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. Along the way we generalize the result of McNaughton-Papert [10] to show a counter-free automata characterization of FO-definable finitely varying functions.

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, 8], 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 [6, 3], which make use of the "eventual operator" $\Diamond_a$ inspired by Metric Temporal Logic (MTL). In [7, 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 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 graphs have no counters in the classical sense. Our results hold for both the "pointwise" and "continuous" semantics for timed formalisms.

For the pointwise semantics, we can simply factor through the classical results of Kamp and McNaughton-Papert, and the translations set up in [7]. For the continuous

* 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`

case we first prove an analogue of the McNaughton-Papert result for finitely varying functions, by characterizing the first-order definable languages of finitely-varying functions in terms of a counter-free fragment of a class of automata we call ST-NFA's. Once we have this result, we can essentially factor through the translations for continuous time set up in [3].

We emphasize that this general result gives us automata characterizations for several of the timed logics proposed in the literature, including EventClockTL [8], 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.

In the sequel we concentrate on the continuous semantics. The details for the pointwise semantics and other arguments not included here for space reasons, can be found in the technical report [5].

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. The set of non-negative reals and rationals will be denoted by $\mathbb{R}_{\geq 0}$ and $\mathbb{Q}_{\geq 0}$ respectively. We will deal with intervals of non-negative reals, i.e. convex subsets of $\mathbb{R}_{\geq 0}$, and denote by $\mathcal{I}_{\mathbb{R}_{\geq 0}}$ and $\mathcal{I}_{\mathbb{Q}_{\geq 0}}$ the set of such intervals with end-points in $\mathbb{R}_{\geq 0} \cup \{\infty\}$ and $\mathbb{Q}_{\geq 0} \cup \{\infty\}$ respectively. Two intervals I and J will be called *adjacent* if $I \cap J = \emptyset$ and $I \cup J$ is an interval.

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, which in this case is r. A point $t \in (0, r)$ is a point of *continuity* of f if there exists $\epsilon > 0$ such that f is constant in the interval $(t - \epsilon, t + \epsilon)$. All other points in $[0, r]$ are points of *discontinuity* of f. We say f is *finitely varying* if it has only a finite number of discontinuities. We denote by $FVF(A)$ the set of all finitely varying functions over A.

An *interval representation* for a finitely varying function $f : [0, r] \to A$ is a sequence of the form $(a_0, I_0) \cdots (a_n, I_n)$, with $a_i \in A$ and $I_i \in \mathbb{I}_{\mathbb{R}_{\geq 0}}$, satisfying the conditions that the union of the intervals is $[0, r]$, each I_i and I_{i+1} are adjacent, and for each i, f is constant and equal to a_i in the interval I_i. We can obtain a *canonical* interval representation for f by putting each point of discontinuity in a singular interval by itself. Thus the above interval representation for f is canonical if n is even, for each even i I_i is singular (i.e. of the form $[t, t]$), and for no even i such that $0 < i < n$ is $a_{i-1} = a_i = a_{i+1}$.

A canonical interval representation for a function gives us a canonical way of "untiming" the function: thus if $(a_0, I_0) \cdots (a_{2n}, I_{2n})$ is the canonical interval representation for a function f, then we define $untiming(f)$ to be the string $a_0 \cdots a_{2n}$ in A^*. The untiming thus captures explicitly the value of the function at its points of discontinuity and the open intervals between them. Note that strings which represent the untiming of a function will always be of odd length and for no even position i will the letters at positions $i-1, i$, and $i+1$ be the same. We call such words *canonical*. A canonical word w can be "timed" to get a function in a natural way: thus a function f is in $timing(w)$ if $untiming(f) = w$. We extend the definition of *timing* and *untiming* to languages of functions and words in the expected way.

We now turn to some notions regarding classical automata and a variant we introduce. 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 NFA is said to be *counter-free* if it does not contain a counter. A regular language is said to be *counter-free* if there exists a counter-free NFA for it. We will call a sequence of words in A^* $\langle w_i \rangle = w_0, w_1, \cdots$ *periodic* if there exist strings u, v, w in A^* such that $w_i = uv^iw$ for each i. We say a language $L \subseteq A^*$ is *ultimately stable* (with respect to periodic sequences of words) if for each periodic sequence $\langle w_i \rangle$ there exists a $k \geq 0$, such that for all $i \geq k$, $w_i \in L$ or for all $i \geq k$, $w_i \notin L$.

Proposition 2.1 *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) L is ultimately stable with respect to periodic sequences of words.* □

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

We now define a variant of NFA's called *state-transition-labeled* NFA*'s* or ST-NFA$'s$ for short, which are convenient for generating finitely varying functions. An ST-NFA over A is a structure $\mathcal{A} = (Q, s, \delta, F, l)$ similar to an NFA over A, except that $l : Q \longrightarrow A$ labels states with letters from A. The ST-NFA $\mathcal{A}$ accepts strings of the form $A(AA)^*$. A run of $\mathcal{A}$ on a string $w = a_0a_1 \cdots a_{2n}$ in $A(AA)^*$, is a sequence of states $q_0, \ldots, q_{n+1}$ satisfying $q_0 = s$, $(q_i, a_{2i}, q_{i+1}) \in \delta$ for $i \in \{0, \cdots, n\}$ and $l(q_i) = a_{2i-1}$ for each $i \in \{1, \ldots, n\}$; it is accepting if $q_{n+1} \in F$. We define $L_{sym}(\mathcal{A})$ to be the set of strings $w \in A^*$ on which $\mathcal{A}$ has an accepting run.

