On timed automata with input-determined guards

Deepak D'Souza   Nicolas Tabareau

IISc-CSA-TR-2004-1
http://archive.csa.iisc.ernet.in/TR/2004/1/

Computer Science and Automation
Indian Institute of Science, India

April 2004

On timed automata with input-determined guards

Deepak D'Souza[1] and Nicolas Tabareau[2]

[1] Dept. of Computer Science & Automation
Indian Institute of Science, Bangalore, India.
deepakd@csa.iisc.ernet.in
[2] Ecole Normale Superieure de Cachan, Cachan, France.
Nicolas.Tabareau@dptmaths.ens-cachan.fr

Abstract. We consider a general notion of timed automata with *input-determined* guards and show that they admit a robust logical framework along the lines of [6], in terms of a monadic second order logic characterisation and an expressively complete timed temporal logic. We then generalize these automata using the notion of recursive operators introduced by Henzinger, Raskin, and Schobbens [9], and show that they admit a similar logical framework. These results hold in the "pointwise" semantics. We finally use this framework to show that the real-time logic MITL of Alur et al [2] is expressively complete with respect to an MSO corresponding to an appropriate set of input-determined operators.

1 Introduction

The timed automata of Alur and Dill [1] are a popular model for describing timed behaviours. While these automata have the plus point of being very expressive and having a decidable emptiness problem, they are neither determinizable nor closed under complementation. This is a drawback from a couple of points of view. Firstly, one cannot carry out model checking in the framework where a system is modeled as a timed transition system $\mathcal{T}$ and a specification of timed behaviours as a timed automaton $\mathcal{A}$, and where one asks "is $L(\mathcal{T}) \subseteq L(\mathcal{A})$?". This would normally involve *complementing* $\mathcal{A}$ and then checking if its intersection with $\mathcal{T}$ is non-empty. One can get around this problem to some extent by using determinizable specifications, or specifying directly the negation of the required property. A second reason why lack of closure properties may concern us is that it precludes the existence of an unrestricted logical characterisation of the class of languages accepted by timed automata. The existence of a monadic second order logic (MSO) characterisation of a class of languages is a strong endorsement of the "regularity" of the class. It also helps in identifying expressively complete temporal logics, which are natural to use as specification languages and have relatively efficient model checking algorithms.

The event clock automata of [3] was one of the first steps towards identifying a subclass of timed automata with the required closure properties. They were shown to be determinizable in [3], and later to admit a robust logical framework in terms of an MSO characterisation and an expressively complete timed temporal logic [6]. Similar results were shown in [14], [9] and [7]. A common technique used in all these results was the idea of "implicit" clocks, whose values are determined solely by the timed word being read. For example the event recording clock x_a records the time since the last a action w.r.t. the current position in a timed word, and is thus implicitly reset with each a action. The truth of a guard over these clocks at a point in a timed word is thus completely determined by the word itself, unlike in a timed automaton where the value of a clock depends on the path taken in the automaton.

In this paper we generalize the notion of an implicit clock to that of an *input determined operator*. An input determined operator Δ identifies for a given timed word and position in it, a set of intervals in which it is "satisfied". The guard $I \in \Delta$ is then satisfied at a point in a timed word if the set of intervals identified by Δ contains I. For example, the event recording clock x_a can be modeled as an input determined operator $\lhd_a$ which identifies at a given point in a timed word, the (infinite) set of intervals containing the distance to the last a action. The guard $(x_a \in I)$ now translates to $(I \in \lhd_a)$. As an example to show that this framework is more general than implicit clocks, consider the input determined operator $\Diamond_a$ inspired by the Metric Temporal logic (MTL) of [11, 4]. This operator identifies the set of all intervals I for which there is a future occurrence of an a at a distance which lies in I. The guard $I \in \Diamond_a$ is now true iff there is a future occurrence of an a action, at a distance which lies in I.

Timed automata which use guards based on a set of input determined operators are what we call *input determined automata*. We show that input determined automata form a robust class of timed languages, in that they are (a) determinizable, (b) effectively closed under boolean operations, (c) admit a logical characterisation via an unrestricted MSO, and (d) identify a natural expressively complete timed temporal logic.

We then go over to a more expressive framework using the idea of *recursive* event clocks from [9]. In the recursive version of our input determined operator, the operators now expect a third parameter (apart from the timed word and a position in it) which identifies a set of positions in the timed word. This argument could be (recursively) another input determined automaton, or as is better illustrated, a temporal logic formula θ. The formula θ naturally identifies a set of positions in a timed word where the formula is satisfied. Thus a recursive operator Δ along with the formula θ, written Δ_θ, behaves like an input determined operator above, and the guard $I \in \Delta_\theta$ is true iff the set of intervals identified by Δ_θ contains I. These recursive input determined automata are also shown to admit similar robust logical properties above.

We should be careful to point out here that, firstly, these results hold in the *pointwise* semantics, where formulas are evaluated only at the "action points" in a timed word (used e.g. in [16]), and not at arbitrary points in between actions in a timed word as allowed in the *continuous* semantics of [2, 9]. Secondly, we make no claims about the existence of *decision procedures* for these automata and logics. In fact it can be seen the operator $\Diamond_a$ above takes us out of the class of timed automata as we can define the language of timed sequences of a's in which no two a's are a distance 1 apart, with a single state input determined automaton which has the guard $\neg([1,1] \in \diamondsuit_a)$. Similar versions can be seen to have undecidable emptiness problems and correspondingly undecidable logics [4]. Thus the contribution of this paper should be seen more in terms of a general framework for displaying logical characterisations of timed automata, and proving expressive completeness of temporal logics related to these automata. Many of the results along these lines from [7, 6] and some in the pointwise semantics from [13] follow from the results in this paper.

As a new application of this framework, we provide an expressive completeness result for MITL in the pointwise semantics, by showing that it is expressively equivalent to the first order fragment of an MSO based on recursive operators. This answers an open question from [13], apart from identifying an interesting class of timed automata.

The techniques used in this paper essentially build on those from [7] and [6] which use the notion of *proper* symbolic alphabets and factor through the results of Büchi [5] and Kamp [10]. The idea of using recursive operators comes from [9], who show a variety of expressiveness results, including an expressive completeness for MITL in the continuous semantics. Their result for MITL is more interesting in that it uses event-clock modalities, while we use essentially the same modalities as MITL. However, our MSO is more natural as unlike the MSO in [9] it has unrestricted second order quantification.

2 Input determined automata

We use $\mathbb{N}$ to denote the set of natural numbers $\{0, 1, \ldots\}$, and $\mathbb{R}^{\geq 0}$ and $\mathbb{Q}^{\geq 0}$ to denote the set of non-negative reals and rationals respectively. The set of finite and infinite words over an alphabet A will be denoted by A^* and A^ω respectively. We use the notation $X \to Y$ to denote the set of functions from X to Y.

An *(infinite) timed word* over an alphabet Σ is an element σ of $(\Sigma \times \mathbb{R}^{\geq 0})^\omega$ satisfying the following conditions. Let $\sigma = (a_0, t_0)(a_1, t_1) \cdots$. Then:

1. (*monotonicity*) for each $i \in \mathbb{N}$, $t_i \leq t_{i+1}$,
2. (*progressiveness*) for each $t \in \mathbb{R}^{\geq 0}$ there exists $i \in \mathbb{N}$ such that $t_i > t$.

Let $T\Sigma^\omega$ denote the set of infinite timed words over Σ. Where convenient, we will use the representation of σ as (α, τ) where $\alpha \in \Sigma^\omega$ and $\tau : \mathbb{N} \to \mathbb{R}^{\geq 0}$ is a time sequence satisfying the conditions above.

We will use rational bounded intervals to specify timing constraints. These intervals can be open or closed, and we allow ∞ as an open right end. These intervals denote a subset of reals in the usual manner – for example $[2, \infty)$ denotes the set $\{t \in \mathbb{R}^{\geq 0} \mid 2 \leq t\}$. The set of all such intervals is denoted $\mathcal{I}_{\mathbb{Q}}$.

