X}, \mathscr{P})$ to denote the set of all clock constraints over $\mathscr{X}$ and $\mathscr{P}$. A *clock valuation* is a function from $\mathscr{X}$ to $\mathbb{R}_{\geq 0}$. For $\delta \in \mathbb{R}_{\geq 0}$, we define $\nu + \delta$ to be $(\nu + \delta)(x) = \nu(x) + \delta$ for each $x \in \mathscr{X}$. For $\lambda \subseteq \mathscr{X}$, we define $\nu[\lambda := 0]$ by $(\nu[\lambda := 0])(x) = 0$ if $x \in \lambda$, and otherwise $(\nu[\lambda := 0])(x) = \nu(x)$.

A parameter valuation is a function $\pi : \mathscr{P} \to \mathbb{Q}_{\geq 0}$ assigning a non-negative rational to each parameter.

A clock valuation ν and a parameter valuation π satisfy a clock constraint ϕ, written $(\nu, \pi) \models \phi$, if the expression obtained from ϕ by replacing each parameter p by $\pi(p)$ and each clock x by $\nu(x)$ evaluates to true.

A *parametric timed automaton* is a tuple $\mathscr{A} = (\Sigma, \mathscr{L}, \mathscr{L}_0, \mathscr{X}, \mathscr{P}, E, \mathscr{L}_F)$, where

- $\mathscr{L}$ is a finite set of locations,
- $\mathscr{L}_0 \subseteq \mathscr{L}$ is the set of *initial* locations,
- $E \subseteq \mathscr{L} \times \Sigma \times \Phi(\mathscr{X}, \mathscr{P}) \times 2^{\mathscr{X}} \times \mathscr{L}$ is a finite set of *edges*,
- $\mathscr{L}_F \subseteq \mathscr{L}$ is the set of *final* locations.

Each edge $(l, a, \phi, \lambda, l')$ represents a discrete transition from l to l' on the input symbol a. The clock constraint ϕ specifies the bounds on the value of the clocks, and the set λ specifies the clocks to be reset to zero.

A *global state* of $\mathscr{A}$ is a pair (l, ν), where $l \in \mathscr{L}$ represents the current location, and the clock valuation ν represents the current values of all clocks. The behaviour of $\mathscr{A}$ depends upon the current global state and the parameter valuation. Each parameter valuation π induces a $(\Sigma, \mathbb{R}_{\geq 0})$-labelled transition relation τ_π over the set of all global states of $\mathscr{A}$ as follows: $\langle (l, \nu), (a, \delta), (l', \nu') \rangle \in \tau_\pi$, where $a \in \Sigma$ and $\delta \in \mathbb{R}_{\geq 0}$, if, and only if, there is an edge $(l, a, \phi, \lambda, l') \in E$ such that for all clocks $x \in \mathscr{X}$ we have $(\nu(x) + \delta, \pi) \models \phi$, and $\nu' = (\nu(x) + \delta)[\lambda := 0]$. A π-run of $\mathscr{A}$ is a finite sequence $\Pi_{1 \leq i \leq k} \langle (l_{i-1}, \nu_{i-1}), (a_i, \delta_i), (l_i, \nu_i) \rangle$ such that $\langle (l_{i-1}, \nu_{i-1}), (a_i, \delta_i), (l_i, \nu_i) \rangle \in \tau_\pi$ for every $i \in \{1, \ldots, k\}$. A π-run is *successful* if $l_0 \in \mathscr{L}_0$, $\nu_0(x) = 0$, and $l_k \in \mathscr{L}_F$.

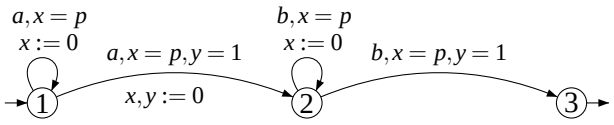

Figure 1: A parametric timed automaton $\mathscr{A}$.

A *timed word* is a non-empty finite sequence $(a_1,t_1)\dots(a_k,t_n) \in (\Sigma \times \mathbb{R}_{\geq 0})^+$ such that the sequence $t_1,\dots,t_n$ of timestamps is non-decreasing. We say that a timed word is *strictly monotonic* if $t_1,\dots,t_n$ is strictly increasing. We use $T\Sigma^+$ to denote the set of finite timed words over Σ. A set $L \subseteq T\Sigma^+$ is called a *timed language*.

Given a parametric timed automaton $\mathscr{A}$ and a parameter valuation π, we associate with each π-run $\Pi_{1\leq i\leq k}\langle(l_{i-1},v_{i-1}),(a_i,\delta_i),(l_i,v_i)\rangle$ the timed word $(a_1,\delta_1)(a_2,\delta_1+\delta_2)\dots(a_k,\sum_{1\leq i\leq k}\delta_k)$. We define $L_\pi(\mathscr{A})$ to be the set of timed words w for which there is a successful π-run of $\mathscr{A}$ that is associated with w. A parameter valuation π is *consistent with* $\mathscr{A}$ if $L_\pi(\mathscr{A})$ is not empty. We use $\Pi(\mathscr{A})$ to denote the set of parameter valuations that are consistent with $\mathscr{A}$.

We say that a parametric timed automaton $\mathscr{A}$ is *deterministic* if $\mathscr{L}_0$ is a singleton, and whenever $(l,a,\phi_1,\lambda_1,l_1)$ and $(l,a,\phi_2,\lambda_2,l_2)$ are two different edges in $\mathscr{A}$, then for all parameter valuations π and clock valuations v we have $(v,\pi) \not\models \phi_1 \wedge \phi_2$.