An ST-NFA $\mathcal{A}$ also generates functions in a natural way: we begin by taking a transition emanating from the start state, emitting its label, and then spend time at the resulting state emitting its label all the while, before taking a transition again; and so on. For convenience we will stick to *canonical* ST-NFA's which are ST-NFA$'s$ in which we never have an a-labelled transition between an a-labelled source and target state, for any $a \in A$. The language of finitely-varying functions defined by a canonical ST-NFA $\mathcal{A}$ is defined to be $timing(L_{sym}(\mathcal{A}))$. Its not difficult to see that any ST-NFA can be converted to a canonical one whose function language is the same. We note that for a canonical ST-NFA $\mathcal{A}$, $untiming(F(\mathcal{A})) = L_{sym}(\mathcal{A})$.

We now define the counter-free version of ST-NFA's. 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 the label of the path $q_0 \stackrel{a_0}{\to} q_1 \stackrel{a_1}{\to} \cdots q_n \stackrel{a_n}{\to} q_{n+1}$ is $l(q_0)a_0l(q_1)a_1 \cdots l(q_n)a_n$. We note that a counter-free ST-NFA can define a language (e.g. the single state, single transition ST-NFA over $\{a\}$ which defines the language $a(aa)^*$) which is *not* counter-free in the classical sense. However, if we consider ST-NFA's over a *partitioned* alphabet, where the alphabet A is partitioned into A_1 and A_2 which label transitions and states respectively, then we can show that:

Proposition 2.2 *Let (A_1, A_2) be a partitioned alphabet. A regular subset of $A_1(A_2A_1)^*$ is counter-free iff there exists a counter-free* ST-NFA *over (A_1, A_2) accepting it.* $\square$

Finally, by going over to an alternating alphabet (say by renaming transition labels) using the closure properties of classical and counter-free languages, and then coming back, we can verify that:

Proposition 2.3 *Each of the classes of function languages definable by* ST-NFA*'s and counter-free* ST-NFA*'s over an alphabet A are closed under boolean operations.* $\square$

3 Counter-free continuous input-determined automata

We begin with some notation. 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 $t_0 < t_1 < \cdots < t_n$. We write $dur(\sigma)$ to denote the duration of σ, i.e. t_n above. We denote the set of timed words over Σ to be $T\Sigma^*$. Given timed words $\sigma = (a_0, t_0) \cdots (a_n, t_n)$ and $\sigma' = (a'_0, t'_0) \cdots (a'_k, t'_k)$, we define their concatenation $\sigma \cdot \sigma'$ in the standard way to be $(a_0, t_0) \cdots (a_n, t_n)(a'_0, t_n + t'_0) \cdots (a'_k, t_n + t'_k)$.

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 $I \in \mathcal{I}_{\mathbb{Q}_{\geq 0}}$. 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.

For example, the operator $\Delta_{\mathbb{Q}}$ 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 $\triangleleft_a$ 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 $I \in \mathcal{I}_{\mathbb{Q}_{\geq 0}}$, and each $\sigma \in T\Sigma^*$, the characteristic function $f^{\sigma}_{\Delta^I} : [0, dur(\sigma)] \to \{0, 1\}$ of Δ^I, defined as $f^{\sigma}_{\Delta^I}(t)$ is 1 if $\sigma, t \models \Delta^I$, and 0 otherwise, is finitely varying. Of the example operators above, $\Diamond_a$ and $\triangleleft_a$ are finitely varying, while $\Delta_{\mathbb{Q}}$ is not.

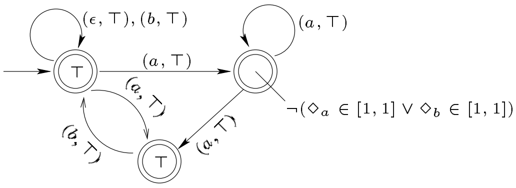
Let Σ be an alphabet and Op be a set of input determined operators over Σ. We call (Γ_1, Γ_2) a *symbolic alphabet* over (Σ, Op), if Γ_1 is a finite subset of $(\Sigma \cup \{\epsilon\}) \times \mathcal{G}(Op)$ and Γ_2 is a finite subset of $\mathcal{G}(Op)$. We define the set of timed words over Σ associated with a function f in $FVF(\Gamma_1 \cup \Gamma_2)$, denoted $tw(f)$, as follows. If $untiming(f) \notin \Gamma_1(\Gamma_2\Gamma_1)^*$, 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 $\mathcal{A}$ defines a language of functions $F(\mathcal{A})$. We are however more interested in the timed language it accepts, denoted $L(\mathcal{A})$, and defined to be $tw(F(\mathcal{A}))$.

Here is a concrete example of a *CIDA* over the set of "eventual operators" $Op = \{\Diamond_a \,|\, a \in \Sigma\}$. The diagram 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 is no time point from which there is an a or a b at a distance of one time unit in the future.



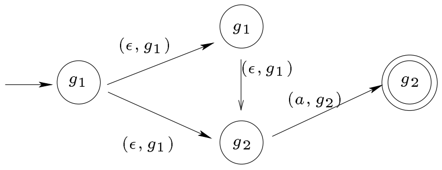
We now define *proper CIDA*'s which are a time-deterministic form of *CIDA*'s, and which we will use to define our counter-free *CIDA*'s. Let G be a finite set of atomic guards over Op. We call (Γ_1, Γ_2) the *proper* symbolic alphabet over (Σ, Op) based on G, if $\Gamma_1 = (\Sigma \cup \{\epsilon\})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

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 $\gamma \in \Gamma_1(\Gamma_2\Gamma_1)^*$ 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 3.1 ([3]) *CIDA's over* (Σ, Op) *and fully canonical proper CIDA's over* (Σ, Op) *define the same class of timed languages.* □

The class of counter-free *CIDA*'s we are interested in this paper is the class of counter-free *CIDA*'s over (Σ, Op), is the class of fully canonical proper *CIDA*'s over (Σ, Op) whose underlying ST-NFA is counter-free. We denote this class by $CFCIDA(\Sigma, Op)$.

