Counter-free input-determined timed automata *

Fabrice Chevalier Deepak D'Souza Pavithra Prabhakar

IISc-CSA-TR-2007-1
http://archive.csa.iisc.ernet.in/TR/2007/1/

Computer Science and Automation
Indian Institute of Science, India

June 2007

*This work was partly supported by P2R Timed-DISCOVERI.

Counter-free input-determined timed automata
*

Fabrice Chevalier[†] Deepak D'Souza[‡] Pavithra Prabhakar[§]

Abstract

We identify a class of timed automata, which we call counter-free input-determined automata, which characterize the class of timed languages definable by several timed temporal logics in the literature, including MTL [1, 11]. We make use of this characterization to show that MTL satisfies an "ultimate stability" property with respect to periodic sequences of timed words. Our results hold for both the pointwise and continuous semantics.

1 Introduction

A number of classes of timed automata based on "input-determined" distance operators have been proposed in the literature. These include the event-recording automata of [2, 7], which make use of the operators $\lhd_a$ and $\rhd_a$ (which measure the distance to the last and next a's respectively), state-clock automata [13], and eventual timed automata [5, 3], which make use of the "eventual operator" $\Diamond_a$ inspired by Metric Temporal Logic (MTL). In [6, 3] these operators were abstracted into a general notion of an input-determined operator, and the corresponding classes of timed automata called input-determined automata or IDA's (parameterized by a set of input-determined operators) were shown to have robust logical properties, including a monadic second-order logic characterization, and expressively complete (with respect to the first-order fragment of the corresponding MSO logics) timed temporal logics based on these operators. However, an important link that remained unexplored was a characterization of the class of automata which correspond to these temporal logics, along the lines of the classical characterization via

*This work was partly supported by P2R Timed-DISCOVERI.
[†]LSV, ENS de Cachan, France fabrice.chevalier@lsv.ens-cachan.fr
[‡]CSA, IISc, Bangalore, India deepakd@csa.iisc.ernet.in
[§]Dept. of Comp. Sci., UIUC, USA pprabha2@uiuc.edu

counter-free automata for discrete temporal logic, which follows from the work of Kamp and McNaughton-Papert [9, 10].

In this paper our aim is to fill this gap. We identify a class of counter-free IDA's (again parameterized by a set of input-determined operators) that precisely characterize the class of timed languages definable by the timed temporal logics based on these operators. Our class of counter-free IDA's comprise "proper" IDA's (which are in a sense a determinized form of IDA's) whose underlying graph have no counters in the classical sense. Our results hold for both the "pointwise" and "continuous" semantics for timed formalisms. The technique we use factors through the classical results of Kamp and McNaughton-Papert, as well as some of the translations set up in [6, 3].

We emphasize that this general result gives us automata characterizations for several of the timed logics proposed in the literature, including EventClockTL [7], Metric Interval Temporal Logic (MITL) [1], and MTL [1, 11], for both the pointwise and continuous semantics. Among other applications, such characterizations can be useful in arguing expressiveness results for these logics.

In the final part of this paper we make use of this characterization to prove a property of timed languages definable by MTL, namely that they must satisfy an "ultimate stability" property with respect to a periodic sequence of timed words. Thus, given a periodic sequence of finite timed words of the form uv^iw, the truth of an MTL formula at any real time point, must eventually stabilize (i.e. become always true, or always false) along the models in the sequence. This is a stronger result than a property for MTL proved in [12], in that it holds for MTL with past operators, and furthermore does not depend on the "duration" of the timed word v in the periodic sequence.

2 Preliminaries

For an alphabet A, we use A^* to denote the set of finite words over A. For a word w in A^*, we use $|w|$ to denote its length. We denote the set of non-negative real numbers by $\mathbb{R}_{\geq 0}$. We use $\mathcal{I}_{\mathbb{R}_{\geq 0}}$ to denote the set of intervals, where an interval is a convex subset of $\mathbb{R}_{\geq 0}$. Intervals I and J are *adjacent* if $I \cap J = \emptyset$ and $I \cup J$ is an interval. We use $\mathcal{I}_{\mathbb{Q}}$ to denote the set of intervals whose end-points are rational or ∞.

Let A be an alphabet and let $f : [0, r] \to A$ be a function, where $r \in \mathbb{R}_{\geq 0}$. We use $dur(f)$ to denote the duration of f, in this case r. We call f a *finitely varying* function over A, if there exist a word $a_0a_1 \cdots a_{2n}$ in A^*, and an interval sequence $I_0I_1 \cdots I_{2n}$, such that $0 \in I_0$, I_i and I_{i+1} are adjacent for each i, I_i is singular if i is even, and for all $t \in [0, r]$, $f(t) = a_i$ if $t \in I_i$. We then call $(a_0, I_0) \cdots (a_{2n}, I_{2n})$ an *interval representation* of f. We call a word $a_0a_1 \cdots a_n$

in A^* *canonical*, if n is even, and there does not exist an even i such that $0 < i < n$ and $a_{i-1} = a_i = a_{i+1}$. An interval representation $(b_0, I_0) \cdots (b_{2n}, I_{2n})$ of f is called *canonical*, if $b_0 \cdots b_{2n}$ is canonical. Note that every finitely varying function has a unique canonical interval representation. We define $FVF(A)$ to be the set of all finitely varying functions over A.

Let $f \in FVF(A)$ and let $(a_0, I_0) \cdots (a_{2n}, I_{2n})$ be its canonical interval representation. We denote the untiming of the function as a sequence which captures explicitly the points of discontinuities and the intervals between them. The untiming of the above f, denoted $untiming(f)$, is defined to be $a_0 \cdots a_{2n}$. Note that the untiming of a function is always canonical. Given a word w in A^*, we define its timing to be a set of functions: $timing(w) = \emptyset$ if $|w|$ is even, otherwise $f \in timing(w)$ if $w = a_0 a_1 \cdots a_{2n}$ and $(a_0, I_0)(a_1, I_1) \cdots (a_{2n}, I_{2n})$ is an interval representation of f. We can extend the definitions of $timing$ and $untiming$ to languages of functions in the expected way.

3 Classical counter-free automata

Recall that a (non-deterministic) finite state automaton (NFA) over an alphabet A is a structure $\mathcal{A} = (Q, s, \delta, F)$, where Q is a finite set of states, s is the initial state, $\delta \subseteq Q \times A \times Q$ is the transition relation, and $F \subseteq Q$ is the set of final states. A run of $\mathcal{A}$ on a word $w = a_0 \cdots a_n \in A^*$ is a sequence of states $q_0, \ldots, q_{n+1}$ such that $q_0 = s$, and $(q_i, a_i, q_{i+1}) \in \delta$ for each $i \leq n$. The run is accepting if $q_{n+1} \in F$. The symbolic language accepted by $\mathcal{A}$, denoted $L_{sym}(\mathcal{A})$, is the set of words in A^* over which $\mathcal{A}$ has an accepting run. Languages accepted by NFA's are called *regular* languages. We say the NFA $\mathcal{A}$ is *deterministic* (and call it a DFA) if the transition relation δ is a function from $Q \times A$ to Q. A well-known fact is that every regular language L has a unique (up to isomorphism) minimal state DFA accepting it, which we refer to as $\mathcal{A}_L$.

A *counter* in an NFA $\mathcal{A}$ is a sequence of distinct states $q_0, \ldots, q_n$ with $n \geq 1$, along with a word $u \in A^*$, such that there is a path labeled u in $\mathcal{A}$ from q_i to q_{i+1} (for each $i \in \{0, \ldots, n-1\}$) and from q_n to q_0. An automaton is said to be *counter-free* (*CFA*) if it does not contain a counter. A regular language is said to be *counter-free* if there exists a counter-free automaton for it.

Proposition 3.1 *Let L be a regular language over an alphabet A. Then the following are equivalent:*

1. *L is counter-free.*

2. *$\mathcal{A}_L$ is counter-free.*

3. There do not exist strings u, v, w in A^*, such that for infinitely many i we have $uv^iw \in L$, and for infinitely many i we have $uv^iw \notin L$.

Proof We show that $(2) \Rightarrow (1) \Rightarrow (3) \Rightarrow (2)$.

$(2) \Rightarrow (1)$ is obvious.

$(1) \Rightarrow (3)$: Let $\mathcal{A}$ be a counter-free automaton recognizing L. Suppose by contradiction that there exist strings u, v, w in A^*, such that for infinitely many i we have $uv^iw \in L$, and for infinitely many i we have $uv^iw \notin L$.

Let n be the number of states of $\mathcal{A}$. There exists $i_0 \geq n$ such that $uv^{i_0}w \in L$. Consider a run of $\mathcal{A}$ on $uv^{i_0}w$. If we consider the states reached after reading each v, we find that there exists a state q such that there is a path from q to q on reading v^i for some $i > 0$. But since $\mathcal{A}$ has no counters there is a path from q to q on v itself. Hence $uv^iw \in L$ for all $i \geq i_0$ as the word v can be pumped using the path from q to q. Contradiction.

$(3) \Rightarrow (2)$: We show the contrapositive. Suppose $\mathcal{A}_L = (Q, s, \delta, F)$ has a counter. So there exist $u \in A^*$ and distinct states $q_0, \cdots, q_m \in Q$, where $m \geq 1$, such that there is a path from q_i on v to q_{i+1} for all $0 \leq i < m$ and from q_m to q_0. Since $\mathcal{A}_L$ is minimal, every state is reachable, in particular q_0. Let u be a word such that there is a path from s to q_0 on u. Further since q_0 is not the same as q_1, there exists w such that q_0 on w goes to a final state and q_1 on w does not. Hence $uv^{km+1}w$ is in L and $uv^{km+2}w$ is not in L for every $k \in \mathbb{N}$. $\square$

Using the above proposition, it follows that counter-free regular languages are closed under the boolean operations of union, intersection and complement.

