On continuous timed automata with input-determined guards

Fabrice Chevalier[1], Deepak D'Souza[2], Pavithra Prabhakar[2]

[1] LSV, ENS de Cachan
61 Av. Pres. Wilson, Cachan Cedex 94235, France.
fabrice.chevalier@lsv.ens-cachan.fr
[2] Department of Computer Science and Automation
Indian Institute of Science, Bangalore 560012, India.
deepakd,pavithra@csa.iisc.ernet.in

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 [2–5]. 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 $\triangleleft_a$ of [2] which measures the distance to the last time an event a occurred. Similarly the "eventual" operator $\diamondsuit_a$ [6, 5] inspired by the well-known timed logic Metric Temporal Logic (MTL) [7, 2, 8], measures the time to "some" future occurrence of an a event.

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 $\diamondsuit\diamondsuit_{[1,1]}a$ 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 [9, 10].

In the pointwise semantics, the work in [6] 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 [11].

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 [11] for the class of recursive event clock automata (ECA's), 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 [6] 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 [13].

The technique used in [11, 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. Details of proofs omitted due to lack of space can be found in the technical report [15].

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 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.

We denote the set of non-negative and positive real numbers by $\mathbb{R}_{\geq 0}$. We use $\mathcal{I}_{\mathbb{R}_{\geq 0}}$ to denote the set of intervals, where an interval is a convex subset of $\mathbb{R}_{\geq 0}$. 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 $\mathit{length}(f)$. 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 $a_0 a_1 \cdots a_n$ 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 $w = a_0 a_1 \cdots a_{2n}$ and $(a_0, I_0)(a_1, I_1) \cdots (a_{2n}, I_{2n})$ is an interval representation of f. We can extend the definitions of timing and $\mathit{untiming}$ to languages of functions in the expected way.

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 denote the set of all timed words over Σ by $T\Sigma^*$. 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 $\Delta_{\mathbb{Q}}$, 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 time points

in σ after t at which an event a occurs, and the event-recording operator $\lhd_a$ 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 $f_\Delta : [0, length(\sigma)] \to \{0, 1\}$ defined as, $f_\Delta(t)$ is 1 if $\sigma, t \models \Delta^I$, and 0 otherwise, is finitely varying. The operators $\Diamond_a$ and $\lhd_a$ are finitely varying, whereas $\Delta_{\mathbb{Q}}$ 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 $func(\Gamma_1 \cup \Gamma_2)$, denoted $tw(f)$, as follows. If $untiming(f) \notin \Gamma_1 \cdot (\Gamma_2 \cdot \Gamma_1)^*$, then $tw(f) = \emptyset$. Otherwise, a timed word $\sigma = (a_1, t_1) \cdots (a_n, t_n)$ is in $tw(f)$, provided for all $t \in [0, length(f)]$,

- $f(t) = (a, g)$, 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 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

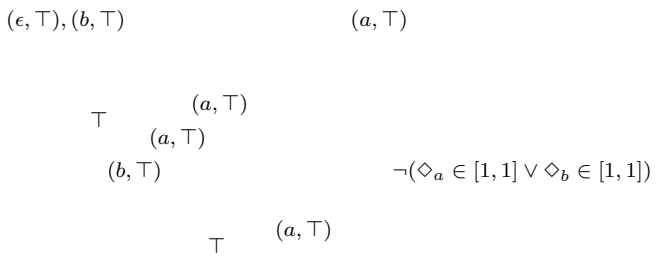
Let Σ be an alphabet and Op be a set of input determined operators based on Σ. A *Continuous Input Determined Automaton* (*CIDA*) $\mathcal{A}$ over (Σ, Op) is a structure (Q, s, δ, F, inv) on a symbolic alphabet (Γ_1, Γ_2) over (Σ, Op), where Q is a finite set of states, $s \in Q$ is the start state, $\delta \subseteq Q \times \Gamma_1 \times Q$ 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. Note that a language L is a regular subset of $\Gamma_1 \cdot (\Gamma_2 \cdot \Gamma_1)^*$ iff it is the symbolic language of a $CIDA$.

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

Theorem 1. *The class of $CIDA$'s over (Σ, Op) is closed under union, intersection and complementation.*

Proof. Union of $CIDA$'s is equivalent to the union of their symbolic languages. For complementation, using lemma 1 we can give an equivalent fully canonical proper $CIDA$ $\mathcal{A}'$ for a given $CIDA$ $\mathcal{A}$. 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. □

4 Continuous Monadic Second Order Logic

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

Recall that for an alphabet A, B$\ddot{u}$chi'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.

