On continuous timed automata with input-determined guards *

Fabrice Chevalier Deepak D'Souza Pavithra Prabhakar

IISc-CSA-TR-2006-7
http://archive.csa.iisc.ernet.in/TR/2006/7/

Computer Science and Automation
Indian Institute of Science, India

June 2006

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

On continuous timed automata with input-determined guards *

Fabrice Chevalier† Deepak D'Souza‡ Pavithra Prabhakar§

Abstract

We consider a general class of timed automata parameterized by a set of "input-determined" operators, in a continuous time setting. We show that for any such set of operators, we have a monadic second order logic characterization of the class of timed languages accepted by the corresponding class of automata. Further, we consider natural timed temporal logics based on these operators, and show that they are expressively equivalent to the first-order fragment of the corresponding MSO logics. As a corollary of these general results we obtain an expressive completeness result for the continuous version of MTL.

1 Introduction

Timed automata are a popular model of real-time systems, introduced by Alur and Dill in the early nineties [1]. Since then there have been several variants of these automata based on *input-determined* guards [2, 15, 6, 4]. Unlike the explicit clock based guards of timed automata, an input-determined guard is based on a distance operator whose value is completely determined by the input timed word and a time point in it. This property leads to robust logical properties including closure under complementation which timed automata lack. A good example of an input-determined operator is the event-recording operator $\lhd_a$ of [2] which measures the distance to the last time an event a occurred. Similarly the "eventual" operator $\Diamond_a$ [5, 4] inspired by the well-known timed logic Metric Temporal Logic (MTL) [9, 2, 10], measures the time to "some" future occurrence of an a event.

*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

§CSA, IISc, Bangalre, India pavithra@csa.iisc.ernet.in

There have been two natural ways of employing these operators in automata and logical formalisms in the literature. One is the traditional "pointwise" interpretation in which guards are asserted only at "action-points" in a timed word. The other is the so-called "continuous" interpretation in which assertions can be made at *any* time point along the timed word. The two interpretations are well illustrated by the MTL formula $\Diamond\Diamond_{[1,1]}a$ which states that there is a point in future such that an a occurs exactly one time unit later. In the pointwise semantics, the formula is not satisfied by the timed word which comprises a b at time 1 followed by an a at time 3, but is satisfied in the continuous semantics. In general, the continuous semantics is strictly more expressive than the pointwise semantics [3, 11].

In the pointwise semantics, the work in [5] provides a general framework for showing determinizability, closure properties, and monadic second-order (MSO) logic characterizations, for classes of timed automata based on input-determined operators, called input-determined automata (IDA's). It also identifies natural timed temporal logics based on these operators which are expressively complete with respect to the corresponding automata classes.

In this paper we show a similar general framework for the continuous semantics. Thus we first define an appropriate "continuous" version of these automata called continuous input-determined automata (CIDA's) which are parameterized by a set of input-determined operators. These CIDA's extend IDA's by allowing epsilon-transitions and state invariants. We show that these classes of automata are determinizable and closed under boolean operations. They also admit logical characterizations via natural MSO logics based on the input-determined operators, and interpreted over continuous time. Further, the continuous version of the natural timed temporal logics based on these operators are shown to be expressively complete, in that they correspond to the first-order fragments of the associated MSO logics. These results generalize to the corresponding *recursive* formalisms where the input-determined operators take as arguments logical formulas or "floating" automata, as originally used in the work of [7].

This framework can be used as a general technique for showing such results for any class of automata and logics based on input-determined operators. In particular, the results of [7] for the class of recursive event clock automata (ECA's), (which correspond in our framework to the class of recursive CIDA's based on the operators $\rhd$ and $\lhd$), pertaining to the MSO characterization via the logic MinMaxML and the expressive completeness of recursive Event Clock Temporal Logic (ECTL), follow as corollaries of our results.

As a new application, we obtain an expressive completeness result for MTL in the continuous semantics. MTL can be viewed as the recursive timed temporal logic based on the operator $\Diamond$, and hence corresponds to the first-order fragment of recursive CIDA's and the MSO based on the operator $\Diamond$.

The techniques used to prove our results are similar to [5] in that we also make use of the notion of *proper* alphabets. These alphabets help in determinizing CIDA's and showing closure properties. For the MSO characterization we use proper alphabets to translate formulas into a continuous version of Büchi's MSO logic, which preserves, in a sense, the original models of the formula. Now we need to make use of the fact that the "untiming" of continuous MSO formulas is regular in order to obtain a CIDA for the original MSO formula. We give an automata-theoretic proof of this result which was independently proved by Rabinovich in [12] using a translation to classical MSO. For the expressive completeness result concerning our timed temporal logics we factor through the well-known result of Kamp for classical LTL [8].

The technique used in [7, 14] for event clock automata is similar in that they factor through Kamp's theorem to prove their expressive completeness result. However the MSO characterization is obtained differently by showing that quantified ECTL is expressively equivalent to recursive ECA's.

In this paper we deal with finite timed words, though the results can be easily extended to infinite words as well.

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 make use of the standard notations for regular expressions, with '$\cdot$' for concatenation and '$*$' for Kleene closure.

A *finite state automaton (FSA)* $\mathcal{A}$ over a finite 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 set of transitions, and $F \subseteq Q$ is the set of final states. A run ρ of $\mathcal{A}$ on a word $w = a_1 \cdots a_n \in A^*$ is a mapping from $\{0, \cdots, n\} \to Q$ such that $(\rho(i), a_{i+1}, \rho(i+1)) \in \delta$ for each $i < n$, and $\rho(0) = s$. The run is accepting if $\rho(n) \in F$. The 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.

We denote the set of non-negative and positive real numbers by $\mathbb{R}_{\geq 0}$ and $\mathbb{R}_{>0}$, respectively. We use $\mathcal{I}_{\mathbb{R}_{\geq 0}}$ to denote the set of intervals, where an interval is a convex subset of $\mathbb{R}_{\geq 0}$. Two interval 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 denote r by $length(f)$. We call f a *finitely varying* function over A, if there exist a word $a_0 a_1 \cdots a_{2n}$ in A^*, and an interval sequence $I_0 I_1 \cdots I_{2n}$, 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 canonical interval representation. We define $\mathit{func}(A)$ to be the set of all finitely varying functions over A.

Let $f \in \mathit{func}(A)$ 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 $\mathit{untiming}(f)$, is defined as $a_0\cdots a_{2n}$. 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: $\mathit{timing}(w) = \emptyset$ if $|w|$ is even, otherwise $f \in \mathit{timing}(w)$ if $w = a_0a_1\cdots a_{2n}$ 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 $\mathit{untiming}$ to languages of functions in the expected way.

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 $t \in [0, \mathit{length}(\sigma)]$. 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 $t \in [0, \mathit{length}(\sigma)]$, denoted $\sigma, t \models g$, as $\sigma, t \models \Delta^I$ iff $\Delta(\sigma, t) \cap I \neq \emptyset$, and in the usual way for the boolean operators. For example Δ_r, 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 time points in σ after t at which an event a occurs, and the event-recording operator $\triangleleft_a$ which maps (σ, t) to the set containing the time point which corresponds to the last occurrence of the event a before time t.

We call an input-determined operator Δ over Σ *finitely varying* if for all $\sigma \in T\Sigma^*$ and $I \in \mathcal{I}_{\mathbb{Q}}$, the function $f_\Delta : [0, \mathit{length}(\sigma)] \to \{0, 1\}$ defined as, $f_\Delta(t)$ is 1 if $\sigma, t \models \Delta^I$, and 0 otherwise, is finitely varying. The operators $\Diamond_a$ and $\triangleleft_a$ are finitely varying, whereas Δ_r 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 $\mathit{func}(\Gamma_1 \cup \Gamma_2)$, denoted $\mathit{tw}(f)$, as follows. If $\mathit{untiming}(f) \notin \Gamma_1 \cdot (\Gamma_2 \cdot \Gamma_1)^*$, then $\mathit{tw}(f) = \emptyset$. Otherwise, a timed word $\sigma = (a_1, t_1)\cdots(a_n, t_n)$ is in $\mathit{tw}(f)$, provided for all $t \in [0, \mathit{length}(f)]$,

- $f(t) = (a, g)$, for some $a \in \Sigma$ and $g \in \mathcal{G}(Op)$, if there exists i such that $i \in \{1, \cdots, n\}$, $t_i = t$ and $a_i = a$, and if $\sigma, t \models g$, and
- $f(t) = (\epsilon, g)$ or g, for some $g \in \mathcal{G}(Op)$, if there does not exist i such that

$i \in \{1, \cdots, n\}$, $t_i = t$, and if $\sigma, t \models g$.

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

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 $\Gamma_2 = 2^G$. A proper word is a word over a proper symbolic alphabet. Further we call a proper word γ over $(\Gamma_1 \cup \Gamma_2)$ *fully canonical*, if $\gamma \in \Gamma_1 \cdot (\Gamma_2 \cdot \Gamma_1)^*$ and no subword of γ is of the form $g \cdot (\epsilon, g) \cdot g$. If $f \in func(\Gamma_1 \cup \Gamma_2)$, then we associate with it the set of timed words obtained by interpreting $g \subseteq G$ as the guard $\bigwedge_{h \in g} h \wedge \bigwedge_{h \in G-g} \neg h$.

Example 2.1 *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* $tw(f_2) = \emptyset$.

3 Continuous Input Determined Automata

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 a structure (Q, s, δ, F, inv) on a symbolic alphabet (Γ_1, Γ_2) over (Σ, Op), where

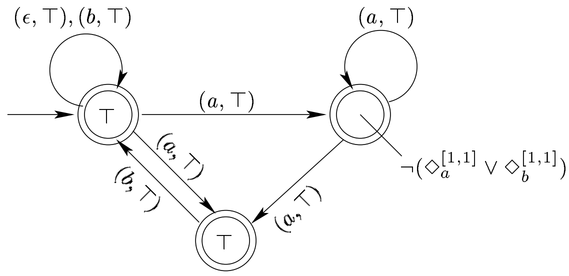
- Q is a finite set of states,
- $s \in Q$ is the start state,
- $\delta \subseteq Q \times \Gamma_1 \times Q$ is the transition relation,
- $inv : Q \to \Gamma_2$ is the labelling function for the states, and
- $F \subseteq Q$ is the set of accepting states.