As an example, let $\Sigma = \{a\}$, $Op = \{\Diamond_a\}$ and $G = \{\Diamond_a^{[1,1]}\}$. The *CFCIDA* over (Σ, Op) alongside recognizes timed words comprising exactly one a, which occurs in the interval $[1, 2]$. In the diagram, $g_1 = \{\Diamond_a^{[1,2]}\}$ and $g_2 = \emptyset$.



4 Counter-free ST-NFA's and FO-definable functions

In this section we show that over a partitioned alphabet (A_1, A_2), the class of first-order definable languages of finitely-varying functions (for a natural FO logic we will introduce soon) is precisely the class of function languages defined by counter-free ST-NFA's over (A_1, A_2).

For an alphabet A, the syntax of the first order logic $\mathrm{FO}^c(A)$, is given by:

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

where $a \in A$, and x and y are variables.

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 each variable 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. 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 φ (a formula without free variables) in $\mathrm{FO}^c(A)$, the interpretation does not play any role, and we set the language of functions defined by φ to be $F(\varphi) = \{f \in FVF(A) \mid f \models \varphi\}$.

As an example, the formula $\varphi_{cont} = \exists y \exists z(y < x \wedge x < z \wedge \bigvee_{a \in A} \forall u(y < u \wedge u < z \Rightarrow Q_a(u) \wedge Q_a(x)))$ asserts that the point x is a point of continuity. As another example, for the partitioned symbolic alphabet (Γ_1, Γ_2) based on some (Σ, Op), the $\mathrm{FO}^c(\Gamma_1 \cup \Gamma_2)$ formula $\varphi_{fc} = \forall x(\varphi_{disc}(x) \Rightarrow \neg(\bigvee_{(\epsilon,g) \in \Gamma_1} Q_{(\epsilon,g)}(x) \wedge \exists y \exists z(y < x \wedge x < z \wedge \forall u(u \neq x \wedge y < u \wedge u < z \Rightarrow Q_g(u)))))$, (where $\varphi_{disc} = \neg\varphi_{cont}$) asserts that the untiming of the function is fully canonical.

For a partitioned alphabet (A_1, A_2) we call a finitely varying function f in $FVF(A_1 \cup A_2)$ *alternating* if $untiming(f) \in A_1(A_2A_1)^*$ (thus the discontinuities are labelled by symbols in A_1 and continuities by labels in A_2). Let *alt*-$FVF(A_1, A_2)$ denote the class of alternating finitely varying functions over (A_1, A_2).

Theorem 4.1 *Let (A_1, A_2) be a partitioned alphabet with $A = A_1 \cup A_2$. Then the class of* $\mathrm{FO}^c(A)$*-definable languages of alternating finitely-varying functions over (A_1, A_2) is precisely the class of function languages definable by counter-free* ST-NFA*'s over (A_1, A_2).*

The rest of this section is devoted to a proof of this theorem. We recall briefly the logic LTL and its two interpretations, one over discrete words and the other over functions. The syntax of $\mathrm{LTL}(A)$ is given by:

$$\theta ::= a \mid (\theta U \theta) \mid (\theta S \theta) \mid \neg\theta \mid (\theta \vee \theta),$$

where $a \in A$. The logic is interpreted over words in A^*, with the following semantics. Given a word $w = a_0 \cdots a_n$ in A^* and a position $i \in \{0, \ldots, n\}$, we say $w, i \models a$ iff $a_i = a$; and $w, i \models \theta U \eta$ iff there exists j such that $i < j \leq n$, $w, j \models \eta$ and for all k such that $i < k < j$, $w, k \models \theta$. The "since" operator S is defined in a symmetric way to U in the past, and the boolean operators in the usual way. We denote by $L_{sym}(\theta)$ the set $\{w \in A^* \mid w, 0 \models \theta\}$.

The logic LTL can also be interpreted over functions as done in [9]. Here we restrict the models to finitely-varying functions in $FVF(A)$, and we denote this logic by $\mathrm{LTL}^c(A)$. Given a function $f \in FVF(A)$, $t \in [0, dur(f)]$ and $\theta \in \mathrm{LTL}^c(A)$, the satisfaction relation $f, t \models \theta$ is defined as follows:

$f, t \models a$	iff	$f(t) = a$.
$f, t \models \theta U \eta$	iff	$\exists t' : t < t' \leq dur(f),\ f, t' \models \eta$, and $\forall t'' : t < t'' < t', f, t'' \models \theta$.
$f, t \models \theta S \eta$	iff	$\exists t' : 0 \leq t' < t,\ f, t' \models \eta, \forall t'' : t' < t'' < t,\ f, t'' \models \theta$.

The boolean operators are interpreted in the expected way. We set $F(\theta) = \{f \in FVF(A) \mid f, 0 \models \theta\}$. As an example, the $\mathrm{LTL}^c(A)$ formulas $\theta_{cont} = \bigvee_{a \in A}(a \wedge (aSa) \wedge (aUa))$ and $\theta_{disc} = \neg\theta_{cont}$ characterize the points of continuity and discontinuity respectively in a function over A.