Example 2.1 *Figure 1 shows a parametric timed automaton over the alphabet $\Sigma = \{a,b\}$ using a parametric clock x and a clock y, and one parameter p. Assume $\pi(p) = n^{-1}$ for some $n \in \mathbb{N}$. Then $L_\pi(\mathscr{A})$ contains a single timed word, namely $(a,\pi(p))(a,2\pi(p))\dots(a,n\pi(p))(b,(n+1)\pi(p))\dots(b,2n\pi(p))$. For all other parameter valuations π, $L_\pi(\mathscr{A}) = \emptyset$, i.e., they are not consistent with $\mathscr{A}$. Hence we have $\Pi(\mathscr{A}) = \{\pi \mid \pi(p) = n^{-1}$ for some $n \in \mathbb{N}\}$. Note that $\mathscr{A}$ is not deterministic, but it can be made deterministic by adding the clock constraint $y < 1$ to the loops in locations 1 and 2.*

3 Metric Temporal Logic

The set of MTL formulae is built up from Σ by boolean connectives and a constraining version of the *until* modality:

$$\varphi ::= a \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \mathsf{U}_I \varphi_2$$

where $a \in \Sigma$ and $I \subseteq \mathbb{R}_{\geq 0}$ is an open, closed, or half-open interval with endpoints in $\mathbb{N} \cup \{\infty\}$. Note that we do *not* allow parameters as endpoints. If $I = \mathbb{R}_{\geq 0}$, then we may omit the annotation I on U_I.

We interprete MTL formulae in the *pointwise semantics*, *i.e.*, over finite timed words over Σ. Let $w = (a_1,t_1)(a_2,t_2)\dots(a_n,t_n)$ be a timed word, and let $i \in \{1,\dots,n\}$. We define the *satisfaction relation for MTL*, denoted by $\models$, inductively as follows:

$$\begin{aligned}
(w,i) \models a &\;\Leftrightarrow\; a_i = a \\
(w,i) \models \neg\varphi &\;\Leftrightarrow\; (w,i) \not\models \varphi, \\
(w,i) \models \varphi_1 \wedge \varphi_2 &\;\Leftrightarrow\; (w,i) \models \varphi_1 \text{ and } (w,i) \models \varphi_2, \\
(w,i) \models \varphi_1 \mathsf{U}_I \varphi_2 &\;\Leftrightarrow\; \exists j.i < j \leq |w| : (w,j) \models \varphi_2 \text{ and } t_j - t_i \in I, \text{ and } \forall k.i < k < j : (w,k) \models \varphi_1.
\end{aligned}$$

We say that a timed word $w \in T\Sigma^+$ satisfies an MTL formula φ, written $w \models \varphi$, if $(w,1) \models \varphi$. Given an MTL formula φ, we define $L(\varphi) := \{w \in T\Sigma^+ \mid w \models \varphi\}$. We use the following syntactical abbreviations:

$\varphi_1 \vee \varphi_2 := \neg(\neg\varphi_1 \wedge \neg\varphi_2)$, $\varphi_1 \rightarrow \varphi_2 := \neg\varphi_1 \vee \varphi_2$, $\mathtt{true} := p \vee \neg p$, $\mathtt{false} := \neg\mathtt{true}$, $\mathsf{X}_I\varphi := \mathtt{false}\mathsf{U}_I\varphi$, $\mathsf{F}_I\varphi := \mathtt{true}\mathsf{U}_I\varphi$, $\mathsf{G}_I\varphi := \neg\mathsf{F}_I\neg\varphi$. Observe that the use of the *strict* semantics for the until modality is essential to derive the next modality.

MTL-Model Checking Problem for Parametric Timed Automata

INPUT: A parametric timed automaton $\mathscr{A}$, an MTL formula φ.
QUESTION: Is there some parameter valuation π such that for every $w \in L_\pi(\mathscr{A})$ we have $w \models \varphi$?

In general, the MTL-model checking problem is undecidable for parametric timed automata. This follows from the undecidability of the emptiness problem for parametric timed automata with three or more parametric clocks [6]. In the next section, we prove the undecidability of the MTL-model checking problem for parametric timed automata using one parametric clock and one parameter.

4 Main Result

Theorem 4.1 *The MTL-model checking problem for parametric timed automata is undecidable, even if the automaton uses only one clock and one parameter and is deterministic.*

The remainder of this section is devoted to the proof of Theorem 4.1. The proof is a reduction of the control state reachability problem for channel machines, which we introduce in the following.

4.1 Channel Machines

Let Γ be a finite alphabet. We use ε to denote the *empty word* over Γ. Given two finite words $x, y \in \Gamma^*$, we use $x \cdot y$ to denote the *concatenation* of x any y. We define the order $\leq$ over the set of finite words over Γ by $x_1x_2\ldots x_m \leq y_1y_2\ldots y_n$ if there exists a strictly increasing function $f : \{1,\ldots,m\} \rightarrow \{1,\ldots,n\}$ such that $x_i = y_{f(i)}$ for every $i \in \{1,\ldots,m\}$.

A *channel machine* consists of a finite-state automaton acting on an unbounded fifo channel. Formally, a channel machine is a tuple $\mathscr{C} = (S, s_I, M, \Delta)$, where

- S is a finite set of *control states*,
- $s_I \in S$ is the initial control state,
- M is a finite set of *messages*,
- $\Delta \subseteq S \times L \times S$ is the transition relation over the label set $L = \{m!, m? \mid m \in M\} \cup \{\varepsilon\}$.

A *configuration* of $\mathscr{C}$ is a tuple (s,x), where $s \in S$ is the control state and $x \in M^*$ represents the contents of the channel. The rules in Δ induce an L-labelled transition relation $\rightarrow$ over the set of configurations of $\mathscr{C}$ as follows:

- $\langle (s,x), m!, (s',x') \rangle \in \rightarrow$ if, and only if, there exists some transition $(s, m!, s') \in \Delta$, $x \in \Sigma^*$, and $x' = x \cdot m$, *i.e.*, m is added to the tail of the channel.
- $\langle (s,x), m?, (s',x') \rangle \in \rightarrow$ if, and only if, there exists some transition $(s, m?, s') \in \Delta$, $x' \in \Sigma^*$, and $x = m \cdot x'$, *i.e.*, m is the head of the current channel content.
- $\langle (s,x), \varepsilon, (s',x') \rangle \in \rightarrow$ if, and only if, there exists some transition $(s, \varepsilon, s') \in \Delta$ and $x = \varepsilon$, *i.e.*, the channel is empty, and $x' = x$.