While defining our continuous timed automata it will be useful to use a variant of NFA's which we call *state-transition-labeled automata* or ST-NFA$'s$ for short. Let A be an alphabet partitioned into A_1 and A_2. An ST-NFA over (A_1, A_2) is a structure $\mathcal{A} = (Q, s, \delta, F, l)$ similar to an NFA over A, except that $\delta \subseteq Q \times A_1 \times Q$, and $l : Q \longrightarrow A_2$ is a function that labels states with letters from A_2. The ST-NFA $\mathcal{A}$ accepts strings of the form $A_1 \cdot (A_2 \cdot A_1)^*$. A run of $\mathcal{A}$ on a string $w = a_0a_1 \cdots a_{2n}$ in $A_1 \cdot (A_2 \cdot A_1)^*$, is a sequence of states $q_0, \ldots, q_n$ satisfying $q_0 = s$, $(q_i, a_{2i}, q_{i+1}) \in \delta$ and $l(q_i) = a_{2i-1}$ for each $i \in \{1, \ldots, n-1\}$; it is accepting if $q_n \in F$. We define $L_{sym}(\mathcal{A})$ to be the set of strings $w \in A^*$ on which $\mathcal{A}$ has an accepting run.

The class of languages accepted by ST-NFA's over (A_1, A_2) is precisely the class of regular languages which are subsets of $A_1 \cdot (A_2 \cdot A_1)^*$. To show this, given an ST-NFA $\mathcal{A} = (Q, s, \delta, F, l)$ over (A_1, A_2) we can construct an NFA $\mathcal{A}'$ over A from $\mathcal{A}$ as follows. For every state $q \in Q$ we add a new state q', and a new transition $(q, l(q), q')$. All outgoing transitions from q are transferred to q'. We add a new start state t with the same outgoing transitions as s. Final states of $\mathcal{A}'$ are the same as of $\mathcal{A}$.

Conversely, given an NFA $\mathcal{A} = (Q, s, \delta, F)$ over A accepting a subset of $A_1 \cdot (A_2 \cdot A_1)^*$, we can construct a language equivalent ST-NFA $\mathcal{A}'$ over (A_1, A_2) as follows. The states of $\mathcal{A}'$ are the transitions in $\mathcal{A}$ which are in $Q \times A_2 \times Q$. There is a transition from a state (p, a, q) to a state (r, c, t) on b if (q, b, r) is a transition in $\mathcal{A}$. The final states are states of the form (p, a, q) where $p \in F$. There is a new start state s', which has transitions of the form $(s', b, (p, c, q))$ whenever (s, b, p) is a transition in $\mathcal{A}$. Every state (p, a, q) is labelled by a and s' is labelled by some arbitrary symbol in the alphabet. It is not hard to convince oneself that the symbolic language of $\mathcal{A}'$ is the same as that of $\mathcal{A}$.

We will now define the counter-free counterpart of the ST-NFA. A counter in an ST-NFA is similar to one in an NFA, except that by the "label" of a path in the automaton we mean the sequence of alternating state and transition labels along the path. Thus we count the label of the first state in the path, but ignore the label of the last state in the path.

Using the fact that the constructions above do not introduce a counter if there was not one already, it follows that:

Proposition 3.2 *A regular subset of* $A_1 \cdot (A_2 \cdot A_1)^*$ *is counter-free iff there exists a counter-free* ST-NFA *over* (A_1, A_2) *accepting it.*

4 Counter-free continuous input-determined automata

In this section we introduce the continuous input-determined automata and their counter-free version. Below are some notations which will help us in formalizing the definitions of these automata.

We define a timed word σ over an alphabet Σ to be an element of $(\Sigma \times \mathbb{R}_{\geq 0})^*$, such that $\sigma = (a_0, t_0)(a_1, t_1) \cdots (a_n, t_n)$ and $t_0 < t_1 < \cdots < t_n$. Given timed words $\sigma = (a_0, t_0)(a_1, t_1) \cdots (a_n, t_n)$ and $\sigma' = (a'_0, t'_0)(a'_1, t'_1) \cdots (a'_k, t'_k)$, we define the concatenation of the timed words $\sigma \cdot \sigma'$ in the classical way to be $(a_0, t_0)(a_1, t_1) \cdots (a_n, t_n)(a'_0, t_n + t'_0)(a'_1, t_n + t'_1) \cdots (a'_k, t_n + t'_k)$.

We define an *input-determined operator* Δ over an alphabet Σ as a partial function from $(T\Sigma^* \times \mathbb{R}_{\geq 0})$ to $2^{\mathbb{R}_{\geq 0}}$, which is defined for all pairs (σ, t), where $t \in [0, dur(\sigma)]$. Thus an input-determined operator identifies a set of "distances" for a given timed word and a time point in it. Given a set of input-determined operators Op, we define the set of guards over Op, denoted by $\mathcal{G}(Op)$, inductively as $g ::= \top \mid \Delta^I \mid \neg g \mid g \vee g \mid g \wedge g$, where $\Delta \in Op$ and $I \in \mathcal{I}_{\mathbb{Q}}$. Guards of the form Δ^I are called atomic. Given a timed word σ, we define the satisfiability of a guard g at time $t \in [0, dur(\sigma)]$, denoted $\sigma, t \models g$, as follows: $\sigma, t \models \Delta^I$ iff $\Delta(\sigma, t) \cap I \neq \emptyset$, with boolean operators treated in the expected way.

Example 4.1 *$\Delta_{\mathbb{Q}}$, which maps (σ, t) to $\{1\}$ if t is rational and to $\{0\}$ otherwise, is an input-determined operator. Other examples include the eventual operator $\Diamond_a$, inspired by* MTL, *which maps (σ, t) to the set of distances d such that an a occurs at time $t + d$ in σ; and the event-recording operator $\lhd_a$ which maps (σ, t) to the (empty or singleton) set of distances to the last occurrence of the event a before time t.*

We call an input-determined operator Δ over Σ *finitely varying* if for each $I \in \mathcal{I}_{\mathbb{Q}}$, and each $\sigma \in T\Sigma^*$, the characteristic function $f_{\Delta^I} : [0, dur(\sigma)] \to \{0, 1\}$ of Δ^I, defined as $f_{\Delta^I}(t)$ is 1 if $\sigma, t \models \Delta^I$, and 0 otherwise, is finitely varying. The operators $\Diamond_a$ and $\lhd_a$ are finitely varying, while $\Delta_{\mathbb{Q}}$ is not.

Let Σ be an alphabet and Op be a set of input determined operators over Σ. We call (Γ_1, Γ_2) a *symbolic alphabet* over (Σ, Op), if Γ_1 is a finite subset of $(\Sigma \cup \{\epsilon\}) \times \mathcal{G}(Op)$ and Γ_2 is a finite subset of $\mathcal{G}(Op)$. We define the set of timed words over Σ associated with a function f in $FVF(\Gamma_1 \cup \Gamma_2)$, denoted $tw(f)$, as follows. If $untiming(f) \notin \Gamma_1 \cdot (\Gamma_2 \cdot \Gamma_1)^*$, then $tw(f) = \emptyset$. Otherwise, a timed word $\sigma = (a_0, t_0) \cdots (a_n, t_n)$ is in $tw(f)$, provided for all $t \in [0, dur(f)]$,

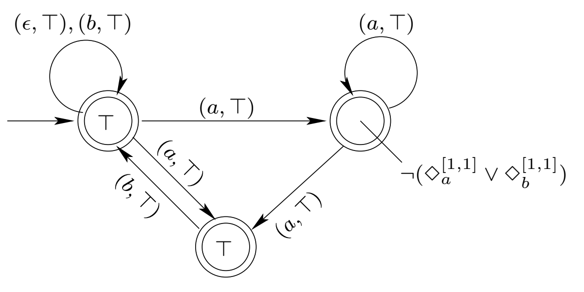
- If $f(t) = (a, g)$, for some $a \in \Sigma$ and $g \in \mathcal{G}(Op)$, then $\sigma, t \models g$, and there exists i in $\{0, \cdots, n\}$, with $t_i = t$ and $a_i = a$.

- If $f(t) = (\epsilon, g)$ or g, for some $g \in \mathcal{G}(Op)$, then $\sigma, t \models g$, and there does not exist i in $\{0, \cdots, n\}$ with $t_i = t$.

Note that for any f, $tw(f)$ is either empty or a singleton set. We extend the definition of tw to sets of functions, as the union of the timed words corresponding to each function in the set.

Let Σ be an alphabet and Op be a set of input-determined operators based on Σ. A *Continuous Input Determined Automaton* (*CIDA*) $\mathcal{A}$ over (Σ, Op) is simply an ST-NFA over a symbolic alphabet (Γ_1, Γ_2) based on (Σ, Op). As an ST-NFA $\mathcal{A}$ accepts a symbolic language $L_{sym}(\mathcal{A})$. We can also define the language of functions accepted by $\mathcal{A}$, denoted $F(\mathcal{A})$, to be $timing(L_{sym}(\mathcal{A}))$. We are however most interested in the timed language it accepts, denoted $L(\mathcal{A})$, and defined to be $tw(F(\mathcal{A}))$.

Example 4.2 *We give below a concrete example of a CIDA over the set of "eventual operators" $Op = \{\Diamond_a \mid a \in \Sigma\}$. The diagram below shows a CIDA over $(\{a, b\}, Op)$ which recognizes the language L_{ni} (for "no insertions"), which consists of timed words in which between any two consecutive a's, there does not exist a time point from which there is an a or a b at a distance of one time unit in the future.*



We now define *proper CIDA*'s which are a time-deterministic form of *CIDA*'s, and which we will use to define our counter-free *CIDA*'s. Let G be a finite set of atomic guards over Op. We call (Γ_1, Γ_2) the *proper* symbolic alphabet over (Σ, Op) based on G, if $\Gamma_1 = (\Sigma \cup \{\epsilon\}) \times 2^G$ and $\Gamma_2 = 2^G$. We interpret $h \subseteq G$ as a guard which specifies precisely the guards in h that are true. Thus h is interpreted as the guard $\bigwedge_{g \in h} g \wedge \bigwedge_{g \in G-h} \neg g$.