We now define the *symbolic language* accepted by the *CIDA* $\mathcal{A}$. Let $\gamma \in \Gamma_1 \cdot (\Gamma_2 \cdot \Gamma_1)^*$ and let $\gamma = \gamma_0 \gamma_1 \cdots \gamma_{2n}$. Let $N = \{0, \cdots, n+1\}$. A run of $\mathcal{A}$ over γ is a map $\rho : N \to Q$ such that

- $\rho(0) = s$,
- $(\rho(i), \gamma_{2i}, \rho(i+1)) \in \delta$ for $i = 0, \cdots, n$, and
- $inv(\rho(i)) = \gamma_{2i-1}$ for all $1 \leq i \leq n$.

We say ρ is accepting if $\rho(n+1) \in F$. The symbolic language defined by $\mathcal{A}$, denoted $L_{sym}(\mathcal{A})$, is the set of words in $\Gamma_1 \cdot (\Gamma_2 \cdot \Gamma_1)^*$ over which $\mathcal{A}$ has an accepting run. We define the language of functions accepted by the *CIDA* $\mathcal{A}$, denoted $F(\mathcal{A})$, as $timing(L_{sym}(\mathcal{A}))$. The timed language of the *CIDA* $\mathcal{A}$, denoted $L(\mathcal{A})$, is defined as $tw(F(\mathcal{A}))$.

We give below a concrete example of a *CIDA*, which we call *Continuous Eventual Timed Automata* (*CETA*). A *CETA* over an alphabet Σ is a *CIDA* over (Σ, Op), where $Op = \{\Diamond_a \mid a \in \Sigma\}$ is the set of eventual operators based on Σ. The diagram below gives a *CETA* over $\{a, b\}$ which recognizes the language L_{ni} (for "no insertion"), which consists of timed words in which between any two consecutive a's, there does not exist a time point from which at time distance one in the future there is an a or a b.



We define a *proper CIDA* to be a structure similar to *CIDA* except that it is over a proper symbolic alphabet instead of a symbolic alphabet. The language of a proper *CIDA* is defined as for *CIDA*'s. We call a proper *CIDA fully canonical* if its symbolic language consists of fully canonical proper words. We show below the closure of *CIDA*'s under the boolean operations. Let Σ be an alphabet and Op be a set of finitely varying operators.

Lemma 3.1 *CIDA's over* (Σ, Op) *and proper CIDA's over* (Σ, Op) *define the same class of timed languages.*

Proof Given a proper *CIDA* over a proper symbolic alphabet (Γ_1, Γ_2) based on a set of atomic guards G, we can get a *CIDA* over (Γ'_1, Γ'_2), where Γ'_1 and Γ'_2 are obtained from Γ_1 and Γ_2 by replacing a subset g of G in them by $\bigwedge_{h \in g} h \wedge \bigwedge_{h \in G-g} \neg h$.

We now show that for every *CIDA* $\mathcal{A} = (Q, s, \delta, F, inv)$ we can give a proper *CIDA* $\mathcal{A}'$ such that $L(\mathcal{A}) = L(\mathcal{A}')$. Let $\mathcal{A}$ be based on a symbolic alphabet (Γ_1, Γ_2) and let G be the set of atomic guards in it. Let (Γ'_1, Γ'_2) be the proper symbolic alphabet over (Σ, Op) based on G. We construct $\mathcal{A}'$ based on (Γ'_1, Γ'_2) as follows. $\mathcal{A}' = (Q', s', \delta', F', inv')$, where
$Q' = \{s'\} \cup \{(q, g) \mid q \in Q, g \subseteq G, g \Rightarrow inv(q)\}$,
$\delta' = \{(s', (c, g_1), (q, g_2)) \mid (s, (c, g), q) \in \delta, g_1 \Rightarrow g\} \cup$

$\{((p, g_1), (c, g_2), (q, g_3)) \mid (p, (c, g), q) \in \delta, g_2 \Rightarrow g\} \cup$
$\{((p, g_1), (\epsilon, g_2), (p, g_3)) \mid g_2 \Rightarrow inv(p)\}$,
$F' = \{(f, g) \in Q' \mid f \in F\}$, and $inv'(s') = \top$ and $inv'((q, g)) = g$. □

Lemma 3.2 *Proper CIDA's over* (Σ, Op) *and fully canonical proper CIDA's over* (Σ, Op) *define the same class of timed languages.*

Proof Given a proper *CIDA* $\mathcal{A}$ based on a proper symbolic alphabet (Γ_1, Γ_2) we give a fully canonical proper *CIDA* $\mathcal{A}'$ based on the same proper symbolic alphabet such that $L(\mathcal{A}) = L(\mathcal{A}')$.

Let $\mathcal{A} = (Q, s, \delta, F, inv)$. We first compute the smallest set δ' such that $\delta \subseteq \delta'$, and if $(q, (\epsilon, g), r) \in \delta'$ such that $inv(q) = inv(r) = g$, then for all $(p, \gamma, q) \in \delta'$, $(p, \gamma, r) \in \delta'$. This set is computable since one can add transitions one at a time until no more can be added and this process terminates since the total number of transitions that can be added is finite. We now argue that adding such a transition does not change the language of the proper *CIDA*. Suppose γ is a new word in its symbolic language, then it has run over γ which takes the newly added transition zero or more times. Then there is a run in the original automaton over γ' got by replacing every $g_1(c, g_2)g_3$ corresponding to the newly added transition by $g_1(c, g_2)g_3(\epsilon, g_3)g_3$. But it is easy to see that the $tw(\gamma) = tw(\gamma')$. Hence the newly added word does not add any more timed words to the language of the proper *CIDA*.

We now remove the transitions of the form $(p, (\epsilon, g), q)$ where $inv(p) = inv(q)$ = g. We can argue similarly that the removal of a transition leaves the language of the proper *CIDA* unchanged. Since no word in the symbolic language of the final automaton has $g(\epsilon, g)g$ as a substring, it can be seen that the resulting *CIDA* accepts only fully canonical proper words. □

We define a *CIDA-language* L over (Σ, Op) to be a regular subset of $\Gamma_1 \cdot (\Gamma_2 \cdot \Gamma_1)^*$ for some proper symbolic alphabet (Γ_1, Γ_2) over (Σ, Op).

Proposition 3.1 *Let* (Γ_1, Γ_2) *be a proper symbolic alphabet over* (Σ, Op), *and let L be a language over* $(\Gamma_1 \cup \Gamma_2)$. *Then L is the symbolic language of a proper CIDA iff L is a CIDA-language.*

Theorem 3.1 *The class of CIDA's over* (Σ, Op) *is closed under union, intersection and complementation.*

Proof From lemma 3.1, it follows that it is enough to show that the class of proper *CIDA*'s is closed under the boolean operations.

Union of proper CIDA's is equivalent to the union of their symbolic languages. Since their symbolic languages are CIDA-languages and CIDA-languages are closed under union, proper CIDA's are themselves closed under union.

For complementation, using 3.2 we can give an equivalent fully canonical proper CIDA $\mathcal{A}'$ for a given proper CIDA $\mathcal{A}$. But the set of timed words associated with two distinct fully canonical proper words is disjoint. Hence we can complement the timed language of $\mathcal{A}'$ by complementing its symbolic language with respect to the set of fully canonical proper words (which is a CIDA-language). $\square$

4 Continuous Monadic Second Order Logic

In this section, we interpret Büchi's monadic second order logic over finitely varying functions and show that the untiming of a language of functions definable in the logic is regular.

Recall that for an alphabet A, Büchi's monadic second order logic (denoted here by $\mathrm{MSO}^c(A)$) is given as follows:

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

where $a \in A$, and x and X are first and second order variables, respectively. We use the convention that the small letters are first order variables and capital letters are second order variables.

We interpret a formula of the logic over a finitely varying function f in $\mathit{func}(A)$, along with an interpretation $\mathbb{I}$ with respect to f, which assigns to a first order variable x, a value in $[0, \mathit{length}(f)]$, and to a set variable X, a finite subset of $[0, \mathit{length}(f)]$. We use $X \subseteq_{\mathit{fin}} Y$ to denote that X is a finite subset of Y. The relation $<$ is interpreted as the usual ordering of $\mathbb{R}_{\geq 0}$ and the predicate Q_a as a boolean function which maps a real number to true iff f maps the same to a.

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. Similarly, $\mathbb{I}[B/X]$ denotes the modification of $\mathbb{I}$ which maps the set variable X to B and the rest to the same as that by $\mathbb{I}$. We also use the notation $[t/x]$ to denote an interpretation which sends x to t when the rest of the interpretation is irrelevant.

We now define the semantics of $\mathrm{MSO}^c(A)$. Given a formula $\varphi \in \mathrm{MSO}^c(A)$, $f \in \mathit{func}(A)$ 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 as:

$$
\begin{array}{lll}
f, \mathbb{I} \models \mathrm{Q}_a(x) & \text{iff} & f(\mathbb{I}(x)) = a. \\
f, \mathbb{I} \models x \in X & \text{iff} & \mathbb{I}(x) \in \mathbb{I}(X). \\
f, \mathbb{I} \models x < y & \text{iff} & \mathbb{I}(x) < \mathbb{I}(y). \\
f, \mathbb{I} \models \neg\varphi & \text{iff} & f, \mathbb{I} \not\models \varphi. \\
f, \mathbb{I} \models \varphi_1 \vee \varphi_2 & \text{iff} & f, \mathbb{I} \models \varphi_1 \text{ or } f, \mathbb{I} \models \varphi_2. \\
f, \mathbb{I} \models \exists x\varphi & \text{iff} & \exists t \in [0, \mathit{length}(f)] : f, \mathbb{I}[t/x] \models \varphi. \\
f, \mathbb{I} \models \exists X\varphi & \text{iff} & \exists B \subseteq_{\mathit{fin}} [0, \mathit{length}(f)] : f, \mathbb{I}[B/X] \models \varphi.
\end{array}
$$

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

The following theorem relates FSA's and MSO^c.

