Sādhanā Vol. 27, Part 2, April 2002, pp. 181–208. © Printed in India

Product interval automata

DEEPAK D'SOUZA and P S THIAGARAJAN

Chennai Mathematical Institute, 92 G N Chetty Road, Chennai 600 017, India
e-mail: {deepak,pst}@cmi.ac.in

Abstract. We identify a subclass of timed automata called product interval automata and develop its theory. These automata consist of a network of timed agents with the key restriction being that there is just one clock for each agent and the way the clocks are read and reset is determined by the distribution of shared actions across the agents. We show that the resulting automata admit a clean theory in both logical and language theoretic terms. We also show that product interval automata are expressive enough to model the timed behaviour of asynchronous digital circuits.

Keywords. Timed automata; distributed systems; logic.

1. Introduction

Timed automata as formulated by Alur & Dill (1994) have become a canonical model for describing timed behaviours. It is well-known that these automata are very powerful in language-theoretic terms. Their languages are closed under union and intersection but not under complementation. Further, their language inclusion problem is undecidable and hence cannot be reduced to the emptiness problem which is decidable. Consequently, the verification problem which can be often phrased as whether $L(\mathcal{A}_{Pr}) \subseteq L(\mathcal{A}_{spec})$ cannot be reduced to whether $L(\mathcal{A}_{Pr} \cap \mathcal{A}_{\neg spec}) = \emptyset$. Here $\mathcal{A}_{Pr}$ is the timed automaton modelling a real time program Pr and $\mathcal{A}_{spec}$ is the automaton capturing the specification so that $\mathcal{A}_{\neg spec}$ is the complement of $\mathcal{A}_{spec}$. To get around this, one must use deterministic timed automata for specifications (since they can be easily complemented) or one must work with a restricted class of timed automata that possess the desired closure properties.

Here we follow the second route and propose a subclass of timed automata called product interval automata (PI automata). Such an automaton consists of a network of timed agents $\|_{i=1}^{K}\ \mathcal{A}_i$ where each $\mathcal{A}_i$ operates over an alphabet Σ_i of events. Further, there is a *single* clock c_i associated with each agent i. The agents communicate by synchronising on the timed executions of common events. Suppose a is an event in which the agents $\{1, 3, 4\}$ participate. Then the timing constraint governing each a-execution only involves the clocks $\{c_1, c_3, c_4\}$. Moreover, the set of clocks that is reset at the end of each a-execution is $\{c_1, c_3, c_4\}$. Thus the distribution $\widetilde{\Sigma} = \{\Sigma_i\}_{i=1}^{K}$ of events over the agents canonically determines the usage of clocks; so much so, we can avoid mentioning the clocks altogether once we fix $\widetilde{\Sigma}$.

This method of structuring timed automata has a number of advantages. In particular, one can provide naturally decomposed and succinct presentations of timed automata with large

(control) state spaces. The technique of presenting a global timed automaton as a product of component timed automata has been used by many authors starting from Alur & Dill (1994). What is new here, as explained above, is that our decomposed presentation places a corresponding restriction on the manner in which clocks are read and reset. A related class of hybrid systems is mentioned in passing by Henzinger *et al* (1995) the timed versions of which boil down to PI automata in which there is *no* communication between the agents. Yet another piece of related work is by Yi & Jonsson (1994) in the framework of timed CSP. Their model can be easily represented as PI automata. Their main result, in our terms, is that the language inclusion problem for PI automata is decidable. But in their setting, timing constraints are stated in terms of a *single* integer value whereas we use, as is usual, *intervals* with rational bounds. We establish a variety of results concerning PI automata which subsume the decidability of the language inclusion problem.

Structurally the underlying (symbolic) automata can of course be viewed as labelled Petri nets and hence a PI automaton can also be interpreted as a kind of timed Petri net. The classical timed Petri net model (Merlin & Faber 1976) however uses implicit clocks which record the time since a transition was enabled. For modelling PI automata, one needs to attach clocks to places or – due to the fact we are dealing with 1-safe Petri nets – attach clocks to the individual tokens. The semantics we attach to our automata is strictly along the lines of the literature on timed automata whereas the semantics one traditionally uses for timed Petri nets – with earliest and latest firing times for the transitions – is somewhat different.

A final aspect of PI automata is that due to the disciplined use of clocks across components partial order reduction techniques that are under development (Bengtsson *et al* 1998; Minea 1999) can be readily applied to our automata. See D'Souza (2000a) for further discussion on this.

In pragmatic terms, PI automata – despite their severely restricted usage of clocks – still seem to have a good deal of modelling power. To bring this out, we consider the networks of timed automata that communicate through shared variables used by Maler & Pnueli (1995) to model and analyse the timed behaviour of asynchronous circuits. We show here that PI automata suffice for implementing this very useful modelling technique. Consequently the logical framework accompanying PI automata (detailed below) can be applied to the study of asynchronous circuits. We admit however that much more work needs to be done on the experimental front to test the practical applicability of the models and techniques presented here.

From a theoretical standpoint, PI automata are strictly less expressive than event clock automata due to Alur *et al* (1994) and their state-based version (Raskin & Schobbens 1997) which in turn are strictly less powerful than general timed automata. As a result, the logics we develop here will also be strictly less expressive than the corresponding logics presented by Henzinger *et al* (1998) for a generalisation of event clock automata called recursive event clock automata. Nevertheless we feel that PI automata are of independent interest due to the reasons sketched earlier. They also admit a smoother logical characterisation. In particular, the monadic second order logic presented by Henzinger *et al* (1998) permits only restricted (second-order) quantification. This is not the case for the logical characterisation we obtain. (For basic information about timed automata and their logics see Alur & Henzinger (1992) and Henzinger (1998) and references therein.)

In the next section we begin with some preliminary notions. In § 3 we examine interval automata, which are essentially the components of PI automata. The properties of these automata will play an important role in our study of PI automata in the subsequent section. In § 5 we first show that a monadic second order logic denoted $\mathrm{TMSO}^{\otimes}$ captures the timed regular

languages recognised by PI automata. We then formulate a linear time temporal logic denoted TLTL$^{\otimes}$ and provide automata-theoretic solutions to the satisfiability and model checking problems for TLTL$^{\otimes}$ in terms of PI automata. Section 8 contains a detailed description of how we can model asynchronous circuits using PI automata, as well as some properties that we can specify and verify in our logical framework.

It turns out that all our ideas can be extended smoothly to a larger setting in which the underlying "symbolic" automata are asynchronous Büchi automata (Gastin & Petit 1992). The resulting timed automata are called *distributed interval automata*. We also consider the natural timed extension of "cellular" asynchronous automata, called *cellular interval automata*. These automata can be studied with the help of powerful results available in the theory of Mazurkiewicz traces (Diekert & Rozenberg 1995). Due to space limitations we do not present these extensions here. Details can be found in (D'Souza & Thiagarajan 1998; D'Souza 2000a).

2. Preliminaries

We begin with some useful notions about timed words and timed automata.

As usual, for an alphabet A we will use A^* and A^ω to denote the set of finite and infinite words over A respectively. We will use A^∞ to denote the set $A^* \cup A^\omega$.

It will be necessary for us to deal with both finite and infinite words, and in this regard it is convenient to use prefixes to play the role of positions in a word. For a word σ in A^∞, $prf(\sigma)$ will be used to denote the set of finite prefixes of σ. The strict and non-strict prefix relations on finite words will be denoted by $\prec$ and $\preceq$ respectively. We will use $|\sigma|$ to denote the length of a word σ. The empty word will be denoted by ϵ.