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:

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

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

Proof. We construct the automaton for a formula $\varphi \in \mathrm{MSO}^c(A)$, inductively. Let $X = (x_1, x_2, \cdots, x_n)$ and $Y = (X_1, X_2, \cdots, X_m)$ be the free variables in φ. We give an automaton $\mathcal{A}_\varphi^{X,Y}$ over $A' = A \times \{0,1\}^{n+m}$, which is related to φ as follows. Let $f \in \mathit{func}(A)$ and $\mathbb{I}$ be an interpretation of the variables in (X, Y) with respect to f. Then $f, \mathbb{I} \models \varphi$ iff $\mathit{untiming}(f_{\mathbb{I}}^{(X,Y)}) \in \mathcal{A}_\varphi^{X,Y}$. The function $f_{\mathbb{I}}^{(X,Y)} : [0, \mathit{length}(f)] \to A'$ is defined 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. Let $\mathcal{A}_{canon}^i$ be the automaton which accepts canonical words over $A \times \{0,1\}^i$. We consider here the cases when φ is $Q_a(x)$ and $\exists x\varphi$, and the detailed proof can be found in [15].

If $\varphi = Q_a(x)$, then the automaton $\mathcal{A}_\varphi^{X,Y}$ is the intersection of $\mathcal{A}_{canon}^1$ with:

Suppose $\varphi = \exists x \eta$. 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 words in which there is exactly one symbol with a 1 for its x-component at some even position (assuming indices start from 0). We then project away the x-components of the labels on the transitions in the automaton. Next we canonicalize the resulting automaton in two steps. First we convert the automaton to one that is in the form of a bipartite graph in which the transitions are only from the states in one set to the other. We then add transitions as described below repeatedly until no more can be added. A transition (p, a, r) is added if there exist transitions (p, a, q), (q, a, q') and (q', a, r). The above construction relies on the fact that if $f_{\mathbb{I}}^{(X,Y)}$ is in the timing of w, then $f_{\mathbb{I}'}^{(X',Y)}$ is in the timing of w', where w' is obtained from w by projecting away its x-component and $\mathbb{I}'$ is an interpretation to the variables in (X', Y) which agrees with $\mathbb{I}$ on the common variables. Finally we intersect the automaton with $\mathcal{A}_{canon}^{n+m}$ where m is the number of variables in 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)($\mathrm{TMSO}^c(\Sigma, Op)$) 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 $\varphi \in \mathrm{TMSO}^c(\Sigma, Op)$, 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 $t \in [0, \mathit{length}(\sigma)]$ and a second order variable X to $B \subseteq_{fin} [0, \mathit{length}(\sigma)]$, we define the satisfaction relation $\sigma, \mathbb{I} \models \mathrm{Q}_a(x)$ as $\exists i : a_i = a, t_i = \mathbb{I}(x)$, and $\sigma, \mathbb{I} \models \Delta^I(x)$ as $\Delta(\sigma, \mathbb{I}(x)) \cap I \neq \emptyset$, and the rest of the cases are similar to that of MSO^c over functions. 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 3. *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 (Γ_1, Γ_2) based on a set of atomic guards G over Op. 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 the timed words. Let $\delta = \{e_1, \cdots, e_m\}$ be the set of transitions. We set $(e, e') \in \mathit{consec}$ if and only if there exists q such that

$e = (p, \gamma, q)$ and $e' = (q, \gamma', r)$. We use $action(x)$ 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. Let $between(x, y, z) = x < y \wedge y < z$, $first(x) = \neg \exists y (y < x)$, $last(x) = \neg \exists y (x < y)$ and $next(x, y, X) = x \in X \wedge y \in X \wedge \neg \exists w (x < w \wedge w < y \wedge w \in X)$.