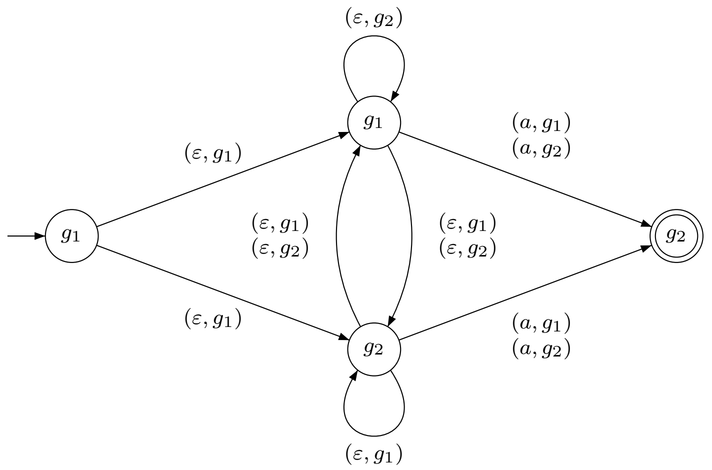
Example 4.3 *Let $\Sigma = \{a\}$, $Op = \{\Diamond_a\}$ and $G = \{\Diamond_a^{[1,1]}\}$. The proper alphabet (Γ_1, Γ_2) is given by $\Gamma_1 = (\Sigma \cup \{\epsilon\}) \times 2^G$ and $\Gamma_2 = 2^G$. Let $f_1 : [0, 2] \to \Gamma_1 \cup \Gamma_2$ such that $f_1(0) = f_1(2) = (a, \emptyset)$, $f_1(1) = (\epsilon, \{\Diamond_a^{[1,1]}\})$ and $f(t) = \emptyset$ if $t \neq 0, 1, 2$. We then have $tw(f_1) = \{(a, 0)(a, 2)\}$. Let $f_2 : [0, 2] \to \Gamma_1 \cup \Gamma_2$ defined by $f_2(1) = (\epsilon, \emptyset)$ and $f_2(t) = f_1(t)$ if $t \neq 1$. Then $tw(f_2) = \emptyset$.*

We define a *proper CIDA* over (Σ, Op) to be a ST-NFA over a proper symbolic alphabet based on (Σ, Op). The symbolic, function, and timed languages defined by a proper *CIDA* are defined similarly to *CIDA*'s. We call a word γ over a symbolic alphabet (Γ_1, Γ_2) *fully canonical*, if $\gamma \in \Gamma_1 \cdot (\Gamma_2 \cdot \Gamma_1)^*$ and no subword of γ is of the form $g \cdot (\epsilon, g) \cdot g$. We call a proper *CIDA fully canonical* if its symbolic language consists of fully canonical proper words. The class of languages defined by *CIDA*'s and fully canonical proper *CIDA*'s coincide:

Lemma 4.1 ([3]) *CIDA's over (Σ, Op) and fully canonical proper CIDA's over (Σ, Op) define the same class of timed languages.*

We define the class of counter-free *CIDA* languages over (Σ, Op), denoted *CFCIDA*(Σ, Op), to be the class of timed languages definable by fully canonical proper *CIDA*'s over (Σ, Op) whose underlying ST-NFA is counter-free.

Example 4.4 *Let $\Sigma = \{a\}$, $Op = \{\Diamond_a\}$ and $G = \{\Diamond_a^{[1,1]}\}$. The following CFCIDA over (Σ, Op) recognize timed words which have exactly one a in the interval $[1, 2]$. In the diagram, $g_1 = \{\Diamond_a^{[1,2]}\}$ and $g_2 = \emptyset$.*



5 A logical characterization of *CFCIDA* languages

In this section we give a logical characterization of *CFCIDA* in terms of a first-order logic parameterized by a set of input-determined guards. We first relate the first-order fragment of Büchi's monadic second order logic (MSO) and counter-free automata, which will then be used in giving a logical characterization of *CFCIDA*.

For an alphabet A, the syntax of the first order fragment of Büchi's MSO logic, denoted here by $\mathrm{FO}^c(A)$, is given by:

$$\varphi ::= Q_a(x) \,|\, x < y \,|\, \neg\varphi \,|\, (\varphi \vee \varphi) \,|\, \exists x\varphi,$$

where $a \in A$, and x is a first order variable.

We interpret a formula φ of the logic over a finitely varying function f in $FVF(A)$, along with an interpretation $\mathbb{I}$ with respect to f, which assigns to a first order variable x, a value in $[0, dur(f)]$. For an interpretation $\mathbb{I}$, we use the notation $\mathbb{I}[t/x]$ to denote the interpretation which sends x to t and agrees with $\mathbb{I}$ on all other variables.

We now define the semantics of $\mathrm{FO}^c(A)$. Given a formula $\varphi \in \mathrm{FO}^c(A)$, $f \in FVF(A)$ and an interpretation $\mathbb{I}$ with respect to f to the variables in φ, the satisfaction relation $f, \mathbb{I} \models \varphi$, is defined inductively (with boolean operators handled in the usual way) as:

$$\begin{array}{lll} f, \mathbb{I} \models \mathrm{Q}_a(x) & \text{iff} & f(\mathbb{I}(x)) = a, \text{ where } a \in A. \\ f, \mathbb{I} \models x < y & \text{iff} & \mathbb{I}(x) < \mathbb{I}(y). \\ f, \mathbb{I} \models \exists x\varphi & \text{iff} & \exists t \in [0, dur(f)] : \, f, \mathbb{I}[t/x] \models \varphi. \end{array}$$

For a sentence, a formula without free variables, the interpretation does not play any role. Hence, for a sentence φ in $\mathrm{FO}^c(A)$, we set the language defined by φ to be $F(\varphi) = \{f \in FVF(A) \,|\, f \models \varphi\}$.

We relate the class of languages defined by FO^c formulas and counter-free regular languages below. Let A_1 and A_2 be two disjoint alphabets and let $A = A_1 \cup A_2$. Let φ be a sentence in $\mathrm{FO}^c(A)$. Let $alt\text{-}FVF(A) = \{f \in FVF(A) \mid untiming(f) \in A_1 \cdot (A_2 \cdot A_1)^*\}$. We construct a *CFA* A_φ such that for all $f \in alt\text{-}FVF(A)$, $f \in F(\varphi)$ iff $untiming(f) \in A_\varphi$.

We will represent models for formulas with free variables in them, as functions with the interpretations built into them. We assume an ordering on the countable set of first-order variables given by $x_1, x_2, \cdots$. For a formula φ with free variables among $X = \{x_{i_1}, \cdots, x_{i_m}\}$ (in order), we represent a function f and an interpretation $\mathbb{I}$ as a function $f_{\mathbb{I}}^X : [0, dur(f)] \to A \times \{0,1\}^m$ given by $f_{\mathbb{I}}^X(t) = (f(t), b_1, \cdots, b_m)$, where $b_k = 1$ iff $\mathbb{I}(x_{i_k}) = t$. Thus for a formula φ with free variables in X we have a notion of X-models of φ. Further we call $f_{\mathbb{I}}^X$ an *alternating* X-model of φ, if $f \in alt\text{-}FVF(A)$.

Proposition 5.1 *Let φ be a $\mathrm{FO}^c(A)$ formula with free variables in $X = \{x_{i_1}, \cdots, x_{i_m}\}$. Let X' be a finite set of first order variables which is a superset of X. Suppose that we have a CFA $\mathcal{A}$ accepting the untiming of the alternating X-models of φ. Then we can construct a CFA $\mathcal{A}'$ accepting precisely the untiming of alternating X'-models of φ.*

Proof Let us assume that $X' = X \cup \{x_{i_{m+1}}\}$, where $i_{m+1} > i_m$. The transitions in $\mathcal{A}$ are labeled with symbols from $A \times \{0,1\}^m$. We replace each of these labels by a symbol from $A \times \{0,1\}^{m+1}$ such that its last component is a 0 and by projecting away the last component we get back the earlier label on the transition. Let us call the resulting automaton $\mathcal{A}_1$.

We note that an X'-model is an X-model extended with a 1 at exactly one time point. So we construct the automaton $\mathcal{A}_2$ by making two copies of $\mathcal{A}_1$ and non-deterministically guessing a transition from the first copy to the second corresponding to the 1. If $(q, (a, b_1, \cdots, b_m, 0), q')$ is a transition in $\mathcal{A}_1$ then we have $((q, 0), (a, b_1, \cdots, b_m, 1), (q', 1))$ as a transition in $\mathcal{A}_2$, when either a belongs to A_1 or one of the b_i's is a 1. This corresponds to the last 1 being at a time which is a point of discontinuity in the alternating X-model. When the 1 happens to be at a place of continuity in the X-model, it creates a new point of discontinuity, in which case we do the following. When a belongs to A_2 and all the b_i's are 0's, we break the transition into three parts, we go from $(q, 0)$ on $(a, b_1, \cdots, b_m, 0)$ to a new state q_1, from q_1 on $(a, b_1, \cdots, b_m, 1)$ to another new state q_2 and from q_2 on $(a, b_1, \cdots, b_m, 0)$ to $(q', 1)$.

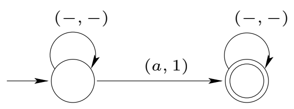
We can now generalize the construction for any X' such that $X \subseteq X'$.

$\square$

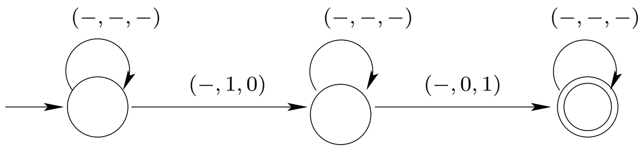
Lemma 5.1 *Let $\varphi \in \mathrm{FO}^c(A)$ with X being the set of free variables in it. We can give a CFA $\mathcal{A}_\varphi$ such that for all $f \in$ alt-$FVF(A)$ and $\mathbb{I}$ an interpretation to X, $f, \mathbb{I} \models \varphi$ iff $untiming(f_{\mathbb{I}}^X) \in L_{sym}(\mathcal{A}_\varphi)$.*

Proof The idea of the proof is similar to the one in [3], except that now we need to also ensure that the automaton we obtain is counter-free. We denote by $\mathcal{A}_\varphi^Y$, where Y contains the free variables in φ, the automaton which generates the language conisting of untiming of alternating Y-models of φ. Let $\mathcal{A}_{valid}^Y$ denote the automaton which consists of the untiming of all alternating Y-models. We can give a counter-free automaton $\mathcal{A}_{valid}^Y$, since we can give a *CFA* for the untiming of alt-$FVF(A)$ and then by Proposition 5.1 extend it to the untiming of the alternating Y-models. We will construct a counter-free automaton $\mathcal{A}_\varphi^X$ by induction on the structure of φ.