It will be helpful to recall the definition of Büchi automata. The reader is referred to Thomas (1990) for a comprehensive treatment of this subject.

DEFINITION 1

Let A be a finite alphabet. A (mixed) Büchi automaton over the alphabet A is a structure $\mathcal{A} = (Q, \longrightarrow, Q_{in}, F, G)$ where

- Q is a finite set of states,
- $\longrightarrow \subseteq Q \times A \times Q$ is the transition relation,
- $Q_{in} \subseteq Q$ is a set of initial states, and
- $F, G \subseteq Q$ are, respectively, the finitary and infinitary acceptance state sets.

Let $\sigma \in A^\infty$. A run of $\mathcal{A}$ over σ is a map $\rho : prf(\sigma) \to Q$ which satisfies:

- $\rho(\epsilon) \in Q_{in}$.
- $\rho(\tau) \stackrel{a}{\longrightarrow} \rho(\tau a)$ for every $\tau a \in prf(\sigma)$. ($q \stackrel{a}{\longrightarrow} q'$ is alternate notation for $(q, a, q') \in \longrightarrow$).

We say ρ is an *accepting* run on σ iff either

- σ is finite and $\rho(\sigma) \in F$, or,
- σ is infinite and $\rho(\tau) \in G$ for infinitely many $\tau \in prf(\sigma)$.

The set of words accepted by $\mathcal{A}$, denoted (for reasons that will soon be clear) $L_{sym}(\mathcal{A})$, is defined to be the set of words in A^∞ on which $\mathcal{A}$ has an accepting run. Following established

convention, we term a subset L of A^∞ *regular* if $L = L_{sym}(\mathcal{A})$ for some Büchi automaton $\mathcal{A}$ over A.

The notion of a *timed word* is central to this paper. In what follows we will use $\mathbb{R}^{>0}$ and $\mathbb{R}^{\geq 0}$ to denote the set of positive and non-negative reals respectively. The non-negative rationals will be denoted by $\mathbb{Q}^{\geq 0}$.

DEFINITION 2

Let Σ be a finite alphabet of actions. A *timed word* over Σ is a member σ of $(\Sigma \times \mathbb{R}^{>0})^\infty$ such that

(1) for all prefixes $\tau(a,t)(b,t')$ of σ we have $t < t'$ (strict monotonicity).
(2) if σ is infinite, then for each $t \in \mathbb{R}^{>0}$ there exists a prefix $\tau(a,t')$ of σ such that $t' > t$ (progressiveness).

We use $T\Sigma^*$ and $T\Sigma^\omega$ to denote the set of finite and infinite timed words over Σ, respectively, and set $T\Sigma^\infty = T\Sigma^* \cup T\Sigma^\omega$.

For a finite timed word τ we will use $time(\tau)$ to denote the time of occurrence of the last action in τ. Formally, $time(\epsilon) = 0$, and $time(\tau(a,t)) = t$. Analogously, for a non-empty finite timed word τ we will use $action(\tau)$ to denote the last action in τ.

In what follows, we will use intervals with rational bounds to specify timing constraints (and use ∞ as the upper bound to capture unbounded intervals). These intervals will be of the form (l,r), $[l,r)$, $(l,r]$, or $[l,r]$, where $l, r \in \mathbb{Q}^{\geq 0} \cup \{\infty\}$ with $l \leq r$. For an interval of the form $(l,r]$ or $[l,r]$ we require $r \neq \infty$. Further, to avoid empty intervals, unless an interval is of the form $[l,r]$, we require $l < r$. An interval will denote a non-empty, convex subset of reals in the obvious way. For example the interval $[1,\infty)$ denotes the set $\{t \in \mathbb{R} \mid 1 \leq t\}$. The set of all intervals will be denoted by $\mathcal{I}\mathbb{R}$.

DEFINITION 3

A (mixed) *timed Büchi automaton*, TBA for short, over an alphabet Σ is a structure $\mathcal{A} = (Q, \longrightarrow, C, Q_{in}, F, G)$ where:

- Q is a finite set of states,
- $Q_{in} \subseteq Q$ is a set of initial states,
- $F, G \subseteq Q$ are sets of finitary and infinitary accepting states,
- C is a finite set of clocks, and
- $\longrightarrow$, the transitions of $\mathcal{A}$, is a finite subset of $Q \times \Sigma \times 2^C \times G_C \times Q$ where G_C is the set of clock constraints (guards) which are conjunctions of atomic guards of the form $(x \in I)$, where x ranges over C and I ranges over $\mathcal{I}\mathbb{R}$.

In what follows, a transition (q, a, X, g, q') will be written as $q \xrightarrow[X]{a,g} q'$. The manner in which the timed automaton accepts a timed word is defined in terms of clock valuations.

A C-valuation is a map $v : C \to \mathbb{R}^{\geq 0}$. Where C is clear from the context we will say valuation instead of C-valuation. We let val_C stand for the set of C-valuations. Let v be a valuation and $t \in \mathbb{R}^{\geq 0}$. Then $v + t$ is the valuation given by:

$$(v+t)(x) = v(x) + t \text{ for every } x \in C.$$

Suppose v is a valuation, $t \in \mathbb{R}^{\geq 0}$ and $X \subseteq C$. Then the valuation $v[t/X]$ is given by:

$$v[t/X](y) = \begin{cases} t, & \text{if } y \in X. \\ v(y), & \text{otherwise.} \end{cases}$$

Finally, **0** is the null-valuation given by:

$$\mathbf{0}(x) = 0, \text{ for every } x \in C.$$

Next, the notion of a valuation v satisfying a clock constraint g is denoted by $v \models g$ and is defined via:

- $v \models x \in I$, iff $v(x) \in I$
- $v \models \varphi \wedge \varphi'$, iff $v \models \varphi$ and $v \models \varphi'$.

Let $\sigma \in T\Sigma^\infty$. Then a run of $\mathcal{A}$ over σ is a pair of maps (ρ, ν) where $\rho : prf(\sigma) \to Q$ and $\nu : prf(\sigma) \to val_C$ are such that the following conditions are satisfied:

- $\rho(\epsilon) \in Q_{in}$ and $\nu(\epsilon) = \mathbf{0}$
- For every $\tau(a,t) \in prf(\sigma)$, there exists a transition $\rho(\tau) \xrightarrow{a,g}_{X} \rho(\tau(a,t))$ such that $\nu(\tau) + t \models g$ and $\nu(\tau(a,t)) = (\nu(\tau) + t)[0/X]$.

The run (ρ, ν) is an accepting run iff either

(1) σ is finite and $\rho(\sigma) \in F$, or,
(2) σ is infinite and $\rho(\tau) \in G$ for infinitely many prefixes τ of σ.

$L(\mathcal{A}) \subseteq T\Sigma^\infty$ the language of timed words accepted by $\mathcal{A}$ is then given by:

$$L(\mathcal{A}) = \{\sigma \mid \exists \text{ an accepting run of } \mathcal{A} \text{ over } \sigma\}.$$

For a timed word $\sigma \in T\Sigma^\infty$ let $untime(\sigma)$ be the word $\widehat{\sigma} \in \Sigma^\infty$ obtained by projecting away the time-stamps from σ. For a timed language $L \subseteq T\Sigma^\infty$, $untime(L) \subseteq \Sigma^\infty$ will denote the set $\{untime(\sigma) \mid \sigma \in L\}$.