Returning now to the proof of Theorem 4.1, the route we follow is given schematically in the figure below.

$$\begin{tikzcd}
\overset{\mathcal{A}_\varphi}{\text{CF-ST-NFA}(A_1, A_2)} \arrow[dr, "(b)"', "\mathcal{A}"] & & & & \overset{\varphi}{\mathrm{FO}^c(A)} \arrow[llll, "(a)"'] \\
& \text{CF-NFA}(A) \arrow[dr, "\text{McN–P+Kamp (c)}"', "\mathcal{B}"] & & \mathrm{LTL}^c(A) \arrow[ur, "\text{Kamp (e)}"', "\varphi_{\mathcal{A}}"] & \\
& & \overset{\theta}{\mathrm{LTL}(A)} \arrow[ur, "(d)"', "\widehat{\theta}"] & &
\end{tikzcd}$$

Step (a): Let φ be a sentence in $\mathrm{FO}^c(A)$. We show how to construct a counter-free ST-NFA $\mathcal{A}_\varphi$ over A, such that $F(\mathcal{A}_\varphi) = F(\varphi)$.

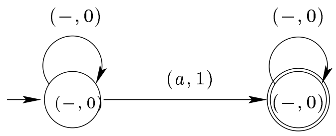
The proof proceeds in a similar manner to the one in [3]. We will represent models of formulas with free variables in them, as functions with the interpretations built into them. We assume an ordering on the countable set of first-order variables given by $x_1, x_2, \cdots$. For a formula φ with free 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.

Proposition 4.1 *Let φ be an $\mathrm{FO}^c(A)$ formula with free variables X and let $\mathcal{A}$ be a counter-free ST-NFA accepting the X-models of φ. Then for any set of variables X' which contains X, we can construct a counter-free ST-NFA $\mathcal{A}'$ accepting precisely the X'-models of φ.* □

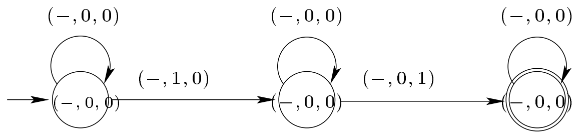
Lemma 4.1 *Let φ be an $\mathrm{FO}^c(A)$ formula and let X be the set of free variables in it. Then we can construct a counter-free* ST-NFA $\mathcal{A}_\varphi^X$ *which accepts precisely the X-models of φ.*

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. For a set of variables Y, let $\mathcal{A}_{valid}^Y$ denote the ST-NFA which accepts all "valid" Y-models. It is easy to construct this ST-NFA and to check that it is counter-free. We construct the counter-free ST-NFA $\mathcal{A}_\varphi^X$ by induction on the structure of φ.

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



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



3. $\varphi = \neg\psi$: Let $\mathcal{A}_\psi^X$ be the automaton for ψ, where X is the set of free variables in ψ. Then $\mathcal{A}_\varphi^X$ is the intersection of $\mathcal{A}_{valid}^X$ with the counter-free ST-NFA that recognizes the complement of the function language of $\mathcal{A}_\psi^X$ (cf. Prop 2.3).

4. $\varphi = \psi \vee \nu$: Let $\mathcal{A}_\psi^{X'}$ be the counter-free ST-NFA for ψ, where X' is the set of free variables in ψ, and let $\mathcal{A}_\nu^{X''}$ be the counter-free ST-NFA for ν, where X'' is the set of free variables in ν. Let $X = X' \cup X''$. By Prop. 4.1 we obtain ST-NFA's $\mathcal{A}_\psi^X$ and $\mathcal{A}_\nu^X$. Then $\mathcal{A}_\varphi^X$ is the union of $\mathcal{A}_\psi^X$ and $\mathcal{A}_\nu^X$.

5. $\varphi = \exists x\psi$: Let X' be the set of free variables in ψ so that $X = X' - \{x\}$. Let $\mathcal{A}_\psi^{X'}$ be a counter-free ST-NFA for ψ. Without loss of generality we can assume $\mathcal{A}_\psi^{X'}$ has no "useless" states (i.e. those which cannot be reached from the start state or cannot reach a final state). Now we simply project away the component corresponding to x in the symbols on the transitions of $\mathcal{A}_\psi^{X'}$ to obtain the required counter-free ST-NFA $\mathcal{A}_\varphi^X$. It is easy to see that $\mathcal{A}_\varphi^X$ must be counter-free, since if it had a counter, the counter must contain a transition with a 1 in the x-component in the original ST-NFA $\mathcal{A}_\psi^{X'}$. But then by our assumption on the structure of $\mathcal{A}_\psi^{X'}$, it would accept non-valid X' models having multiple 1's in the x-component.

□

From the above lemma it now follows that for a sentence $\varphi \in \mathrm{FO}^c(A)$ we have a counter-free ST-NFA $\mathcal{A}_\varphi$ such that $F(\varphi) = F(\mathcal{A}_\varphi)$. In particular, if we are interested in the alternating function language of φ, we can conjunct φ with the $\mathrm{FO}^c(A)$ formula $\varphi_{alt} = \forall x((\varphi_{disc} \Rightarrow \bigvee_{a\in A_1} Q_a(x)) \wedge (\varphi_{cont} \Rightarrow \bigvee_{a\in A_2} Q_a(x)))$ which forces models to be alternating. The resulting ST-NFA will also be alternating.

Steps (b) to (d) prove that we can go from an arbitrary counter-free ST-NFA $\mathcal{A}$ over the partitioned alphabet (A_1, A_2) to an equivalent $\mathrm{FO}^c(A)$-sentence $\varphi_{\mathcal{A}}$.

Step (b): By Prop. 2.2, for a counter-free ST-NFA $\mathcal{A}$ over (A_1, A_2) we can give a classical counter-free NFA $\mathcal{B}$ such that $L_{sym}(\mathcal{A}) = L_{sym}(\mathcal{B})$.