Our input determined automata will use guards of the form "$I \in \Delta$", where I is an interval and Δ is an operator which determines for a given timed word σ and a position i in it, a set of intervals "satisfying" it at that point. We then say that σ at position i satisfies the guard "$I \in \Delta$" if I belongs to the set of intervals identified by Δ. By a "position" in the timed word we mean one of the "action points" or instants given by the time-stamp sequence, and use natural numbers i (instead of the time $\tau(i)$) to denote these positions. More formally, an input determined operator Δ (w.r.t. the alphabet Σ) has a semantic function $[\![\Delta]\!] : (T\Sigma^\omega \times \mathbb{N}) \to 2^{\mathcal{I}_\mathbb{Q}}$. The guard $I \in \Delta$ is satisfied at position i in $\sigma \in T\Sigma^\omega$ iff $I \in [\![\Delta]\!](\sigma, i)$.

The transitions of our input determined automata are labeled by symbolic actions of the form (a, g) where a is an action, and g is a guard which is a boolean combination of atomic guards of the form $I \in \Delta$. The set of guards over a finite set of input determined operators Op is denoted by $\mathcal{G}(Op)$ and given by the syntax $g ::= \top \mid I \in \Delta \mid \neg g \mid g \vee g \mid g \wedge g$. The satisfaction of a guard g in a timed word σ at position i, written $\sigma, i \models g$, is given in the expected way: we have $\sigma, i \models \top$ always, $\sigma, i \models I \in \Delta$ as above, and the boolean operators $\neg$, $\vee$, and $\wedge$ interpreted as usual.

A *symbolic alphabet* Γ based on (Σ, Op) is a finite subset of $\Sigma \times \mathcal{G}(Op)$. An infinite word γ in Γ^ω specifies in a natural way a subset of timed words $tw(\gamma)$ defined as follows. Let $\gamma(i) = (a_i, g_i)$ for each $i \in \mathbb{N}$. Let $\sigma \in T\Sigma^\omega$ with $\sigma(i) = (b_i, t_i)$ for each $i \in \mathbb{N}$. Then $\sigma \in tw(\gamma)$ iff for each $i \in \mathbb{N}$, $b_i = a_i$ and $\sigma, i \models g_i$. We extend the map tw to work on subsets of Γ^ω in the natural way. Thus, for $\widehat{L} \subseteq \Gamma^\omega$, we define $tw(\widehat{L}) = \bigcup_{\gamma \in \widehat{L}} tw(\gamma)$. Finally, we denote the vocabulary of intervals mentioned in Γ by $ivoc(\Gamma)$.

Recall that a Büchi automaton over an alphabet A is a structure $\mathcal{A} = (Q, s, \longrightarrow, F)$ where Q is a finite set of states, $s \in Q$ is an initial state, $\longrightarrow \subseteq Q \times A \times Q$ is the transition relation, and $F \subseteq Q$ is a set of accepting states. Let $\alpha \in A^\omega$. A run of $\mathcal{A}$ over α is a map $\rho : \mathbb{N} \to Q$ which satisfies: $\rho(0) = s$ and $\rho(i) \xrightarrow{\alpha(i)} \rho(i+1)$ for every $i \in \mathbb{N}$. We say ρ is an *accepting* run of $\mathcal{A}$ on α if $\rho(i) \in F$ for infinitely many $i \in \mathbb{N}$. The set of words accepted by $\mathcal{A}$, denoted here as $L_{sym}(\mathcal{A})$ (for the "symbolic" language accepted by $\mathcal{A}$), is defined to be the set of words in A^ω on which $\mathcal{A}$ has an accepting run.

We are now in a position to define an input determined automaton. An *input determined automaton* (IDA for short) over an alphabet Σ and a set of operators Op, is simply a Büchi automaton over a symbolic alphabet based on (Σ, Op). Viewed as a Büchi automaton over a symbolic alphabet Γ, an input determined automaton $\mathcal{A}$ accepts the language $L_{sym}(\mathcal{A}) \subseteq \Gamma^\omega$ which we call the symbolic language accepted by $\mathcal{A}$. However, we will be more interested in the timed language accepted by $\mathcal{A}$: this is denoted $L(\mathcal{A})$ and is defined to be $tw(L_{sym}(\mathcal{A}))$.

To give a concrete illustration of input determined automata, we show how the event clock automata of [3] can be realized in the above framework. Take Op to be the set of operators $\{\lhd_a, \rhd_a \mid a \in \Sigma\}$, where the operators $\lhd_a$ and $\rhd_a$ essentially record the time since the last a action, and the time to the next a action. The operator $\lhd_a$ (and similarly $\rhd_a$) can be defined here by setting $[\![\lhd_a]\!](\sigma, i)$ to be

$$\{I \in \mathcal{I}_\mathbb{Q} \mid \exists j < i : \sigma(j) = a,\, \tau(i) - \tau(j) \in I, \text{ and } \forall k : j < k < i,\, \sigma(k) \neq a\}.$$

As another example which we will use later in the paper, consider the operator $\Diamond_a$ related to MTL [11,4]. The guard $\Diamond_a \in I$ is meant to be true in a word σ at time i iff there is a future instant j labeled a and the distance to it lies in I – i.e. $\tau(j) - \tau(i) \in I$. The guard $\diamondminus_a \in I$ makes a similar assertion about the *past* of σ w.r.t. the current position. An input determined automaton based on these operators can be defined by taking $Op = \{\Diamond_a, \diamondminus_a \mid a \in \Sigma\}$, and where, for example, $[\![\Diamond_a]\!](\sigma, i) = \{I \mid \exists j \geq i: \ \sigma(j) = a, \text{ and } \tau(j) - \tau(i) \in I\}$.

We now want to show that the class of timed languages accepted by input determined automata (for a given choice of Σ and Op) is closed under boolean operations. The notion of a *proper* symbolic alphabet will play an important role here and subsequently. A *proper symbolic alphabet* based on (Σ, Op) is of the form $\Gamma = \Sigma \times (Op \to 2^{\mathcal{I}})$ where $\mathcal{I}$ is a finite subset of $\mathcal{I}_{\mathbb{Q}}$. An element of Γ is thus of the form (a, h), where the set of intervals specified by $h(\Delta)$ is interpreted as the *exact* subset of intervals in $ivoc(\Gamma)$ which are satisfied by Δ. This is formalised in the following definition of tw_Γ for a proper symbolic alphabet Γ. Let $\gamma \in \Gamma^\omega$ with $\gamma(i) = (a_i, h_i)$. Let $\sigma \in T\Sigma^\omega$ with $\sigma(i) = (b_i, t_i)$. Then $\sigma \in tw_\Gamma(\gamma)$ iff for each $i \in \mathbb{N}$: $b_i = a_i$ and for each $\Delta \in Op$, $h_i(\Delta) = [\![\Delta]\!](\sigma, i) \cap ivoc(\Gamma)$.

Let Γ be a proper symbolic alphabet based on (Σ, Op). Then a Büchi automaton $\mathcal{A}$ over Γ, which we call a *proper* IDA over (Σ, Op), determines a timed language over Σ given by $tw_\Gamma(L_{sym}(\mathcal{A}))$.