$\varphi_{\mathcal{A}}$ is given by: $\exists X \exists X_{e_1} \cdots \exists X_{e_m} (\varphi_1 \wedge \varphi_2 \wedge \varphi_3 \wedge \varphi_4 \wedge \varphi_5 \wedge \varphi_6 \wedge \varphi_7)$, where:

$\varphi_1 : \forall x (\bigvee_{e \in \delta} x \in X_e \Leftrightarrow x \in X) \bigwedge \forall x \bigwedge_{i,j \in \{1, \cdots, m\}, i \neq j} (x \in X_{e_i} \Rightarrow x \notin X_{e_j})$.
$\varphi_2 : \forall x (first(x) \Rightarrow \bigvee_{(s,\gamma,q) \in \delta} x \in X_{(s,\gamma,q)})$.
$\varphi_3 : \forall x (last(x) \Rightarrow \bigvee_{(q,\gamma,f) \in \delta, f \in F} x \in X_{(q,\gamma,f)})$.
$\varphi_4 : \forall x \forall y (next(x, y, X) \Rightarrow \bigvee_{e, e' \in consec} (x \in X_e \wedge y \in X_{e'}))$.
$\varphi_5 : \forall x \bigwedge_{(p,(a,g),q) \in \delta} (x \in X_{(p,(a,g),q)} \Rightarrow (Q_a(x) \wedge g(x)))$.
$\varphi_6 : \forall x \bigwedge_{(p,(\epsilon,g),q) \in \delta} (x \in X_{(p,(\epsilon,g),q)} \Rightarrow (\neg action(x) \wedge g(x)))$.
$\varphi_7 : \forall x \forall y \forall z ((next(y, z) \wedge between(y, x, z)) \Rightarrow$
$(\bigwedge_{(p,a,q) \in \delta} (y \in X_{(p,a,q)} \Rightarrow (\neg action(x) \wedge [inv(q)](x)))))$.

In the other direction we reduce a TMSO^c formula to an MSO^c formula and then factor through theorem 2 to get an *FSA* over $\Gamma_1 \cup \Gamma_2$. Let $\varphi \in \mathrm{TMSO}^c(\Sigma, Op)$ and let $G = \{\Delta^I \mid \Delta^I(x) \text{ is a subformula of } \varphi\}$. 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 the $\mathrm{MSO}^c(\Gamma)$ formula obtained by replacing every atomic formula $Q_a(x)$ by $\bigvee_{(a,g) \in \Gamma} Q_{(a,g)}(x)$ and $\Delta^I(x)$ by $\bigvee_{(c,g) \in \Gamma, \Delta^I \in g} Q_{(c,g)}(x) \vee \bigvee_{g \in \Gamma, \Delta^I \in g} Q_g(x)$.

Theorem 4. *Given a sentence $\varphi \in \mathrm{TMSO}^c(\Sigma, Op)$, $L(\varphi) = tw(F(tmso\text{-}mso\ (\varphi)))$.*

$\mathrm{TMSO}^c - \varphi$ $\qquad$ $CIDA - \mathcal{A}'$

$\mathrm{MSO}^c - \tilde{\varphi}$ $\qquad$ $FSA - \mathcal{A}_{\tilde{\varphi}}$

We can now complete the proof by taking the route in the diagram above. From theorem 4, $L(\varphi) = tw(F(\tilde{\varphi}))$, where $\tilde{\varphi} = tmso\text{-}mso(\varphi)$. By theorem 2 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 $t \in [0, \mathit{length}(\sigma)]$. 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 $\exists t' : t < t' \leq \mathit{length}(\sigma), \sigma, t' \models \eta, \forall t'' : t < t'' < t', \sigma, t'' \models \theta$.
$\sigma, t \models \theta S \eta$ iff $\exists t' : 0 \leq t' < t, \sigma, t' \models \eta, \forall t'' : t' < t'' < t, \sigma, t'' \models \theta$.

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

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 5. *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. The proof of the forward direction is similar to the classical translation of LTL to MSO. 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: $\theta ::= a \mid (\theta U \theta) \mid (\theta S \theta) \mid \neg\theta \mid (\theta \vee \theta)$, 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 $f, t \models a$ is defined as $f(t) = a$, and for the rest of the cases it is defined as for TLTL^c. Let $\mathrm{FO}^c(A)$ denote the first order fragment of $\mathrm{MSO}^c(A)$. Then the result due to Kamp [13] states that:

Theorem 6 ([13]). $\mathrm{LTL}^c(A)$ *is expressively equivalent to* $\mathrm{FO}^c(A)$.