Theorem 4.1 *Given a sentence φ in $\mathrm{MSO}^c(A)$, we can give a finite state automaton $\mathcal{A}_\varphi$ such that $F(\varphi) = \mathit{timing}(L_{\mathit{sym}}(\mathcal{A}_\varphi))$.*

In the rest of the section we present a proof of the above theorem. Let $X = (x_1, \cdots, x_n)$ and $Y = (X_1, \cdots, X_m)$. Let $f \in \mathit{func}(A)$ and let $\mathbb{I}$ be an interpretation to the variables in X and Y with respect to f. We define an (X, Y)-extension of f with respect to $\mathbb{I}$ to be a function from $[0, \mathit{length}(f)]$ to $A \times \{0, 1\}^{n+m}$ (A if $n + m$ = 0), denoted $f_{\mathbb{I}}^{(X,Y)}$, as: $f_{\mathbb{I}}^{(X,Y)}(t) = (f(t), i_1, i_2, \cdots, i_n, j_1, j_2, \cdots, j_m)$, where $i_k = 1$ if $\mathbb{I}(x_k) = t$, and 0 otherwise, and $j_k = 1$ if $t \in \mathbb{I}(X_k)$, and 0 otherwise. A function f' is an (X, Y)-extension of f if $f' = f_{\mathbb{I}}^{(X,Y)}$ for some $\mathbb{I}$. Let X' and Y' be such that they contain the variables in X and Y. We call a function f_1 an (X, Y, X', Y')-extension of f_2 if there exist $f, \mathbb{I}$ an interpretation to (X, Y) and $\mathbb{I}'$ an interpretation to (X', Y') which agrees with $\mathbb{I}$ on the common variables, such that $f_2 = f_{\mathbb{I}}^{(X,Y)}$ and $f_1 = f_{\mathbb{I}'}^{(X',Y')}$. We will prove the following lemma which will help us in the construction of $\mathcal{A}_\varphi$.

Lemma 4.1 *Let $X = (x_1, \cdots, x_n)$ and $Y = (X_1, \cdots, X_m)$. Let $\mathcal{A}$ be an automaton over $A \times \{0, 1\}^{n+m}$ accepting $\mathit{untiming}(L_1)$ where L_1 is a subset of the (X, Y)-extensions of the functions in func(A). Let $X' = (x_1, \cdots, x_n, x_{n+1})$. Then we can give an automaton $\mathcal{A}'$ over $A \times \{0, 1\}^{n+1+m}$ accepting $\mathit{untiming}(L_2)$ where L_2 is the set of (X, Y, X', Y)-extensions of the functions in L_1.*

Proof Let $\mathcal{A} = (Q, s, \delta, F)$.

We first construct an automaton $\mathcal{A}_1$, which consists of two copies of the states of the above automaton and transitions only between state of different copies, and accepts the set of odd length words accepted by the original automaton (any word which is the untiming of some f is of odd length). $\mathcal{A}_1 = (Q_1, s_1, \delta_1, F_1)$, where $Q_1 = Q \times \{0,1\}$, $s_1 = (s,0)$, $\delta_1 = \{((p,0),a,(q,1)) \mid (p,a,q) \in \delta\} \cup \{((p,1),a,(q,0)) \mid (p,a,q) \in \delta\}$ and $F_1 = F \times \{1\}$.

Next we give an automaton $\mathcal{A}_2$ such that if $w = a_0 b_1 a_1 \cdots b_l a_l$ is accepted by $\mathcal{A}_1$ then every w' obtained by replacing b_i by $b_i^{k_i}$, where k_i is odd, is accepted by $\mathcal{A}_2$. Let $\delta_o = \{((p,1),a,(q,0)) \in \delta_1\}$. $\mathcal{A}_2 = (Q_2, s_2, \delta_2, F_2)$, where $Q_2 = Q_1 \cup (\delta_o \times \{0,1\})$, $s_2 = s_1$, $\delta_2 = \delta_1 \cup \{(p,a,(e,0)) \mid e = (p,a,q), e \in \delta_o\} \cup \{((e,1),a,q) \mid e = (p,a,q), e \in \delta_o\} \cup \{((e,0),a,(e,1)) \mid e = (p,a,q), e \in \delta_o\} \cup \{((e,1),a,(e,0)) \mid e = (p,a,q), e \in \delta_o\}$, and $F_2 = F_1$.

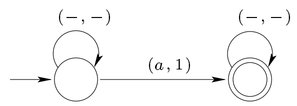
Finally we construct the automaton $\mathcal{A}'$ by replacing every transition labelled by a symbol from $A \times \{0,1\}^{n+m}$ in $\mathcal{A}_2$ by transitions corresponding to every possible extension of the symbol. $\mathcal{A}' = (Q', s', \delta', F')$ where $Q' = Q$, $s' = s$, $\delta' = \{(p,(i_1,\cdots,i_n,k,j_1,\cdots,j_m),q) \mid (p,(i_1,\cdots,i_n,j_1,\cdots,j_m),q) \in \delta_2, k \in \{0,1\}\}$, and $F' = F$. $\square$

We note that we can give an automaton in general which accepts the untimings of all (X,Y,X',Y')-extensions of a language containing (X,Y)-extensions of functions whose untiming is given as an automaton.

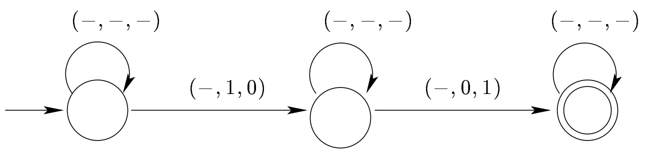
We now construct the automaton for $\varphi \in \mathrm{MSO}^c(A)$, inductively. Let $X = (x_1, x_2, \cdots, x_n)$ and $Y = (X_1, X_2, \cdots, X_m)$. We denote by $\mathcal{A}_\varphi^{X,Y}$ an automaton whose symbolic language is $\{untiming(f_{\mathbb{I}}^{(X,Y)}) \mid f, \mathbb{I} \models \varphi\}$.

The inductive construction below gives an automaton $\mathcal{A}_\varphi^{X,Y}$ for φ where X and Y consist of the free variables in φ. Let $\mathcal{A}_{canon}^i$ be the automaton which accepts canonical words over $A \times \{0,1\}^i$. We use $|X|$ to denote the number of elements in X.

1. $\varphi = Q_a(x)$: The automaton $\mathcal{A}_\varphi^{X,Y}$ corresponding to $X = (x)$ and $Y = \emptyset$ is the intersection of $\mathcal{A}_{canon}^1$ with:

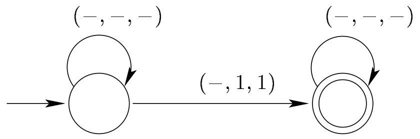


2. $\varphi = x < y$: The automaton $\mathcal{A}_\varphi^{X,Y}$ corresponding to $X = (x,y)$ and $Y = \emptyset$ is the intersection of $\mathcal{A}_{canon}^2$ with:



We can also give one for $X = (y,x)$.

3. $\varphi = x \in X_i$: The automaton $\mathcal{A}_\varphi^{X,Y}$ corresponding to $X = (x)$ and $Y = (X_i)$ is the intersection of $\mathcal{A}_{canon}^2$ with:



4. $\varphi = \neg\eta$: Let $\mathcal{A}_\eta^{X,Y}$ be the automaton for η, where (X, Y) consists of the free variables in η. Then $\mathcal{A}_\varphi^{X,Y}$ is the intersection of $\mathcal{A}_{canon}^{|X|+|Y|}$ with complement of $\mathcal{A}_\eta^{X,Y}$.

5. $\varphi = \eta \vee \psi$: Let $\mathcal{A}_\eta^{X',Y'}$ be the automaton for η where (X', Y') are the free variable in η and let $\mathcal{A}_\psi^{X'',Y''}$ be the automaton for ψ where (X'', Y'') are the free variable in ψ. Let (X, Y) be the free variables in either (X', Y') or (X'', Y''). We can use the construction in lemma 4.1 to get the automata $\mathcal{A}_\eta^{X,Y}$ and $\mathcal{A}_\psi^{X,Y}$. Then $\mathcal{A}_\varphi^{X,Y}$ is the intersection of $\mathcal{A}_{canon}^{|X|+|Y|}$ with the union of $\mathcal{A}_\eta^{X,Y}$ and $\mathcal{A}_\psi^{X,Y}$.

6. $\varphi = \exists x\eta$: Let $\mathcal{A}_\eta^{X',Y}$ be the automaton for η, where (X', Y) are the free variables in η and $X' = (x, x_1, \cdots, x_n)$. Let $X = (x_1, \cdots, x_n)$. We first intersect $\mathcal{A}_\eta^{X',Y}$ with $\mathcal{A}_{valid}$, which accepts all words in which the x-component has a single 1 and the 1 occurs at an even position (assuming indices start from 0), to obtain the automaton $\mathcal{A}$.

   Let w be a word of odd length over A and w' in A^*. Then $w' = canonical(w)$ if $w' = a_0 b_1 a_1 \cdots b_n a_n$ is a canonical word and w can be obtained from w' by replacing each b_i by $b_i^{k_i}$ where k_i is odd.

   Given a word w over $A \times \{0,1\}^{n+m+1}$, we define $proj(w)$ to be a word over $A \times \{0,1\}^{n+m}$, which projects away its x-component. Similarly for an interpretation $\mathbb{I}$ with respect to (X', Y), $proj(\mathbb{I})$ is an interpretation with respect to (X, Y) which agrees with $\mathbb{I}$ on the common variables.

   Proposition 4.1 *Let $f \in func(A)$ and $\mathbb{I}$ be an interpretation of the variables in (X', Y) with respect to f. Then $untiming(f_{proj(\mathbb{I})}) = canonical(proj(untiming(f_{\mathbb{I}})))$.*

   We first project away the x-components of the labels on the transitions in $\mathcal{A}$. We then use the construction in lemma 4.1 used for the construction of $\mathcal{A}_1$ to get the automaton $\mathcal{A}'$. We now add transitions $((p,1), a, (q,0))$ if there exist transitions $((p,1),a,(r,0))$, $((r,0),a,(t,1))$ and $((t,1),a,(q,0))$, repeatedly as done in lemma 3.2 to get $\mathcal{A}''$. We intersect it with $\mathcal{A}_{canon}^{|X|+|Y|}$ to get $\mathcal{A}_{\varphi}^{X,Y}$.