The class of timed languages defined by IDA's and proper IDA's over (Σ, Op) coincide. An IDA over a symbolic alphabet Γ can be converted to an equivalent one (in terms of the timed language they define) over a proper symbolic alphabet $\Gamma' = \Sigma \times (Op \to 2^{ivoc(\Gamma)})$. Firstly, each transition label (a, g) in Γ can be written in a disjunctive normal form $(c_1 \vee \cdots \vee c_k)$, with each c_i being a conjunction of literals $I \in \Delta$ or $\neg(I \in \Delta)$. Thus each transition labeled (a, g) can be replaced by a set of transitions labeled (a, c_i), one for each i. Now each transition labeled (a, c), with c a conjunct guard, can be replaced by a set of transitions (a, h), one for each h "consistent" with c: i.e. h should satisfy the condition that if $I \in \Delta$ is one of the conjuncts in c then $I \in h(\Delta)$, and if $\neg(I \in \Delta)$ is one of the conjuncts in c then $I \notin h(\Delta)$. In the other direction, to go from a proper IDA to an IDA, a label (a, h) of a proper symbolic alphabet can be replaced by the guard

$$\bigwedge_{\Delta \in Op} \Big(\bigwedge_{I \in h(\Delta)} (I \in \Delta) \ \wedge \bigwedge_{I \in ivoc(\Gamma) - h(\Delta)} \neg(I \in \Delta) \Big).$$

The following property of proper symbolic alphabets will play a crucial role.

Lemma 1. *Let Γ be a proper symbolic alphabet based on Σ. Then for any $\sigma \in T\Sigma^\omega$ there is a* unique *symbolic word γ in Γ^ω such that $\sigma \in tw_\Gamma(\gamma)$.*

Proof. Let $\sigma(i) = (a_i, t_i)$. Then the only possible symbolic word γ we can use must be given by $\gamma(i) = (a_i, h_i)$, where $h_i(\Delta) = [\![\Delta]\!](\sigma, i) \cap ivoc(\Gamma)$. ☐

In the light of lemma 1, going from a symbolic alphabet to a proper one can be viewed as a step towards determinizing the automaton with respect to its timed language. From here one can simply use classical automata theoretic techniques to determinize the automaton w.r.t. its *symbolic* language. (Of course, since we deal with infinite words we will need to go from a Büchi to a Muller or Rabin acceptance condition [15]).

Theorem 1. *The class of IDA's over (Σ, Op) are effectively closed under the boolean operations of union, intersection, and complement.*

Proof. It is sufficient to address union and complementation. Given automata $\mathcal{A}$ and $\mathcal{B}$ over symbolic alphabets Γ and Λ respectively, we can simply construct an automaton over $\Gamma \cup \Lambda$ which accepts the union of the two symbolic languages. For complementing the timed language of $\mathcal{A}$, we can go over to an equivalent proper IDA $\mathcal{A}'$ over a proper symbolic alphabet Γ', and now simply complement the symbolic language accepted by $\mathcal{A}'$ to get an automaton $\mathcal{C}$. It is easy to verify, using the uniqueness property of proper alphabets given in Lemma 1, that $L(\mathcal{C}) = T\Sigma^\omega - L(\mathcal{A}')$. In the constructions above we have made use of the closure properties of ω-regular languages [15]. $\square$

3 A logical characterisation of IDA's

We now show that input determined automata admit a natural characterisation via a timed MSO in the spirit of [5]. Recall that for an alphabet A, Büchi's monadic second order logic (denoted here by $\mathrm{MSO}(A)$) is given as follows:

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

The logic is interpreted over a word $\alpha \in A^\omega$, along with an interpretation $\mathbb{I}$ which assigns individual variables x a position in α (i.e. an $i \in \mathbb{N}$), and to set variables X a set of positions $S \subseteq \mathbb{N}$. The relation $<$ is interpreted as the usual ordering of natural numbers, and the predicate Q_a (one for each $a \in A$) as the set of positions in α labeled a.

The formal semantics of the logic is given below. For an interpretation $\mathbb{I}$ we use the notation $\mathbb{I}[i/x]$ to denote the interpretation which sends x to i and agrees with $\mathbb{I}$ on all other variables. Similarly, $\mathbb{I}[S/X]$ denotes the modification of $\mathbb{I}$ which maps the set variable X to a subset S of $\mathbb{N}$. Later we will also use the notation $[i/x]$ to denote the interpretation with sends x to i when the rest of the interpretation is irrelevant.

$$\begin{array}{lll}
\alpha, \mathbb{I} \models Q_a(x) & \text{iff} & \alpha(\mathbb{I}(x)) = a. \\
\alpha, \mathbb{I} \models x \in X & \text{iff} & \mathbb{I}(x) \in \mathbb{I}(X). \\
\alpha, \mathbb{I} \models x < y & \text{iff} & \mathbb{I}(x) < \mathbb{I}(y). \\
\alpha, \mathbb{I} \models \exists x\varphi & \text{iff} & \text{there exists } i \in \mathbb{N} \text{ such that } \sigma, \mathbb{I}[i/x] \models \varphi. \\
\alpha, \mathbb{I} \models \exists X\varphi & \text{iff} & \text{there exists } S \subseteq \mathbb{N} \text{ such that } \sigma, \mathbb{I}[S/X] \models \varphi.
\end{array}$$

For a sentence φ (i.e. a formula without free variables) in $\mathrm{MSO}(A)$ we set $L(\varphi) = \{\sigma \in A^\omega \mid \sigma \models \varphi\}$. Büchi's result then states that a language $L \subseteq A^\omega$ is accepted by a Büchi automaton over A iff $L = L(\varphi)$ for a sentence φ in $\mathrm{MSO}(A)$.

We define a timed MSO called TMSO(Σ, Op), parameterised by the alphabet Σ and set of input determined operators Op, whose syntax is given by:

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

In the predicate "$I \in \Delta(x)$", I is an interval in $\mathcal{I}_{\mathbb{Q}}$, $\Delta \in Op$, and x is a variable.

The logic is interpreted in a similar manner to MSO, except that models are now timed words over Σ. In particular, for a timed word $\sigma = (\alpha, \tau)$, we have:

$$\begin{array}{ll} \sigma, \mathbb{I} \models Q_a(x) & \text{iff } \alpha(\mathbb{I}(x)) = a \\ \sigma, \mathbb{I} \models I \in \Delta(x) & \text{iff } I \in [\![\Delta]\!](\sigma, \mathbb{I}(x)). \end{array}$$

Given a sentence φ in TMSO(Σ) we define $L(\varphi) = \{\sigma \in T\Sigma^\omega \mid \sigma \models \varphi\}$.

Theorem 2. *A timed language $L \subseteq T\Sigma^\omega$ is accepted by an input determined automaton over (Σ, Op) iff $L = L(\varphi)$ for some sentence φ in* TMSO(Σ, Op).

Proof. Given an IDA $\mathcal{A}$ over (Σ, Op) we can give a TMSO sentence φ which describes the existence of an accepting run of $\mathcal{A}$ on a timed word. Following [15], for $\mathcal{A} = (Q, q_0, \longrightarrow, F)$ with $Q = \{q_0, \ldots q_n\}$, we can take φ to be the sentence

$$\begin{array}{ll} & \exists X_0 \cdots \exists X_n \ (\ 0 \in X_0 \ \wedge \ \bigwedge_{i \neq j} \forall x (x \in X_i \ \Rightarrow \ \neg(x \in X_j)) \\ (*) & \quad \wedge\ \forall x \bigvee_{q_i \xrightarrow{(a,g)} q_j} \ (x \in X_i \ \wedge \ (x+1) \in X_j \ \wedge Q_a(x) \wedge g') \\ & \quad \wedge \bigvee_{q_i \in F} \forall x \exists y (x < y \wedge y \in X_i)). \end{array}$$

Here g' denotes the formula obtained by replacing each $I \in \Delta$ in g by $I \in \Delta(x)$. Further, "$0 \in X_0$" abbreviates $\forall x\, (zero(x) \ \Rightarrow \ x \in X_0)$ where $zero(x)$ in turn stands for $\neg\exists y(y < x)$. Similarly $x+1 \in X_j$ can be expressed via $\forall y(succ_x(y) \Rightarrow y \in X_j)$, where $succ_x(y)$ is the formula $x < y \ \wedge \ \neg\exists z(x < z \ \wedge \ z < y)$.