Let φ be a $\mathrm{TFO}^c(\Sigma, Op)$ sentence. By theorem 4 the function $\mathit{tmso\text{-}mso}$ maps a $\mathrm{TFO}^c(\Sigma, Op)$ formula to an $\mathrm{FO}^c(\Gamma)$ formula $\tilde{\varphi} = \mathit{tmso\text{-}mso}(\varphi)$ such that $L(\varphi) = \mathit{tw}(F(\tilde{\varphi}))$. By Kamp's result, there exists a mapping *fo-ltl* such that $F(\tilde{\varphi}) = F(\mathit{fo\text{-}ltl}(\tilde{\varphi}))$. Let $\theta = \mathit{fo\text{-}ltl}(\tilde{\varphi})$. Let $a \in \Sigma$, $g \in G$ and $(a, g) \in \Gamma$. Let $\theta_g = \bigwedge_{h \in g} h \wedge \bigwedge_{h \in G - g} \neg h$. We define the function *ltl-tltl* which maps an $\mathrm{LTL}^c(\Gamma)$ formula θ to a $\mathrm{TLTL}^c(\Sigma, Op)$ formula obtained by replacing each atomic formula (a, g) by $a \wedge \theta_g$, (ϵ, g) by $\neg \bigvee_{c \in \Sigma} c \wedge \theta_g \wedge \neg(gSg \wedge gUg)$ and g by $\neg \bigvee_{c \in \Sigma} c \wedge \theta_g \wedge (gSg \wedge gUg)$. We then have $L(\varphi) = \mathit{tw}(F(\tilde{\varphi})) = \mathit{tw}(F(\theta)) = L(\mathit{ltl\text{-}tltl}(\theta))$. So $\mathit{ltl\text{-}tltl}(\theta)$ is the $\mathrm{TLTL}^c(\Sigma, Op)$ formula equivalent to φ.

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 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 $fw(L(\mathcal{B}))$.

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 $pos(\sigma, \mathcal{B}) = \{t \in [0, \mathit{length}(\sigma)] \mid (\sigma, t) \in L^{fl}(\mathcal{B})\}$. 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-$\mathit{CIDA}(\Sigma, Rop)$ is a $\mathit{CIDA}(\Sigma, Op)$ which uses at least one operator of level i. And a level $i + 1$ frec-$\mathit{CIDA}(\Sigma, Rop)$ is a floating $\mathit{CIDA}(\Sigma, Op)$ which uses at least one operator of level i.

We now introduce the recursive version of TMSO^c and show that it characterizes the class of timed languages defined by rec-CIDA. Given an alphabet Σ and a set of recursive operators Rop, the set of formulas of rec-$\mathit{TMSO}^c(\Sigma, Rop)$ 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 $rec\text{-}TMSO^c(\Sigma, Rop)$ 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.

A $rec\text{-}TMSO^c(\Sigma, Rop)$ sentence φ defines the language $L(\varphi) = \{\sigma \in T\Sigma^* \mid \sigma \models \varphi\}$. A $rec\text{-}TMSO^c(\Sigma, Rop)$ formula ψ with one free variable z defines the floating language $L^{fl}(\psi) = \{(\sigma, t) \mid \sigma, [t/z] \models \psi\}$. We have the following characterization.

Theorem 7. *Let Rop be a set of finitely varying recursive operators and Σ be a finite alphabet. $L \subseteq T\Sigma^*$ is accepted by a $rec\text{-}CIDA$ over (Σ, Rop) iff L is definable by a $rec\text{-}TMSO^c(\Sigma, Rop)$ sentence.*

We now define a recursive timed temporal logic along the lines of [6] 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 $rec\text{-}TLTL^c(\Sigma, Rop)$. 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 $a \in \Sigma$, $\Delta \in Rop$ and $I \in \mathcal{I}_\mathbb{Q}$. The logic is interpreted over timed words in a manner similar to TLTL^c, where the satisfaction of the predicate Δ^I_θ by σ at t is equivalent to $\Delta(pos(\sigma, \theta), \sigma, t) \cap I = \emptyset$, and $pos(\sigma, \theta) = \{t \in \mathbb{R}_{\geq 0} \mid \sigma, t \models \theta\}$. Let us denote by $rec\text{-}TFO^c(\Sigma, Rop)$ the first order fragment of the logic $rec\text{-}TMSO^c(\Sigma, Rop)$. Then we have the following expressiveness result:

Theorem 8. *$rec\text{-}TLTL^c(\Sigma, Rop)$ is expressively equivalent to $rec\text{-}TFO^c(\Sigma, Rop)$.*

8 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 [7] is expressively equivalent to $rec\text{-}TFO^c$ 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 $t \in [0, length(\sigma)]$.