Next, we define another L-labelled transition relation $\rightsquigarrow$ over the set of configurations of $\mathscr{C}$. The relation $\rightsquigarrow$ is a superset of $\rightarrow$. It contains some additional transitions which result from *insertion errors*. We define $\langle (s,x_1),l,(s,x_1')\rangle \in \rightsquigarrow$, if, and only if, $\langle (s,x),l,(s',x')\rangle \in \rightarrow$, $x_1 \leq x$, and $x' \leq x_1'$. A computation of $\mathscr{C}$ is a finite sequence $\Pi_{1\leq i\leq k}\langle (s_{i-1},x_{i-1}),l_i,(s_i,x_i)\rangle$ such that $\langle (s_{i-1},x_{i-1}),l_i,(s_i,x_i)\rangle \in \rightsquigarrow$ for every $i \in \{1,\ldots,k\}$. We say that a computation is *error-free* if for all $i \in \{1,\ldots,k\}$ we have $\langle (s_{i-1},x_{i-1}),l_i,(s_i,x_i)\rangle \in \rightarrow$. Otherwise, we say that the computation is *faulty*.

Control State Reachability Problem for Channel Machines

INPUT: A channel machine $\mathscr{C}$ with control states S, a control state $s_F \in S$.
QUESTION: Is there an error-free computation of $\mathscr{C}$ from (s_I,ε) to (s_F,x) for some $x \in M^*$?

The control state reachability problem is undecidable for channel machines, because channel machines are Turing-powerful [9, 1].

4.2 Encoding Faulty Computations

For the remainder of this section, let $\mathscr{C} = (S,s_I,M,\Delta)$ be a channel machine and let $s_F \in S$. We construct an MTL formula $\varphi_{\mathscr{C}}$ that is satisfiable if, and only if, there exists some $x \in M^*$ such that $\mathscr{C}$ has a computation from (s_I,ε) to (s_F,x) that may be faulty. Later we are going to define a parametric timed automaton $\mathscr{A}_{\mathscr{C}}$ with one clock and one parameter to exclude faulty computations from $L(\varphi_{\mathscr{C}})$.

Let $\Sigma = S \cup M \cup L \cup \{\#,\star\}$, where $\#$ and $\star$ do not occur in $S \cup M \cup L$. We start with defining a timed language $L(\mathscr{C})$ over Σ that consists of all timed words that encode (potentially faulty) computations of $\mathscr{C}$ from (s_I,ε) to (s_F,x) for some $x \in M^*$. The definition of $L(\mathscr{C})$ follows the ideas presented in [18]. Let $\gamma := \Pi_{1\leq i\leq k}\langle (s_{i-1},x_{i-1}),l_i,(s_i,x_i)\rangle$ be a computation of $\mathscr{C}$ with $s_0 = s_I$, $x_0 = \varepsilon$, and $s_k = s_F$. Each configuration (s_i,x_i) occurring in γ is encoded by a timed word of duration one starting with s_0 at time δ for some arbitrary $\delta \in \mathbb{R}_{\geq 0}$. Every symbol s_i is followed by l_{i+1} after one time unit, and by s_{i+1} after two time units. The content x_i of the channel is stored in the time interval between s_i and l_{i+1}. Note that due to the denseness of the time domain we can indeed store the channel content without any restriction on its length. An important detail of the definition of $L(\mathscr{C})$ is that for every message symbol m between s_i and l_{i+1}, there is a copy in the encoding of the next configuration exactly two time units later, unless the label of the current transition is $m?$. In that case, the symbol m is simply removed from the encoding of the configuration.

For our reduction to work, we have to change the idea in some details. First, we define a timed language $L(\mathscr{C},n)$ for every $n \in \mathbb{N}$, where n is non-deterministically chosen and is supposed to represent the expected maximum length of the channel content during a computation. The empty channel in the initial configuration will be represented by a timed word with n hash symbols between s_0 and l_1. Second, we put a stronger condition on the copy policy of the messages. We require that for every hash symbol between s_0 and l_1 there is a message or hash symbol with *the same fractional part* between s_i and l_{i+1} for every $i \in \{1,\ldots,k-1\}$. In Fig. 2, we present some examples to explain the details. (a) If the current instruction is of the form $m_1!$ for some $m_1 \in M$, then in the encoding of the next configuration, the first hash symbol between the control state symbol and the next label symbol is replaced by m_1. (b) If in the encoding of the current configuration there is no hash symbol left, *i.e.*, the expected maximum length of the channel content is exceeded, then a new symbol m_1 is inserted at the end of the encoding of the next configuration. The timestamp of the newly inserted event can be any time strictly between the timestamps of the last message symbol and the next label symbol. (c) If the current instruction is of the form $m_1?$ and the first symbol in the encoding of the current configuration is m_1, then we replace

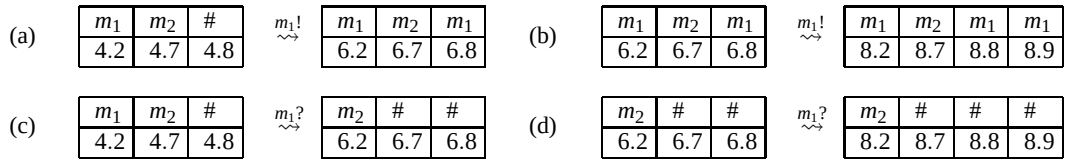

Figure 2: Encoding of the channel content

m_1 by a new hash symbol at the end of the encoding of the next configuration, and additionally shift the fractional parts of the timestamps of the copies of all remaining symbols for one position to the right. (d) If the first symbol is not m_1, *i.e.*, an insertion error is occurring, then we insert a new hash symbol at the end of the encoding of the next configuration. Next, we give the formal definition of $L(\mathscr{C},n)$. Let $n \in \mathbb{N}$. The timed language $L(\mathscr{C},n)$ consists of all timed words w over Σ that satisfy the following conditions:

- w must be strictly monotonic.
- In w, every control state symbol s different from s_F is followed by a label symbol l after one time unit, and by a control state symbol s' after two time units, provided that $(s,l,s') \in \Delta$. The symbol s_F is followed by $\star$ after one time unit. Control state symbols, label symbols and the symbol $\star$ must not occur anywhere else in w.
- Symbols in $M \cup \{\#\}$ may occur in w between a control state symbol and a label symbol. They may not occur anywhere else in w.
- Between a control state and a label symbol, hash symbols # may only occur after message symbols $m \in M$.
- The (untimed) prefix of w must be of the form $s_I \#^n l s$ for some $l \in L, s \in S$.
- w must contain s_F.