1. $\varphi = Q_a(x)$: The automaton $\mathcal{A}_\varphi^{\{x\}}$ is the intersection of $\mathcal{A}_{valid}^{\{x\}}$ with:



2. $\varphi = x < y$: The automaton $\mathcal{A}_\varphi^{\{x,y\}}$ (assuming x occurs before y in the variable ordering) is the intersection of $\mathcal{A}_{valid}^{\{x,y\}}$ with:



3. $\varphi = \neg\eta$: Let $\mathcal{A}_\eta^X$ be the automaton for η, where X consists of the free variables in η. Then $\mathcal{A}_\varphi^X$ is the intersection of $\mathcal{A}_{valid}^X$ with complement of $\mathcal{A}_\eta^X$.

4. $\varphi = \eta \vee \psi$: Let $\mathcal{A}_\eta^{X'}$ be the automaton for η, where X' is the set of free variables in η, and let $\mathcal{A}_\psi^{X''}$ be the automaton for ψ, where X'' is the set of free variables in ψ. Let $X = X' \cup X''$. We can use the construction in Proposition 5.1 to get the automata $\mathcal{A}_\eta^X$ and $\mathcal{A}_\psi^X$. Then $\mathcal{A}_\varphi^X$ is the intersection of $\mathcal{A}_{valid}^X$ with the union of $\mathcal{A}_\eta^X$ and $\mathcal{A}_\psi^X$.

5. $\varphi = \exists x\eta$: Let $\mathcal{A}_\eta^{X'}$ be the automaton for η, where X' is the set of free variables in η and $X = X' - \{x\}$. ¿From Proposition 3.1, we can assume that $\mathcal{A}_\eta^{X'}$ is deterministic and minimal. First we project away the component

corresponding to x in the symbols on the transitions of $\mathcal{A}_\eta^{X'}$. We then add a transition from a state q_1 to a state q_2 on the symbol a' if there exist states p_1 and p_2 such that the following transitions were present in $\mathcal{A}_\eta^{X'}$: q_1 on a 0-extension of a' to p_1, p_1 on a 1-extension of a' to p_2 and p_2 on a 0-extension of a' to q_2, where the extensions are with respect to x. Since a symbol having a 1 in one of its component cannot occur in a loop (except that it can occur as a self-loop on a dead state), the above operations do not introduce counters. We then intersect the resulting automaton with $\mathcal{A}_{valid}^X$ to obtain $\mathcal{A}_\varphi^X$.

$\square$

We now introduce *continuous timed first-order logic* (TFO^c) and show that it characterizes the class *CFCIDA*. Let Σ be an alphabet and Op be a set of input determined operators over Σ. The syntax of $\mathrm{TFO}^c(\Sigma, Op)$ is given by:

$$\varphi ::= Q_a(x) \,|\, \Delta^I(x) \,|\, x < y \,|\, \neg\varphi \,|\, (\varphi \vee \varphi) \,|\, \exists x\varphi,$$

where $a \in \Sigma$, $\Delta \in Op$, $I \in \mathcal{I}_{\mathbb{Q}}$, and x is a first-order variable.

We interpret the logic over timed words in $T\Sigma^*$. Given a formula $\varphi \in \mathrm{TFO}^c(\Sigma, Op)$, a timed word $\sigma = (a_1, t_1) \cdots (a_n, t_n)$ in $T\Sigma^*$, and an interpretation $\mathbb{I}$ with respect to σ, which maps a first order variable x to $t \in [0, dur(\sigma)]$, we define the satisfaction relation $\sigma, \mathbb{I} \models \mathrm{Q}_a(x)$ as $\exists i : a_i = a, t_i = \mathbb{I}(x)$, and $\sigma, \mathbb{I} \models \Delta^I(x)$ as $\Delta(\sigma, \mathbb{I}(x)) \cap I \neq \emptyset$, and the rest of the cases are similar to that of FO^c over functions. For a sentence φ in $\mathrm{TFO}^c(\Sigma, Op)$, we set the timed language defined by φ to be $L(\varphi) = \{\sigma \in \mathrm{T}\Sigma^* \,|\, \sigma \models \varphi\}$.

The main result of this section states that TFO^c-definable languages and *CFCIDA* coincide:

Theorem 5.1 *Let Σ be an alphabet and Op a set of finitely varying input-determined operators over Σ. A timed language $L \subseteq T\Sigma^*$ is in CFCIDA(Σ, Op) iff it is definable by a $\mathrm{TFO}^c(\Sigma, Op)$ sentence.*

The rest of the section is devoted to a proof of the above theorem. The translation from TFO^c to *CFCIDA* relies on Proposition 5.1. In the backward direction, we factor through linear temporal logic over continuous time and use Kamp's theorem [9].

From TFO^c to *CFCIDA*

Lemma 5.2 *Let Σ be an alphabet and Op a set of finitely varying input-determined operators over Σ. Let φ be a $\mathrm{TFO}^c(\Sigma, Op)$ sentence, then there exists a CFCIDA $\mathcal{A}$ over (Σ, Op) such that $L(\varphi) = L(\mathcal{A})$.*

Proof Let φ be a $\mathrm{TFO}^c(\Sigma, Op)$ sentence. There exist a proper symbolic alphabet (Γ_1, Γ_2) over (Σ, Op) and a $\mathrm{FO}^c(\Gamma_1 \cup \Gamma_2)$ sentence $\tilde{\varphi}$ such that $L(\varphi) = tw(F(\tilde{\varphi}))$. The symbolic alphabet (Γ_1, Γ_2) is based on the set of guards $\{\Delta^I | \Delta^I(x)$ is a sub-formula of $\varphi\}$. $\tilde{\varphi}$ is then obtained from φ by replacing each $Q_a(x)$ by $\bigvee_{(a,h)\in\Gamma} Q_{(a,h)}(x)$ and $\Delta^I(x)$ by $\bigvee_{(c,h)\in\Gamma, \Delta^I\in h} Q_{(c,h)}(x) \vee \bigvee_{h\in\Gamma,\Delta^I\in h} Q_h(x)$, where $\Gamma = \Gamma_1 \cup \Gamma_2$, and taking its conjunction with φ_{canon}, which is satisfied by functions whose untiming is a fully canonical proper word. Now from the definitions of tw and $alt\text{-}FVF$, it follows that $tw(F(\tilde{\varphi})) = tw(F(\tilde{\varphi}) \cap alt\text{-}FVF(\Gamma))$. ¿From Lemma 5.1, there exists a *CFA* $\mathcal{A}'$ such that $F(\tilde{\varphi}) \cap alt\text{-}FVF(\Gamma) = timing(L_{sym}(\mathcal{A}'))$. Further $L_{sym}(\mathcal{A}') \subseteq \Gamma_1 \cdot (\Gamma_2 \cdot \Gamma_1)^*$. Hence from Proposition 3.2 there exists a counter-free ST-NFA $\mathcal{A}$ such that $L_{sym}(\mathcal{A}) = L_{sym}(\mathcal{A}')$. Putting everything together we have $L(\varphi) = L(\mathcal{A})$. Since $\mathcal{A}$ consists of only fully canonical proper words, $\mathcal{A}$ is the required *CFCIDA*. □

From *CFCIDA* to TFO^c

We show the containment of *CFCIDA* in TFO^c by showing their containment in a timed linear temporal logic which is equivalent to TFO^c. The logic is denoted $\mathrm{TLTL}^c(\Sigma, Op)$, parameterized by the alphabet Σ and the set of input-determined operators Op over Σ. The formulas of TLTL^c are given by:

$$\theta ::= a \,|\, \Delta^I \,|\, (\theta U \theta) \,|\, (\theta S \theta) \,|\, \neg\theta \,|\, (\theta \vee \theta),$$

where $a \in \Sigma$, $\Delta \in Op$ and $I \in \mathcal{I}_{\mathbb{Q}}$. We interpret $\mathrm{TLTL}^c(\Sigma, Op)$ formulas over timed words over Σ. Let φ be a $\mathrm{TLTL}^c(\Sigma, Op)$ formula. Let $\sigma \in \mathrm{T}\Sigma^*$, with $\sigma = (a_1, t_1) \cdots (a_n, t_n)$ and let $t \in [0, dur(\sigma)]$. Then the satisfaction relation $\sigma, t \models \varphi$ is given by:

$$\begin{array}{lll}
\sigma, t \models a & \text{iff} & \exists i : a_i = a, t_i = t. \\
\sigma, t \models \Delta^I & \text{iff} & \Delta(\sigma, t) \cap I \neq \emptyset. \\
\sigma, t \models \theta U \eta & \text{iff} & \exists t' : t < t' \leq dur(\sigma), \sigma, t' \models \eta, \forall t'' : t < t'' < t', \sigma, t'' \models \theta. \\
\sigma, t \models \theta S \eta & \text{iff} & \exists t' : 0 \leq t' < t, \sigma, t' \models \eta, \forall t'' : t' < t'' < t, \sigma, t'' \models \theta.
\end{array}$$

It is defined in the usual manner for the boolean combinations. The language defined by a $\mathrm{TLTL}^c(\Sigma, Op)$ formula θ is given by $L(\theta) = \{\sigma \in T\Sigma^* \,|\, \sigma, 0 \models \theta\}$. We state the following theorem from [3] which relates TLTL^c and TFO^c.

Theorem 5.2 ([3]) *Let Σ be an alphabet and Op be a set of finitely varying input-determined operators over Σ. A timed language $L \subseteq T\Sigma^*$ is definable by a* $\mathrm{TLTL}^c(\Sigma, Op)$ *formula θ iff it is definable by a sentence φ in* $\mathrm{TFO}^c(\Sigma, Op)$.

We will first give two interpretations of the classical linear-time temporal logic (LTL) and interpret the formulas of the logic over two kinds of models, namely, finite words and finitely varying functions. We will then give a translation from a formula accepting finite words to that accepting the functions which are timings of those words.

Recall that the syntax of LTL(A) is given by: $\theta ::= a \mid (\theta U \theta) \mid (\theta S \theta) \mid \neg\theta \mid (\theta \vee \theta)$, where $a \in A$. Given a function $f \in FVF(A)$, $t \in [0, dur(f)]$ and $\theta \in \mathrm{LTL}^c(A)$, the satisfaction relation $f, t \models a$ is defined as $f(t) = a$, and for the rest of the cases it is defined as for TLTL^c. Given a word $w = w_0 \cdots w_n \in A^*$ and $0 \leq i \leq n$, $w, i \models a$ if $w_i = a$ and $w, i \models \theta U \eta$ if there exists j such that $i < j$, $w, j \models \eta$ and for all k such that $i < k < j$, $w, k \models \theta$. S is defined symmetrical to U and the boolean operators in the usual way. $L(\theta) = \{w \in A^* | w, 0 \models \theta\}$ and $F(\theta) = \{f \in FVF(A) | f, 0 \models \theta\}$.