$\sigma, t \models \theta U_I \eta$ iff $\exists t' \geq t : t' - t \in I, \sigma, t' \models \eta$, and $\forall t'' : t < t'' < t', \sigma, t'' \models \theta$.
$\sigma, t \models \theta S_I \eta$ iff $\exists t' \leq t : t - t' \in I, \sigma, t' \models \eta$, and $\forall t'' : t' < t'' < t, \sigma, t'' \models \theta$.

We first 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 $\Diamond\!\!\!\!-_I$, 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 $\Diamond\!\!\!\!-_I \theta = \top S_I \theta$. To show the equivalence we need to consider only the cases when $I = [l,l]$ and $I = (l,r)$. If $I = [l,l]$, then $\theta U_I \eta = \neg\Diamond_{(0,l)}\neg\theta \wedge \Diamond_{[l,l]}\eta$, otherwise $I = (l,r)$ in which case $\theta U_I \eta = \neg\Diamond_{(0,l]}\neg\theta \wedge \Diamond_{[l,l]}(\theta U \eta) \wedge \Diamond_{(l,r)}\eta$. Next we consider the logic $rec\text{-}TLTL^c(\Sigma, \{\Diamond, \Diamond\!\!\!\!-\})$ where the semantics of $\Diamond$ and $\Diamond\!\!\!\!-$ is defined as $\Diamond(X,\sigma,t) = \{t' - t \,|\, t' \geq t, t \in X\}$ and $\Diamond\!\!\!\!-(X,\sigma,t) = \{t - t' \,|\, t' \leq t, t \in X\}$. The logic $\mathrm{MTL}^{c\Diamond}(\Sigma)$ is clearly expressively equivalent to $rec\text{-}TLTL^c(\Sigma, \{\Diamond, \Diamond\!\!\!\!-\})$, since the predicates $\Diamond_I \theta$ and $\Diamond_\theta^I$ are equivalent. Further $\Diamond$ and $\Diamond\!\!\!\!-$ are finitely varying recursive operators. Hence,

Theorem 9. $\mathrm{MTL}^c(\Sigma)$ *is expressively equivalent to* $rec\text{-}TFO^c(\Sigma, \{\Diamond, \Diamond\!\!\!\!-\})$.

References

1. R. Alur and D.L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.
2. R. Alur, T. Feder and T.A. Henzinger. The Benefits of Relaxing Punctuality. *Journal of the ACM*, 43(1):116–146, 1996.
3. J. F. Raskin and P. Y. Schobbens. State Clock Logic: A Decidable Real-Time Logic. In HART, pages 33–47, 1997.
4. D. D'Souza and P. S. Thiagarajan. Product Interval Automata: A Subclass of Timed Automata. In FSTTCS, pages 60–71, 1999.
5. D. D'Souza and M. Raj Mohan. Eventual Timed Automata. In FSTTCS, pages 322-334, 2005.
6. D. D'Souza and N. Tabareau. On timed automata with input-determined guards. *FORMATS/FTRTFT*, volume 3253 of *LNCS*, 68-83, Springer, 2004.
7. R. Koymans. Specifying Real-Time Properties with Metric Temporal Logic. *Real-Time Systems*, 2(4):255-299, 1990.
8. J. Ouaknine and J. Worrel. On the Decidability of Metric Temporal Logic. In LICS, pages 188-197. IEEE Computer Society, 2005.
9. P. Bouyer, F. Chevalier, and N. Markey. On the expressiveness of TPTL and MTL. In FSTTCS, pages 432-443, 2005.
10. P. Prabhakar and D. 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/`
11. T. A. Henzinger and J. F. Raskin and P. Y. Schobbens. The Regular Real-Time Languages. In ICALP, volume 1443 of LNCS, pages 580-591. Springer, 1998.
12. A. M. Rabinovich. Finite variability interpretation of monadic logic of order. Theor. Comput. Sci., 275(1-2):111-125, 2002.
13. J. A. W. Kamp. Tense Logic and the Theory of Linear Order. PhD thesis, University of California, Los Angeles, California, 1968.
14. J. F. Raskin. Logics, Automata and Classical Theories for Deciding Real-Time. PhD thesis, FUNDP, Belgium, 1999.
15. F. Chevalier and D. D'Souza and P. 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/`