In the converse direction we take the route used in [6] as it will be useful in the sequel. Let φ be a formula in TMSO(Σ, Op), and let Γ be a *proper* symbolic alphabet with the same interval vocabulary as φ. We give a way of translating φ to a formula $t\text{-}s(\varphi)$ in MSO(Γ) in such a way that the timed languages are preserved. The translation $t\text{-}s$ is done with respect to Γ and simply replaces each occurrence of

$$Q_a(x) \ \text{ by } \bigvee_{(b,h) \in \Gamma,\ b=a} Q_{(b,h)}(x) \ \text{ and } \ I \in \Delta(x) \ \text{ by } \bigvee_{(a,h) \in \Gamma,\ I \in h(\Delta)} Q_{(a,h)}(x).$$

The translation preserves the timed models of a formula φ in the following sense:

Lemma 2. *Let $\sigma \in T\Sigma^\omega$, $\gamma \in \Gamma^\omega$, and $\sigma \in tw_\Gamma(\gamma)$. Let $\mathbb{I}$ be an interpretation for variables. Then $\sigma, \mathbb{I} \models \varphi$ iff $\gamma, \mathbb{I} \models t\text{-}s(\varphi)$.* $\square$

The lemma is easy to prove using induction on the structure of the formula φ and making use of the properties of proper symbolic alphabets. From the lemma it immediately follows now that for a sentence φ in TMSO(Σ, Op), we have $L(\varphi) = tw_\Gamma(L(t\text{-}s(\varphi)))$, and this is the sense in which the translation preserves timed languages.

We can now argue the converse direction of Theorem 2 using this translation and factoring through Büchi's theorem. Let φ be a sentence in TMSO(Σ, Op) and let $\widehat{\varphi} = t\text{-}s(\varphi)$. Then by Büchi's theorem we have an automaton $\mathcal{A}$ over Γ which recognizes exactly $L(\widehat{\varphi})$. Thus $\mathcal{A}$ is our required proper IDA since $L(\mathcal{A}) = tw_\Gamma(L_{sym}(\mathcal{A})) = tw_\Gamma(L(\widehat{\varphi})) = L(\varphi)$. $\square$

4 An expressively complete timed LTL

In this section we identify a natural, expressively complete, timed temporal logic based on input determined operators. The logic is denoted TLTL(Σ, Op), parameterized by the alphabet Σ and set of input determined operators Op. The formulas of TLTL(Σ, Op) are given by:

$$\theta ::= \ a \mid I \in \Delta \mid O\theta \mid \Theta\theta \mid (\theta U \theta) \mid (\theta S \theta) \mid \neg\theta \mid (\theta \vee \theta).$$

Here we require $a \in \Sigma$, $I \in \mathcal{I}_\mathbb{Q}$, and $\Delta \in Op$. The models for TLTL(Σ, Op) formulas are timed words over Σ. Let $\sigma \in T\Sigma^\omega$, with $\sigma = (\alpha, \tau)$, and let $i \in \mathbb{N}$. Then the satisfaction relation $\sigma, i \models \varphi$ is given by

$$\begin{array}{lll}
\sigma, i \models a & \text{iff} & \alpha(i) = a \\
\sigma, i \models I \in \Delta & \text{iff} & I \in [\![\Delta]\!](\sigma, i) \\
\sigma, i \models O\theta & \text{iff} & \sigma, i+1 \models \theta \\
\sigma, i \models \Theta\theta & \text{iff} & i > 0 \text{ and } \sigma, i-1 \models \theta \\
\sigma, i \models \theta U \eta & \text{iff} & \exists k \geq i : \sigma, k \models \eta \text{ and } \forall j : i \leq j < k,\ \sigma, j \models \theta \\
\sigma, i \models \theta S \eta & \text{iff} & \exists k \leq i : \sigma, k \models \eta \text{ and } \forall j : k < j \leq i,\ \sigma, j \models \theta.
\end{array}$$

We define $L(\theta) = \{\sigma \in T\Sigma^\omega \mid \sigma, 0 \models \varphi\}$.

Let us denote by TFO(Σ, Op) the first-order fragment of TMSO(Σ, Op) (i.e. the fragment we get by disallowing quantification over set variables). The logics TLTL and TFO are *expressively equivalent* in the following sense:

Theorem 3. *A timed language $L \subseteq T\Sigma^\omega$ is definable by a* TLTL(Σ, Op) *formula θ iff it is definable by a sentence φ in* TFO(Σ, Op).

Proof. Given a TLTL(Σ, Op) formula θ we can associate an TFO(Σ, Op) formula φ which has a single free variable x, and satisfies the property that $\sigma, i \models \theta$ iff $\sigma, [i/x] \models \varphi$. This can be done in a straightforward inductive manner as follows. For the atomic formulas a and $I \in \Delta$ we can take φ to be $Q_a(x)$ and $I \in \Delta(x)$ respectively. In the inductive step, assuming we have already translated θ and η into φ and ψ respectively, we can translate $\theta U \eta$ into

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

Here $\psi[y/x]$ denotes the standard renaming of the free variable x to y in ψ. The remaining modalities are handled in a similar way, and we can verify that if φ is the above translation of θ then $\sigma, i \models \theta$ iff $\sigma, [i/x] \models \varphi$. It also follows that $\sigma, 0$ satisfies θ iff σ satisfies the sentence φ_0 given by $\forall x(zero(x) \Rightarrow \varphi)$. Hence we have that $L(\theta) = L(\varphi_0)$.

In the converse direction a more transparent proof is obtained by factoring through Kamp's result for classical LTL. Recall that the syntax of LTL(A) is given by:

$$\theta ::= \; a \mid O\theta \mid \Theta\theta \mid (\theta U \theta) \mid (\theta S \theta) \mid \neg\theta \mid (\theta \vee \theta)$$

where $a \in A$. The semantics is given in a similar manner to TLTL, except that models are words in A^ω. In particular the satisfaction relation $\alpha, i \models \theta$ for the atomic formula a is given by: $\sigma, i \models a$ iff $\alpha(i) = a$. Let FO(A) denote the first-order fragment of MSO(A). Then the result due to Kamp [10] states that:

Theorem 4 ([10]). LTL(A) *is expressively equivalent to* FO(A). □

Consider now a proper symbolic alphabet Γ based on (Σ, Op). We can define a timed language preserving translation of an LTL(Γ) formula $\widehat{\theta}$ to a formula $s\text{-}t(\widehat{\theta})$ in TLTL(Σ, Op). In the translation $s\text{-}t$ we replace subformulas (a, h) by

$$a \wedge \bigwedge_{\Delta \in Op} \Big(\bigwedge_{I \in h(\Delta)} (I \in \Delta) \;\wedge \bigwedge_{I \in ivoc(\Gamma) - h(\Delta)} \neg(I \in \Delta)\Big).$$

It is easy to argue along the lines of Lemma 1 that

Lemma 3. *Let $\sigma \in T\Sigma^\omega$ and $\gamma \in \Gamma^\omega$ with $\sigma \in tw_\Gamma(\gamma)$. Then $\sigma, i \models s\text{-}t(\widehat{\theta})$ iff $\gamma, i \models \widehat{\theta}$.* □

Hence we have $L(s\text{-}t(\widehat{\theta})) = tw_\Gamma(L(\widehat{\theta}))$.