We use the following abbreviations: $cont$ (for a point of continuity) is defined to be $\bigvee_{a\in\Sigma}((aUa)\wedge(aSa))$, and $discont$ (for a point of discontinuity) is defined to be $\neg cont$. We define the translation $ltl\text{-}ltlc$ as follows (we use $\hat{\varphi}$ for $ltl\text{-}ltlc(\varphi)$):

$$
\begin{array}{lcl}
ltl\text{-}ltlc(a) & = & a. \\
ltl\text{-}ltlc(\neg\varphi) & = & \neg\hat{\varphi}. \\
ltl\text{-}ltlc(\varphi_1 \vee \varphi_2) & = & \hat{\varphi}_1 \vee \hat{\varphi}_2. \\
ltl\text{-}ltlc(\varphi_1 U \varphi_2) & = & \big(discont \Rightarrow (\hat{\varphi}_1 U \hat{\varphi}_2)\big) \bigwedge \big(cont \Rightarrow \\
 & & ((\neg discont) U (discont \wedge (\hat{\varphi}_2 \vee (\hat{\varphi}_1 \wedge \hat{\varphi}_1 U \hat{\varphi}_2))))\big) \\
ltl\text{-}ltlc(\varphi_1 S \varphi_2) & = & \big(discont \Rightarrow (\hat{\varphi}_1 S \hat{\varphi}_2)\big) \bigwedge \big(cont \Rightarrow \\
 & & ((\neg discont) S (discont \wedge (\hat{\varphi}_2 \vee (\hat{\varphi}_1 \wedge \hat{\varphi}_1 S \hat{\varphi}_2))))\big)
\end{array}
$$

Lemma 5.3 *Let A be an alphabet and w be a canonical word in A^*. Let $f \in timing(w)$ with a canonical interval representation $(a_0, I_0) \cdots (a_{2n}, I_{2n})$. Then for all $i \in \{0, \cdots, 2n\}$ and for all $t \in I_i$, $w, i \models \varphi \Leftrightarrow f, t \models ltl\text{-}ltlc(\varphi)$.*

Proof We do the proof by induction on φ. The base case and induction step for boolean operators are straightforward. We now show the induction step for $\varphi = \varphi_1 U \varphi_2$ (the proof is similar for $\varphi = \varphi_1 S \varphi_2$).

Left to right implication :

Let $t \in I_i$ and suppose $w, i \models \varphi_1 U \varphi_2$.

Then $\exists j > i$ such that $w, i \models \varphi_2$ and $\forall i < i' < j \;\; w, i' \models \varphi_1$. Consider $t' \in I_j$. Remark that by induction hypothesis we have that $f, t' \models \hat{\varphi}_2$, $\forall t'' \in I_j \;\; f, t'' \models \hat{\varphi}_2$ and $\forall t'' \in \cup_{i'|i<i'<j} I_{i'} \;\; f, t'' \models \hat{\varphi}_1$.

We distinguish two cases :

- i is even : Then $f, t \models discont$. We have that $(t, t') \subseteq \cup_{i'|i<i'<j} I_{i'} \cup I_j$. By the remark $\forall t < t'' < t' \;\; f, t'' \models \hat{\varphi}_1 \vee \hat{\varphi}_2$, so $f, t \models \hat{\varphi}_1 U \hat{\varphi}_2$.
- i is odd : Then $f, t \models cont$. Let t_1 such that $I_{i+1} = \{t_1\}$. We have that $\forall t_1 < t'' < t' \;\; f, t'' \models \hat{\varphi}_1 \vee \hat{\varphi}_2$. So $f, t_1 \models discont \wedge (\hat{\varphi}_2 \vee (\hat{\varphi}_1 \wedge \hat{\varphi}_1 U \hat{\varphi}_2))$.

Moreover $\forall t < t'' < t_1 \ f, t'' \models \neg discont$ so $f, t \models (\neg discont)U(discont \wedge (\hat{\varphi}_2 \vee (\hat{\varphi}_1 \wedge \hat{\varphi}_1 U \hat{\varphi}_2)))$.

Right to left implication :

Let $t \in I_i$ and suppose $f, t \models ltl\text{-}ltlc(\varphi)$. We distinguish two cases :

- i is even : Then $f, t \models discont$ so $f, t \models \hat{\varphi}_1 U \hat{\varphi}_2$. It follows that $\exists t' > t$ such that $f, t' \models \varphi_2$ and $\forall t < t'' < t' \ f, t'' \models \hat{\varphi}_1 \vee \hat{\varphi}_2$.

  Let j such that $t' \in I_j$. By induction hypothesis $w, j \models \varphi_2$ and $\forall i < i' < j \ w, i' \models \varphi_1$ so $w, i \models \varphi_1 U \varphi_2$.

- i is odd : Then $f, t \models cont$ and so $f, t \models (\neg discont)U(discont \wedge (\hat{\varphi}_2 \vee (\hat{\varphi}_1 \wedge \hat{\varphi}_1 U \hat{\varphi}_2)))$. Then it exists $t_1 > t$ such that $\forall t < t'' < t_1 \ \ f, t'' \models \neg discont$ and $f, t_1 \models discont \wedge (\hat{\varphi}_2 \vee (\hat{\varphi}_1 \wedge \hat{\varphi}_1 U \hat{\varphi}_2))$. Necessarily $I_{i+1} = \{t_1\}$.

  Either $f, t_1 \models \hat{\varphi}_2$ then by induction hypothesis $w, i+1 \models \varphi_2$ and $w, i \models \varphi_1 U \varphi_2$. The other case is that $f, t_1 \models \hat{\varphi}_1$ (so by induction hypothesis $w, i+1 \models \varphi_1$) and $f, t_1 \models \hat{\varphi}_1 U \hat{\varphi}_2$. Now $i+1$ is even and we can apply the case i even to get that $w, i+1 \models \varphi_1 U \varphi_2$. So we can deduce $w, i \models \varphi_1 U \varphi_2$.

$\Box$

The following lemma gives the desired translation from *CFCIDA* to TLTL^c.

Lemma 5.4 *Let Σ be an alphabet and Op be a set of finitely varying input-determined operators over Σ. Let $\mathcal{A}$ be a CFCIDA over (Σ, Op). Then there exists a $\mathrm{TLTL}^c(\Sigma, Op)$ formula θ such that $L(\mathcal{A}) = L(\theta)$.*

Proof Given *CFCIDA* $\mathcal{A}$, $L_{sym}(\mathcal{A})$ is a counter-free regular language. From [10] and [9], there exists a formula θ_1 in LTL(A) such that $L_{sym}(\mathcal{A}) = L(\theta_1)$. Since every $w \in L_{sym}(\mathcal{A})$ is canonical, from Lemma 5.3 we have $timing(L(\theta_1)) = F(\theta_2)$ where $\theta_2 = ltl\text{-}ltlc(\theta_1))$. We then use the translation $ltl\text{-}tltl$ from [3] which maps θ_1 to θ such that $L(\theta) = tw(F(\theta_1))$ if the untiming of functions in $F(\theta_1)$ are fully canonical proper words. Since we began with *CFCIDA* $\mathcal{A}$ whose symbolic language consisted of fully canonical proper words, the above condition is satisfied. Hence we have $L(\mathcal{A}) = L(\theta)$. $\Box$

Putting together Lemma 5.4 and Theorem 5.2, we obtain:

Theorem 5.3 *Let Σ be an alphabet and Op a set of finitely varying input-determined operators over Σ. Let $\mathcal{A}$ a CFCIDA over (Σ, Op). Then there exists a $\mathrm{TFO}^c(\Sigma, Op)$ formula φ such that $L(\mathcal{A}) = L(\varphi)$.*

6 Counter-free Recursive $CIDA$'s

We now consider "recursive" (or "hierarchical") $CIDA$'s, an idea introduced in [8], which allows us to increase the power of timed temporal logics by passing formulas as arguments to distance operators. We give a first-order characterization of the counter-free version of these recursive $CIDA$'s and then show that it can be applied to obtain a counter-free automata characterization for MTL.

Below are a few preliminaries. A *floating timed word* over Σ is a pair (σ, t), where σ in $T\Sigma^*$ and $t \in [0, dur(\sigma)]$. We denote the set of floating timed words over Σ by $FT\Sigma^*$. We will represent a floating word over Σ as timed word over the alphabet $\Sigma' = (\Sigma \cup \{\epsilon\}) \times \{0, 1\}$. For a timed word σ' over Σ', let σ denote the timed word obtained from σ' by projecting away the $\{0, 1\}$ component from each pair and then dropping any ϵ's in the resulting word. Then a timed word σ' over Σ' which contains exactly one symbol from $(\Sigma \cup \{\epsilon\}) \times \{1\}$, and whose last symbol is from $\Sigma \times \{0, 1\}$, represents the floating timed word (σ, t), where t is the time of the unique action which has a 1-extension. We use fw to denote the (partial) map which given a timed word σ' over Σ' returns the floating word (σ, t) represented by it, and extend it to apply to timed languages over Σ' in the natural way.

Let Σ be an alphabet and Op be a set of input determined operators. Given $\Delta \in Op$, we use the notation Δ' for the operator over Σ' with the semantics $\Delta'(\sigma', t) = \Delta(\sigma, t)$. We use the notation Op' to denote the set $\{\Delta' \mid \Delta \in Op\}$. We now define a *floating* $CIDA$ over (Σ, Op) to be a $CIDA$ over (Σ', Op'). We define the floating language of a floating $CIDA$ $\mathcal{B}$, denoted $L^{fl}(\mathcal{B})$, as $fw(L(\mathcal{B}))$.

A *recursive* input-determined operator Δ over an alphabet Σ is a partial function from $(2^{FT\Sigma^*} \times T\Sigma^* \times \mathbb{R}_{\geq 0})$ to $2^{\mathbb{R}_{\geq 0}}$, which is defined for tuples (M, σ, t) where M is a floating language over Σ, $\sigma \in T\Sigma^*$, and $t \in [0, dur(\sigma)]$. Thus, given a floating language M, we obtain an input-determined operator Δ_M whose semantics is given by $\Delta_M(\sigma, t) = \Delta(M, \sigma, t)$. For a floating $CIDA$ $\mathcal{B}$, we write $\Delta_{\mathcal{B}}$ for the operator $\Delta_{L^{fl}(\mathcal{B})}$.