Assume that w contains the infix $(s,\delta)(\sigma_1,\delta+\delta_1)(\sigma_2,\delta+\delta_2)\dots(\sigma_m,\delta+\delta_m)(l,\delta+1)$ for some $s \in S\backslash\{s_F\}$, $l \in L$, $\delta \in \mathbb{R}_{\geq 0}$ and $0 < \delta_1 < \delta_2 < \dots < \delta_m < 1$.

- If $l = \varepsilon$, then $\sigma_i = \#$ for all $i \in \{1,\dots,m\}$ (*i.e.*, the channel is indeed empty), and for each σ_i there is a copy two time units later.
- If $l = m!$, then we distinguish between two cases: If there is some $i \in \{1,\dots,m\}$ such that $\sigma_i = \#$, then *replace* σ_j by m two time units later, where $j \in \{1,\dots,m\}$ is the smallest number such that $\sigma_j = \#$. For each $k \in \{1,\dots,m\}\backslash\{j\}$, there is a copy of σ_k two time units later. Otherwise, *i.e.*, if for all $i \in \{1,\dots,m\}$ we have $\sigma_i \neq \#$, then for each $i \in \{1,\dots,m\}$, there is a copy of σ_i two time units later. Further, a new symbol m is added between the copy of σ_m and the following symbol in $L \cup \{\star\}$. Note that this corresponds to the case where n has been chosen too small to capture the maximum length of the channel content during the computation.
- If $l = m?$, then we distinguish between two cases: If $\sigma_1 = m$, then for each $i \in \{2,\dots,m\}$, there is a copy of σ_i two time units after the occurrence of σ_{i-1}. Further there is a new hash symbol two time units after the occurrence of σ_m. Otherwise, *i.e.*, if $\sigma_1 \neq m$, then there is a copy of σ_i two time units later for every $i \in \{1,\dots,m\}$. Further, the encoding of the next configuration contains an additional hash symbol between the copy of σ_m and the next symbol in $L \cup \{\star\}$. Note that this case corresponds to an *insertion error*.

Let $w_1 = (a_1,t_1)\dots(a_k,t_k)$ and $w_2 = (a'_1,t'_1)\dots(a'_{k'},t'_{k'})$ be two timed words. If $t_k \le t'_1$, then we define the *concatenation* of w_1 and w_2, denoted by $w_1 \cdot w_2$, to be the timed word $(a_1,t_1)\dots(a_k,t_k)(a'_1,t'_1)\dots(a'_{k'},t'_{k'})$. Let $w \in L(\mathscr{C},n)$. We use $\max(w)$ to denote the maximum number of symbols in $M \cup \{\#\}$ that occur in w between a control state symbol and a symbol in $L \cup \{\star\}$. Clearly, every timed word in $L(\mathscr{C},n)$ is of the form

$$(s_0,\delta)\cdot w_1 \cdot (l_1,\delta+1)(s_1,\delta+2)\cdot w_2 \cdot (l_2,\delta+3)\dots(s_F,\delta+N)\cdot w_N \cdot (\star,\delta+N+1)$$

for some $\delta \in \mathbb{R}_{\ge 0}$ and $N \in \mathbb{N}$, where $s_0 = s_I$ and for every $i \in \{1,\dots,N\}$, w_i is of the form

$$w_i = (\sigma^i_1,\delta+2(i-1)+\delta^i_1)(\sigma^i_2,\delta+2(i-1)+\delta^i_2)\dots(\sigma^i_{n_i},\delta+2(i-1)+\delta^i_{n_i})$$

for some $n_i \in \mathbb{N}$ with $n_1 = n$, and $0 < \delta^i_1 < \delta^i_2 < \dots < \delta^i_{n_i} < 1$. In the following, whenever we refer to a timed word $w \in L(\mathscr{C},n)$, we assume that w is of this form. The next lemma states that the fractional parts of the initial time delays $\delta^1_1,\dots,\delta^1_{n_1}$ are not lost. This will be important later.

Lemma 4.2 *Let $n \in \mathbb{N}$ and let $w \in L(\mathscr{C},n)$. For every $i \in \{1,\dots,N-1\}$ there exists a strictly increasing function $f_i : \{1,\dots,n_i\} \to \{1,\dots,n_{i+1}\}$ such that $\delta^i_j = \delta^{i+1}_{f_i(j)}$ for every $j \in \{1,\dots,n_i\}$.*

Proof The proof is by induction on N. (Induction base:) Observe that $\sigma^1_i = \#$ for every $i \in \{1,\dots,n_1\}$. Assume $l_1 = \varepsilon$. Then for every $j \in \{1,\dots,n_1\}$, there is a copy of σ^1_j two time units later. If $l_1 = m!$, then for every $j \in \{2,\dots,n_1\}$, there is a copy of σ^1_j two time units later, and σ^1_1 is replaced by m two time units later. If $l_1 = m?$, then for every $j \in \{1,\dots,n_1\}$, there is a copy of σ^1_j two time units later, and there is an additional symbol # between the copy of $\sigma^1_{n_1}$ and l_2. Whatever case, the definition of $L(\mathscr{C},n)$ does not exclude that new symbols in $M \cup \{\#\}$ are inserted somewhere between s_1 and l_2. Thus we have $n_1 \le n_2$. Moreover, since there is a copy for each symbol two time units later, there exists a strictly increasing function $f : \{1,\dots,n_1\} \to \{1,\dots,n_2\}$ such that $\delta^1_j = \delta^2_{f(j)}$ for every $j \in \{1,\dots,n_1\}$. (Induction step) Assume that the claim holds for all $i \in \{1,\dots,k\}$. We prove it also holds for $k+1$. We only treat the two remaining cases. First, assume $l_{k+1} = m?$ and $\sigma^{k+1}_1 = m$. By definition, for every $j \in \{2,\dots,n_{k+1}\}$, there is a copy of σ^{k+1}_j two time units after the occurrence of symbol σ^{k+1}_{j-1}. Further, the first symbol m is replaced by a new hash symbol two time units after the occurrence of $\sigma^{k+1}_{n_{k+1}}$. Second, assume $l_{k+1} = m!$ and we have $\sigma^{k+1}_j \ne \#$ for every $j \in \{1,\dots,n_{k+1}\}$. Then, for each $j \in \{1,\dots,n_{k+1}\}$, there is a copy of σ^{k+1}_j two time units later, and a new symbol m is added after the copy of $\sigma^i_{n_{k+1}}$. Whatever case, the definition of $L(\mathscr{C},n)$ does not exclude that new symbols in $M \cup \{\#\}$ are inserted between s_{k+2} and l_{k+2}. Hence $n_{k+1} \le n_{k+2}$. Since for every $j \in \{1,\dots,n_{k+1}\}$ the symbol σ^{k+1}_j is copied or replaced two time units later, there exists a strictly increasing function $f : \{1,\dots,n_{k+1}\} \to \{1,\dots,n_{k+2}\}$ such that $\delta^{k+1}_j = \delta^{k+2}_{f_{k+1}(j)}$ for every $j \in \{1,\dots,n_{k+1}\}$. □