7. $\varphi = \exists Z\eta$: Let $\mathcal{A}_{\eta}^{X,Y'}$ be the automaton for η, where (X,Y') are the free variables in η and $Y' = (Z, X_1, \cdots, X_n)$. Let $Y = (X_1, \cdots, X_n)$. The construction is similar to that done for the previous case except that now the automaton $\mathcal{A}_{valid}$ needs to be replaced by one which checks that a 1 always occurs at an even position. We intersect it with $\mathcal{A}_{canon}^{|X|+|Y|}$ to get $\mathcal{A}_{\varphi}^{X,Y}$.

5 A logical characterization of *CIDA*'s

In this section we give a logical characterization of *CIDA*'s in terms of a monadic second order logic parameterized by a set of input-determined operators. Let Σ be an alphabet and Op be a set of input determined operators over Σ. We define the syntax of *continuous timed monadic second order logic* over (Σ, Op), denoted $\mathrm{TMSO}^c(\Sigma, Op)$, as follows. The formulas of $\mathrm{TMSO}^c(\Sigma, Op)$ are inductively defined as:

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

where $a \in \Sigma$, $\Delta \in Op$, $I \in \mathcal{I}_{\mathbb{Q}}$, and x and X are first and second order variables.

We interpret the logic over timed words in $T\Sigma^*$. Given a formula $\varphi \in \mathrm{TMSO}^c$ (Σ, Op), a timed word $\sigma = (a_1,t_1)\cdots(a_n,t_n)$ in $T\Sigma^*$, and an interpretation $\mathbb{I}$ to the variables in φ with respect to σ, which maps a first order variable x to $t \in [0, \mathit{length}(\sigma)]$ and a second order variable X to $B \subseteq_{\mathit{fin}} [0, \mathit{length}(\sigma)]$, we define the satisfaction relation $\sigma, \mathbb{I} \models \varphi$, inductively as follows:

$$\begin{array}{lll}
\sigma,\mathbb{I} \models \mathrm{Q}_a(x) & \text{iff} & \exists i : a_i = a, t_i = \mathbb{I}(x). \\
\sigma,\mathbb{I} \models \Delta^I(x) & \text{iff} & \Delta(\sigma, \mathbb{I}(x)) \cap I \neq \emptyset. \\
\sigma,\mathbb{I} \models x \in X & \text{iff} & \mathbb{I}(x) \in \mathbb{I}(X). \\
\sigma,\mathbb{I} \models x < y & \text{iff} & \mathbb{I}(x) < \mathbb{I}(y). \\
\sigma,\mathbb{I} \models \neg\varphi & \text{iff} & \sigma,\mathbb{I} \not\models \varphi. \\
\sigma,\mathbb{I} \models \varphi_1 \vee \varphi_2 & \text{iff} & \sigma,\mathbb{I} \models \varphi_1 \text{ or } \sigma,\mathbb{I} \models \varphi_2. \\
\sigma,\mathbb{I} \models \exists x\varphi & \text{iff} & \exists t \in [0, \mathit{length}(\sigma)],\ \sigma, \mathbb{I}[t/x] \models \varphi. \\
\sigma,\mathbb{I} \models \exists X\varphi & \text{iff} & \exists B \subseteq_{\mathit{fin}} [0, \mathit{length}(\sigma)] :\ \sigma, \mathbb{I}[B/X] \models \varphi.
\end{array}$$

For a sentence φ in $\mathrm{TMSO}^c(\Sigma, Op)$, we set the timed language defined by φ to be $L(\varphi) = \{\sigma \in \mathrm{T}\Sigma^* \mid \sigma \models \varphi\}$.

We now show that TMSO^c characterizes *CIDA*'s.

Theorem 5.1 *Let Σ be a finite alphabet and Op be a set of finitely varying input-determined operators based on Σ. Let L be a timed language over Σ. Then L is accepted by a CIDA over (Σ, Op) iff it is definable by a $\mathrm{TMSO}^c(\Sigma, Op)$ sentence.*

We devote the rest of the section for a proof of the above theorem. As a proof of the forward direction, we show that the class of languages defined by proper $CIDA$'s over (Σ, Op) is a subset of the class of languages defined by $\mathrm{TMSO}^c(\Sigma, Op)$ sentences.

Let $\mathcal{A} = (Q, s, \delta, F, inv)$ be a proper $CIDA$ over (Σ, Op) based on a set of atomic guards G. Let the proper alphabet associated with it be (Γ_1, Γ_2). We give a formula $\varphi_{\mathcal{A}}$ such that $L(\mathcal{A}) = L(\varphi_{\mathcal{A}})$. The formula essentially checks for the existence of a valid run of $\mathcal{A}$ over a timed word. Let $\delta = \{e_1, \cdots, e_m\}$ be the set of transitions. We define the set $consec \subseteq \delta \times \delta$, where $(e, e') \in consec$ if and only if there exists q such that $e = (p, \gamma, q)$ and $e' = (q, \gamma', r)$. We define below some abbreviations which will help us in the construction of the formula $\varphi_{\mathcal{A}}$.

- $\mathit{first}(x) = \neg\exists y(y < x)$,
- $\mathit{last}(x) = \neg\exists y(x < y)$,
- $\mathit{next}(x, y, X) = x \in X \wedge y \in X \wedge \neg\exists w(x < w \wedge w < y \wedge w \in X)$, and
- $\mathit{between}(x, y, z) = x < y \wedge y < z$.

We use $\mathit{action}(x)$ for $\bigvee_{a\in\Sigma} Q_a(x)$. Given $g \subseteq G$, we will use $g(x)$ to denote the TMSO^c formula $\bigwedge_{\Delta^I\in g} \Delta^I(x) \wedge \bigwedge_{\Delta^I\in G-g} \neg\Delta^I(x)$. The second order variables $X_{e_1}, \cdots, X_{e_m}$ are used to capture the points in the timed words which correspond to the transitions $e_1, \cdots, e_m$, respectively, and X to capture their union. $\varphi_{\mathcal{A}}$ is given by: $\exists X \exists X_{e_1} \cdots \exists X_{e_m}(\varphi_1 \wedge \varphi_2 \wedge \varphi_3 \wedge \varphi_4 \wedge \varphi_5 \wedge \varphi_6 \wedge \varphi_7 \wedge \varphi_8)$, where:

1. φ_1: It says that X is the union of $X_{e_1}, \cdots, X_{e_m}$.

$$\forall x(\bigvee_{e\in\delta} x \in X_e \Leftrightarrow x \in X).$$

2. φ_2: It guarantees that the sets $X_{e_1}, \cdots, X_{e_m}$ are disjoint.

$$\forall x \bigwedge_{i,j\in\{1,\cdots,m\}, i\neq j} (x \in X_{e_i} \Rightarrow x \notin X_{e_j}).$$

3. φ_3: It ensures that the time instant 0 belongs to the set corresponding to some transition from the start state.

$$\forall x(\mathit{first}(x) \Rightarrow \bigvee_{(s,\gamma,q)\in\delta} x \in X_{(s,\gamma,q)}).$$

4. φ_4: It ensures that the time of the last action of the timed word is in a set corresponding to some transition into the final state.

$$\forall x(last(x) \Rightarrow \bigvee_{(q,\gamma,f)\in\delta, f\in F} x \in X_{(q,\gamma,f)}).$$

5. φ_5: It says that any two consecutive points in the set X belong to consecutive transitions.

$$\forall x \forall y(next(x, y, X) \Rightarrow \bigvee_{e,e'\in consec} (x \in X_e \wedge y \in X_{e'})).$$

6. φ_6: It says that if a point in a timed word corresponds to a transition labelled (a, g) where $a \in \Sigma$, then the timed word at that point contains an a and satisfies the guard g.

$$\forall x \bigwedge_{(p,(a,g),q)\in\delta} (x \in X_{(p,(a,g),q)} \Rightarrow (Q_a(x) \wedge g(x))).$$

7. φ_7: It says that if a point in a timed word corresponds to a transition labelled (ϵ, g), then the timed word at that point does not contain an action and satisfies the guard g.

$$\forall x \bigwedge_{(p,(\epsilon,g),q)\in\delta} (x \in X_{(p,(\epsilon,g),q)} \Rightarrow (\neg action(x) \wedge g(x))).$$

8. φ_8: It says that all points which lie between the points corresponding to two consecutive transition in the run of a timed word should not contain any action and should satisfy the guard on the corresponding state.

$$\forall x \forall y \forall z((next(y, z) \wedge between(y, x, z)) \Rightarrow$$

$$(\bigwedge_{(p,a,q)\in\delta} (y \in X_{(p,a,q)} \Rightarrow (\neg action(x) \wedge [inv(q)](x))))).$$

In the other direction we reduce a TMSO^c formula to an MSO^c formula and then factor through theorem 4.1 to get an FSA over $\Gamma_1 \cup \Gamma_2$.

Let $\sigma \in T\Sigma^*$ and let G be a finite set of atomic guards over Op. We define the function $f_\sigma^G : [0, length(\sigma)] \to 2^G$ as: for all t, $f_\sigma^G(t) = \{h \mid h \in G, \sigma, t \models h\}$. Let $(g_0, I_0)(g_1, I_1) \cdots (g_{2m}, I_{2m})$ be the canonical interval representation of f_σ^G (the operators are finitely varying). Let (Γ_1, Γ_2) be the set of proper guards based on G and let $\Gamma = \Gamma_1 \cup \Gamma_2$. We define the function $f_\sigma^\Gamma : [0, length(\sigma)] \to \Gamma$ as follows. Let $t \in [0, length(\sigma)]$ and let $t \in I_j$. If $t = t_i$ for some $i \in \{i, \cdots, n\}$, then $f_\sigma^G(t) = (a_i, g_j)$, otherwise $f_\sigma^G(t) = (\epsilon, g_j)$ if j is even, and $f_\sigma^G(t) = g_j$ if j is odd. Note that $\sigma \in tw(f_\sigma^\Gamma)$.