We call a floating language M over Σ finitely varying, if for each timed word σ, the characteristic function of the set $pos(M, \sigma) = \{t \mid (\sigma, t) \in M\}$ is finitely varying in $[0, dur(\sigma)]$. We say a recursive operator Δ is *finitely varying* if for every finitely varying floating language M, the operator Δ_M is finitely varying.

We are now ready to define the recursive version of our $CIDA$'s. We define the class of *recursive* $CIDA$'s (*rec*-$CIDA$'s), and the class of *recursive floating* $CIDA$'s (*frec*-$CIDA$'s) over an alphabet Σ and a set of recursive operators Rop based on Σ, as the union over $i \in \mathbb{N}$, of level-i *rec*-$CIDA$'s over (Σ, Rop) and level i *frec*-$CIDA$'s over (Σ, Rop), which are defined inductively below:

- A level-0 *rec*-$CIDA$ over (Σ, Rop) is a $CIDA$ $\mathcal{A}$ over Σ that uses only the guard $\top$. It accepts the timed language accepted by $\mathcal{A}$ viewed as a $CIDA$ – i.e. $L(\mathcal{A})$. A level-0 *frec*-$CIDA$ over (Σ, Rop) is a floating $CIDA$ $\mathcal{B}$ over Σ which uses only the guard $\top$. It accepts the floating language $L^{fl}(\mathcal{B})$ (i.e by viewing it as a floating $CIDA$ over Σ).

- A level-$i+1$ *rec*-$CIDA$ over (Σ, Rop) is a $CIDA$ $\mathcal{A}$ over Σ and finite set of operators Op of the form $\Delta_{\mathcal{B}}$, where $\Delta \in Rop$ and $\mathcal{B}$ is a level-i or less *frec*-$CIDA$ over (Σ, Rop). We require that $\mathcal{A}$ uses at least one operator of the form $\Delta_{\mathcal{B}}$ wit $\mathcal{B}$ a level-i *frec*-$CIDA$. The timed language $L(\mathcal{A})$ accepted by $\mathcal{A}$ is defined to be the timed language accepted by $\mathcal{A}$ viewed as a $CIDA$ over (Σ, Op).

  A level-$i+1$ *frec*-$CIDA$ over (Σ, Rop) is a floating $CIDA$ $\mathcal{B}$ over Σ and finite set of operators Op of the form $\Delta_{\mathcal{C}}$, where $\Delta \in Rop$ and $\mathcal{C}$ is a level-i or less *frec*-$CIDA$ over (Σ, Rop). We require that $\mathcal{B}$ uses at least one operator of the form $\Delta_{\mathcal{C}}$ wit $\mathcal{C}$ a level-i *frec*-$CIDA$. The floating language $L^{fl}(\mathcal{B})$ accepted by $\mathcal{B}$ is defined to be the floating language accepted by $\mathcal{B}$ viewed as a floating $CIDA$ over (Σ, Op).

We now define the counter-free versions of these automata, by induction on the level in which they occur. A level-0 *rec*-$CIDA$ (respectively *frec*-$CIDA$) is counter-free if the underlying ST-NFA is counter-free. A level-$i+1$ *rec*-$CIDA$ (resp. *frec*-$CIDA$) is counter-free if it only uses operators of the form $\Delta_{\mathcal{B}}$ where $\mathcal{B}$ is a counter-free *frec*-$CIDA$ of level-i or less, and the underlying ST-NFA is counter-free.

We extend these definitions to proper *rec*-$CIDA$'s and *frec*-$CIDA$'s in the obvious way. We define the class of counter-free *rec*-$CIDA$ languages over (Σ, Rop), denoted *rec*-$CFCIDA(\Sigma, Rop)$, to be the class of timed languages definable by counter-free fully canonical proper *rec*-$CIDA$'s over (Σ, Rop).

A logical characterization of *rec*-$CFCIDA$ languages

We now introduce the recursive version of TFO^c and show that it characterizes the class of timed languages defined by *rec*-$CFCIDA$. Given an alphabet Σ and a set of recursive operators Rop, the set of formulas of *rec*-$TFO^c(\Sigma, Rop)$ are defined inductively as:

$$\varphi ::= Q_a(x) \mid \Delta^I_\psi(x) \mid x < y \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists x \varphi$$

where $a \in \Sigma$, $\Delta \in Rop$, $I \in \mathcal{I}_{\mathbb{Q}}$ and ψ is a *rec*-TFO^c formula with a single free variable z.

The *rec*-TFO^c formulas are interpreted similar to TFO^c formulas where the operator Δ_ψ is defined by $\Delta_\psi(\sigma, t) = \Delta(L^{fl}(\psi), \sigma, t)$ and $L^{fl}(\psi) = \{(\sigma, t) \mid \sigma, [t/z] \models$

$\psi\}$. A $rec\text{-}TMSO^c(\Sigma, Rop)$ sentence φ defines the language $L(\varphi) = \{\sigma \in T\Sigma^* \mid \sigma \models \varphi\}$.

We can now show that $rec\text{-}TFO^c$-definable languages and $rec\text{-}CFCIDA$ and the same:

Theorem 6.1 *$L \subseteq T\Sigma^*$ is in $rec\text{-}CFCIDA(\Sigma, Rop)$ iff L is definable by a $rec\text{-}TFO^c(\Sigma, Rop)$ sentence.*

To prove theorem 6.1 we first need to prove a more general version of theorem 5.1: *fCFCIDA* corresponds to TFO^c formula with one free variable.

Proposition 6.1 *Let Σ be an alphabet and Op a set of finitely varying input-determined operators over Σ. A floating timed language $L \subseteq T\Sigma^* \times \mathbb{R}$ is accepted by a fCFCIDA over (Σ, Op) iff it is definable by a $\mathrm{TFO}^c(\Sigma, Op)$ formula with one free variable.*

We split the proof of proposition 6.1 into the two following lemmas:

Lemma 6.1 *Let Σ be an alphabet and Op a set of finitely varying input-determined operators over Σ. Let φ be a $\mathrm{TFO}^c(\Sigma, Op)$ formula with one free variable. Then there exists a fCFCIDA $\mathcal{B}$ over (Σ, Op) such that $L^{fl}(\varphi) = L^{fl}(\mathcal{B})$.*

Proof Let z be the free variable into φ. The proof essentially relies on lemma 5.1: we translate $\varphi(z)$ into a formula of FO^c with one free variable, and lemma 5.1 provides an equivalent *fCFCIDA*.

Let Γ be the proper symbolic alphabet associated to $\varphi(z)$. $tmso\text{-}mso(\varphi)$ is a $\mathrm{FO}^c(\Gamma)$ formula. Let $\mathcal{B}_{\varphi(z)}$ be the associated CFFA by lemma 5.1.

We have that $L^{fl}(\varphi(z)) = L^{fl}(\mathcal{B}_{\varphi(z)})$. Indeed

$$\begin{array}{rcll}
(\sigma, t) \in L^{fl}(\varphi(z)) & \Leftrightarrow & \sigma, t/z \models \varphi & \\
& \Leftrightarrow & f^{\Gamma}_{\sigma}, t/z \models tmso\text{-}mso(\varphi)(z) & \text{by lemma 5.1 of [4]} \\
& \Leftrightarrow & untiming(f^{\Gamma}_{\sigma,t/z}) \in L_{sym}(\mathcal{B}_{\varphi(z)}) & \text{by lemma 5.1} \\
& \Leftrightarrow & f^{\Gamma}_{\sigma,t/z} \in timing(L_{sym}(\mathcal{B}_{\varphi(z)})) & \\
& \Leftrightarrow & (\sigma, t) \in fw(timing(L_{sym}(\mathcal{B}_{\varphi(z)}))) & \\
& \Leftrightarrow & (\sigma, t) \in L^{fl}(\mathcal{B}_{\varphi(z)}) &
\end{array}$$

□

Lemma 6.2 *Let Σ be an alphabet and Op a set of finitely varying input-determined operators over Σ. Let $\mathcal{B}$ be a fCFCIDA over (Σ, Op). Then there exists a $\mathrm{TFO}^c(\Sigma, Op)$ formula with one free variable φ such that $L^{fl}(\mathcal{B}) = L^{fl}(\varphi)$.*

Proof Let $\mathcal{B}$ be a *fCFCIDA* over (Σ, Op). $\mathcal{B}$ can be seen as a *CFCIDA* over $(\Sigma\times\{0,1\}, \mathit{Op})$. So by corollary 5.3 there exists a $\mathrm{TFO}^c(\Sigma\times\{0,1\}, \mathit{Op})$ formula ψ such that $L(\mathcal{B}) = L(\psi)$.

¿From ψ we then construct a $\mathrm{TFO}^c(\Sigma, \mathit{Op})$ formula with one free variable $\varphi(z)$ as follows: for every variable x and every $a \in \Sigma$ we replace each occurrence $Q_{(a,1)}(x)$ by $Q_a(x) \wedge x = z$ and each occurrence of $Q_{(a,0)}(x)$ by $Q_a(x) \wedge \neg(x = z)$.

Recall that we note Σ' for $(\Sigma \cup \{\varepsilon\}) \times \{0, 1\}$. Let $\mathit{valid}(\Sigma'))$ be the set of timed words over Σ' which contain exactly one symbol from $(\Sigma \cup \{\varepsilon\}) \times \{1\}$. Let $\mathit{first}(\sigma)$ be the timed word obtained from σ by projecting away the $\{0,1\}$ component.
By induction on ψ we can easily prove the following lemma:
$\forall \sigma' \in \mathit{valid}(\Sigma \times \{0,1\})\ \forall \mathbb{I}\ \ \sigma', \mathbb{I} \models \psi \Leftrightarrow \mathit{first}(\sigma'), \mathbb{I}[\mathit{dur}(\sigma')/z] \models \varphi(z)$

Then we can deduce that $\mathit{fw}(L(\psi)) = L^{\mathit{fl}}(\varphi(z))$.
We thus have that $L^{\mathit{fl}}(\mathcal{B}) = \mathit{fw}(L(\mathcal{B})) = \mathit{fw}(L(\psi)) = L^{\mathit{fl}}(\varphi(z))$. $\square$

We can now end the proof of theorem 6.1:
Proof We show that for each i, the class of *rec-CFCIDA*'s of level i corresponds to the sentences of *rec-TFO*^{c}.