Let $\gamma := \Pi_{1\le i\le k}\langle (s_{i-1},x_{i-1}),l_i,(s_i,x_i)\rangle$ be a finite computation of $\mathscr{C}$. We use $\max(\gamma)$ to denote the maximum length of the channel content occurring in γ, formally: $\max(\gamma) := \max\{|x_i| \mid 0 \le x_i \le k\}$.

Lemma 4.3 *For each error-free computation γ of $\mathscr{C}$ from (s_I,ε) to (s_F,x) for some $x \in M^*$, and every $\delta \in \mathbb{R}_{\ge 0}$, $0 < \delta_1 < \delta_2 < \dots < \delta_{\max(\gamma)} < 1$, there exists some timed word $w \in L(\mathscr{C},\max(\gamma))$ such that the prefix of w is of the form $(s_I,\delta)(\#,\delta+\delta_1)\dots(\#,\delta+\delta_{\max(\gamma)})(l_1,\delta+1)$ for some $l_1 \in L$, and $\max(w) = \max(\gamma)$.*

Proof Let γ be an errror-free computation of $\mathscr{C}$ of the form $\Pi_{1\le i\le k}\langle (s_{i-1},x_{i-1}),l_i,(s_i,x_i)\rangle$ where $s_0 = s_I$, $x_0 = \varepsilon$ and $s_k = s_F$. Further let $n = \max(\gamma)$. Now assume $\delta \in \mathbb{R}_{\ge 0}$ and $0 < \delta_1 < \delta_2 < \dots < \delta_n < 1$. Clearly there is some $w \in L(\mathscr{C},n)$ whose prefix is of the form $u_1 = (s_I,\delta)(\#,\delta+\delta_1)\dots(\#,\delta+\delta_n)(l_1,\delta+1)$. We

prove that there exists some $w \in L(\mathscr{C},n)$ such that u_1 is the prefix of w and $\max(w) = n$, *i.e.*, for every $i \in \{1,\ldots,k\}$, the number of symbols in $M \cup \{\#\}$ between s_{i-1} and l_i (and between s_k and $\star$) is equal to n. The proof is by induction on k.

(Induction base:) Assume $l_1 = \varepsilon$. By definition, there must be a copy for each # exactly two time units later. The addition of new symbols is not required. If $l_1 = m!$, then by definition the first occurrence of # is *replaced* by m exactly two time units later, and for each of the remaining # there is a copy two time units later. The addition of new symbols is not required. Note that the case $m?$ cannot occur because γ is error-free. Hence, there exists some timed word $w \in L(\mathscr{C},n)$ whose prefix is of the form $u_1 \cdot u_2$, where $u_2 = (s_1, 2+\delta)(\sigma_1^2, 2+\delta+\delta_1)(\#, 2+\delta+\delta_2)\ldots(\#, 2+\delta+\delta_n)(l_2, 2+\delta+1)$ for some $\sigma_1^2 \in M \cup \{\#\}$.

(Induction step:) Assume there is some timed word $w \in L(\mathscr{C},n)$ whose prefix is of the form $u_1 \cdot \cdots \cdot u_p$ for some $p < k$, where for every $i \in \{1,\ldots,p\}$, u_i is of the form

$$(s_{i-1}, 2(i-1)+\delta)(\sigma_1^i, 2(i-1)+\delta+\delta_1)(\sigma_2^i, 2(i-1)+\delta+\delta_2)\ldots(\sigma_n^i, 2(i-1)+\delta+\delta_n)(l_i, 2(i-1)+\delta+1)$$

for some $\sigma_1^i,\ldots,\sigma_n^i \in M \cup \{\#\}$.

Assume $l_p = m?$ for some $m \in M$. By the fact that γ is error-free, we know $\sigma_1^p = m$. By definition, there is a copy of σ_i^p two time units after the occurrence of σ_{i-1}^p for every $i \in \{2,\ldots,n\}$, and there is a new hash symbol inserted two time units after the occurrence of σ_n^p. The addition of new symbols is not required.

Assume $l_p = m!$ for some $m \in M$. Recall that $n = \max(\gamma)$ is the maximum length of the channel content in γ. Hence there must be some $j \in \{1,\ldots,n\}$ such that $\sigma_j^p = \#$. By definition, the smallest $j \in \{1,\ldots,n\}$ with $\sigma_j^p = \#$ is replaced by m exactly two time units later. For each of the remaining symbols there is a copy two time units later. The addition of new symbols is not required.

Assume $l_p = \varepsilon$. We can proceed as above, concluding that the addition of new symbols is not required.

Hence, there exists some timed word $w \in L(\mathscr{C},n)$ whose prefix is of the form $u_1 \cdot u_2 \cdot \ldots u_p \cdot u_{p+1}$, where $u_{p+1} = (s_p, 2p+\delta)(\sigma_1^{p+1}, 2p+\delta+\delta_1)(\sigma_2^{p+1}, 2p+\delta+\delta_2)\ldots(\sigma_n^{p+1}, 2p+\delta+\delta_n)(l_{p+1}, 2p+\delta+1)$ for some $\sigma_1^{p+1},\ldots,\sigma_n^{p+1} \in M \cup \{\#\}$.