Step (c): For a counter-free NFA $\mathcal{B}$, by the McNaughton-Papert result [10] we can give an $\mathrm{FO}(A)$ formula ψ, where the logic $\mathrm{FO}(A)$ is the discrete version of $\mathrm{FO}^c(A)$ defined in a similar manner to $\mathrm{LTL}(A)$, such that $L_{sym}(\psi) = L_{sym}(\mathcal{B})$. From Kamp's result for discrete LTL [9], we have an equivalent $\mathrm{LTL}(A)$ formula θ such that $L_{sym}(\psi) = L_{sym}(\theta)$.

Step (d): For a formula θ in $\mathrm{LTL}(A)$ we construct a formula *ltl-ltlc*(θ) in $\mathrm{LTL}^c(A)$ which is such that $F(\textit{ltl-ltlc}(\theta)) = \textit{timing}(L_{sym}(\theta))$.

We will use the abbreviation $\theta_1 U_d \theta_2$ to mean that at a point of discontinuity "$\theta_1 U \theta_2$" is true in an untimed sense, and define it to be $(\theta_2 U \theta_2) \vee (\theta_1 U(\theta_{disc} \wedge (\theta_2 \vee (\theta_1 \wedge (\theta_2 U \theta_2)))))$. Symmetrically we use $\theta_1 S_d \theta_2$ for $(\theta_2 S \theta_2) \vee (\theta_1 S(\theta_{disc} \wedge (\theta_2 \vee (\theta_1 \wedge (\theta_2 S \theta_2)))))$.

The translation *ltl-ltlc* is defined as follows (we use $\widehat{\eta}$ for *ltl-ltlc*(η) for brevity):

$$
\begin{array}{lcl}
\textit{ltl-ltlc}(a) & = & a. \\
\textit{ltl-ltlc}(\neg\theta_1) & = & \neg\widehat{\theta}_1. \\
\textit{ltl-ltlc}(\theta_1 \vee \theta_2) & = & \widehat{\theta}_1 \vee \widehat{\theta}_2. \\
\textit{ltl-ltlc}(\theta_1 U \theta_2) & = & (\theta_{disc} \Rightarrow (\widehat{\theta}_1 U_d \widehat{\theta}_2)) \wedge \\
 & & \qquad (\theta_{cont} \Rightarrow (\theta_{cont}\, U(\theta_{disc} \wedge (\widehat{\theta}_2 \vee (\widehat{\theta_1} \wedge (\widehat{\theta}_1 U_d \widehat{\theta}_2)))))). \\
\textit{ltl-ltlc}(\theta_1 S \theta_2) & = & (\theta_{disc} \Rightarrow (\widehat{\theta}_1 S_d \widehat{\theta}_2)) \wedge \\
 & & \qquad (\theta_{cont} \Rightarrow (\theta_{cont}\, S(\theta_{disc} \wedge (\widehat{\theta}_2 \vee (\widehat{\theta_1} \wedge (\widehat{\theta}_1 S_d \widehat{\theta}_2)))))).
\end{array}
$$

Lemma 4.2 *Let θ be an $\mathrm{LTL}(A)$ formula. Let w be a canonical word in A^*. Let $f \in \textit{timing}(w)$ 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 $t \in I_i$, we have $w, i \models \theta \iff f, t \models \textit{ltl-ltlc}(\theta)$.* $\square$

From the above lemma it follows that $F(\textit{ltl-ltlc}(\theta)) = \textit{timing}(L_{sym}(\theta))$.

Step (e): Using Kamp's theorem [9] for a given $\mathrm{LTL}^c(A)$ formula $\widehat{\theta}$ we can give an equivalent $\mathrm{FO}^c(A)$ formula φ such that $F(\widehat{\theta}) = F(\varphi)$.

To summarize this direction of the proof: given a counter-free ST-NFA $\mathcal{A}$ over (A_1, A_2) by steps (b) and (c) we have an $\mathrm{LTL}(A)$ formula θ such that $L_{sym}(\mathcal{A}) = L_{sym}(\theta)$. By steps (c) and (d) we have an $\mathrm{FO}^c(A)$ formula $\varphi_{\mathcal{A}}$ such that $\textit{timing}(L_{sym}(\theta)) = F(\varphi_{\mathcal{A}})$. It follows that $F(\mathcal{A}) = F(\varphi_{\mathcal{A}})$. Further, by the alternating nature of the symbolic language of $\mathcal{A}$ it follows that the function models and alternating function models of $\varphi_{\mathcal{A}}$ are the same.

This completes the proof of Theorem 4.1. $\square$

5 Counter-free *CIDA*'s and TFO^c

We can now prove the main result of this paper which is a general characterization of timed first-order definable languages (again, for a natural first-order logic based on input-determined operators) via counter-free *CIDA*'s.

We recall the definition of the *continuous timed first-order logic* (TFO^c) based on (Σ, Op) from [3]. The syntax of the logic $\mathrm{TFO}^c(\Sigma, \mathit{Op})$ is given by:

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

where $a \in \Sigma$, $\Delta \in Op$, $I \in \mathcal{I}_{\mathbb{Q}_{\geq 0}}$, and x and y are first-order variables.