We show by induction on i that:

1. $L \subseteq \mathrm{T}\Sigma^*$ is accepted by a level i *rec-CFCIDA* over (Σ, Rop) iff L is definable by a level i *rec-TFO*$^c(\Sigma, \mathit{Rop})$ sentence

2. A floating language L over Σ is accepted by a level i frec-CFIDA over (Σ, Rop) iff L is definable by a level i *rec-TFO*$^c(\Sigma, \mathit{Rop})$ formula with one free variable

Base case for 1.

Level 0 automata only use the guard $\top$ so they are Büchi automata. Similarly , level 0 sentences do not use any Δ predicates and hence they are simply $\mathrm{MSO}(\Sigma)$ sentences. By Büchi's theorem we have that level 0 automata and sentences are expressively equivalent.

Base case for 2.

Level 0 frec-CFIDA are only using the guard $\top$. Level 0 *rec-TFO*^{c}are not using Δ predicates. We can use the proof of proposition 6.1 in a particular case to show they are equivalent.

Induction step for 1.

Let $\mathcal{A}$ be a level $i+1$ *rec-CFCIDA* over (Σ, Rop). Let Op be the set of top-level operators in $\mathcal{A}$. Since $\mathcal{A}$ is a *CFCIDA* over (Σ, Op), by theorem 5.1

there is a $\mathrm{TFO}^c(\Sigma, Op)$ sentence φ such that $L(\mathcal{A}) = L(\varphi)$. Now for each $\Delta_{\mathcal{B}} \in Op$, $\mathcal{B}$ is of level i or lower; so by induction hypothesis there is a corresponding $\textit{rec-TFO}^c(\Sigma, \mathit{Rop})$ formula ψ with one free variable, of the same level as $\mathcal{B}$, with $L^{fl}(\mathcal{B}) = L^{fl}(\psi)$. Thus each $\Delta_{\mathcal{B}}$ has a semantically equivalent operator Δ_ψ. We can now replace each occurrence of $\Delta_{\mathcal{B}}$ in φ by Δ_ψ to get an equivalent sentence φ' which is in $\textit{rec-TFO}^c(\Sigma, \mathit{Rop})$.

Conversely, let φ be a level $i+1$ sentence in $\textit{rec-TFO}^c(\Sigma, \mathit{Rop})$. Let Op be the set of top-level operators in φ. φ is a $\mathrm{TFO}^c(\Sigma, Op)$ sentence, and hence by theorem 5.1 we have an equivalent CFCIDA $\mathcal{A}$ over (Σ, Op). Once again, for each $\Delta_\psi \in Op$, the formula ψ is of level i or lower, and hence by induction hypothesis we have a frec-CFIDA$\mathcal{B}$ over (Σ, Rop), of the same level as ψ, and accepting the same floating language. The operators Δ_ψ and $\Delta_{\mathcal{B}}$ are now equivalent, and we can replace each Δ_ψ in $\mathcal{A}$ by the corresponding $\Delta_{\mathcal{B}}$ to get a language equivalent input determined automaton. This automaton is now the required level $i+1$ $\textit{rec-CFCIDA}$ over (Σ, Rop) which accepts the same language as φ.

Induction step for 2. Similar to induction step for 1. and makes use of proposition 6.1. ☐

Counter-free automata characterization of MTL

As an application of our results we give an automata characterization for MTL^c in terms of $\textit{rec-CFCIDA}$ over the operators $\diamond$ ("eventually") and $\diamondminus$("eventually in the past"). The logic $\mathrm{MTL}^c(\Sigma)$ is defined inductively as:

$$\theta ::= a \,|\, (\theta U_I \theta) \,|\, (\theta S_I \theta) \,|\, \neg\theta \,|\, (\theta \vee \theta),$$

where $a \in \Sigma$ and $I \in \mathcal{I}_{\mathbb{Q}}$. modalities U_I and S_I are interpreted over a timed word σ at time $t \in [0, dur(\sigma)]$ as follows:

$$\begin{array}{lll} \sigma, t \models \theta U_I \eta & \text{iff} & \exists t' \geq t : t' - t \in I, \sigma, t' \models \eta, \text{ and } \forall t'' : t < t'' < t', \sigma, t'' \models \theta. \\ \sigma, t \models \theta S_I \eta & \text{iff} & \exists t' \leq t : t - t' \in I, \sigma, t' \models \eta, \text{ and } \forall t'' : t' < t'' < t, \sigma, t'' \models \theta. \end{array}$$

The recursive operators $\diamond$ and $\diamondminus$are defined as $\diamond(M, \sigma, t) = \{t' - t \,|\, t' \geq t, t \in pos(M, \sigma)\}$ and $\diamondminus(M, \sigma, t) = \{t - t' \,|\, t' \leq t, t \in pos(M, \sigma)\}$. In [3], $\mathrm{MTL}^c(\Sigma)$ is shown to be expressively equivalent to $\textit{rec-TFO}^c(\Sigma, \{\diamond, \diamondminus\})$, and Theorem 6.1 shows that every $\textit{rec-TFO}^c(\Sigma, \{\diamond, \diamondminus\})$ corresponds to a $\textit{rec-CFCIDA}$ over $(\Sigma, \{\diamond, \diamondminus\})$. Thus we have:

Theorem 6.2 $\mathrm{MTL}(\Sigma)$ *is expressively equivalent to* $\textit{rec-CFCIDA}(\Sigma, \{\diamond, \diamondminus\})$.

A similar result for MITL [1], by simply restricting intervals to be non-singular.

7 Ultimate stability of MTL

In section 6 we showed that every MTL^c language is recognized by a recursive counter-free *CIDA*. We will now use this characterization to show the *ultimate stability* of MTL^c, which states that, if a language L is expressible in MTL^c, any periodic sequence of timed words will be ultimately in L or will ultimately stay out of L. In this section we assume that Σ is an alphabet and Rop a set of finitely varying recursive operators.

A *periodic sequence* is a triple $s = (u, v, w) \in (T\Sigma^*)^3$, which denotes the sequence of timed words $s_0, s_1, \cdots$ where $s_i = uv^iw$.

A language of timed words $L \subseteq T\Sigma^*$ is said to be *ultimately stable* w.r.t. a periodic sequence s if there exists $i_0 \in \mathbb{N}$ such that either $\forall i \geq i_0,\ s_i \in L$ or $\forall i \geq i_0,\ s_i \notin L$. A language $L \subseteq T\Sigma^*$ is said to be *ultimately stable* if it is ultimately stable w.r.t. all periodic sequences.

Theorem 7.1 *Let φ be an MTL formula. Then $L(\varphi)$ is ultimately stable.*

The rest of this section is devoted to the proof of the above theorem. By theorem 6.2 we just need to show that all languages recognized by *rec-CFCIDA* are ultimately stable.

We first introduce the concept of middle zone: it represents the set of time points "in the middle" of a timed word. A *middle zone* is a couple $Z = (l, r)$ with $l, r \in \mathbb{R}$.
Given a timed word w we define $Z(w) = (l, dur(w) - r)$.

Figure 1: A middle zone

A floating language $L \subseteq T\Sigma^* \times \mathbb{R}$ is said to be *well-behaved* w.r.t. a periodic sequence $s = (u, v, w)$ if there exist a middle zone Z and an index i_0 such that the following conditions hold:

$$\begin{aligned}&\forall i\ \forall i' \geq i\ \forall t \in Z(s_i),\ (s_i, t) \in L \Leftrightarrow (s_i, t + dur(v)) \in L \text{ and}\\ &\quad (s_i, t) \in L \Leftrightarrow (s_{i'}, t) \in L.\end{aligned} \tag{1}$$

$$\forall i \geq i_0\ \forall i' \geq i\ \forall t < l,\ (s_i, t) \in L \Leftrightarrow (s_{i'}, t) \in L. \tag{2}$$

$$\forall i \geq i_0\ \forall i' \geq i\ \forall t < r,\ (s_i, dur(s_i) - t) \in L \Leftrightarrow (s_{i'}, dur(s_{i'}) - t) \in L. \tag{3}$$

A floating language L is said to be *well-behaved* if it is *well-behaved* w.r.t. all periodic sequences.

Note that a guard Δ^I defines a floating language given by $\{(\sigma, t) \mid \sigma, t \models \Delta^I\}$. We say that a floating *CFCIDA* (resp. a guard) is *well-behaved* w.r.t. a periodic sequence if its associated language is. Similarly we say that a floating *CFCIDA* (resp. a guard) is *well-behaved* if its associated language is.

Given a proper symbolic alphabet Γ and a timed word s we note γ_s^Γ the unique symbolic word $\gamma \in \Gamma^*$ such that $s \in tw(\gamma)$.

Lemma 7.1 *Let G be a finite set of guards and Γ be a proper symbolic alphabet over (Σ, Rop) based on G. Let s be a periodic sequence. If for all $g \in G$, g is well-behaved w.r.t. s then there exists an integer i_0 and $\gamma_1, \gamma_2, \gamma_3 \in \Gamma^*$ such that for all $i \geq i_0$, $\gamma_{s_i}^\Gamma = \gamma_1 \gamma_2^{i-i_0} \gamma_3$.*

Note that this lemma shows that if for all $g \in G$, g is well-behaved w.r.t. s then any floating automaton based on G is well-behaved w.r.t. s.

Proof Let $G = \{g_1, \ldots, g_k\}$. As $g_1, \ldots, g_k$ are well-behaved there exist middle zones $Z_1, \ldots, Z_k$ and integers $i_1, \ldots, i_k$ such that properties (1), (2) and (3) hold.

We define the maximum of a finite set of middle zones as the maximum component-wise. Let $Z = \max\limits_{1 \leq j \leq k} Z_j$ and $i = \max\limits_{1 \leq j \leq k} i_j$. The middle zone Z and index i satisfy properties (1), (2) and (3) for all $g \in G$.