Let $\varphi \in \mathrm{TMSO}^c(\Sigma, Op)$ and let $G = \{\Delta^I \mid \Delta^I(x)$ is a subformula of $\varphi\}$. Let (Γ_1, Γ_2) be the proper alphabet over (Σ, Op) based on G, and let $\Gamma = \Gamma_1 \cup \Gamma_2$. We now give the function $tmso\text{-}mso$, which maps a $\mathrm{TMSO}^c(\Sigma, Op)$ formula φ to an $\mathrm{MSO}^c(\Gamma)$ formula. We define $tmso\text{-}mso$, inductively as follows:

$$\begin{array}{lcl}
tmso\text{-}mso(Q_a(x)) & = & \bigvee_{(a,g)\in\Gamma} Q_{(a,g)}(x). \\
tmso\text{-}mso(\Delta^I(x)) & = & \bigvee_{(c,g)\in\Gamma, \Delta^I\in g} Q_{(c,g)}(x) \vee \bigvee_{g\in\Gamma, \Delta^I \in g} Q_g(x). \\
tmso\text{-}mso(x \in X) & = & x \in X. \\
tmso\text{-}mso(x < y) & = & x < y. \\
tmso\text{-}mso(\neg\psi) & = & \neg tmso\text{-}mso(\psi). \\
tmso\text{-}mso(\eta \vee \psi) & = & tmso\text{-}mso(\eta) \vee tmso\text{-}mso(\psi). \\
tmso\text{-}mso(\exists x\, \psi) & = & \exists x\, tmso\text{-}mso(\psi). \\
tmso\text{-}mso(\exists X\, \psi) & = & \exists X\, tmso\text{-}mso(\psi).
\end{array}$$

Lemma 5.1 *Let $\sigma \in T\Sigma^*$ and $\mathbb{I}$ be an interpretation to the variables in φ with respect to σ. Let $f \in func(\Gamma)$ such that $\sigma \in tw(f)$. Then $\sigma, \mathbb{I} \models \varphi$ iff $f, \mathbb{I} \models tmso\text{-}mso(\varphi)$.*

Proof Proof is by a straightforward induction on the structure of φ. □

Theorem 5.2 *Let φ be a sentence in $\mathrm{TMSO}^c(\Sigma, Op)$. Then we have $L(\varphi) = tw(F(tmso\text{-}mso(\varphi)))$.*

Proof Let $tmso\text{-}mso(\varphi) = \tilde{\varphi}$. Suppose $\sigma \in tw(F(\tilde{\varphi}))$. Then there exists f such that $\sigma \in tw(f)$ and $f \in F(\tilde{\varphi})$. Hence $f \models \tilde{\varphi}$, which implies that $\sigma \models \varphi$ from lemma 5.1. Therefore $\sigma \in L(\varphi)$.

Suppose $\sigma \in L(\psi)$. $\sigma \models \varphi$. Since $\sigma \in tw(f_\sigma^\Gamma)$, $f_\sigma^\Gamma \models \tilde{\varphi}$. Hence $f_\sigma^\Gamma \in F(\tilde{\varphi})$. Therefore, $\sigma \in tw(F(\tilde{\varphi}))$. □

We can now complete the proof by taking the route in the diagram below.

$$\begin{tikzcd}
\mathrm{TMSO}^c - \varphi \arrow[r, dotted] \arrow[d] & CIDA - \mathcal{A}' \\
\mathrm{MSO}^c - \tilde{\varphi} \arrow[r] & FSA - \mathcal{A}_{\tilde{\varphi}} \arrow[u]
\end{tikzcd}$$

From theorem 5.2, $L(\varphi) = tw(F(\tilde{\varphi}))$, where $\tilde{\varphi} = tmso\text{-}mso(\varphi)$. By theorem 4.1 there exists an FSA $\mathcal{A}_{\tilde{\varphi}}$ such that $F(L_{sym}(\mathcal{A}_{\tilde{\varphi}})) = F(\tilde{\varphi})$. Hence $L(\varphi) = tw(F(L_{sym}(\mathcal{A}_{\tilde{\varphi}})))$. We can assume that $L_{sym}(\mathcal{A}_{\tilde{\varphi}}) \subseteq \Gamma_1 \cdot (\Gamma_2 \cdot \Gamma_1)^*$ as words not in $\Gamma_1 \cdot (\Gamma_2 \cdot \Gamma_1)^*$ do not have any timed words associated with them. Thus we can give a $CIDA$ $\mathcal{A}'$ such that $L_{sym}(\mathcal{A}') = L_{sym}(\mathcal{A}_{\tilde{\varphi}})$. It now follows that $L(\varphi) = L(\mathcal{A}')$.

6 Continuous Timed Linear Temporal Logic

In this section we identify a natural, expressively complete, timed linear temporal logic based on a set of input-determined operators. 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:

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

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 $t \in [0, length(\sigma)]$. Then the satisfaction relation $\sigma, t \models \varphi$ is given by:

$\sigma, t \models a$	iff	$\exists i : a_i = a, t_i = t.$
$\sigma, t \models \Delta^I$	iff	$\Delta(\sigma, t) \cap I \neq \emptyset.$
$\sigma, t \models \theta U \eta$	iff	$\exists t' : t < t' \leq length(\sigma), \sigma, t' \models \eta, \forall t'' : t < t'' < t', \sigma, t'' \models \theta.$
$\sigma, t \models \theta S \eta$	iff	$\exists t' : 0 \leq t' < t, \sigma, t' \models \eta, \forall t'' : t' < t'' < t, \sigma, t'' \models \theta.$
$\sigma, t \models \neg\theta$	iff	$\sigma, t \not\models \theta.$
$\sigma, t \models (\theta \vee \eta)$	iff	$\sigma, t \models \theta$ or $\sigma, t \models \eta.$

The language defined by a $\mathrm{TLTL}^c(\Sigma, Op)$ formula θ is given by $L(\theta) = \{\sigma \in T\Sigma^* \mid \sigma, 0 \models \theta\}$.

We show that TLTL^c is expressively equivalent to the first order fragment of TMSO^c. Let us denote by $\mathrm{TFO}^c(\Sigma, Op)$ the first order fragment of $\mathrm{TMSO}^c(\Sigma, Op)$ (i,e, the fragment we get by disallowing quantification over set variables). The logics TLTL^c and TFO^c are expressively equivalent in the following sense:

Theorem 6.1 *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* $\mathrm{TFO}^c(\Sigma, Op)$.

Proof ($\Rightarrow$) Given a $\mathrm{TLTL}^c(\Sigma, Op)$ formula θ we can associate with it a $\mathrm{TFO}^c(\Sigma, Op)$ formula φ with a single free variable x, such that $\sigma, t \models \theta$ iff $\sigma, [t/x] \models \varphi$. This can be done in a straightforward inductive manner as follows. For the atomic formulas a and Δ^I, we can take φ to be $Q_a(x)$ and $\Delta^I(x)$ respectively. In the inductive step, assuming we have already translated θ_1 and θ_2 into η and ψ respectively, such that they do not use the variables y and z, we can translate $\theta_1 U \theta_2$ into

$$\exists y(x < y \wedge \psi[y/x] \wedge \forall z((x < z \wedge z < y) \Rightarrow \eta[z/x])).$$

Here $\psi[y/x]$ denotes the standard renaming of the free variable x by y in ψ. We can similarly write the translations for S and the boolean operators. It is not difficult to verify that if φ is the above translation of θ then $\sigma, t \models \theta$ iff $\sigma, [t/x] \models \varphi$. It also follows that $\sigma, 0 \models \theta$ iff σ satisfies the sentence φ_0 given by $\forall x(\mathit{first}(x) \Rightarrow \varphi)$. Hence we have that $L(\theta) = L(\varphi_0)$.

($\Leftarrow$) In the converse direction a more transparent proof is obtained by factoring through Kamp's result for classical LTL^c. Recall that the syntax of $\mathrm{LTL}^c(A)$ is given by:

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

where $a \in A$.

The logic is interpreted over functions $f \in \mathit{func}(A)$. Given $t \in [0, \mathit{length}(f)]$ and $\theta \in \mathrm{LTL}^c(A)$, the satisfaction relation $f, t \models \theta$ is inductively defined as:

$f, t \models a$	iff	$f(t) = a$.
$f, t \models \theta U \eta$	iff	$\exists t' : t < t' \leq \mathit{length}(f), f, t' \models \eta, \forall t'' : t < t'' < t', f, t'' \models \theta$.
$f, t \models \theta S \eta$	iff	$\exists t' : 0 \leq t' < t, f, t' \models \eta, \forall t'' : t' < t'' < t, f, t'' \models \theta$.
$f, t \models \neg\theta$	iff	$f, t \not\models \theta$.
$f, t \models (\theta \vee \eta)$	iff	$f, t \models \theta$ or $f, t \models \eta$.

The language defined by an $\mathrm{LTL}^c(A)$ formula θ is given by $F(\theta) = \{f \in \mathit{func}(A) \,|\, f, 0 \models \theta\}$. Let $\mathrm{FO}^c(A)$ denote the first order fragment of $\mathrm{MSO}^c(A)$. Then the result due to Kamp [8] states that:

Theorem 6.2 ([8]) $\mathrm{LTL}^c(A)$ *is expressively equivalent to* $\mathrm{FO}^c(A)$.

Let φ be a $\mathrm{TFO}^c(\Sigma, Op)$ sentence. Let (Γ_1, Γ_2) be the proper symbolic alphabet over (Σ, Op) based on $G = \{\Delta^I \,|\, \Delta^I(x) \text{ is a subformula of } \varphi\}$. Let $\Gamma = \Gamma_1 \cup \Gamma_2$ and let $\sigma \in T\Sigma^*$. The function *tmso-mso* maps a $\mathrm{TFO}^c(\Sigma, Op)$ formula to an $\mathrm{FO}^c(\Gamma)$ formula. And it follows from lemma 5.1 that $f_\sigma^\Gamma \models \mathit{tmso}\text{-}\mathit{mso}(\varphi)$ iff $\sigma \models \varphi$.