Alur & Dill (1994) showed the following result:

Theorem 1. *Given a timed automaton $\mathcal{A}$ over an alphabet Σ one can effectively construct a Büchi automaton $\mathcal{A}'$ over Σ such that $L(\mathcal{A}') = untime(L(\mathcal{A}))$.*

Using the above construction, we can check if the timed language accepted by a TBA $\mathcal{A}$ is empty in time

$$O(|Q| + |E|) \cdot 2^{O(|C|)} \cdot |C|! \cdot \prod_{x \in C}(c_x + 1),$$

where Q and E are the state and edge sets of $\mathcal{A}$ respectively, C is the set of clocks used in $\mathcal{A}$, and for each $x \in C$, c_x is the largest normalised interval bound appearing in $\mathcal{A}$ in a guard of the form $(x \in I)$.

Using standard convention we let $|\mathcal{A}|$ denote the size of the representation of $\mathcal{A}$, using binary encoding for the numeric constants. Then the above expression can be seen to bounded by $2^{(|\mathcal{A}|^2)}$.

3. Interval automata

Product interval automata are essentially a network of very simple timed automata called interval automata. It is convenient to first examine these automata and establish some results about them which help us to study product interval automata.

Interval automata are timed automata, with a single clock which must be reset along every transition. Effectively, these automata can only measure the time elapsed since the last action performed.

The notion of an *interval alphabet* will be useful in representing these timed languages symbolically. Let Σ be a finite alphabet. Then an *interval alphabet* based on Σ is a finite subset of $\Sigma \times \mathcal{I}\mathbb{R}$.

Given an interval alphabet Γ over Σ and a word $\widehat{\sigma} \in \Gamma^\infty$, $\widehat{\sigma}$ naturally induces a set of timed words over Σ which we denote $tw(\widehat{\sigma})$. It is defined as follows. Let $\sigma \in T\Sigma^\infty$. Then $\sigma \in tw(\widehat{\sigma})$ iff

(1) $|\sigma| = |\widehat{\sigma}|$.
(2) For each prefix $\tau(a, t)$ of σ, and for each prefix $\widehat{\tau}(b, I)$ of $\widehat{\sigma}$ such that $|\tau| = |\widehat{\tau}|$, we have $a = b$ and $t - time(\tau) \in I$.

For $L \subseteq \Gamma^\infty$ we set

$$tw(L) = \bigcup_{\widehat{\sigma} \in L} tw(\widehat{\sigma}).$$

DEFINITION 4

An *interval automaton* over Σ is simply a Büchi automaton over an interval alphabet based on Σ.

Thus an interval automaton $\mathcal{A}$ over Σ has edges of the form $q \xrightarrow{(a,I)} q'$, where a is a Σ-action and I is an interval. Viewed as a Büchi automaton over an interval alphabet Γ, $\mathcal{A}$ accepts the symbolic language $L_{sym}(\mathcal{A}) \subseteq \Gamma^\infty$. What is more interesting to us however, is the timed language accepted by $\mathcal{A}$, denoted $L(\mathcal{A})$, which we define as

$$L(\mathcal{A}) = tw(L_{sym}(\mathcal{A})).$$

We will say a timed language $L \subseteq T\Sigma^\infty$ is a *regular interval language* if $L = L(\mathcal{A})$ for some interval automaton $\mathcal{A}$ over Σ.

Example 1. Figure 1 shows an interval automaton $\mathcal{A}$ over the alphabet $\Sigma = \{a, b\}$. Here we have $\mathcal{A} = (\{q_0, q_1\}, \longrightarrow, \{q_0\}, \{q_1\}, \{q_1\})$. In the figure we use the convention that the initial states are indicated with incoming double arrows, while final states are indicated by two concentric circles. The automaton accepts all timed words $\sigma \in T\Sigma^\infty$ which begin with a (possibly empty) sequence of a actions, unit times apart, followed by a b action at some time between 1 and 2 units from the last action, and finally a sequence of a's again, without any time restrictions. □

We can also define the notion of a run of an interval automaton directly over a timed word. Let $\mathcal{A} = (Q, \longrightarrow, Q_{in}, F, G)$ be an interval automaton over Σ, and let $\sigma \in T\Sigma^\infty$. Then a run of $\mathcal{A}$ on σ is map $\rho : prf(\sigma) \to Q$ such that

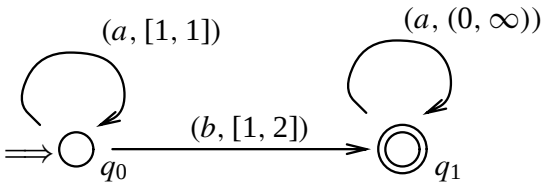

Figure 1. An interval automaton over $\{a, b\}$.

(1) $\rho(\epsilon) \in Q_{in}$,

(2) for each $\tau(a,t) \in prf(\sigma)$, there exists I, such that $\rho(\tau) \xrightarrow{(a,I)} \rho(\tau(a,t))$ and $(t - time(\tau)) \in I$.

As usual, the run ρ will be termed accepting if either σ is finite and $\rho(\sigma) \in F$, or, σ is infinite and $\rho(\tau) \in G$ for infinitely many $\tau \in prf(\sigma)$.

It is not difficult to see that this is an equivalent way of defining the timed language of $\mathcal{A}$ – in the sense that $\sigma \in tw(L_{sym}(\mathcal{A}))$ iff there exists an accepting run of $\mathcal{A}$ on σ, in the sense defined above.

We now show that the class of regular interval languages is closed under boolean operations. It is easy to see that this class is closed under union. For closure under complementation the notion of a *proper* interval set proves useful. This notion also plays an important role in subsequent sections.

We say a set of intervals $\mathcal{I} \subseteq \mathcal{I}\mathbb{R}$ is *proper* if it forms a finite partition of $\mathbb{R}^{\geq 0}$. Thus, if $\mathcal{I}$ is a proper interval set, then for each $t \in \mathbb{R}^{\geq 0}$ there exists an $I \in \mathcal{I}$ such that $t \in I$, and for each $I, I' \in \mathcal{I}$, $I \cap I' \neq \emptyset$ implies $I = I'$. An interval alphabet Γ is termed proper if for each $a \in \Sigma$ the set $\Gamma_a = \{I \mid (a, I) \in \Gamma\}$ is a proper interval set. We say an interval set $\mathcal{I}$ *covers* an interval set $\mathcal{I}'$ if every interval in $\mathcal{I}'$ is the union of some subset of intervals in $\mathcal{I}$. Finally, an interval alphabet Γ covers the interval alphabet Γ' if Γ_a covers Γ'_a for each $a \in \Sigma$.

Each interval alphabet Γ induces, in a canonical way, a proper interval alphabet, denoted $prop(\Gamma)$, with the property that it covers Γ. It is given by

$$prop(\Gamma) = \{(a, I) \mid a \in \Sigma,\ I \in prop(\Gamma_a)\}$$

where for each a, the set $prop(\Gamma_a)$ is obtained from Γ_a by the procedure outlined below.