Let i_0 be an integer such that $i_0 \geq i$ and $Z(s_{i_0}) \neq \emptyset$. Let $\gamma_1 \in \Gamma^*$ such that there exist a prefix p_{i_0} of s_{i_0} with $dur(p_{i_0}) \in Z(s_{i_0})$ and $p_{i_0} \in tw(\gamma_1)$. Let γ_3 such that $\gamma_{s_i}^\Gamma = \gamma_1 \gamma_3$. Let γ_2 such that $\gamma_{s_{i+1}}^\Gamma = \gamma_1 \gamma_2 \gamma_3$. Then by properties (1), (2) and (3) $\forall i \geq i_0\ \gamma_{s_i}^\Gamma = \gamma_1 \gamma_2^{i-i_0} \gamma_3$. $\square$

Lemma 7.2 *Let $\mathcal{B}$ be a proper canonical fCFIDA over (Σ, Rop) and I an interval. If $\mathcal{B}$ is well-behaved, then the guards $\Diamond_B^I$ and $\diamondminus_B^I$ are also well-behaved.*

Proof Let $s = (u, v, w)$ a periodic sequence, we show that $\Diamond_B^I$ is well-behaved w.r.t. s, the case of $\diamondminus_B^I$ is similar.

We have to show that there exists a middle zone $Z = (l, r)$ and an index i_0 such that:

$$\begin{aligned} &\forall i\ \forall i' \geq i\ \forall t \in Z(s_i) \quad s_i, t \models \Diamond_{\mathcal{B}}^I \Leftrightarrow s_i, t + dur(v) \models \Diamond_{\mathcal{B}}^I \text{ and} \\ &\quad s_i, t \models \Diamond_{\mathcal{B}}^I \Leftrightarrow s_{i'}, t \models \Diamond_{\mathcal{B}}^I. \end{aligned} \tag{1}$$

$$\forall i \geq i_0\ \forall i' \geq i\ \forall t < l\ s_i, t \models \Diamond_{\mathcal{B}}^I \Leftrightarrow s_{i'}, t \models \Diamond_{\mathcal{B}}^I. \tag{2}$$

$$\forall i \geq i_0\ \forall i' \geq i\ \forall t < r\ s_i, dur(s_i) - t \models \Diamond_{\mathcal{B}}^I \Leftrightarrow s_{i'}, dur(s_{i'}) - t \models \Diamond_{\mathcal{B}}^I. \tag{3}$$

$\mathcal{B}$ is well-behaved by hypothesis. Let $Z_{\mathcal{B}} = (l_{\mathcal{B}}, r_{\mathcal{B}})$ be the middle zone for $\mathcal{B}$. Let M be a real greater than the bounds of I. We define $Z = (l_{\mathcal{B}}, r_{\mathcal{B}} + M + dur(v))$.

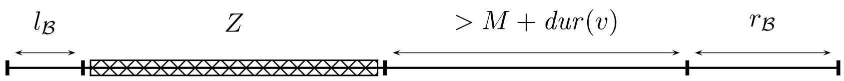
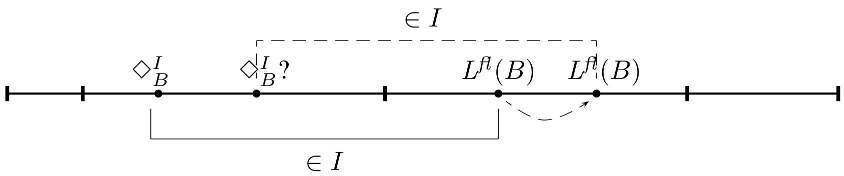


Figure 2: Choice of Z

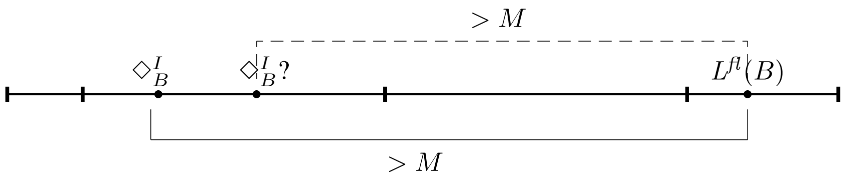
Let i, t be such that $t \in Z(s_i)$ and suppose $s_i, t \models \Diamond^I_{\mathcal{B}}$. We show that $s_i, t + dur(v) \models \Diamond^I_{\mathcal{B}}$ (other direction is similar).

We have that $\exists t_1 \geq t$ with $(s_i, t_1) \in L^{\mathit{fl}}(\mathcal{B})$ and $t_1 - t \in I$. We distinguish two cases:

- $t_1 \in Z_{\mathcal{B}}(s_i)$
  As $\mathcal{B}$ is well-behaved $(s_i, t_1 + dur(v)) \in L^{\mathit{fl}}(\mathcal{B})$ and we have that $t_1 + dur(v) - (t + dur(v)) = t_1 - t \in I$. Thus $s_i, t + dur(v) \models \Diamond^I_{\mathcal{B}}$



- $t_1 \notin Z_{\mathcal{B}}(s_i)$
  Then $t_1 - t > M$, so necessarily $I = (c, +\infty)$ or $I = [c, +\infty)$. As $t_1 - (t + dur(v)) > M$, $s_i, t + dur(v) \models \Diamond^I_{\mathcal{B}}$.



A similar reasoning shows that $s_{i'}, t \models \Diamond^I_{\mathcal{B}}$.

We now show how to choose i_0 and prove property (2) (showing property (3) is similar). We take $i_0 \in \mathbb{N}$ such that $i_0 \geq i_{\mathcal{B}}$ and $dur(Z(y_{i_0})) \geq M$ (so that the time of the middle zone is greater than M for $i \geq i_0$).

Let $i \geq i_0, i' \geq i, t < l$ and suppose $s_i, t \models \Diamond^I_{\mathcal{B}}$. Then $\exists t_1 \geq t$ with $(s_i, t_1) \in L^{\mathit{fl}}(\mathcal{B})$ and $t_1 - t \in I$.

We distinguish two cases:

- $t_1 \leq dur(s_i) - r$
  As $\mathcal{B}$ is well-behaved $(s_{i'}, t_1) \in L^{\mathit{fl}}(\mathcal{B})$ and so $s_{i'}, t \models \Diamond^I_{\mathcal{B}}$

- $t_1 > dur(s_i) - r$
  Then $t_1 - t > M$ and necessarily $I = (c, +\infty)$ or $I = [c, +\infty)$.

Set $t'_1 = dur(s_i) - t_1$. $(s_i, dur(s_i) - t'_1) \in L^{fl}(\mathcal{B})$ so as $\mathcal{B}$ is well-behaved $(s_{i'}, dur(s_{i'}) - t'_1) \in L^{fl}(\mathcal{B})$. Moreover $(dur(s_{i'}) - t'_1) - t \geq (dur(s_i) - t'_1) - t = t_1 - t > M$ thus $s_{i'}, t \models \Diamond^I_{\mathcal{B}}$.

□

Returning to the proof of theorem 7.1 : let L be recognized by some *rec-CFCIDA* $\mathcal{A}$ and s be a periodic sequence. Let Γ be the proper symbolic alphabet of $\mathcal{A}$; by lemma 7.1 and 7.2 there exists an integer i_0 and $\gamma_1, \gamma_2, \gamma_3 \in \Gamma^*$ such that for all $i \geq i_0$, $\gamma^{\Gamma}_{s_i} = \gamma_1 \gamma_2^{i-i_0} \gamma_3$. Let n be the number of states of $\mathcal{A}$. γ_2 cannot be a counter for $\mathcal{A}$ so for i greater than $i_0 + n$, the run of $\mathcal{A}$ on s_i ends in the same state. Thus L is ultimately stable w.r.t. s. □

References

[1] Rajeev Alur, Tomás Feder, and Thomas A. Henzinger. The Benefits of Relaxing Punctuality. *Journal of the ACM*, 43(1):116–146, 1996.

[2] Rajeev Alur, Limor Fix, and Thomas A. Henzinger. A Determinizable Class of Timed Automata. In David L. Dill, editor, *CAV*, volume 818 of *Lecture Notes in Computer Science*, pages 1–13. Springer, 1994.

[3] Fabrice Chevalier, Deepak D'Souza, and Pavithra Prabakhar. On continuous timed automata with input-determined guards. In Naveen Garg and S. Arun-Kumar, editors, *FSTTCS*, volume 4337 of *Lecture Notes in Computer Science*, pages 369–380, Kolkata, India, December 2006. Springer.

[4] Fabrice Chevalier, Deepak D'Souza, and Pavithra Prabhakar. On continuous timed automata with input-determined guards. Technical Report IISc-CSA-TR-2006-7, Indian Institute of Science, Bangalore 560012, India, June 2006. URL: `http://archive.csa.iisc.ernet.in/TR/2006/7/`.

[5] Deepak D'Souza and M. Raj Mohan. Eventual Timed Automata. In R. Ramanujam and Sandeep Sen, editors, *FSTTCS*, volume 3821 of *Lecture Notes in Computer Science*, pages 322–334. Springer, 2005.

[6] Deepak D'Souza and Nicolas Tabareau. On timed automata with input-determined guards. In Yassine Lakhnech and Sergio Yovine, editors, *FORMATS/FTRTFT*, volume 3253 of *Lecture Notes in Computer Science*, pages 68–83. Springer, 2004.

[7] Thomas A. Henzinger, Jean-François Raskin, and Pierre-Yves Schobbens. The Regular Real-Time Languages. In Kim Guldstrand Larsen, Sven Skyum, and Glynn Winskel, editors, *ICALP*, volume 1443 of *Lecture Notes in Computer Science*, pages 580–591. Springer, 1998.

[8] Thomas A. Henzinger, Jean-Francois Raskin, and Pierre-Yves Schobbens. Axioms for Real-Time Logics. Research Report MPI-I-1999-3-005, Max-Planck-Institut für Informatik, Stuhlsatzenhausweg 85, 66123 Saarbrücken, Germany, August 1999.

[9] Johan Anthony Willem Kamp. *Tense Logic and the Theory of Linear Order*. PhD thesis, University of California, Los Angeles, California, 1968.

[10] Robert McNaughton and Seymour Papert. *Counter-Free Automata*. MIT Press, Cambridge, MA, 1971.

[11] Joël Ouaknine and James Worrell. On the Decidability of Metric Temporal Logic. In *LICS*, pages 188–197. IEEE Computer Society, 2005.

[12] Pavithra Prabhakar and Deepak D'Souza. On the Expressiveness of MTL with Past Operators. In Eugene Asarin and Patricia Bouyer, editors, *FORMATS*, volume 4202 of *Lecture Notes in Computer Science*, pages 322–336. Springer, 2006.

[13] Jean-François Raskin and Pierre-Yves Schobbens. The logic of event clocks: decidability, complexity and expressiveness. *Automatica*, 4(3):247–282, 1999.