The logic is interpreted over timed words in $T\Sigma^*$, in a way similar to the logic FO^c. 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 say $\sigma, \mathbb{I} \models \mathrm{Q}_a(x)$ iff $\exists i : a_i = a$, and $t_i = \mathbb{I}(x)$; $\sigma, \mathbb{I} \models \Delta^I(x)$ iff $\Delta(\sigma, \mathbb{I}(x)) \cap I \neq \emptyset$; and the rest of the cases are similar to that of the logic FO^c defined in the previous section. For a sentence φ in $\mathrm{TFO}^c(\Sigma, Op)$, the timed language defined by φ, denoted $L(\varphi)$, is defined to be $\{\sigma \in T\Sigma^* \mid \sigma \models \varphi\}$.

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 definable by a $\mathrm{TFO}^c(\Sigma, Op)$ sentence iff it is definable by a CFCIDA over (Σ, Op).*

Proof We first show how to go from TFO^c to *CFCIDA*. The route we take is shown in the figure alongside:

$$\begin{tikzcd}
CFCIDA(\Sigma, Op)\ \ \mathcal{A}_{\widehat{\varphi}} & \widehat{\varphi}\ \ \mathrm{FO}^c(\Gamma) \arrow[l, "\text{Thm. 4.1}"'] \\
& \varphi\ \ \mathrm{TFO}^c(\Sigma, Op) \arrow[u, "tfo\text{-}fo\ [3]"']
\end{tikzcd}$$

Let φ be a $\mathrm{TFO}^c(\Sigma, Op)$ sentence. Then there is a proper symbolic alphabet (Γ_1, Γ_2) over (Σ, Op) and a $\mathrm{FO}^c(\Gamma_1 \cup \Gamma_2)$ sentence $\widehat{\varphi}$ such that $L(\varphi) = tw(F(\widehat{\varphi}))$. The symbolic alphabet (Γ_1, Γ_2) is based on the set of guards $\{\Delta^I | \Delta^I(x)$ is a subformula of $\varphi\}$. The formula $\widehat{\varphi}$ 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_{fcp}, which is satisfied by functions whose untimings are fully canonical proper words.

From Theorem 4.1, we have a counter-free ST-NFA $\mathcal{A}_{\widehat{\varphi}}$ over (Γ_1, Γ_2) such that $F(\mathcal{A}_{\widehat{\varphi}}) = F(\widehat{\varphi})$. By construction of $\widehat{\varphi}$ it follows that $\mathcal{A}_{\widehat{\varphi}}$ is a fully canonical proper *CIDA* over (Σ, Op), and hence a *CFCIDA* over (Σ, Op). Since $L(\mathcal{A}_{\widehat{\varphi}}) = tw(F(\mathcal{A}_{\widehat{\varphi}})) = tw(F(\widehat{\varphi})) = L(\varphi)$, we are done.

In the converse direction, the route we follow is:

$$\begin{tikzcd}
CFCIDA(\Sigma, Op)\ \ \mathcal{A} \arrow[r, "\text{Thm. 4.1}"] & \varphi_{\mathcal{A}}\ \ \mathrm{FO}^c(\Gamma) \arrow[d, "fo\text{-}tfo\ [3]"] \\
& \widehat{\varphi}_{\mathcal{A}}\ \ \mathrm{TFO}^c(\Sigma, Op)
\end{tikzcd}$$

Let $\mathcal{A}$ be a *CFCIDA* over (Σ, Op). Thus $\mathcal{A}$ is a counter-free ST-NFA over a proper alphabet (Γ_1, Γ_2) based on (Σ, Op), which accepts a fully canonical function language. By Theorem 4.1 we have a $\mathrm{FO}^c(\Gamma)$ sentence $\varphi_{\mathcal{A}}$ (where $\Gamma = \Gamma_1 \cup \Gamma_2$), such that $F(\varphi_{\mathcal{A}}) = F(\mathcal{A})$. We now use the translation *fo-tfo* from [3] which simply "unpacks" a formula φ in $\mathrm{FO}^c(\Gamma)$ to a formula $\widehat{\varphi}$ in $\mathrm{TFO}^c(\Sigma, Op)$ such that $L(\widehat{\varphi}) = tw(F(\varphi))$. Thus we take $\widehat{\varphi}_{\mathcal{A}}$ to be *fo-tfo*$(\varphi_{\mathcal{A}})$, and we have that $L(\mathcal{A}) = tw(F(\mathcal{A})) = tw(F(\varphi_{\mathcal{A}})) = L(\widehat{\varphi}_{\mathcal{A}})$. $\Box$

6 Counter-free Recursive *CIDA*'s

Our aim is now to extend the counter-free characterization of first-order definable timed languages to "recursive" (or "hierarchical") first-order logic and *CIDA*'s. This will give us a counter-free *CIDA* characterization for many of the timed temporal logics defined in the literature, including MTL^c and MITL.

We begin with a few preliminaries, mostly from [3]. 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-CFIDA*(Σ, Rop), to be the class of timed languages definable by counter-free fully canonical proper *rec-CIDA*'s over (Σ, Rop).

We now introduce the recursive version of TFO^c. 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: $\varphi ::= Q_a(x) \,|\, \Delta_\psi^I(x) \,|\, x < y \,|\, \neg\varphi \,|\, \varphi \vee \varphi \,|\, \exists x\varphi$, where $a \in \Sigma$, $\Delta \in Rop$, $I \in \mathcal{I}_{\mathbb{Q}_{\geq 0}}$ 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 $L^{fl}(\psi) = \{(\sigma, t) \,|\, \sigma, [t/z] \models \psi\}$. A *rec-TMSO*$^c(\Sigma, Rop)$ sentence φ defines the language $L(\varphi) = \{\sigma \in T\Sigma^* \,|\, \sigma \models \varphi\}$.