We can now translate a sentence φ in TFO(Σ, Op) to an equivalent TLTL(Σ, Op) formula θ as follows. Let Γ be the proper symbolic alphabet based on (Σ, Op) with the same interval vocabulary as φ. Let $\widehat{\varphi}$ be the FO(Γ) formula $t\text{-}s(\varphi)$. Note that the translation $s\text{-}t$ preserves first-orderness and hence $\widehat{\varphi}$ belongs to FO(Γ). Now by Theorem 4, we have a formula $\widehat{\theta}$ in LTL(Γ) which is equivalent to $\widehat{\varphi}$. We now use the translation $t\text{-}s$ on the formula $\widehat{\theta}$ to get a TLTL(Σ, Op) formula θ. θ is our required TLTL(Σ, Op) formula. Observe that firstly $L(\theta) = tw_\Gamma(L(\widehat{\theta}))$ by the property of the translation $s\text{-}t$. Next, by Kamp's theorem we have that $L(\widehat{\theta}) = L(\widehat{\varphi})$ and hence $tw_\Gamma(L(\widehat{\theta})) = tw_\Gamma(L(\widehat{\varphi}))$. But by the property of the translation $t\text{-}s$ applied to φ, we have $tw_\Gamma(L(\widehat{\varphi})) = L(\varphi)$, and hence we can conclude that $L(\varphi) = L(\theta)$. This completes the proof of Theorem 3. □

We point out here that the past temporal operators of Θ ("previous") and S ("since") can be dropped from our logic without affecting the expressiveness of the logic. This follows since it is shown in [8] that Theorem 4 holds for the future fragment of LTL. The reason we retain the past operators is because they are needed when we consider a recursive version of the logic in Section 7.

5 Recursive input determined automata

We now consider "recursive" input determined operators. The main motivation is to increase the expressive power of our automata, as well as to characterise the expressiveness of recursive temporal logics which occur naturally in the real-time setting.

To introduce recursion in our operators, we need to consider *parameterised* (or *recursive*) input determined operators. These operators, which we continue to denote by Δ, have a semantic function $[\![\Delta]\!] : (2^{\mathbb{N}} \times T\Sigma^{\omega} \times \mathbb{N}) \to 2^{\mathcal{I}_{\mathbb{Q}}}$, whose first argument is a subset of positions X. Thus Δ with the parameter X determines an input determined operator of the type introduced earlier, whose semantic function is given by the map $(\sigma, i) \mapsto [\![\Delta]\!](X, \sigma, i)$. The set of positions X will typically be specified by a temporal logic formula or a "floating" automaton, in the sense that given a timed word σ, the formula (resp. automaton) will identify a set of positions in σ where the formula is satisfied (resp. automaton accepts). These ideas will soon be made more precise.

We first recall the idea of a "floating" automaton introduced in [9]. These are automata which accept pairs of the form (σ, i) with σ a timed word, and i a position (i.e. $i \in \mathbb{N}$). We will represent a "floating" word (σ, i) as a timed word over $\Sigma \times \{0,1\}$. Thus a timed word ν over $\Sigma \times \{0,1\}$ represents the floating word (σ, i), iff $\nu = (\alpha, \beta, \tau)$, with $\beta \in \{0,1\}^{\omega}$ with a *single* 1 in the i-th position, and $\sigma = (\alpha, \tau)$. We use *fw* to denote the (partial) map which given a timed word ν over $\Sigma \times \{0,1\}$ returns the floating word (σ, i) corresponding to ν, and extend it to apply to timed languages over $\Sigma \times \{0,1\}$ in the natural way.

Let Op be a set of input determined operators w.r.t. Σ. Then a *floating IDA* over (Σ, Op) is an IDA over $(\Sigma \times \{0,1\}, Op')$, where the set of operators Op' w.r.t. $\Sigma \times \{0,1\}$ is defined to be $\{\Delta' \mid \Delta \in Op\}$, with the semantics

$$[\![\Delta']\!](\sigma', i) = [\![\Delta]\!](\sigma, i),$$

where σ' is a timed word over $\Sigma \times \{0,1\}$, with $\sigma' = (\alpha, \beta, \tau)$ and $\sigma = (\alpha, \tau)$. Thus the operator Δ' simply ignores the $\{0,1\}$ component of σ' and behaves like Δ on the Σ component. A floating IDA $\mathcal{B}$ accepts the floating timed language $L^f(\mathcal{B}) = fw(L(\mathcal{B}))$.

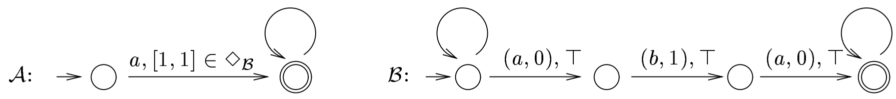
We now give a more precise definition of recursive input determined automata, denoted rec-IDA, and their floating counterparts frec-IDA. Let Rop be a finite set of recursive input determined operators. Then the class of rec-IDA's over (Σ, Rop), and the timed languages they accept, are defined as follows.

- Every IDA $\mathcal{A}$ over Σ that uses only the guard $\top$ is a rec-IDA over (Σ, Rop), and accepts the timed language $L(\mathcal{A})$.
  Similarly, every floating IDA $\mathcal{B}$ over Σ which uses only the guard $\top$ is a frec-IDA over (Σ, Rop), and accepts the floating language $L^f(\mathcal{B})$.
- Let C be a finite collection of frec-IDA's over (Σ, Rop). Let Op be the set of input determined operators $\{\Delta_{\mathcal{B}} \mid \Delta \in Rop,\ \mathcal{B} \in C\}$, where the semantic function of each $\Delta_{\mathcal{B}}$ is given as follows. Let $pos(\sigma, \mathcal{B})$ denote the set of positions i such that $(\sigma, i) \in L^f(\mathcal{B})$. Then $[\![\Delta_{\mathcal{B}}]\!](\sigma, i) = [\![\Delta]\!](pos(\sigma, \mathcal{B}), \sigma, i)$.

Then any IDA $\mathcal{A}$ over (Σ, Op) is a rec-IDA over (Σ, Rop), and accepts the timed language $L(\mathcal{A})$ (defined in Section 2).
Similarly every floating IDA $\mathcal{B}$ over (Σ, Op) is a frec-IDA over (Σ, Rop), and accepts the floating language $L^f(\mathcal{B})$.

Recursive automata fall into a natural "level" based on the level of nesting of operators they use. A rec-IDA is of *level* 0 if the only guard it uses is $\top$. Similarly a frec-IDA is of level 0, if the only guard it uses is $\top$. A rec-IDA is of *level* (i+1) if it uses an operator $\Delta_{\mathcal{B}}$, with $\Delta \in Rop$ and $\mathcal{B}$ a frec-IDA of level i, and no operator $\Delta'_{\mathcal{C}}$ with $\Delta' \in Rop$ and $\mathcal{C}$ of level greater than i. A similar definition of level applies to frec-IDA's.

As an example consider the level 1 rec-IDA $\mathcal{A}$ over the alphabet $\{a, b\}$ below. The floating automaton $\mathcal{B}$ accepts a floating word (σ, i) iff the position i is labeled b and the previous and next positions are labeled a. The recursive input determined operator $\Diamond$ is defined formally in Sec. 8. The rec-IDA $\mathcal{A}$ thus recognizes the set of timed words σ over $\{a, b\}$ which begin with an a and have an occurrence of b – with a's on its left and right – exactly 1 time unit later.



Theorem 5. *The class of rec-IDA's over (Σ, Rop) is closed under boolean operations. In fact, for each i, the class of level i rec-IDA's is closed under boolean operations.*

Proof. Let $\mathcal{A}$ and $\mathcal{A}'$ be two rec-IDA's of level i. Let Op be the union of operators used in $\mathcal{A}$ and $\mathcal{A}'$. Then both $\mathcal{A}$ and $\mathcal{A}'$ are IDA's over (Σ, Op), and hence by Theorem 1 there exists an IDA $\mathcal{B}$ over (Σ, Op) which accepts $L(\mathcal{A}) \cup L(\mathcal{A}')$. Similarly there exists an IDA $\mathcal{C}$ over (Σ, Op), which accepts the language $T\Sigma^\omega - L(\mathcal{A})$. Notice that $\mathcal{B}$ and $\mathcal{C}$ use the same set of operators Op, and hence are also level i automata. $\square$

We note that IDA's over (Σ, Op) are a special case of level 1 rec-IDA's over (Σ, Rop), where the set of recursive operators Rop is taken to be $\{\Delta' \mid \Delta \in Op\}$ with $[\![\Delta']\!](X, \sigma, i) = [\![\Delta]\!](\sigma, i)$. Thus each guard $I \in \Delta$ in an IDA over (Σ, Op) can be replaced by the guard $I \in \Delta'_{\mathcal{B}}$, for any "dummy" level 0 frec-IDA $\mathcal{B}$.