Let $\tilde{\varphi} = \mathit{tmso}\text{-}\mathit{mso}(\varphi)$. By Kamp's result, there exists a mapping *fo-ltl* which maps a $\mathrm{FO}^c(\Gamma)$ formula to an equivalent $\mathrm{LTL}^c(\Gamma)$ formula. Hence $F(\tilde{\varphi}) = F(\mathit{fo}\text{-}\mathit{ltl}(\tilde{\varphi}))$ and therefore $f_\sigma^\Gamma \models \tilde{\varphi}$ iff $f_\sigma^\Gamma \models \mathit{fo}\text{-}\mathit{ltl}(\tilde{\varphi})$.

Let $\theta = \mathit{fo}\text{-}\mathit{ltl}(\tilde{\varphi})$. We define the function *ltl-tltl* which maps an $\mathrm{LTL}^c(\Gamma)$ formula to a $\mathrm{TLTL}^c(\Sigma, Op)$ formula as follows:

Let $a \in \Sigma$, $g \in G$ and $(a, g) \in \Gamma$. Let $\theta_g = \bigwedge_{h \in g} h \wedge \bigwedge_{h \in G-g} \neg h$.

$$\begin{array}{lcl} \mathit{ltl\text{-}tltl}((a,g)) & = & a \wedge \theta_g. \\ \mathit{ltl\text{-}tltl}((\epsilon,g)) & = & \neg \bigvee_{a\in\Sigma} a \wedge \theta_g \wedge \neg(gSg \wedge gUg). \\ \mathit{ltl\text{-}tltl}(g) & = & \neg \bigvee_{a\in\Sigma} a \wedge \theta_g \wedge (gSg \wedge gUg). \\ \mathit{ltl\text{-}tltl}(\theta U \eta) & = & \mathit{ltl\text{-}tltl}(\theta) U \mathit{ltl\text{-}tltl}(\eta). \\ \mathit{ltl\text{-}tltl}(\theta S \eta) & = & \mathit{ltl\text{-}tltl}(\theta) S \mathit{ltl\text{-}tltl}(\eta). \\ \mathit{ltl\text{-}tltl}(\neg\theta) & = & \neg \mathit{ltl\text{-}tltl}(\theta). \\ \mathit{ltl\text{-}tltl}(\theta \vee \eta) & = & \mathit{ltl\text{-}tltl}(\theta) \vee \mathit{ltl\text{-}tltl}(\eta). \end{array}$$

Lemma 6.1 *Let $\sigma \in T\Sigma^*$. Then $f_\sigma^\Gamma, t \models \theta$ iff $\sigma, t \models \mathit{ltl\text{-}tltl}(\theta)$.*

Putting everything together we have $\sigma \models \varphi$ iff $f_\sigma^\Gamma \models \mathit{tmso\text{-}mso}(\varphi)$ iff $f_\sigma^\Gamma \models \mathit{fo\text{-}ltl}(\mathit{tmso\text{-}mso}(\varphi))$ iff $\sigma \models \mathit{ltl\text{-}tltl}(\mathit{fo\text{-}ltl}(\mathit{tmso\text{-}mso}(\varphi)))$. Hence we have, $L(\varphi) = L(\mathit{ltl\text{-}tltl}(\mathit{fo\text{-}ltl}(\mathit{tmso\text{-}mso}(\varphi))))$. $\square$

7 Recursive continuous input determined automata

We now consider "recursive" *CIDA*'s. The main motivation is to increase the expressive power of our automata, as well as to characterize the expressiveness of recursive temporal logics which occur naturally in the real-time settings.

We define a *recursive* input-determined operator Δ over an alphabet Σ as a partial function from $(2^{\mathbb{R}_{\geq 0}} \times T\Sigma^* \times \mathbb{R}_{\geq 0})$ to $2^{\mathbb{R}_{\geq 0}}$, which is defined for tuples (X, σ, t) where $X \subseteq \mathbb{R}_{\geq 0}$, $\sigma \in T\Sigma^*$ and $t \in [0, \mathit{length}(\sigma)]$. Given a recursive operator Δ and a set $X \subseteq \mathbb{R}_{\geq 0}$, We denote by Δ_X, the operator whose semantics is given by $\Delta_X(\sigma, t) = \Delta(X, \sigma, t)$. We call a set X finitely varying if there exists a finitely varying function $f : [0, r] \to \{0, 1\}$ such that $X \subseteq [0, r]$ and $f(t) = 1$ if and only if $t \in X$. We call a recursive operator Δ *finitely varying* if for every finitely varying set X, Δ_X is a finitely varying operator.

Given a timed word σ in $T\Sigma^*$ and a $t \in [0, \mathit{length}(\sigma)]$ we call the pair (σ, t) a *floating timed word* over Σ. A floating timed language is then a set of floating timed words. We will use the notation Σ' for $(\Sigma \cup \{\epsilon\}) \times \{0, 1\}$. Given $\sigma' \in T\Sigma'^*$, we denote by σ 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. A timed word σ' over the alphabet Σ' which contains exactly one symbol from $(\Sigma \cup \{\epsilon\}) \times \{1\}$, and whose last symbol is from $\Sigma \times \{0, 1\}$, defines 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 (σ, t), and extend it to apply to timed languages over Σ' in a 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 $fw(L(\mathcal{B}))$.

We define the recursive continuous input determined automata (rec-$CIDA$'s) and the floating recursive continuous input determined automata ($frec$-$CIDA$'s) over an alphabet Σ and a set of recursive operators Rop based on Σ, as the union of level i rec-$CIDA$'s and level i $frec$-$CIDA$'s, for all $i \in \mathbb{N}$, respectively.

- A level 0 rec-$CIDA$ $\mathcal{A}$ is a $CIDA$ over Σ that uses only the guard $\top$. It accepts the timed language $L(\mathcal{A})$. A level 0 $frec$-$CIDA$ $\mathcal{B}$ is a floating $CIDA$ over Σ which uses only the guard $\top$. It accepts the floating language $L^{fl}(\mathcal{B})$.

- Let C be a finite collection of $frec$-$CIDA$'s of level i or less over (Σ, Rop). Let Op be the set of operators $\{\Delta_{\mathcal{B}} \mid \Delta \in Rop, \mathcal{B} \in C\}$, where the semantics of each $\Delta_{\mathcal{B}}$ is defined as follows. Let $pos(\sigma, \mathcal{B}) = \{t \in [0, length(\sigma)] \mid (\sigma, t) \in L^{fl}(\mathcal{B})\}$. Then $\Delta_{\mathcal{B}}(\sigma, t) = \Delta(pos(\sigma, \mathcal{B}), \sigma, t)$. We say that an operator $\Delta_{\mathcal{B}}$ is of level j if $\mathcal{B}$ is a level j $frec$-$CIDA$.

  A level $i+1$ rec-$CIDA(\Sigma, Rop)$ is a $CIDA(\Sigma, Op)$ which uses at least one operator of level i. And a level $i+1$ $frec$-$CIDA(\Sigma, Rop)$ is a floating $CIDA(\Sigma, Op)$ which uses at least one operator of level i.

8 MSO characterization of rec-$CIDA$

In this section we introduce the recursive version of TMSO^c and show that it characterizes the class of timed languages defined by rec-$CIDA$. The logic is parameterised by an alphabet Σ and a set of recursive operators Rop, and denoted by rec-$TMSO^c(\Sigma, Rop)$. Given an alphabet Σ and a set of recursive operators Rop, the set of formulas of rec-$TMSO^c(\Sigma, Rop)$ are defined inductively as:

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

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

The logic is interpreted over timed words in $T\Sigma^*$. If φ contains no predicates of the form "$\Delta^I_\psi(x)$", then $\sigma, \mathbb{I} \models \varphi$ is defined as for TMSO^c. Inductively we assume that $\sigma, \mathbb{I} \models \psi$ is defined where ψ has a single free variable z. Let $pos(\sigma, \psi) = \{t \mid \sigma, [t/z] \models \psi\}$ be the set of interpretations of z which make ψ true in σ. We then consider Δ_ψ as an operator with the semantics $\Delta_\psi(\sigma, t) =$

$\Delta(pos(\sigma, \psi), \sigma, t)$. The rest of the interpretation is similar to TMSO^c. We note that each *rec-*$TMSO^c(\Sigma, Rop)$ formula can be viewed as a $\mathrm{TMSO}^c(\Sigma, Op)$ formula, where Op is the set of Δ_ψ's which have a *top-level* occurrence, i.e., they are not in the scope of any other Δ operator.

We now define the level of a *rec-*$TMSO^c(\Sigma, Rop)$ formula.

- A level 0 *rec-*$TMSO^c(\Sigma, Rop)$ formula is a formula in $\mathrm{TMSO}^c(\Sigma, \emptyset)$.
- Let C be the set of *rec-*$TMSO^c(\Sigma, Rop)$ formulas of level i or less with exactly one free variable z. Then a level $i+1$ *rec-*$TMSO^c(\Sigma, Rop)$ formula is a formula in $\mathrm{TMSO}^c(\Sigma, Op)$, where $Op = \{\Delta_\psi \mid \Delta \in Rop, \psi \in C\}$, which uses at least one level i operator, i.e., operator of the form Δ_ψ where ψ is a level i formula.

Note that a variable z which is free in ψ is not free in the formula $\Delta^I_\psi(x)$. A *rec-*$TMSO^c(\Sigma, Rop)$ sentence φ defines the language $L(\varphi) = \{\sigma \in T\Sigma^* \mid \sigma \models \varphi\}$. A *rec-*$TMSO^c(\Sigma, Rop)$ formula ψ with one free variable z defines the floating language $L^{fl}(\psi) = \{(\sigma, t) \mid \sigma, [t/z] \models \psi\}$.

We first show the equivalence of the floating automata and TMSO^c formulas with one free variable.