Using a similar technique to the proof of Theorem 5.1 we can show that *rec-TFO*^{c}-definable and *rec-CFIDA*-definable languages are the same:

Theorem 6.1 *Let Σ be an alphabet and Rop be a set of finitely-varying recursive operators based on Σ. Then a timed language $L \subseteq T\Sigma^*$ is definable by a rec-TFO$^c(\Sigma, Rop)$ sentence iff it is definable by a rec-CFIDA over (Σ, Rop).* □

We recall the definition of the recursive timed temporal logic based on (Σ, Rop) from [3], denoted *rec*-$\mathrm{TLTL}^c(\Sigma, Rop)$. The syntax of the logic is given by

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

where $a \in \Sigma$, $\Delta \in Rop$ and $I \in \mathcal{I}_{\mathbb{Q}_{\geq 0}}$. The logic is interpreted over timed words in a manner similar to TLTL^c, where the operator Δ_θ is defined by $\Delta_\theta(\sigma, t) = \Delta(L^{fl}(\theta), \sigma, t)$, and $L^{fl}(\theta) = \{(\sigma, t) \,|\, \sigma, t \models \theta\}$. From [3] we know that:

Theorem 6.2 ([3]) *Let Σ be an alphabet and Rop be a set of finitely-varying recursive operators based on Σ. Then a timed language $L \subseteq T\Sigma^*$ is definable by a rec-TFO$^c(\Sigma, Rop)$ sentence iff it is definable by a rec-$\mathrm{TLTL}^c(\Sigma, Rop)$ formula.* □

Putting Theorems 6.1 and 6.2 together we obtain counter-free *CIDA* characterizations for many timed temporal logics based on input-determined operators, proposed in the literature. In particular we obtain a counter-free *CIDA* characterization for the logic MTL^c (with past operators). Recall that the syntax of the logic $\mathrm{MTL}^c(\Sigma)$ is:

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

where $a \in \Sigma$ and $I \in \mathcal{I}_{\mathbb{Q}_{\geq 0}}$. The logic is interpreted over timed words in $T\Sigma^*$, and the modalities U_I (and symmetrically S_I) is as follows:

$$\sigma, t \models \theta U_I \eta \quad \text{iff} \quad \exists t' \geq t : t' - t \in I, \sigma, t' \models \eta, \text{ and } \forall t'' : t < t'' < t', \sigma, t'' \models \theta.$$

We recall that MTL^c was shown to be expressively equivalent to the logic *rec*-$\mathrm{TLTL}^c(\Sigma, \{\diamond, \overleftarrow{\diamond}\})$ in [3], where the recursive operators $\diamond$ and $\overleftarrow{\diamond}$ are defined as $\diamond(M, \sigma, t) = \{t' - t \,|\, t' \geq t, t \in pos(M, \sigma)\}$ and $\overleftarrow{\diamond}(M, \sigma, t) = \{t - t' \,|\, t' \leq t, t \in pos(M, \sigma)\}$. Thus we have:

Theorem 6.3 *The class of timed languages definable by* $\mathrm{MTL}^c(\Sigma)$ *and rec-CFIDA*$(\Sigma, \{\Diamond, \diamondminus\})$ *are the same.*

Restricting to non-singular intervals we obtain a similar result for the logic MITL^c [1].

7 Ultimate stability of MTL^c

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 with respect to periodic sequences of timed words. In this section we assume that Σ is an alphabet and *Rop* a set of finitely-varying recursive operators.

A *periodic sequence* of timed words $\langle \sigma_i \rangle$ is of the form $uw, uvw, uv^2w, \cdots$ for some timed words u, v and w in $T\Sigma^*$. We represent $\langle \sigma_i \rangle$ above via the triple (u, v, w). A language of timed words $L \subseteq T\Sigma^*$ is said to be *ultimately stable* w.r.t. a periodic sequence $\langle \sigma_i \rangle$ if there exists $i_0 \in \mathbb{N}$ such that either $\forall i \geq i_0,\ \sigma_i \in L$ or $\forall i \geq i_0,\ \sigma_i \notin L$. The language L is said to be *ultimately stable* if it is ultimately stable w.r.t. all periodic sequences of timed words.

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

The rest of this section is devoted to the proof of the above theorem. By theorem 6.3 we just need to show that all languages recognized by *rec-CFIDA* 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 $\langle \sigma_i \rangle = (u, v, w)$ if there exist a middle zone Z and an index i_0 such that the following conditions hold:

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

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

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

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 σ we note γ_σ^Γ the unique symbolic word $\gamma \in \Gamma^*$ such that $\sigma \in tw(\gamma)$.

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 $\langle\sigma_i\rangle$ be a periodic sequence. If for all $g \in G$, g is well-behaved w.r.t. $\langle\sigma_i\rangle$ then there exists an integer i_0 and $\gamma_1, \gamma_2, \gamma_3 \in \Gamma^*$ such that for all $i \geq i_0$, $\gamma^{\Gamma}_{\sigma_i} = \gamma_1 \gamma_2^{i-i_0} \gamma_3$.* □

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

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 $\Diamond^I_B$ and $\diamondminus^I_B$ are also well-behaved.*

Proof Let $\langle\sigma_i\rangle = (u, v, w)$ a periodic sequence, we show that $\Diamond^I_B$ is well-behaved w.r.t. $\langle\sigma_i\rangle$, the case of $\diamondminus^I_B$ is similar. We have to show that there exists a middle zone $Z = (l, r)$ and an index i_0 such that:

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

$$\forall i \geq i_0\ \forall i' \geq i\ \forall t < l\ \sigma_i, t \models \Diamond^I_{\mathcal{B}} \Leftrightarrow \sigma_{i'}, t \models \Diamond^I_{\mathcal{B}}. \tag{2}$$

$$\forall i \geq i_0\ \forall i' \geq i\ \forall t < r\ \sigma_i, dur(\sigma_i) - t \models \Diamond^I_{\mathcal{B}} \Leftrightarrow \sigma_{i'}, dur(\sigma_{i'}) - t \models \Diamond^I_{\mathcal{B}}. \tag{3}$$

$\mathcal{B}$ 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 $Z = (l_{\mathcal{B}}, r_{\mathcal{B}} + M + dur(v))$.