We thus have proved that there exists some $w \in L(\mathscr{C},n)$ with $\max(w) = n$. □

Lemma 4.4 *For each $n \in \mathbb{N}$ and $w \in L(\mathscr{C},n)$ with $\max(w) = n$, there exists some error-free computation γ of $\mathscr{C}$ from (s_I,ε) to (s_F,x) for some $x \in M^*$ with $\max(\gamma) \leq n$.*

Proof Let $n \in \mathbb{N}$ and let $w \in L(\mathscr{C},n)$ such that $\max(w) = n$. Hence the number of symbols in $M \cup \{\#\}$ between every control state symbol and the following label symbol (or the symbol $\star$ if the state symbol is s_F) in w is constantly equal to n. This implies that (1) whenever a control state symbol s is followed by a label symbol $m?$ one time unit later, then the next symbol after s must be m, which will be replaced by a new hash symbol; (2) whenever a state symbol s is followed by a label symbol $m!$ one time unit later, then there must exist some hash symbol in between, and the first such hash symbol will be replaced by m; and (3) w does not contain any spontaneously inserted symbols. From (1) and (3) we can conclude that w encodes an error-free computation. From (2) we can conclude that the choice of n is big enough to capture the maximum length of the channel content. Hence there exists some error-free computation of $\mathscr{C}$ from (s_I,ε) to (s_F,x) for some $x \in M^*$ with $\max(\gamma) \leq n$. □

4.3 Excluding Faulty Computations

Next we define a parametric timed automaton $\mathscr{A}_{\mathscr{C}}$ over $\Sigma_{\mathscr{C}}$ such that $L(\mathscr{C},n) \cap L(\mathscr{A}_{\mathscr{C}})$ consists of all timed words that encode *error-free* computations of $\mathscr{C}$ from (s_I,ε) to (s_F,x) for some $x \in M^*$. The

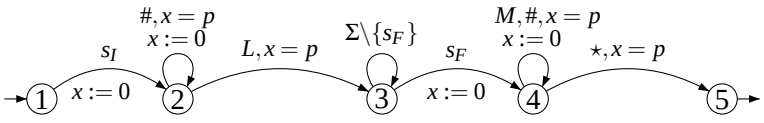

Figure 3: The parametric timed automaton $\mathscr{A}_{\mathscr{C}}$ that excludes insertion errors.

parametric timed automaton $\mathscr{A}_{\mathscr{C}}$ is shown in Fig. 3. It uses one clock x, parametrically constrained by a single parameter p. Note that $\mathscr{A}_{\mathscr{C}}$ is deterministic.

Theorem 4.5 *$\mathscr{C}$ has an error-free computation from (s_0,ε) to (s_F,x) for some $x \in M^*$, if, and only if, there exist $n \in \mathbb{N}$ and a parameter valuation π such that $L(\mathscr{C},n) \cap L_\pi(\mathscr{A}_{\mathscr{C}}) \neq \emptyset$.*

Proof For the direction from left to right, let $\gamma := \Pi_{1\leq i\leq k}\langle (s_{i-1},x_{i-1}), l_i, (s_i,x_i)\rangle$ be an error-free computation of $\mathscr{C}$ such that $s_0 = s_I$, $x_0 = \varepsilon$ and $s_k = s_F$. Define $n = \max(\gamma)$. Let $\delta \in \mathbb{R}_{\geq 0}$, and define $\delta_i = \frac{i}{(n+1)}$ for every $i \in \{1,\ldots,n\}$. By Lemma 4.3, there exists $w \in L(\mathscr{C},n)$ such that the prefix of w is of the form

$$(s_I,\delta)(\#,\delta+\delta_1)\ldots(\#,\delta+\delta_n)(l_1,\delta+1)$$

and $\max(w) = n$. This together with Lemma 4.2 implies that the suffix of w is of the form

$$(s_F,2k+\delta)(\sigma_1,2k+\delta+\delta_1)\ldots(\sigma_n,2k+\delta+\delta_n)(\star,2k+\delta+1)$$

for some $\sigma_1,\ldots,\sigma_n \in M \cup \{\#\}$. Note that in both the prefix and the suffix of w the time delay between every symbol is δ_1. Define $\pi(p) = \delta_1$. It is easy to see that $w \in L_\pi(\mathscr{A}_{\mathscr{C}})$. Hence $L(\mathscr{C},n) \cap L_\pi(\mathscr{A}_{\mathscr{C}}) \neq \emptyset$.

For the direction from right to left, assume there exist $n \in \mathbb{N}$ and a parameter valuation π such that $L(\mathscr{C},n) \cap L_\pi(\mathscr{A}_{\mathscr{C}}) \neq \emptyset$. Let $w \in L(\mathscr{C},n) \cap L_\pi(\mathscr{A}_{\mathscr{C}})$. By definition of $L(\mathscr{C},n)$, the prefix of w is of the form

$$(s_I,\delta)(\#,\delta+\delta_1)(\#,\delta+\delta_2)\ldots(\#,\delta+\delta_n)(l,\delta+1)$$

for some $\delta \in \mathbb{R}_{\geq 0}$, $0 < \delta_1 < \delta_2 < \cdots < \delta_n < 1$, and $l \in L$. The clock constraints at the loop in location 2 and at the edge from location 2 to 3 implies $\delta_i = \frac{i}{(n+1)}$ for every $i \in \{1,\ldots,n\}$ and $\pi(p) = \delta_1$. By Lemma 4.2, the suffix of w must be of the form

$$(s_F,N+\delta)(\sigma_n,N+\delta+\delta'_1)\ldots(\sigma_m,N+\delta+\delta'_m)(\star,N+\delta+1)$$

for some $N \in \mathbb{N}$, $0 < \delta'_1 < \delta'_2 < \cdots < \delta'_m < 1$ such that $n \leq m$, and there exists a strictly increasing function $f : \{1,\ldots,n\} \to \{1,\ldots,m\}$ such that $\delta_i = \delta'_{f(i)}$. Note that $\star$ occurs exactly one time unit after s_F. This, together with the clock constraints at the loop in location 4 and at the edge from 4 to the final location 5, implies $m = n$ (and $\delta'_i = \delta_i$ for every $i \in \{1,\ldots,n\}$). By Lemma 4.2, we further know that the number of symbols between a control state symbol and a symbol in $L \cup \{\star\}$ cannot *decrease*, and hence it follows that $\max(w) = n$. By Lemma 4.4, there exists an error-free computation of $\mathscr{C}$ from (s_0,ε) to (s_F,x) for some $x \in M^*$. □