Lemma 8.1 *Let L be a floating timed language over an alphabet Σ. Let Op be a set of finitely varying operators based on Σ. Then $L = L^{fl}(\mathcal{B})$ for some floating-CIDA $\mathcal{B}$ over (Σ, Op) iff $L = L^{fl}(\psi)$ for some formula ψ with one free variable.*

Proof $(\Rightarrow)$ Let $\mathcal{B} = (Q, s, \delta, F, inv)$ be a floating *CIDA* over (Σ, Op) such that $L = L^{fl}(\mathcal{B})$. Then $\mathcal{B}$ is a *CIDA* over (Σ', Op'). Let $\delta = \{e_1, \cdots, e_m\}$. We now give the $\mathrm{TMSO}^c(\Sigma, Op)$ formula $\psi_\mathcal{B}$ such that $L^{fl}(\mathcal{B}) = L^{fl}(\psi_\mathcal{B})$. The construction is similar to that of theorem 5.1 except for the following changes. Let $g(x)$ be obtained from g by replacing each Δ'^I by $\Delta^I(x)$. We replace φ_6 and φ_7 by the following and take the conjunction of the resulting formula with φ_9.

$$\varphi_6 : \forall x \bigwedge_{(p,((a,i),g),q)\in\delta} (x \in X_{(p,((a,i),g),q)} \Rightarrow (Q_a(x) \wedge g(x))).$$

$$\varphi_7 : \forall x \bigwedge_{(p,(\epsilon,g),q)\in\delta} (x \in X_{(p,(\epsilon,g),q)} \Rightarrow (\neg action(x) \wedge g(x)))\wedge$$

$$\forall x \bigwedge_{(p,((\epsilon,i),g),q)\in\delta} (x \in X_{(p,((\epsilon,i),g),q)} \Rightarrow (\neg action(x) \wedge g(x))).$$

$$\varphi_9 : \forall x((\bigvee_{(p,((a,1),g),q)\in\delta} x \in X_{(p,((a,1),q),q)}) \leftrightarrow x = z).$$

($\Leftarrow$) In the other direction, let ψ be a formula in $\mathrm{TMSO}^c(\Sigma, \mathit{Op})$ with one free variable z. Using lemma 5.1, we get that $\sigma, [t/z] \models \psi$ iff $f, [t/z] \models \mathit{tmso\text{-}mso}(\psi)$, where $\sigma \in tw(f)$. From the construction in theorem 4.1, we can get an automaton $\mathcal{A}_\psi^{(X,Y)}$, where $X = (z)$ and $Y = ()$, such that it contains the untimings of $f_{[t/z]}^{X,Y}$, where $f, [t/z] \models \psi$. The automaton is over $\Gamma \times \{0, 1\}$, where $\Gamma = \Gamma_1 \cup \Gamma_2$ and (Γ_1, Γ_2) is the proper symbolic alphabet over (Σ, Op) based on the G, the atomic guards in ψ. Every symbol in the alphabet is of the form (c, g, i) or (g, i) where $c \in \Sigma \cup \{\epsilon\}$, $g \subseteq G$ and $i \in \{0, 1\}$. But it can be seen that words which are the untiming of some $f_{\mathbb{I}}^{(X,Y)}$ start and end with symbols of the form (c, g, i) and alternate between (c, g, i) and $(g, 0)$. Hence we intersect the FSA with an automaton which removes such words since this does not alter the floating timed language obtained by first timing the words, then interpreting them as pairs (f, t) and finally extracting the timed words. Hence it accepts the floating timed language of ψ. Next we replace every (c, g, i) by $((c, i), g'')$ and (g, i) (note that we remove (g, i)'s where i is 1) by g'', where g'' is obtained from g by replacing every operator Δ by Δ'. This is a CIDA language over (Σ', Op'). We can give a $\mathit{CIDA}(\Sigma', \mathit{Op}')$ or a floating $\mathit{CIDA}(\Sigma, \mathit{Op})$ whose symbolic language is that. We can now argue that the floating timed language of this floating CIDA is same as that of ψ. $\square$

We now prove the finite variability of the formulas. We call a formula ψ in $\mathrm{TMSO}^c(\Sigma, \mathit{Op})$ with one free variable finitely varying if for every $\sigma \in T\Sigma^*$, $pos(\sigma, \psi)$ is a finitely varying set. If ψ is finitely varying then the operator Δ_ψ, where Δ is a finitely varying recursive operator, is finitely varying.

Lemma 8.2 *Let Σ be an alphabet and Op be a set of finitely varying operators. Every formula $\psi \in \mathrm{TMSO}^c(\Sigma, \mathit{Op})$ with a single free variable z is finitely varying.*

Proof Let $\sigma \in T\Sigma^*$. Let $X = pos(\sigma, \psi)$ and $f_X : [0, \mathit{length}(\sigma)] \to \{0, 1\}$ such that $f_X(t) = 1$ iff $t \in X$. To show the finite variability of ψ, it is enough to show that f_X is finitely varying. We will show that the interval sequences of the canonical interval representations of the functions f_X and f_σ^Γ are the same, where $\Gamma = \Gamma_1 \cup \Gamma_2$ and (Γ_1, Γ_2) is the proper symbolic alphabet over (Σ, Op) based on G, the set of atomic guards in ψ. Let the canonical interval representation of f_σ^Γ be $(a_0, I_0) \cdots (a_{2n}, I_{2n})$. We will show that if $t_1, t_2 \in I_i$ then $t_1 \in pos(\sigma, \psi)$ iff $t_2 \in pos(\sigma, \psi)$. $t_1 \in pos(\sigma, \psi)$ iff $\sigma, [t_1/z] \models \psi$ iff $f_\sigma^\Gamma, [t_1/z] \models \mathit{tmso\text{-}mso}(\psi)$ iff $\mathit{untiming}(f_{\sigma\,[t_1/z]}^\Gamma) \in \mathcal{A}_{\mathit{tmso\text{-}mso}(\psi)}$ iff $\mathit{untiming}(f_{\sigma\,[t_2/z]}^\Gamma) \in \mathcal{A}_{\mathit{tmso\text{-}mso}(\psi)}$ (since $\mathit{untiming}(f_{\sigma\,[t_1/z]}^\Gamma) = \mathit{untiming}(f_{\sigma\,[t_2/z]}^\Gamma)$) iff $f_\sigma^\Gamma, [t_2/z] \models \mathit{tmso\text{-}mso}(\psi)$ iff σ, $[t_2/z] \models \psi$ iff $t_2 \in pos(\sigma, \psi)$. $\square$

Theorem 8.1 *Let Σ be a finite alphabet and let Rop be a set of finitely varying recursive operators over Σ. $L \subseteq T\Sigma^*$ is accepted by a rec-$CIDA$ over (Σ, Rop) iff L is definable by a rec-$TMSO^c(\Sigma, Rop)$ sentence.*

Proof We will show that for each i, the class of rec-$CIDA$'s of level i correspond to the sentences of rec-$TMSO^c$ of level i. We will use induction on the level of automata and formulas to argue that

1. $L \subseteq T\Sigma^*$ is accepted by a level i rec-$CIDA(\Sigma, Rop)$ iff L is definable by a level i rec-$TMSO^c(\Sigma, Rop)$ sentence φ.

2. A floating timed language L over Σ is accepted by a level i $frec$-$CIDA$ over (Σ, Rop) iff L is definable by a level i rec-$TMSO^c(\Sigma, Rop)$ formula ψ with one free variable.

3. Level i rec-$TMSO^c(\Sigma, Rop)$ formulas with one free variable and level i $frec$-$CIDA(\Sigma, Rop)$'s are finitely varying, where a $\mathcal{B} \in frec$-$CIDA(\Sigma, Op)$ is finitely varying if $pos(\sigma, \mathcal{B})$ is finitely varying for every $\sigma \in T\Sigma^*$.

For the base case of the first part of the claim, we consider level 0 automata and formulas. They correspond to $CIDA(\Sigma, \emptyset)$'s and $\mathrm{TMSO}^c(\Sigma, \emptyset)$ sentences, which by theorem 5.1 are expressively equivalent.

For the second part of the claim we consider level 0 $frec$-$CIDA(\Sigma, Rop)$'s and rec-$TMSO^c(\Sigma, Rop)$ formulas with one free variable, which correspond to floating $CIDA(\Sigma, \emptyset)$'s and $\mathrm{TMSO}^c(\Sigma, \emptyset)$ formulas with one free variable, which are expressively equivalent by lemma 8.1.

The level 0 rec-$TMSO^c(\Sigma, Rop)$ formulas with one free variable are finitely varying by lemma 8.2, and hence the level 0 $frec$-$CIDA(\Sigma, Rop)$'s are finitely varying.

Turning now to the induction step, let $\mathcal{A}$ be a level $i+1$ rec-$CIDA$ over (Σ, Rop). Then it is a $CIDA(\Sigma, Op)$ for some set of level i or less operators Op, with at least one level i operator. The operators in Op are finitely varying, since they are formed from a finitely varying recursive operator and a level i or less $frec$-$CIDA$ which by induction hypothesis is finitely varying. From theorem 5.1, we can get an equivalent $\mathrm{TMSO}^c(\Sigma, Op)$ sentence. For each $\Delta_\mathcal{B} \in Op$, $\mathcal{B}$ is a level j rec-$CIDA(\Sigma, Rop)$ for some $j < i+1$. Hence by induction hypothesis there exists a level j rec-$TMSO^c(\Sigma, Op)$ formula with one free variable such that $L^{fl}(\psi) = L^{fl}(\mathcal{B})$. Hence $pos(\sigma, \psi) = pos(\sigma, \mathcal{B})$. Therefore the semantics of the operators $\Delta_\mathcal{B}$ and Δ_ψ are the same, and hence we can replace $\Delta_\mathcal{B}$ by Δ_ψ to get a level $i+1$ rec-$TMSO^c(\Sigma, Op)$ sentence.

Conversely, let φ be a level $i+1$ sentence in rec-$TMSO^c(\Sigma, Rop)$. It is then a $\mathrm{TMSO}^c(\Sigma, Op)$ formula for some set of level i or less operators Op. The operators are finitely varying by induction hypothesis. By theorem 5.1, we can get an equivalent $CIDA(\Sigma, Op)$. Now we can replace each Δ_ψ in Op by $\Delta_{\mathcal{B}}$, where the floating timed languages of ψ and $\mathcal{B}$ are same, which exists by lemma 8.1.