6 MSO characterisation of rec-IDA's

We now introduce a recursive version of TMSO which will characterise the class of timed languages defined by rec-IDA's. The logic is parameterised by an alphabet Σ and set of recursive input determined operators Rop, and denoted rec-TMSO(Σ, Rop). The syntax of the logic is given by

$$\varphi ::= Q_a(x) \mid I \in \Delta_\psi(x) \mid x \in X \mid x < y \mid \neg\varphi \mid (\varphi \vee \varphi) \mid \exists x\varphi \mid \exists X\varphi.$$

In the predicate $I \in \Delta_\psi(x)$, we have $I \in \mathcal{I}_{\mathbb{Q}}$, $\Delta \in Rop$, and ψ a rec-TMSO(Σ, Rop) formula with a single free variable z.

The logic is interpreted over timed words in $T\Sigma^\omega$. Its semantics is similar to TMSO except for the predicate "$I \in \Delta_\psi(x)$" which is defined inductively as follows. If ψ is a formula which uses no Δ predicates, then the satisfaction relation $\sigma, \mathbb{I} \models \psi$ is defined as for TMSO. Inductively, assuming the semantics of ψ has already been defined, Δ_ψ is interpreted as an input determined operator as follows. Let $pos(\sigma, \psi)$ denote the set of interpretations for z that make ψ true in the timed word σ – i.e. $pos(\sigma, \psi) = \{i \mid \sigma, [i/z] \models \psi\}$. Then

$$[\![\Delta_\psi]\!](\sigma, i) = [\![\Delta]\!](pos(\sigma, \psi), \sigma, i).$$

Thus we have

$$\sigma, \mathbb{I} \models I \in \Delta_\psi(x) \text{ iff } I \in [\![\Delta]\!](pos(\sigma, \psi), \sigma, \mathbb{I}(x)).$$

Note that the variable z, which is free in ψ, is *not* free in the formula $I \in \Delta_\psi(x)$. A sentence φ in rec-TMSO(Σ, Rop) defines the language $L(\varphi) = \{\sigma \models \varphi\}$, and a rec-TMSO$(\Sigma, Rop)$ formula ψ with one free variable z defines a floating language $L^f(\psi) = \{\sigma, i \mid \sigma, [i/z] \models \psi\}$.

We note that each rec-TMSO(Σ, Rop) formula φ can be viewed as a TMSO(Σ, Op) formula, for a suitably defined set of input determined operators Op. We say an operator Δ_ψ has a *top-level* occurrence in φ if there is an occurrence of Δ_ψ in φ which is *not* in the scope of any Δ' operator. We can now take Op to be the set of all top-level operators Δ_ψ in φ.

Analogous to the notion of level for rec-IDA's we can define the *level* of an rec-TMSO formula φ. The level of φ is 0, if φ uses no Δ predicates. φ has level $i+1$ if it uses a predicate of the form $I \in \Delta_\psi(x)$ with ψ a level i formula, and *no* predicate of the form $I \in \Delta'_\phi(x)$ with ϕ of level greater than i.

As an example the level 1 sentence φ below defines the same timed language as the level 1 rec-IDA $\mathcal{A}$ defined in Section 2. We can take φ to be $\forall x(zero(x) \Rightarrow (Q_a(x) \wedge ([1,1] \in \Diamond_\psi(x))))$, where ψ is the level 0 formula $Q_b(z) \wedge Q_a(z-1) \wedge Q_a(z+1)$.

Theorem 6. *$L \subseteq T\Sigma^\omega$ is accepted by a rec-IDA over (Σ, Rop) iff L is definable by a* rec-TMSO(Σ, Rop) *sentence.*

In fact, we will show that for each i, the class of rec-IDA's of level i correspond to the sentences of rec-TMSO(Σ, Rop) of level i. But first it will be useful to state a characterisation of floating languages along the lines of Theorem 2.

Theorem 7. *Let L be a a floating language over Σ. Then $L = L^f(\mathcal{B})$ for some floating IDA $\mathcal{B}$ over (Σ, Op) iff $L = L^f(\psi)$, for some* TMSO(Σ, Op) *formula ψ with one free variable.*

Proof. Let $\mathcal{B}$ be a floating IDA over (Σ, Op). Keeping in mind that $\mathcal{B}$ runs over the alphabet $\Sigma \times \{0, 1\}$, we define a formula ψ with one free variable z as follows.

ψ is the formula φ given in the proof of Theorem 2, except for the clause (*) which we replace by

$$\wedge\ \forall x((x = z) \Rightarrow \bigvee_{q_i \xrightarrow{((a,1),g)} q_j} (x \in X_i \ \wedge\ (x+1) \in X_j \ \wedge Q_a(x) \wedge g')$$
$$\wedge\ (x \neq z) \Rightarrow \bigvee_{q_i \xrightarrow{((a,0),g)} q_j} (x \in X_i \ \wedge\ (x+1) \in X_j \ \wedge Q_a(x) \wedge g')).$$

The formula ψ satisfies $(\sigma, i) \in L^f(\mathcal{B})$ iff $\sigma, [i/z] \models \psi$.

In the converse direction, let $\varphi(m, n)$ denote a TMSO(Σ, Op) formula with free variables $x_1, \ldots, x_m, X_1 \ldots X_n$. An interpretation $\mathbb{I}$ for these variables is encoded (along with σ) as a timed word over $\Sigma \times \{0,1\}^{m+n}$. We extend the definition of a floating IDA to an IDA which works over such an alphabet, where, in particular, the Δ operators apply only to the Σ component of the timed word. Then we can inductively associate with $\varphi(m, n)$ a floating IDA $\mathcal{B}$ over $\Sigma \times \{0,1\}^{m+n}$ such that $L^f(\mathcal{B}) = L^f(\varphi)$. In the inductive step for $\exists X_n(\varphi(m, n))$ we make use of the fact that the class of languages accepted by floating IDA's over (Σ, Op) are closed under the restricted renaming operation required in this case. The reader is referred to [6] for a similar argument. $\square$

Returning now to the proof of Theorem 6, we use induction on the level of automata and formulas to argue that

1. $L \subseteq T\Sigma^\omega$ is accepted by a level i rec-IDA over (Σ, Rop) iff L is definable by a level i rec-TMSO(Σ, Rop) sentence φ. And
2. A floating language L over Σ is accepted by a level i frec-IDA over (Σ, Rop) iff L is definable by a level i rec-TMSO(Σ, Rop) formula ψ with one free variable.

For the base case we consider level 0 automata and sentences. Since level 0 automata only make use of the guard $\top$, they are simply Büchi automata over Σ. Similarly, level 0 sentences don't use any Δ predicates and hence they are simply MSO(Σ) sentences. By Büchi's theorem, we have that level 0 automata and sentences are expressively equivalent.

For the base case for the second part of the claim, given a level 0 floating automaton $\mathcal{B}$ we can apply the construction in the proof of Theorem 7 to get a TMSO(Σ) formula ψ with one free variable. Since the construction preserves the guards used, ψ has no Δ operators, and hence is a level 0 rec-TMSO(Σ, Rop) formula. Conversely, for a level 0 formula ψ we can apply the construction of Theorem 7 to obtain a floating automaton $\mathcal{B}$ such that $L^f(\mathcal{B}) = L^f(\psi)$. The construction preserves the Δ operators used, and hence $\mathcal{B}$ is a level 0 automaton.