4.4 The Reduction

We define $L(\mathscr{C}) = \cup_{n\in\mathbb{N}} L(\mathscr{C},n)$. Then we obtain

Corollary 4.6 *There exists an error-free computation of $\mathscr{C}$ from (s_I,ε) to (s_F,x) for some $x \in M^*$, if, and only if, there exists some parameter valuation π with $L_\pi(\mathscr{A}_{\mathscr{C}}) \cap L(\mathscr{C}) \neq \emptyset$.*

Next, we define the MTL formula $\varphi_{\mathscr{C}}$ such that $L(\varphi_{\mathscr{C}}) = L(\mathscr{C})$. The formula $\varphi_{\mathscr{C}}$ is the conjunction of a set of formulas, each of them expressing one of the conditions of $L(\mathscr{C})$. We start by defining some auxiliary formulas: $\bigvee S := \bigvee_{s\in S} s$, $\bigvee M := \bigvee_{m\in M} m$, $\bigvee L := \bigvee_{l\in L} l$, $\varphi_{\mathsf{copyM}} := \mathsf{G}_{(0,1)} \bigwedge_{m\in M}(m \to \mathsf{F}_{=2} m)$, and $\varphi_{\mathsf{copy\#}} := \mathsf{G}_{(0,1)}(\# \to \mathsf{F}_{=2}\#)$.

- $\mathsf{G}(\mathsf{X}_{>0}\mathtt{true} \vee \neg \mathsf{X}\mathtt{true})$ (Strict monotonicity)
- $\mathsf{G}\langle \bigwedge_{s\in S\setminus\{s_F\}}(s \to \bigvee_{(s,l,s)\in\Delta}(\mathsf{F}_{=1} l \wedge \mathsf{F}_{=2} s')) \wedge (s_F \to \mathsf{F}_{=1}\star)\rangle$,
  $\mathsf{G}\langle \bigvee S \to ((\mathsf{G}_{<2}\neg \bigvee S) \wedge (\mathsf{G}_{(0,1)\cup(1,2)} \neg \bigvee L)\rangle$ (Conditions on the occurrence of control state symbols and symbols in $L \cup \{\star\}$)
- $\mathsf{G}\langle \bigvee S \to (\mathsf{G}_{(0,1)}(\bigvee M \vee \#) \wedge \mathsf{G}_{[1,2)} \neg(\bigvee M \vee \#))\rangle$, $\mathsf{G}((\# \wedge \mathsf{X} \bigvee M) \to \mathtt{false})$ (Conditions on symbols in $M \cup \{\#\}$)
- $s_I \wedge \bigvee_{(s_I,l,s)\in\Delta}(\#\mathsf{U}(l \wedge \mathsf{X}s))$ (Encoding of the initial configuration)
- $\mathsf{F}s_F$ (Reaching s_F)
- $\mathsf{G} \bigwedge_{\substack{(s,\varepsilon,-)\in\Delta \\ s\neq s_F}} ((s \wedge \mathsf{F}_{=1}\varepsilon) \to ((\mathsf{G}_{(0,1)} \neg \bigvee M) \wedge \varphi_{\mathsf{copy\#}}))$
- $\mathsf{G} \bigwedge_{\substack{\delta=(s,m!,-)\in\Delta \\ s\neq s_F}} ((s \wedge \mathsf{F}_{=1} m!) \to (\varphi_{\mathsf{copyM}} \wedge \varphi_{next\#} \wedge \varphi_{yes\#} \wedge \varphi_{no\#}))$, where
  - $\varphi_{next\#} = \mathsf{X}\# \to (\mathsf{X}\mathsf{F}_{=2} m \wedge \mathsf{X}\varphi_{\mathsf{copy\#}})$
  - $\varphi_{yes\#} = (\mathsf{F}_{<1} \wedge \neg \mathsf{X}\#)\# \to \mathsf{G}_{<1}((\neg\# \wedge \mathsf{X}\#) \to \mathsf{X}\mathsf{F}_{=2} m \wedge \mathsf{X}\varphi_{\mathsf{copy\#}}))$
  - $\varphi_{no\#} = \neg \mathsf{F}_{<1}\# \to \mathsf{G}_{<1}(\mathsf{X}m! \to \mathsf{F}_{=2}(\mathsf{X}m \wedge \mathsf{X}\mathsf{X} \bigvee L))$
- $\mathsf{G} \bigwedge_{\substack{(s,m?,-)\in\Delta \\ s\neq s_F}} ((s \wedge \mathsf{F}_{=1} m?) \to (\varphi_{\mathsf{yesm}} \wedge \varphi_{\mathsf{nom}}))$, where
  - $\varphi_{\mathsf{yesm}} = \mathsf{X}m \to (\varphi_{\mathsf{shift}} \mathsf{U} m?)$, $\varphi_{\mathsf{shift}} = \bigwedge_{m\in M}(\mathsf{X}m \to \mathsf{F}_{=2} m) \wedge (\mathsf{X}\# \to \mathsf{F}_{=2}\#) \wedge (\mathsf{X}m? \to \mathsf{F}_{=2}\#)$
  - $\varphi_{\mathsf{nom}} = \mathsf{X}\neg m \to (\varphi_{\mathsf{copyM}} \wedge \varphi_{\mathsf{copy\#}} \wedge \mathsf{G}_{<1}(\mathsf{X}m? \to \mathsf{F}_{=2}(\mathsf{X}\# \wedge \mathsf{X}\mathsf{X} \bigvee L)))$

Proof of Theorem 4.1 Let $\mathscr{C} = (S, s_0, M, \Delta)$ be a channel machine, let $s_F \in S$. Define the parametric timed automaton $\mathscr{A}_{\mathscr{C}}$ and the MTL formula $\varphi_{\mathscr{C}}$ as above. By Corollary 4.6 we know that there is an error-free computation from (s_I, ε) to (s_F, x) for some $x \in M^*$, if, and only if, there exists some parameter valuation π with $L_\pi(\mathscr{A}_{\mathscr{C}}) \cap L(\varphi_{\mathscr{C}}) \neq \emptyset$. The latter, however, is equivalent to $L_\pi(\mathscr{A}_{\mathscr{C}}) \not\subseteq L(\neg\varphi_{\mathscr{C}})$, *i.e.*, there exists some timed word $w \in L_\pi(\mathscr{A}_{\mathscr{C}})$ such that $w \not\models \neg\varphi_{\mathscr{C}}$. Hence, the MTL-model checking problem for parametric timed automata is undecidable. □