The arguments for inductive part of the second claim are similar.

Let ψ be a level $i+1$ $\mathit{rec}\text{-}\mathit{TMSO}^c(\Sigma, \mathit{Rop})$ formula with one free variable. It is a $\mathrm{TMSO}^c(\Sigma, Op)$ formula with one free variable for some set of operators Op formed from level i or less formulas. Hence by induction hypothesis, the operators in Op are finitely varying. From lemma 8.2, the formula ψ itself in finitely varying. For every level $i+1$ $\mathit{frec}\text{-}\mathit{CIDA}(\Sigma, \mathit{Rop})$ we can give a level $i+1$ $\mathit{rec}\text{-}\mathit{TMSO}^c(\Sigma, \mathit{Rop})$ formula with one free variable. Hence level $i+1$ $\mathit{frec}\text{-}\mathit{CIDA}(\Sigma, \mathit{Rop})$ are finitely varying. $\square$

9 Expressive completeness of recursive continuous timed temporal logics

We now define a recursive timed temporal logic along the lines of [5] and show that it is expressively complete. It is similar to the logic TLTL^c, and is parameterized by an alphabet Σ and a set of recursive input determined operators Rop, and is denoted $\mathit{rec}\text{-}\mathit{TLTL}^c(\Sigma, \mathit{Rop})$. The syntax of the logic is given by

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

where $a \in \Sigma$, $\Delta \in \mathit{Rop}$ and $I \in \mathcal{I}_{\mathbb{Q}}$.

The logic is interpreted over timed words in a manner similar to TLTL^c. The predicate Δ^I_θ is interpreted as follows. If θ does not use a Δ predicate, then the satisfaction relation $\sigma, t \models \theta$ is defined as for TLTL^c. Inductively assuming the semantics of $\mathit{rec}\text{-}\mathit{TLTL}^c(\Sigma, \mathit{Rop})$ formula θ has been defined, and setting $pos(\sigma, \theta) = \{t \in \mathbb{R}_{\geq 0} \mid \sigma, t \models \theta\}$, the operator Δ_θ is interpreted as an input determined operator with the semantics, $\Delta_\theta(\sigma, t) = \Delta(pos(\sigma, \theta), \sigma, t)$. The satisfaction relation $\sigma, t \models \Delta^I_\theta$ is then defined as for TLTL^c.

Once again, since Δ_θ behaves like an input determined operator, each formula in $\mathit{rec}\text{-}\mathit{TLTL}^c(\Sigma, \mathit{Rop})$ is also a $\mathrm{TLTL}^c(\Sigma, Op)$ formula, for an appropriately chosen set of input determined operators Op, containing operators of the form Δ_θ. We define the level of a $\mathrm{TLTL}^c(\Sigma, Op)$ formula in the expected way. A $\mathit{rec}\text{-}\mathit{TLTL}^c$ formula naturally defines both a timed language $L(\theta) = \{\sigma \in T\Sigma^* \mid \sigma, 0 \models \theta\}$ and a floating language $L^{fl}(\theta) = \{(\sigma, t) \mid \sigma, t \models \theta\}$.

Let us use $\mathit{rec}\text{-}\mathit{TFO}^c(\Sigma, \mathit{Rop})$ to denote the first order fragment of the logic $\mathit{rec}\text{-}\mathit{TMSO}^c(\Sigma, \mathit{Rop})$. Then we have the following expressiveness result:

Theorem 9.1 *$\mathit{rec}\text{-}\mathit{TLTL}^c(\Sigma, \mathit{Rop})$ is expressively equivalent to $\mathit{rec}\text{-}\mathit{TFO}^c(\Sigma, \mathit{Rop})$.*

Proof We show by induction on i that

1. A timed language $L \subseteq T\Sigma^*$ is definable by a level i *rec*-$TLTL^c(\Sigma, Rop)$ formula iff it is definable by a level i *rec*-$TFO^c(\Sigma, Rop)$ sentence.
2. A floating timed language over Σ is definable by a level i *rec*-$TLTL^c(\Sigma, Rop)$ formula iff it is definable by a level i *rec*-$TFO^c(\Sigma, Rop)$ formula with one free variable.
3. Level i *rec*-$TLTL^c(\Sigma, Rop)$ formulas and level i *rec*-$TFO^c(\Sigma, Rop)$ formulas with one free variable are finitely varying, where a *rec*-$TLTL^c(\Sigma, Op)$ formula θ is finitely varying if $pos(\sigma, \theta)$ is finitely varying for every σ.

We need to make use of the following result due to Kamp:

Theorem 9.2 ([8]) *For any* $\mathrm{FO}^c(A)$ *formula* ψ *with one free variable* z*, there is an* $\mathrm{LTL}^c(A)$ *formula* θ *such that for each* $f \in func(A)$ *and* $t \in [0, length(f)]$*,* $f, [t/z] \models \psi$ *iff* $f, t \models \theta$.

The proof is now similar to theorem 8.1 and uses theorem 6.1. □

10 Expressive completeness of MTL

As an application of the results in this paper we show that the logic Metric Temporal Logic (MTL^c) in the continuous semantics introduced in [9] is expressively equivalent to *rec*-TFO^c for a suitably defined set of recursive input-determined operators. We define the logic $\mathrm{MTL}^c(\Sigma)$ inductively as below:

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

where $a \in \Sigma$ and $I \in \mathcal{I}_{\mathbb{Q}}$.

The modalities U_I and S_I are interpreted as follows for a timed word σ and $t \in [0, length(\sigma)]$.

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

We observe that $\mathrm{MTL}^c(\Sigma)$ is expressively equivalent to its sublogic $\mathrm{MTL}^{c\Diamond}(\Sigma)$ in which the modalities U_I and S_I are replaced by the modalities U, S, $\Diamond_I$ and $\overleftarrow{\Diamond}_I$, where $\theta U \eta = \theta U_{(0,\infty)} \eta$, $\theta S \eta = \theta S_{(0,\infty)} \eta$, $\Diamond_I \theta = \top U_I \theta$ and $\overleftarrow{\Diamond}_I \theta = \top S_I \theta$. We give below the translation for U_I and it can be written down similarly for S_I.

$$\theta U_I \eta = \begin{cases} \Diamond_I \eta \wedge \neg\Diamond_{(0,a)} \neg\theta \wedge \Diamond_{[a,a]}(\eta \vee (\theta \wedge (\theta U \eta))) & \text{if } I = [a, b\rangle, a > 0 \\ \Diamond_I \eta \wedge \neg\Diamond_{(0,a]} \neg\theta \wedge \Diamond_{[a,a]}(\theta U \eta) & \text{if } I = (a, b\rangle, a > 0 \\ \Diamond_I \eta \wedge (\eta \vee (\theta U \eta)) & \text{if } I = [0, b\rangle \\ \Diamond_I \eta \wedge (\theta U \eta) & \text{if } I = (0, b\rangle \end{cases}$$

Next we consider the logic $\mathit{rec}\text{-}\mathit{TLTL}^c(\Sigma, \{\Diamond, \diamondminus\})$ where the semantics of $\Diamond$ and $\diamondminus$ are defined as $\Diamond(X, \sigma, t) = \{t' - t \mid t' \geq t, t \in X\}$ and $\diamondminus(X, \sigma, t) = \{t - t' \mid t' \leq t, t \in X\}$. The logic $\mathrm{MTL}^{c\Diamond}(\Sigma)$ is clearly expressively equivalent to $\mathit{rec}\text{-}\mathit{TLTL}^c(\Sigma, \{\Diamond, \diamondminus\})$, since the predicates $\Diamond_I\theta$ and $\Diamond^I_\theta$ are equivalent. Further $\Diamond$ and $\diamondminus$ are finitely varying recursive operators. Hence,

Theorem 10.1 $\mathrm{MTL}^c(\Sigma)$ *is expressively equivalent to* $\mathit{rec}\text{-}\mathit{TFO}^c(\Sigma, \{\Diamond, \diamondminus\})$.

References

[1] Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.

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

[3] Patricia Bouyer, Fabrice Chevalier, and Nicolas Markey. On the Expressiveness of TPTL and MTL. In Ramanujam and Sen [13], pages 432–443.

[4] Deepak D'Souza and M. Raj Mohan. Eventual Timed Automata. In Ramanujam and Sen [13], pages 322–334.

[5] 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.

[6] Deepak D'Souza and P. S. Thiagarajan. Product interval automata: A subclass of timed automata. In C. Pandu Rangan, Venkatesh Raman, and R. Ramanujam, editors, *FSTTCS*, volume 1738 of *Lecture Notes in Computer Science*, pages 60–71. Springer, 1999.

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

[9] Ron Koymans. Specifying Real-Time Properties with Metric Temporal Logic. *Real-Time Systems*, 2(4):255–299, 1990.

[10] Joël Ouaknine and James Worrell. On the decidability of metric temporal logic. In *LICS*, pages 188–197. IEEE Computer Society, 2005.

[11] Pavithra Prabhakar and Deepak D'Souza. On the expressiveness of MTL with past operators. Technical Report IISc-CSA-TR-2006-5, Indian Institute of Science, Bangalore 560012, India, May 2006. URL: `http://archive.csa.iisc.ernet.in/TR/2006/5/`.

[12] Alexander Moshe Rabinovich. Finite variability interpretation of monadic logic of order. *Theor. Comput. Sci.*, 275(1-2):111–125, 2002.

[13] R. Ramanujam and Sandeep Sen, editors. *FSTTCS 2005: Foundations of Software Technology and Theoretical Computer Science, 25th International Conference, Hyderabad, India, December 15-18, 2005, Proceedings*, volume 3821 of *Lecture Notes in Computer Science*. Springer, 2005.

[14] J. F. Raskin. *Logics, Automata and Classical Theories for Deciding Real-Time*. PhD thesis, FUNDP, Belgium, 1999.

[15] Jean-Francois Raskin and Pierre-Yves Schobbens. State clock logic: A decidable real-time logic. In *HART*, pages 33–47, 1997.