Turning now to the induction step, let $\mathcal{A}$ be a level $i+1$ automaton over (Σ, Rop). Let Op be the set of top-level Δ operators in $\mathcal{A}$. Now since $\mathcal{A}$ is an IDA over (Σ, Op), by Theorem 2, we have a TMSO(Σ, Op) sentence φ such that $L(\mathcal{A}) = L(\varphi)$. Now for each $\Delta_{\mathcal{B}}$ in Op, $\mathcal{B}$ is of level i or lower, and by our induction hypothesis there is a corresponding rec-TMSO(Σ, Rop) formula ψ with one free variable, of the same level as $\mathcal{B}$, with $L^f(\mathcal{B}) = L^f(\psi)$. Hence for each $\Delta_{\mathcal{B}}$ we have a semantically equivalent operator Δ_ψ. This is because $L^f(\mathcal{B}) = L^f(\psi)$, which implies $pos(\sigma, \mathcal{B}) = pos(\sigma, \psi)$, which in turn implies $[\![\Delta_{\mathcal{B}}]\!] = [\![\Delta_\psi]\!]$. We can now simply replace each occurrence of $\Delta_{\mathcal{B}}$ in φ to get an equivalent sentence φ' which is in rec-TMSO(Σ, Rop). Further, by construction it follows that φ' is also of level $i+1$.

Conversely, let φ be a level $i+1$ sentence in rec-TMSO(Σ, Rop). Let Op be the set of top level Δ operators in φ. Then φ is a TMSO(Σ, Op) sentence, and hence by Theorem 2 we have an equivalent input determined automaton $\mathcal{A}$ over (Σ, Op). Once again, for each Δ_ψ in Op, the formula ψ is of level i or lower, and hence by induction hypothesis we have a frec-IDA $\mathcal{B}$ over (Σ, Rop), of the same level as ψ, and accepting the same floating language. The operators Δ_ψ and $\Delta_{\mathcal{B}}$ are now equivalent, and we can replace each Δ_ψ in $\mathcal{A}$ by the corresponding $\Delta_{\mathcal{B}}$ to get a language equivalent input determined automaton. This automaton is now the required level $i+1$ rec-IDA over (Σ, Rop) which accepts the same language as $L(\varphi)$.

The induction step for part 2 is proved similarly, making use of Theorem 7 and the induction hypothesis. This completes the proof of Theorem 6. $\square$

7 Expressive completeness of rec-TLTL

We now define a recursive timed temporal logic along the lines of [9]. The logic is similar to the logic TLTL defined in Sec. 4. It is parameterized by an alphabet Σ and a set of recursive input determined operators Rop, and denoted rec-TLTL(Σ, Rop). The syntax of the logic is given by

$$\theta ::= \; a \mid I \in \Delta_\theta \mid O\theta \mid \Theta\theta \mid (\theta U \theta) \mid (\theta S \theta) \mid \neg\theta \mid (\theta \vee \theta),$$

where $a \in \Sigma$, and $\Delta \in Rop$.

The logic is interpreted over timed words in a similar manner to TLTL. The predicate $I \in \Delta_\theta$ is interpreted as follows. If θ does not use a Δ predicate, then the satisfaction relation $\sigma, i \models \theta$ is defined as for TLTL. Inductively assuming the semantics of a rec-TLTL(Σ, Rop) formula θ has been defined, and setting $pos(\sigma, \theta) = \{i \in \mathbb{N} \mid \sigma, i \models \theta\}$, the operator Δ_θ is interpreted as an input determined operator with the semantic function

$$[\![\Delta_\theta]\!](\sigma, i) = [\![\Delta]\!](pos(\sigma, \theta), \sigma, i).$$

The satisfaction relation $\sigma, i \models I \in \Delta_\theta$ is then defined as in TLTL.

Once again, since Δ_θ behaves like an input determined operator, each rec-TLTL(Σ, Rop) formula is also a TLTL(Σ, Op) formula, for an appropriately chosen set of input determined operators Op, containing operators of the form Δ_θ. A rec-TLTL(Σ, Rop) formula θ naturally defines both a timed language $L(\theta) = \{\sigma \in T\Sigma^\omega \mid \sigma, 0 \models \theta\}$ and a floating language $L^f(\theta) = \{(\sigma, i) \mid \sigma, i \models \theta\}$.

As an example, the formula $a \wedge ([1, 1] \in \Diamond_\theta)$ where $\theta = b \wedge \Theta a \wedge O a$, restates the property expressed by the rec-TMSO formula in Sec. 6.

Let us denote by rec-TFO(Σ, Rop) the first-order fragment of the logic rec-TMSO(Σ, Rop). Then we have the following expressive completeness result:

Theorem 8. rec-TLTL(Σ, Rop) *is expressively equivalent to* rec-TFO(Σ, Rop).

Proof. As before we show that formulas in the logics are equivalent level-wise (the level of a rec-TLTL formula being defined analogous to rec-TMSO). We show by induction on i that

1. A timed language $L \subseteq T\Sigma^\omega$ is definable by a level i rec-TLTL(Σ, Rop) formula iff it is definable by a level i rec-TFO(Σ, Rop) sentence.
2. A floating timed language over Σ is definable by a level i rec-TLTL(Σ, Rop) formula iff it is definable by a level i rec-TFO(Σ, Rop) formula with one free variable.

The base case for part 1 follows from Theorem 4, since level 0 formulas are simply untimed LTL(Σ) and FO(Σ) formulas. For the base case for part 2, a level 0 rec-TLTL(Σ, Rop) formula θ can be translated to a level 0 rec-TFO(Σ, Rop) formula ψ with one free variable z using the translation given in the proof of Theorem 3. The formula ψ satisfies $\sigma, [i/z] \models \psi$ iff $\sigma, i \models \theta$. The converse direction follows immediately from the following version of Kamp's result:

Theorem 9 ([10]). *For any* FO(A) *formula ψ with one free variable z, there is a* LTL(A) *formula θ s.t. for each $\alpha \in A^\omega$ and $i \in \mathbb{N}$, $\alpha, [i/z] \models \psi$ iff $\alpha, i \models \theta$.*

Turning now to the induction step, let θ be a level $i+1$ rec-TLTL(Σ, Rop) formula. Let Op be the set of top-level Δ operators used in θ. Then θ is a TLTL(Σ, Op) formula, and hence by Theorem 3 we have an equivalent TFO(Σ, Op) sentence φ (i.e. with $L(\theta) = L(\varphi)$). Now each operator in Op is of the form Δ_η where η is a level i or less rec-TLTL(Σ, Rop) formula, and hence by the induction hypothesis we have an equivalent rec-TFO(Σ, Rop) formula ψ with one free variable, such that $L^f(\eta) = L^f(\psi)$. It now follows that the input determined operators Δ_η and Δ_ψ are semantically equivalent, and hence we can replace each Δ_η by Δ_ψ in φ to get an equivalent rec-TFO(Σ, Rop) sentence φ'. By construction, the sentence φ' is also of level $i+1$. The converse direction is argued in a very similar manner, once again factoring through Theorem 3.

For part 2, a level $i+1$ rec-TLTL(Σ, Rop) formula θ is a TLTL(Σ, Op) formula, for the set of operators Op defined above. Now using the translation given in the proof of Theorem 3 we obtain a TFO(Σ, Op) formula ψ with a one free variable, satisfying $L^f(\theta) = L^f(\psi)$. Again, by the induction hypothesis, we can replace each Δ_η in Op with an equivalent Δ_ϕ, to get an equivalent rec-TFO(Σ, Rop) with the required properties.