Let $\mathcal{I}$ be a non-empty finite set of intervals (if it is empty, we simply set $prop(\mathcal{I}) = \{[0, \infty)\}$). Let $V = \{0, v_1, v_2, \ldots, v_n, \infty\}$ where for $1 \leq i \leq n$, $v_i \in V$ iff there exists $I \in \mathcal{I}$ with v_i as the left or right end of I. Without loss of generality, we assume that $n \geq 1$ and $0 < v_1 < v_2 \cdots < v_n \neq \infty$. Now define $prop(\mathcal{I})$ via:

$$prop(\mathcal{I}) = \{[v_j, v_j],\ (v_j, v_{j+1}) \mid 0 \leq j \leq n\}$$

where we set $v_0 = 0$ and $v_{n+1} = \infty$. It is easy to verify that $prop(\mathcal{I})$ is a proper interval set which covers $\mathcal{I}$.

The following is an important property of proper interval alphabets.

Lemma 1. *Let Γ be a proper interval alphabet based on Σ. Then for each $\sigma \in T\Sigma^\infty$ there exists a unique word $\widehat{\sigma} \in \Gamma^\infty$ such that $\sigma \in tw(\widehat{\sigma})$.*

Proof. Let $\sigma \in T\Sigma^\infty$. Since Γ is proper we know that for each $a \in \Sigma$ and $t \in \mathbb{R}^{>0}$ there exists a unique $I \in \Gamma_a$ such that $t \in I$. Consider the word $\widehat{\sigma} \in \Gamma^\infty$ given by its set of prefixes which we define as follows. For each prefix τ of σ we define a corresponding prefix $\widehat{\tau}$ of $\widehat{\sigma}$. The prefix corresponding to ϵ is ϵ itself. The prefix corresponding to the non-empty prefix $\tau(a,t)$ of σ is $\widehat{\tau}(a, I)$ where I is the unique interval in Γ_a such that $(t - time(\tau)) \in I$. It is easy to verify that $\sigma \in tw(\widehat{\sigma})$.

For uniqueness of $\widehat{\sigma}$ suppose $\sigma \in tw(\widehat{\sigma}_1)$ and $\sigma \in tw(\widehat{\sigma}_2)$ for some $\widehat{\sigma}_1, \widehat{\sigma}_2 \in \Gamma^\infty$. If $\widehat{\sigma}_1 \neq \widehat{\sigma}_2$ then there must exist prefixes $\widehat{\tau}_1(a, I)$ of $\widehat{\sigma}_1$ and $\widehat{\tau}_2(a, I')$ of $\widehat{\sigma}_2$, such that $|\widehat{\tau}_1| = |\widehat{\tau}_2|$ and $I \neq I'$. Let $\tau(a,t)$ be a prefix of σ such that $|\tau| = |\widehat{\tau}_1|$. Then we know $(t - time(\tau))$ belongs to both I and I'. Since Γ_a is proper, this would mean $I = I'$ which contradicts our assumption. Thus we must have $\widehat{\sigma}_1 = \widehat{\sigma}_2$. □

Using properties of proper alphabets we can now show closure under complementation. Let L be a regular interval language over Σ. It is not difficult to see that there exists a proper interval alphabet Γ based on Σ and a regular subset $\widehat{L}$ of Γ^∞ such that $L = tw(\widehat{L})$. We now claim that $T\Sigma^\infty - L = tw(\Gamma^\infty - \widehat{L})$. We can prove this easily using lemma 1.

Since regular languages are closed under complement, we know that $\Gamma^\infty - \widehat{L}$ is regular, and hence $T\Sigma^\infty - L$ is a regular interval language. We now have:

Theorem 2. *The class of regular interval languages over an alphabet Σ is closed under the boolean operations of union, intersection and complementation.* □

Next we introduce a monadic second-order logic interpreted over timed words, which characterises the class of regular interval languages. This logic is called TMSO(Σ) and is parameterised by the alphabet Σ.

Here and in the logics to follow, we assume a supply of individual variables $x, y, \ldots$, and set-variables $X, Y, \ldots$. These variables range over prefixes (respectively sets of prefixes) of the timed word in question. We make use of the predicates $Q_a(x)$ (one for each $a \in \Sigma$) and $\Delta(x, I)$, where x is an individual variable and I is an element of $\mathcal{I}\mathbb{R}$. The syntax of TMSO(Σ) is given by:

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

A structure for a formula of the logic is a pair $(\sigma, \mathbb{I})$ where $\sigma \in T\Sigma^\infty$ and $\mathbb{I}$ is an interpretation which assigns to each individual variable a non-empty prefix of σ, and to each set variable a set of non-empty prefixes of σ. We depart slightly from classical monadic logics by using prefixes instead of natural numbers to play the role of positions in a word. Once again, this is more convenient for us given that we are dealing with both finite and infinite words. Correspondingly, $<$ will be interpreted as the the strict prefix relation $\prec$ on finite words.

The satisfaction relation $\sigma \models_{\mathbb{I}} \varphi$ for atomic formulas φ is given as follows:

$$\begin{array}{lll}
\sigma \models_{\mathbb{I}} (x \in X), & \text{iff} & \mathbb{I}(x) \in \mathbb{I}(X), \\
\sigma \models_{\mathbb{I}} (x < y), & \text{iff} & \mathbb{I}(x) \prec \mathbb{I}(y), \\
\sigma \models_{\mathbb{I}} Q_a(x), & \text{iff} & action(\mathbb{I}(x)) = a, \\
\sigma \models_{\mathbb{I}} \Delta(x, I), & \text{iff} & \mathbb{I}(x) \text{ is of the form } \tau(a, t) \text{ and } (t - time(\tau)) \in I.
\end{array}$$

The operators $\neg$, $\vee$, and the existential quantifiers $\exists x$ and $\exists X$ are interpreted in the usual manner: Let $\mathbb{I}$ be an interpretation for variables with respect to σ. Let τ be a prefix of σ. We use the notation $\mathbb{I}[\tau/x]$ to denote the interpretation which maps x to τ and agrees with $\mathbb{I}$ on all other individual and set variables. Similarly, for a set of prefixes S of σ, the notation $\mathbb{I}[S/X]$ denotes the interpretation which sends X to S, and agrees with $\mathbb{I}$ on all other variables.

$$\begin{array}{lll}
\sigma \models_{\mathbb{I}} \neg\varphi, & \text{iff} & \sigma \not\models_{\mathbb{I}} \varphi, \\
\sigma \models_{\mathbb{I}} (\varphi \vee \varphi'), & \text{iff} & \sigma \models_{\mathbb{I}} \varphi \text{ or } \sigma \models_{\mathbb{I}} \varphi', \\
\sigma \models_{\mathbb{I}} \exists x\varphi, & \text{iff} & \text{there exists } \tau \preceq \sigma \text{ such that } \sigma \models_{\mathbb{I}[\tau/x]} \varphi, \\
\sigma \models_{\mathbb{I}} \exists X\varphi, & \text{iff} & \text{there exists } S \subseteq prf(\sigma) \text{ such that } \sigma \models_{\mathbb{I}[S/X]} \varphi.
\end{array}$$

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

Example 2. Let $\Sigma = \{a, b\}$. Then the following TMSO(Σ)-sentence describes the language accepted by the automaton $\mathcal{A}$ in example 1.

$$\begin{aligned}\varphi_1 \;=\; & \exists x(Q_b(x) \wedge \Delta(x,[1,2]) \wedge \\ & \forall y((y < x) \;\Rightarrow\; (Q_a(y) \wedge \Delta(y,[1,1]))) \wedge \\ & \forall y((x < y) \;\Rightarrow\; Q_a(y))).\end{aligned}$$

□

Theorem 3. *Let $L \subseteq T\Sigma^\infty$. Then L is a regular interval language iff $L = L(\varphi)$ for some sentence φ in* TMSO(Σ).