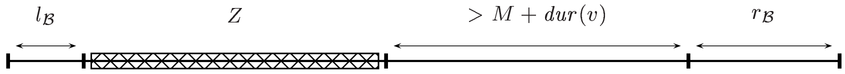
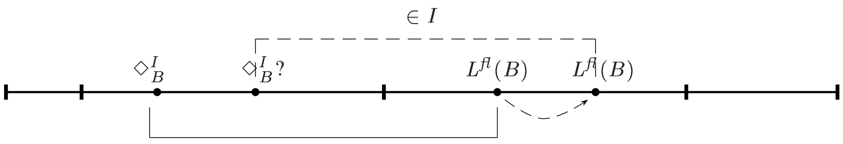


Figure 2: Choice of Z

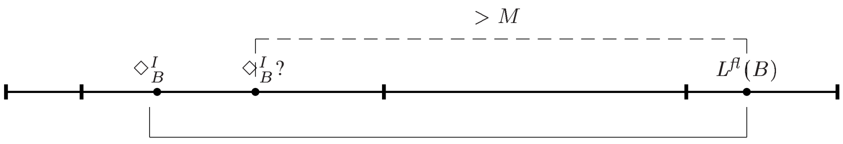
Let i, t be such that $t \in Z(\sigma_i)$ and suppose $\sigma_i, t \models \Diamond^I_{\mathcal{B}}$. We show that $\sigma_i, t + dur(v) \models \Diamond^I_{\mathcal{B}}$ (other direction is similar).

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

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



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



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

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 $\sigma_i, t \models \Diamond^I_{\mathcal{B}}$. Then $\exists t_1 \geq t$ with $(\sigma_i, t_1) \in L^{fl}(\mathcal{B})$ and $t_1 - t \in I$. We distinguish two cases:

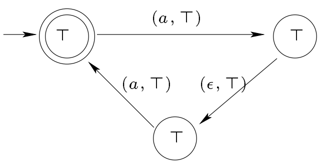
- $t_1 \leq dur(\sigma_i) - r$
  As $\mathcal{B}$ is well-behaved $(\sigma_{i'}, t_1) \in L^{fl}(\mathcal{B})$ and so $\sigma_{i'}, t \models \Diamond^I_{\mathcal{B}}$

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

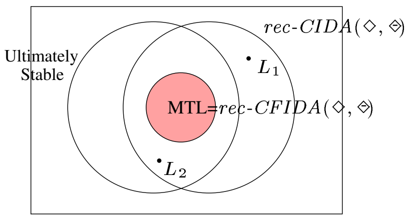
  Set $t'_1 = dur(\sigma_i) - t_1$. $(\sigma_i, dur(\sigma_i) - t'_1) \in L^{fl}(\mathcal{B})$ so as $\mathcal{B}$ is well-behaved $(\sigma_{i'}, dur(\sigma_{i'}) - t'_1) \in L^{fl}(\mathcal{B})$. Moreover $(dur(\sigma_{i'}) - t'_1) - t \geq (dur(\sigma_i) - t'_1) - t = t_1 - t > M$ thus $\sigma_{i'},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

Returning to the proof of theorem 7.1: let L be recognized by some *rec-CFIDA* $\mathcal{A}$ and $\langle \sigma_i \rangle$ 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 $i \geq i_0$, $\gamma^{\Gamma}_{\sigma_i} = \gamma_1 \gamma_2^{i-i_0} \gamma_3$. 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 σ_i ends in the same state. Thus L is ultimately stable w.r.t. $\langle \sigma_i \rangle$. $\square$

We justify here why *CFCIDA*'s were defined to include only fully canonical proper words. Had we allowed words which are not fully canonical, *CFCIDA*'s would not have been equivalent to TFO^c. Alongside is a proper *CIDA* which is not fully canonical but the underlying ST-NFA is counter-free. It accepts the timed language L_1 consisting of timed words with even number of a's. This language is not ultimately stable with respect to the periodic sequence $(\epsilon, (a, 1), \epsilon)$, and hence is not definable in MTL^c and therefore not definable in $\mathrm{TFO}^c(\{a\}, \{\Diamond_a\})$.



We also note that, unlike classical LTL, ultimate stability of a *rec-CIDA*$(\Diamond, \diamondminus)$ language is not a sufficient condition for MTL^c recognizability. Consider the timed language L_2 consisting of timed words ending with an a at time 1 and having even number of b's in the interval $(0, 1)$. This language is recognized by a *rec-CIDA* over $\{\Diamond\}$. However it can be shown to be inexpressible in MTL^c and hence not recognized by a *rec-CFIDA* over $\{\Diamond\}$. Nevertheless it is trivially ultimately stable.



The Venn diagram alongside shows the different classes of timed languages.

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] Fabrice Chevalier, Deepak D'Souza, and Pavithra Prabhakar. Counter-free input-determined timed automata. Technical Report IISc-CSA-TR-2007-1, Indian Institute of Science, Bangalore 560012, India, January 2007. URL: `http://archive.csa.iisc.ernet.in/TR/2007/1/`.

[6] 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.

[7] 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.

[8] 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.

[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.