In the converse direction, let ψ be a level $i+1$ rec-TFO(Σ, Rop) formula with one free variable z. Let Op be set of top-level Δ operators in ψ. Then ψ is also a formula in TFO(Σ, Op). Let Γ be the proper symbolic alphabet induced by ψ. Then we can use the translation *t-s* (cf. Sec 3) on ψ (w.r.t. Γ) to get a formula $\widehat{\psi}$ in FO(Γ) with one free variable z which preserves timed models. By Kamp's theorem above, we have an equivalent LTL(Γ) formula $\widehat{\theta}$ which preserves the floating language accepted. Finally we can apply the translation s-t on $\widehat{\theta}$ to get a TLTL(Σ, Op) formula θ which preserves timed models (cf. Sec. 4). The formula θ satisfies the property that $L^f(\theta) = L^f(\psi)$.

Now using the induction hypothesis each operator Δ_ϕ in θ can be replaced by an equivalent Δ_η operator, with η a TLTL(Σ, Op) formula, to get an equivalent level $i+1$ rec-TLTL(Σ, Rop) formula θ'. This ends the proof of Theorem 8. $\square$

8 Expressive completeness of MITL

As an application of the results in this paper we show that the logic MITL introduced in [2] is expressively equivalent to rec-TFO for a suitably defined set of recursive input determined operators. We point out here that this result is shown for the pointwise semantics of MITL given below. We begin with the logic MTL(Σ) which has the following syntax [4]:

$$\theta ::= \; a \mid O\theta \mid \Theta\theta \mid (\theta U_I \theta) \mid (\theta S_I \theta) \mid \neg\theta \mid (\theta \vee \theta).$$

Here I is an interval in $\mathcal{I}_{\mathbb{Q}}$. When I is restricted to be *non-singular* (i.e. not of the form $[r, r]$) then we get the logic MITL(Σ). The logic is interpreted over timed words in $T\Sigma^\omega$ similarly to TLTL. The modalities U_I and S_I are interpreted as follows, for a timed word $\sigma = (\alpha, \tau)$.

$$\sigma, i \models \theta U_I \eta \text{ iff } \exists k \geq i : \sigma, k \models \eta, \tau(k) - \tau(i) \in I, \textit{ and } \forall j : i \leq j < k, \; \sigma, j \models \theta$$
$$\sigma, i \models \theta S_I \eta \text{ iff } \exists k \leq i : \sigma, k \models \eta, \tau(i) - \tau(k) \in I, \text{ and } \forall j : k < j \leq i, \; \sigma, j \models \theta.$$

We first observe that MTL(Σ) is expressively equivalent to its sublogic MTL$^\diamond$(Σ) in which the modalities U_I and S_I are replaced by the modalities U, S, $\diamond_I$ and $\diamondminus_I$, where U and S are as usual and $\diamond_I\theta = \top U_I \theta$ and $\diamondminus_I\theta = \top S_I \theta$. This is because the formula $\theta U_I \eta$ (and dually $\theta S_I \eta$) can be translated as follows. Here '$\rangle$' denotes either a ']' or ')' interval bracket.

$$\theta U_I \eta = \begin{cases} \diamond_I \eta \wedge \square_{[0,a)}(\theta U(\theta \wedge O\eta)) & \text{if } I = [a, b\rangle, a > 0 \\ \diamond_I \eta \wedge \square_{[0,a]}(\theta U(\theta \wedge O\eta)) & \text{if } I = (a, b\rangle, a > 0 \\ \diamond_I \eta \wedge (\theta U \eta) & \text{if } I = [0, b\rangle \\ \diamond_I \eta \wedge (\theta U(\theta \wedge O\eta)) & \text{if } I = (0, b\rangle. \end{cases}$$

Next we consider the logic rec-TLTL($\Sigma, \{\diamond, \diamondminus\}$) where the semantics of the recursive input determined operators $\diamond$ and $\diamondminus$ are given below (as usual $\sigma \in T\Sigma^\omega$ with $\sigma = (\alpha, \tau)$).

$$[\![\diamond]\!](X, \sigma, i) \; = \{I \in \mathcal{I}_{\mathbb{Q}} \mid \exists j \in X : \; j \geq i, \text{ and } \tau_j - \tau_i \in I\}$$
$$[\![\diamondminus]\!](X, \sigma, i) = \{I \in \mathcal{I}_{\mathbb{Q}} \mid \exists j \in X : \; j \leq i, \text{ and } \tau_i - \tau_j \in I\}.$$

The logic MTL$^\diamond$(Σ) is clearly expressively equivalent to rec-TLTL($\Sigma, \{\diamond, \diamondminus\}$) since the predicates $\diamond_I\theta$ and $I \in \diamond_\theta$ are equivalent. Using Theorem 8 we can now conclude that

Theorem 10. MTL(Σ) *is expressively equivalent to* rec-TFO($\Sigma, \{\diamond, \diamondminus\}$).

Let rec-TFO$_{\neq}$ denote the restriction of rec-TFO to non-singular intervals. Then since the translation of MTL to MTL$^{\diamond}$ does not introduce any singular intervals, and the constructions in Theorem 8 preserve the interval vocabulary of the formulas, we conclude that the logics MITL(Σ) and rec-TFO$_{\neq}(\Sigma, \{\diamond, \diamondminus\})$ are expressively equivalent.

References

1. R. Alur, D. L. Dill: A theory of timed automata, *Theoretical Computer Science* **126**: 183–235 (1994).
2. R. Alur, T. Feder, T. A. Henzinger: The benefits of relaxing punctuality, *J. ACM* **43**, 116–146 (1996).
3. R. Alur, L. Fix, T. A. Henzinger: Event-clock automata: a determinizable class of timed automata, *Proc. 6th International Conference on Computer-aided Verification,* LNCS **818**, 1–13, Springer-Verlag (1994).
4. R. Alur, T. A. Henzinger: Real-time logics: complexity and expressiveness, *Information and Computation* **104**, 35–77 (1993).
5. J. R. Büchi: Weak second-order arithmetic and finite automata, *Zeitschrift für Math. Logik und Grundlagen der Mathematik*, **6**, 66–92 (1960).
6. D. D'Souza: A Logical Characterisation of Event Clock Automata, in *J. Foundations of Computer Science*, **14**, No. 4, World Scientific (2003).
7. D. D'Souza, P. S. Thiagarajan: Product Interval Automata: A Subclass of Timed Automata, *Proc. 19th Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, LNCS **1732** (1999).
8. D. Gabbay, A. Pnueli, S. Shelah, J. Stavi: The Temporal Analysis of Fairness, *Seventh ACM Symposium on Principles of Programming Languages,* 163–173 (1980).
9. T. A. Henzinger, J.-F. Raskin, and P.-Y. Schobbens: The regular real-time languages, *Proc. 25th International Colloquium on Automata, Languages, and Programming 1998*, LNCS **1443**, 580–591 (1998).
10. H. Kamp: Tense Logic and the Theory of Linear Order, PhD Thesis, University of California (1968).
11. R. Koymans: Specifying real-time properties with metric temporal logic, *Real-time Systems*, **2**(4), 255–299 (1990).
12. A. Pnueli: The temporal logic of programs, *Proc. 18th IEEE Symposium on Foundation of Computer Science*, 46–57 (1977).
13. J. -F. Raskin: Logics, Automata and Classical Theories for Deciding Real Time, Ph.D Thesis, FUNDP, Belgium (1999).
14. J. -F. Raskin, P. -Y. Schobbens: State-clock Logic: A Decidable Real-Time Logic, *Proc. HART '97: Hybrid and Real-Time Systems*, LNCS **1201**, 33–47 (1997).
15. W. Thomas: Automata on Infinite Objects, in J. V. Leeuwen (Ed.), *Handbook of Theoretical Computer Science*, Vol. B, 133–191, Elsevier (1990).
16. Th. Wilke: Specifying Timed State Sequences in Powerful Decidable Logics and Timed Automata, in *Proc. Formal Techniques in Real-Time and Fault-Tolerant Systems*, LNCS **863**, 694–715 (1994).