To prove this theorem, we will use Büchi's monadic second-order logic characterisation of regular languages. We recall that for an alphabet A, the syntax of Büchi's monadic second order logic (denoted here by MSO(A)) is:

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

A structure for this logic is a pair of the form $(\sigma, \mathbb{I})$ where $\sigma \in A^\infty$ and $\mathbb{I}$ assigns to individual and set variables, non-empty prefixes and sets of non-empty prefixes of σ respectively. The semantics of the logic is given in a similar manner to that of TMSO(Σ). In particular, the atomic formula $Q_a(x)$ – here a is required to be in A – is interpreted as follows:

$$\sigma \models_{\mathbb{I}} Q_a(x) \text{ iff } \mathbb{I}(x) = \tau a \text{ for some } \tau \preceq \sigma.$$

As usual, for a sentence φ in MSO(A) we set $L(\varphi) = \{\sigma \in A^\infty \mid \sigma \models \varphi\}$. Büchi's result states that a language $L \subseteq A^\infty$ is accepted by a Büchi automaton over the alphabet A iff $L = L(\varphi)$ for some sentence φ in MSO(A) (Büchi 1960; Thomas 1990).

Now, given a formula $\varphi \in$ TMSO(Σ) we show how to translate it to a formula $t\text{-}s(\varphi) \in$ MSO(Γ), for a suitably defined interval alphabet Γ. The translation will preserve – in a sense to be made precise – the timed models of φ. (The name *t-s* is the acronym for "timed-to-symbolic".) Let Γ be any proper interval alphabet over Σ such that for each $a \in \Sigma$, Γ_a covers

$$voc(\varphi) = \{I \mid \varphi \text{ has a subformula of the form } \Delta(x, I)\}.$$

Note that $\Sigma \times prop(voc(\varphi))$ is at least one such Γ. Then $t\text{-}s(\varphi)$ (w.r.t. Γ) is obtained from φ by replacing sub-formulas of the form $Q_a(x)$ by the formula

$$\bigvee_{(b,I)\in\Gamma,\ b=a} Q_{(b,I)}(x),$$

and sub-formulas of the form $\Delta(x, I)$ by the formula

$$\bigvee_{(a,I')\in\Gamma,\ I'\subseteq I} Q_{(a,I')}(x).$$

Lemma 2. *Let $\varphi \in$ TMSO(Σ) and let Γ be a proper interval alphabet based on Σ with the property that for each $a \in \Sigma$, Γ_a covers $voc(\varphi)$. Let $\widehat{\sigma} \in \Gamma^\infty$ and $\sigma \in T\Sigma^\infty$ be such that $\sigma \in tw(\widehat{\sigma})$. Suppose further that $\mathbb{I}$ is an interpretation for variables with respect to σ, and $\widehat{\mathbb{I}}$ is the corresponding interpretation for variables w.r.t. $\widehat{\sigma}$, given by $\widehat{\mathbb{I}}(x) = \widehat{\tau}$ where $\widehat{\tau} \preceq \widehat{\sigma}$ is such that $|\widehat{\tau}| = |\mathbb{I}(x)|$, and $\widehat{\mathbb{I}}(X) = \{\widehat{\mathbb{I}}(x) \mid x \in X\}$. Then*

(1) *$\sigma \models_{\mathbb{I}} \varphi$ iff $\widehat{\sigma} \models_{\widehat{\mathbb{I}}} t\text{-}s(\varphi)$.*
(2) *If φ is a sentence, then $L(\varphi) = tw(L(t\text{-}s(\varphi)))$.*

Proof. (1) We prove the statement by induction on the structure of φ. The interesting cases are $\varphi = Q_a(x)$ and $\varphi = \Delta(x, I)$.

Case $\varphi = Q_a(x)$: We know $\sigma \models_{\mathbb{I}} Q_a(x)$ iff $action(\mathbb{I}(x)) = a$. But since $\sigma \in tw(\widehat{\sigma})$, we know that this holds iff $\widehat{\mathbb{I}}(x) = \widehat{\tau}(a, I)$ for some $\widehat{\tau}$ and I such that $(a, I) \in \Gamma$. This in turn holds iff $\widehat{\sigma} \models_{\widehat{\mathbb{I}}} \bigvee_{(b,I)\in\Gamma,\ b=a} Q_{(b,I)}(x)$.

Case $\varphi = \Delta(x, I)$: Let $\sigma \models_{\mathbb{I}} \Delta(x, I)$. Then we know that $\mathbb{I}(x) = \tau(a, t)$ for some τ, a, t such that $(t - time(\tau)) \in I$. Further, since $\sigma \in tw(\widehat{\sigma})$, we know that $\widehat{\mathbb{I}}(x) = \widehat{\tau}(a, I')$ for some I' such that $(a, I') \in \Gamma$, and $(t - time(\tau)) \in I'$. Using the fact that Γ is proper and covers $voc(\varphi)$, and $(t - time(\tau)) \in I \cap I'$, it must be the case that $I' \subseteq I$. Hence

$$\widehat{\sigma} \models_{\widehat{\mathbb{I}}} \bigvee_{(a,I')\in\Gamma,\ I'\subseteq I} Q_{(a,I')}(x).$$

Conversely, let

$$\widehat{\sigma} \models_{\widehat{\mathbb{I}}} \bigvee_{(a,I')\in\Gamma,\ I'\subseteq I} Q_{(a,I')}(x).$$

Then $\widehat{\mathbb{I}}(x) = \widehat{\tau}(a, I')$ for some $(a, I') \in \Gamma$ with $I' \subseteq I$. Since $\sigma \in tw(\widehat{\sigma})$ it must be the case that $\mathbb{I}(x) = \tau(a, t)$ such that $(t - time(\tau)) \in I'$. Thus $(t - time(\tau)) \in I$, and it follows that $\sigma \models_{\mathbb{I}} \Delta(x, I)$.

(2) This follows easily from (1) above. □

We now show how we can associate a formula *s-t*$(\widehat{\varphi}) \in$ TMSO(Σ) with a formula $\widehat{\varphi} \in$ MSO(Γ), such that the translated formula preserves timed models. The formula *s-t*$(\widehat{\varphi})$ is obtained by replacing atomic sub-formulas in $\widehat{\varphi}$ of the form $Q_{(a,I)}(x)$ by the formula

$$Q_a(x) \wedge \Delta(x, I).$$

Using arguments along the lines of the previous lemma, one can show that:

Lemma 3. *Let Γ be a proper interval alphabet based on Σ and let $\widehat{\varphi} \in$ MSO(Γ). Let $\widehat{\sigma} \in \Gamma^\infty$ and $\sigma \in T\Sigma^\infty$ such that $\sigma \in tw(\widehat{\sigma})$. Suppose further that $\widehat{\mathbb{I}}$ is an interpretation for variables w.r.t. $\widehat{\sigma}$ and let $\mathbb{I}$ be the corresponding interpretation w.r.t. σ. Then*

(1) *$\sigma \models_{\mathbb{I}}$ s-t$(\widehat{\varphi})$ iff $\widehat{\sigma} \models_{\widehat{\mathbb{I}}} \widehat{\varphi}$.*
(2) *If φ is a sentence, then we have L(s-t$(\widehat{\varphi})) = tw(L(\widehat{\varphi}))$.* □

We can now prove theorem 3. Let L be a regular interval language over Σ. We observe again that there exists a proper interval alphabet Γ based on Σ and a regular subset $\widehat{L}$ of Γ^∞ such that $L = tw(\widehat{L})$. Büchi's theorem tells us that there exists an MSO(Γ)-sentence $\widehat{\varphi}$ such that $L(\widehat{\varphi}) = \widehat{L}$. Hence $L = tw(L(\widehat{\varphi}))$. Thus, by lemma 3, we have a TMSO(Σ)-sentence, namely $\varphi =$ *s-t*$(\widehat{\varphi})$, such that $L = L(\varphi)$.