5 Discussion

For our undecidability result we construct a parametric timed automaton using a parametric *equality* constraint of the form $x = p$. Parametric equality constraints seem to be a source of undecidability; they occur in the undecidability proofs of, *eg.*, the emptiness problem for parametric timed automata with three clocks [6], and the satisfiability problem for a parametric extension of LTL [4]. A natural question is thus to consider the MTL-model checking problem for L/U-automata [15], a subclass of parametric timed automata in which parameters are only allowed to occur either as a lower bound or as an upper bound, but not both, and for which the emptiness problem is decidable independent of the number of clocks. We further remark that the proof does not work if we restrict the parameter valuation to be a function mapping each parameter to a non-negative integer.

Acknowledgements I would like to thank James Worrell for pointing me to MTL's capability to encode computations of *Turing machines with insertion errors*, explained in [18].

References

[1] Parosh Aziz Abdulla, Johann Deneux, Joël Ouaknine, Karin Quaas & James Worrell (2008): *Universality Analysis for One-Clock Timed Automata*. *Fundam. Inform.* 89(4), pp. 419–450. Available at `http://iospress.metapress.com/content/xx63231v71037607/`.

[2] Luca Aceto & Anna Ingólfsdóttir, editors (2006): *Foundations of Software Science and Computation Structures, 9th International Conference, FOSSACS 2006, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2006, Vienna, Austria, March 25-31, 2006, Proceedings*. *Lecture Notes in Computer Science* 3921, Springer.

[3] Rajeev Alur & David L. Dill (1994): *A Theory of Timed automata*. *Theor. Comput. Sci.* 126(2), pp. 183–235. Available at `http://dx.doi.org/10.1016/0304-3975(94)90010-8`.

[4] Rajeev Alur, Kousha Etessami, Salvatore La Torre & Doron Peled (2001): *Parametric temporal logic for "model measuring"*. *ACM Trans. Comput. Log.* 2(3), pp. 388–407. Available at `http://doi.acm.org/10.1145/377978.377990`.

[5] Rajeev Alur, Tomás Feder & Thomas A. Henzinger (1996): *The Benefits of Relaxing Punctuality*. *J. ACM* 43(1), pp. 116–146. Available at `http://doi.acm.org/10.1145/227595.227602`.

[6] Rajeev Alur, Thomas A. Henzinger & Moshe Y. Vardi (1993): *Parametric real-time reasoning*. In Kosaraju et al. [16], pp. 592–601. Available at `http://doi.acm.org/10.1145/167088.167242`.

[7] Roberto M. Amadio, editor (2008): *Foundations of Software Science and Computational Structures, 11th International Conference, FOSSACS 2008, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2008, Budapest, Hungary, March 29 - April 6, 2008. Proceedings*. *Lecture Notes in Computer Science* 4962, Springer.

[8] Laura Bozzelli & Salvatore La Torre (2009): *Decision problems for lower/upper bound parametric timed automata*. *Formal Methods in System Design* 35(2), pp. 121–151. Available at `http://dx.doi.org/10.1007/s10703-009-0074-0`.

[9] Daniel Brand & Pitro Zafiropulo (1983): *On Communicating Finite-State Machines*. *J. ACM* 30(2), pp. 323–342, Available at `http://doi.acm.org/10.1145/322374.322380`.

[10] Adrian Horia Dediu, Henning Fernau & Carlos Martín-Vide, editors (2010): *Language and Automata Theory and Applications, 4th International Conference, LATA 2010, Trier, Germany, May 24-28, 2010. Proceedings*. *Lecture Notes in Computer Science* 6031, Springer. Available at `http://dx.doi.org/10.1007/978-3-642-13089-2`.

[11] Stéphane Demri & Ranko Lazić (2009): *LTL with the freeze quantifier and register automata*. *ACM Trans. Comput. Log.* 10(3). Available at `http://doi.acm.org/10.1145/1507244.1507246`.

[12] Stéphane Demri, Ranko Lazić & Arnaud Sangnier (2008): *Model Checking Freeze LTL over One-Counter Automata*. In Amadio [7], pp. 490–504. Available at `http://dx.doi.org/10.1007/978-3-540-78499-9_34`.

[13] Barbara Di Giampaolo, Salvatore La Torre & Margherita Napoli (2010): *Parametric Metric Interval Temporal Logic*. In Dediu et al. [10], pp. 249–260. Available at `http://dx.doi.org/10.1007/978-3-642-13089-2_21`.

[14] Thomas Henzinger (1991): *The temporal specification and verification of real-time systems.* Ph.D. thesis, Stanford University. Technical Report STAN-CS-91-1380.

[15] Thomas Hune, Judi Romijn, Mariëlle Stoelinga & Frits W. Vaandrager (2002): *Linear parametric model checking of timed automata*. *J. Log. Algebr. Program.* 52-53, pp. 183–220. Available at `http://dx.doi.org/10.1016/S1567-8326(02)00037-1`.

[16] S. Rao Kosaraju, David S. Johnson & Alok Aggarwal, editors (1993): *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing, May 16-18, 1993, San Diego, CA, USA*. ACM.

[17] Ron Koymans (1990): *Specifying Real-Time Properties with Metric Temporal Logic*. *Real-Time Systems* 2(4), pp. 255–299. Available at `http://dx.doi.org/10.1007/BF01995674`.

[18] Joël Ouaknine & James Worrell (2006): *On Metric Temporal Logic and Faulty Turing Machines*. In Aceto & Ingólfsdóttir [2], pp. 217–230. Available at `http://dx.doi.org/10.1007/11690634_15`.

[19] Joël Ouaknine & James Worrell (2007): *On the decidability and complexity of Metric Temporal Logic over finite words*. *Logical Methods in Computer Science* 3(1). Available at `http://dx.doi.org/10.2168/LMCS-3(1:8)2007`.