Conversely, let φ be a TMSO(Σ)-sentence. Then, using lemma 2, we know that there exists a proper interval alphabet Γ and a formula $\widehat{\varphi} =$ *t-s*(φ) in MSO(Γ), such that $L(\varphi) = tw(L(\widehat{\varphi}))$. Using Büchi's theorem once more, we are assured of an interval automaton $\mathcal{A}$ over Γ such that $L_{sym}(\mathcal{A}) = L(\widehat{\varphi})$. Thus $\mathcal{A}$ is such that

$$L(\mathcal{A}) = tw(L_{sym}(\mathcal{A})) = tw(L(\widehat{\varphi})) = L(\varphi).$$

□

4. Product interval automata

Product interval automata are essentially a network of interval automata. We have an alphabet of actions which is distributed over locations. Each location runs an interval automaton over its local alphabet. Communication takes place between these automata by enforcing that locations synchronise on common actions.

We will need to set up some notation again. Let $\mathcal{P} = \{1, 2, \ldots, k\}$ be a finite set of agents, or locations. A $\mathcal{P}$-distributed alphabet is a family $\widetilde{\Sigma} = \{\Sigma_i\}_{i \in \mathcal{P}}$ where each Σ_i is a finite set of actions. We set $\Sigma = \bigcup_{i \in \mathcal{P}} \Sigma_i$ and call it the global alphabet induced by $\widetilde{\Sigma}$. The set of agents that participate in each occurrence of the action a will be denoted by $loc(a)$ and is given by: $loc(a) = \{i \in \mathcal{P} \mid a \in \Sigma_i\}$.

Through the rest of this section we fix such a set of agents $\mathcal{P}$ and a $\mathcal{P}$-distributed alphabet $\widetilde{\Sigma}$.

Since we will be considering timed languages in a distributed setting, the assumption that action occurrences in a timed word are separated by a non-zero amount of time is no longer valid. Towards this end we re-define the notion of a timed word over a *distributed* alphabet to allow the simultaneous occurrence of *independent* actions. As the reader may guess, actions a and b will be said to be independent if $loc(a) \cap loc(b) = \emptyset$.

DEFINITION 5

A timed word σ over $\widetilde{\Sigma}$ is an element of $(\Sigma \times \mathbb{R}^{>0})^{\infty}$ such that:

(i) if $\tau(a, t)(b, t')$ is a prefix of σ then $t \leq t'$ (non-decreasing).
(ii) if $\tau(a, t)\tau'(b, t')$ is a prefix of σ with $t = t'$, then $loc(a) \cap loc(b) = \emptyset$ (simultaneous actions must be independent).
(iii) if σ is infinite, then for each $t \in \mathbb{R}^{>0}$ there exists a prefix $\tau(a, t')$ of σ such that $t < t'$ (progressiveness).

We let $T\widetilde{\Sigma}^*$ and $T\widetilde{\Sigma}^{\omega}$ denote the set of finite and infinite timed words over $\widetilde{\Sigma}$ respectively, and set $T\widetilde{\Sigma}^{\infty} = T\widetilde{\Sigma}^* \cup T\widetilde{\Sigma}^{\omega}$.

Let $\sigma \in T\widetilde{\Sigma}^{\infty}$. Then $\sigma{\restriction}i$ is the i-projection of σ. It is the timed word over Σ_i obtained by erasing from σ all appearances of letters of the form (a, t) with $a \notin \Sigma_i$. It is easy to check that $\sigma{\restriction}i$ does indeed belong to $T\Sigma_i^{\infty}$.

For a finite timed word τ, we will use $time_i(\tau)$ to denote the time of occurrence of the last i-action in τ. More formally:

DEFINITION 6

Let $\tau \in T\widetilde{\Sigma}^*$. Then $time_i(\tau)$ is given inductively by:

- $time_i(\epsilon) = 0$.
- $time_i(\tau(a, t)) = t$ if $a \in \Sigma_i$, and equals $time_i(\tau)$ otherwise.

We are now ready to define product interval automata.

DEFINITION 7

A *product interval automaton* over $\widetilde{\Sigma}$ is a structure $(\{\mathcal{A}_i\}_{i \in \mathcal{P}}, Q_{in})$, where for each i, $\mathcal{A}_i$ is a structure $(Q_i, \longrightarrow_i, F_i, G_i)$ where

- Q_i is a finite set of states

- $\longrightarrow_i$ is a finite subset of $Q_i \times (\Sigma_i \times \mathcal{I}\mathbb{R}) \times Q_i$
- $F_i, G_i \subseteq Q_i$ are, respectively, finitary and infinitary acceptance state sets.

$Q_{in} \subseteq Q = Q_1 \times \cdots \times Q_k$ is a set of global initial states.

Let $\mathcal{A} = (\{\mathcal{A}_i\}_{i\in\mathcal{P}}, Q_{in})$ be a product interval automaton over $\widetilde{\Sigma}$ and let $\sigma \in T\widetilde{\Sigma}^\infty$. Then a run of $\mathcal{A}$ over σ is a map $\rho : prf(\sigma) \to Q$ such that

(1) $\rho(\epsilon) \in Q_{in}$
(2) for each prefix $\tau(a,t)$ of σ we have

(a) for each $i \in loc(a)$, there exists a transition $\rho(\tau)[i] \xrightarrow{(a,I)}_i \rho(\tau(a,t))[i]$ with $(t - time_i(\tau)) \in I$.
(b) for each $i \notin loc(a)$ we have $\rho(\tau)[i] = \rho(\tau(a,t))[i]$.

A run ρ of $\mathcal{A}$ on σ is accepting iff for each $i \in \mathcal{P}$ either

(i) $\sigma\restriction i$ is finite and $\rho(\tau)[i] \in F_i$ for any prefix τ of σ such that $\tau\restriction i = \sigma\restriction i$, or
(ii) $\sigma\restriction i$ is infinite and $\rho(\tau)[i] \in G_i$ for infinitely many $\tau \in prf(\sigma)$.

We set $L(\mathcal{A})$ to be the set of words in $T\widetilde{\Sigma}^\infty$ accepted by $\mathcal{A}$ (i.e. those on which $\mathcal{A}$ has an accepting run).

Notice that the components $\mathcal{A}_i$ of $\mathcal{A}$ are interval automata over Σ_i, except that they have no start states. The global initial states determine which combination of states the component automata can start in. Thus, if $\mathcal{A}_i$ are interval automata over Σ_i respectively, with each $\mathcal{A}_i = (Q_i, \longrightarrow_i, Q^i_{in}, F_i, G_i)$, and $Q_{in} \subseteq Q_1 \times \cdots \times Q_k$, then we will often use $(\{\mathcal{A}_i\}_{i\in\mathcal{P}}, Q_{in})$ to denote the product interval automaton $(\{\mathcal{B}_i\}_{i\in\mathcal{P}}, Q_{in})$, where $\mathcal{B}_i = (Q_i, \longrightarrow_i, F_i, G_i)$.

Example 3. Figure 2 shows a product interval automaton over the distributed alphabet $\widetilde{\Sigma} = (\{a,b\},\{b\})$. The language accepted by the automaton is:

$$L(\mathcal{A}) = \{(b,t)(a,t')(b,t'') \in T\Sigma^\infty \mid t'' - t > 1\}.$$

It is not difficult to argue (see D'Souza 2000a) that L cannot be accepted by an interval automaton over the alphabet $\Sigma = \{a,b\}$. □

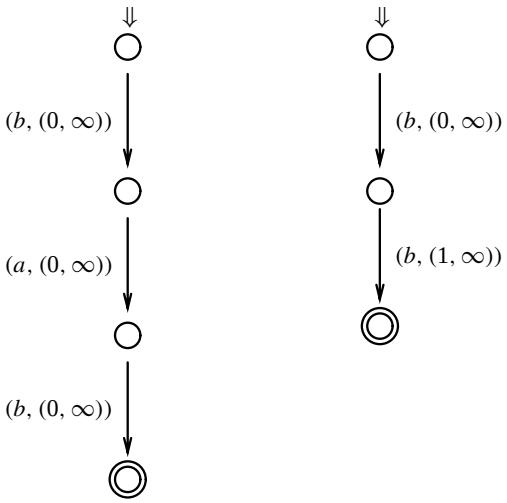

Figure 2. A product interval automaton over $\widetilde{\Sigma} = \{\{a,b\},\{b\}\}$.

4.1 *Checking emptiness*

We show how to simulate a product interval automaton using a timed Büchi automaton. This will then give us a way of checking emptiness for our automata, using the region construction of Alur & Dill (1994).

Let $L_{nd}(\mathcal{A})$ denote the language of non-decreasing timed words accepted by a timed automaton $\mathcal{A}$. The region construction can be modified easily to accept the untiming of $L_{nd}(\mathcal{A})$.

Now let $\mathcal{A} = (\{\mathcal{A}_i\}_{i\in\mathcal{P}}, Q_{in})$ be a product interval automaton, with each $\mathcal{A}_i = (Q_i, \longrightarrow_i, F_i, G_i)$. As a first step we define a TBA $\mathcal{B}$ with the property that $L_{nd}(\mathcal{B}) \cap T\widetilde{\Sigma}^\infty = L(\mathcal{A})$. Without loss of generality, we assume that each local finitary final state is "terminal" in the sense that there are no outgoing edges from them. (This can be done by essentially making a "terminal" copy of each finitary final state.) Define $\mathcal{B} = (S, \longrightarrow, S_{in}, C, F, G)$ where:

- the set of states is $S = (\prod_{i=1}^k Q_i) \times \{0, \ldots, k\}$;
- the set of clocks is $C = \{x_i \mid i \in \mathcal{P}\}$;
- its transition relation is given as follows. We have $(q, l) \xrightarrow[X]{a, g} (q', m)$ iff the following conditions hold:

  (1) $X = \{x_i \mid i \in loc(a)\}$,
  (2) for each $i \in loc(a)$ there exists transitions $q[i] \xrightarrow{(a, I_i)}_i q'[i]$ such that $g = \bigwedge_{i\in loc(a)}(x_i \in I_i)$,
  (3) for each $i \notin loc(a)$, $q[i] = q'[i]$,
  (4) $m = (l + 1) \bmod (k + 1)$ if $q'[l] \in F_l \cup G_l$ or $l = 0$; otherwise $m = l$;

- the initial states are given by $S_{in} = Q_{in} \times \{0\}$;
- the finitary final states F are given by $F = (\prod_{i=1}^k F_i) \times \{0, \ldots, k\}$;
- the infinitary final states are given by $G = \{(q, l) \in S \mid l = 0\}$.

It is not difficult to argue that $L_{nd}(\mathcal{B}) \cap T\widetilde{\Sigma}^\infty = L(\mathcal{A})$.

Next we get rid of words in $L_{nd}(\mathcal{B})$ which are *not* in $T\widetilde{\Sigma}^\infty$. To do this we intersect $\mathcal{B}$ with a TBA $\mathcal{B}'$ which accepts precisely the language $T\widetilde{\Sigma}^\infty$. $\mathcal{B}'$ will have a single clock, and its set of states will be $\{0, 1\}^k$. The clock is used to check whether an action is a zero-time one, while the bits in the state keep track of the components which have already taken part in the last stretch of zero-time actions.

Let $\mathcal{B}''$ be the TBA obtained by intersecting $\mathcal{B}$ and $\mathcal{B}'$. Then $\mathcal{B}''$ is the TBA promised above, with $L_{nd}(\mathcal{B}'') = L(\mathcal{A})$.

Using the check for emptiness of the language accepted by TBA's outlined in § 2, we can check if $L(\mathcal{A})$ is empty or not.

We now analyse the time complexity of checking the emptiness of $\mathcal{A}$ via this route. The number of states and edges in $\mathcal{B}''$ are at most $2^{O(k)}$ times the number of global states and edges of $\mathcal{A}$. Thus the number of states in $\mathcal{B}''$ is at most

$$2^{O(k)} \cdot |Q|,$$

and the number of edges is at most

$$2^{O(k)} \cdot |E|,$$

where $|Q| = \prod_{i=1}^{k} |Q_i|$ and $|E| = \prod_{i=1}^{k} |E_i|$, with Q_i, E_i being the state and edge set of the i-th component of $\mathcal{A}$. Further, the constants used in the clock constraints are the same as in $\mathcal{A}$, and the number of clocks is $k + 1$.

Thus, using the time bound obtained in § 2, the emptiness check for $\mathcal{A}$ takes time

$$|\Sigma| \cdot O(|Q| + |E|) \cdot 2^{O(k)} \cdot k! \cdot \prod_{i=1}^{k} (c_i + 1).$$

(The factor of $|\Sigma|$ comes in as we need to examine the distribution of Σ while generating the global transition relation of $\mathcal{A}$.) A loose upper bound for the above expression can be seen to be $|\Sigma| \cdot 2^{O(|\mathcal{A}|^2)}$.

4.2 *Product interval languages*

We now give a characterisation of languages accepted by product interval automata in terms of a timed version of the parallel composition operator $\otimes$. For convenience, we will continue to use the same symbol $\otimes$ to denote the timed version also. This characterisation plays an important role in the subsequent sections.

Let $L_i \subseteq T\Sigma_i^\infty$ for each $i \in \mathcal{P}$. Then the *direct product* of $L_1, \ldots, L_k$, written $\otimes(L_1, \ldots, L_k)$, is defined as:

$$\otimes(L_1, \ldots, L_k) = \{\sigma \in T\widetilde{\Sigma}^\infty \mid \sigma\restriction i \in L_i \text{ for each } i \in \mathcal{P}\}.$$

$L \subseteq T\widetilde{\Sigma}^\infty$ is a *regular direct product interval language* over $\widetilde{\Sigma}$ if $L = \otimes(L_1, \ldots, L_k)$ for some regular interval languages L_i over Σ_i. Finally, we say $L \subseteq T\widetilde{\Sigma}^\infty$ is a *regular product interval language* over $\widetilde{\Sigma}$ if L is the finite union of regular direct product interval languages over $\widetilde{\Sigma}$.

The main result of this section is:

Theorem 4. *Let $L \subseteq T\widetilde{\Sigma}^\infty$. Then L is a regular product interval language over $\widetilde{\Sigma}$ iff L is accepted by a product interval automaton over $\widetilde{\Sigma}$.*

We first prove a couple of intermediate results.

Lemma 4. *Let $\mathcal{A} = (\{\mathcal{A}_i\}_{i\in\mathcal{P}}, Q_{in})$ be a product interval automaton over $\widetilde{\Sigma}$. Let $Q_{in} = \{q_1, \ldots, q_m\}$ for some $m \geq 1$. For $j \in \{1, \ldots, m\}$ let $\mathcal{A}^j$ denote the product interval automaton $(\{\mathcal{A}_i\}_{i\in\mathcal{P}}, \{q_j\})$. Then*

$$L(\mathcal{A}) = \bigcup_{j\in\{1,\ldots,m\}} L(\mathcal{A}^j).$$

Proof. Follows easily from the definition of the language accepted by a product interval automaton. □

Lemma 5. *For each $i \in \mathcal{P}$ let $\mathcal{A}_i = (Q_i, \longrightarrow_i, Q_{in}^i, F_i, G_i)$ be an interval automaton over Σ_i. Let $\mathcal{A}$ be the product interval automaton*

$$(\{\mathcal{A}_i\}_{i\in\mathcal{P}}, Q_{in}^1 \times \cdots \times Q_{in}^k).$$

Then

$$L(\mathcal{A}) = \otimes(L(\mathcal{A}_1), \ldots, L(\mathcal{A}_k)).$$

Proof. We first show that $L(\mathcal{A}) \subseteq \otimes(L(\mathcal{A}_1), \ldots, L(\mathcal{A}_k))$. Let $\sigma \in L(\mathcal{A})$. Then we know that there exists an accepting run ρ of $\mathcal{A}$ on σ. We can use ρ to define, for each $i \in \mathcal{P}$, a run ρ_i of the interval automaton $\mathcal{A}_i$ on $\sigma\restriction i$. Further, each ρ_i is an accepting run of $\mathcal{A}_i$ on $\sigma\restriction i$. This is again easy to verify given that ρ is accepting. Thus $\sigma\restriction i \in L(\mathcal{A}_i)$ for each i, and hence $\sigma \in \otimes(L(\mathcal{A}_1), \ldots, L(\mathcal{A}_k))$.

Conversely, suppose $\sigma \in \otimes(L(\mathcal{A}_1), \ldots, L(\mathcal{A}_k))$. So there exist accepting runs ρ_i of $\mathcal{A}_i$ on $\sigma\restriction i$ for each $i \in \mathcal{P}$. Using these ρ_i's we can piece together a run ρ of $\mathcal{A}$ on σ, given by: $\rho(\tau)[i] = \rho_i(\tau\restriction i)$. Again, it is routine to check that ρ is indeed a run of $\mathcal{A}$ on σ, and is accepting. □

Returning to the proof of theorem 4, suppose now that L is a regular product interval language over $\widetilde{\Sigma}$. Then there exist $L_i^j \subseteq T\Sigma_i^\infty$ for $i \in \mathcal{P}$ and $j \in \{1, \ldots, m\}$ ($m \geq 1$) such that each L_i^j is a regular interval language over Σ_i and

$$L = \bigcup_{j \in \{1,\ldots,m\}} \otimes(L_1^j, \ldots, L_k^j).$$

Since each L_i^j is a regular interval language over Σ_i, there exist interval automata $\mathcal{A}_i^j$ over Σ_i such that $L(\mathcal{A}_i^j) = L_i^j$. Let each $\mathcal{A}_i^j$ be of the form $(Q_i^j, \longrightarrow_i^j, (Q_{in})_i^j, F_i^j, G_i^j)$. Now let $\mathcal{B}_i$ denote the disjoint union of the automata $\mathcal{A}_i^1, \ldots, \mathcal{A}_i^m$ (viewed as labelled graphs). Let us use $(\mathcal{B}_i, (Q_{in})_i^j)$ to denote the interval automaton with the underlying structure of $\mathcal{B}_i$ and $(Q_{in})_i^j$ as the set of initial states. Then it is easy to see that $L(\mathcal{B}_i, (Q_{in})_i^j) = L(\mathcal{A}_i^j)$. Thus we have

$$\begin{aligned} L &= \textstyle\bigcup_{j \in \{1,\ldots,m\}} \otimes(L_1^j, \ldots, L_k^j) \\ &= \textstyle\bigcup_{j \in \{1,\ldots,m\}} \otimes(L(\mathcal{B}_1, (Q_{in})_1^j), \ldots, L(\mathcal{B}_k, (Q_{in})_k^j)) \\ &= \textstyle\bigcup_{j \in \{1,\ldots,m\}} L(\{\mathcal{B}_i\}_{i \in \mathcal{P}}, (Q_{in})_1^j \times \cdots \times (Q_{in})_k^j) \text{ (using lemma 5)} \\ &= L(\{\mathcal{B}_i\}_{i \in \mathcal{P}}, Q_{in})) \end{aligned}$$

where $Q_{in} = \bigcup_{j \in \{1,\ldots,m\}} (Q_{in})_1^j \times \cdots \times (Q_{in})_k^j$. This last step follows from lemma 4. Thus L is accepted by a product interval automaton.

The converse direction follows in a similar manner. □

The fact that regular product interval languages are closed under union follows directly from the definition of regular product interval languages. To show closure under intersection and complementation, it is sufficient to show that these operations on regular *direct* product interval languages do not take us out of the class of regular product interval languages.

Let $L = \otimes(L_1, \ldots, L_k)$ and $M = \otimes(M_1, \ldots, M_k)$ with each L_i and M_i regular interval languages over Σ_i, respectively. Then it is easy to verify that $L \cap M = \otimes((L_1 \cap M_1), \ldots, (L_k \cap M_k))$. This is once again a regular direct product interval language since regular interval languages are closed under intersection.

To show that $\overline{L} = T\widetilde{\Sigma}^\infty - L$ is a regular product interval language, note that we can write $\overline{L}$ as

$$\overline{L} = \bigcup_{j \in \mathcal{P}} \otimes(W_1^j, \ldots, W_k^j),$$

where for each $i, j \in \mathcal{P}$,

$$W_i^j = \begin{cases} T\Sigma_i^\infty - L_i, & \text{if } i = j, \\ T\Sigma_i^\infty, & \text{otherwise.} \end{cases}$$

Since each W_i^j is a regular interval language, $\overline{L}$ is a regular product interval language.

Thus, we have

Theorem 5. *The class of regular product interval languages over $\widetilde{\Sigma}$ is closed under the boolean operations of union, intersection, and complementation.* □

Our aim now is to formulate a theory of PI automata which mirrors the classical setting of LTL and LTL$^\otimes$. This will help in setting up a verification theory/methodology in our setting.

5. A logical characterisation

The logic TMSO$^\otimes(\widetilde{\Sigma})$ captures the class of regular product interval languages over $\widetilde{\Sigma}$.

The formulas in this logic comprise boolean combinations of TMSO assertions (cf. § 3) about individual components. They are interpreted over timed words over $\widetilde{\Sigma}$. An assertion about the actions of component i is interpreted as a TMSO(Σ_i) sentence over the projection of the word to Σ_i.

The formulas of TMSO$^\otimes(\widetilde{\Sigma})$ are given by the following syntax:

$$\varphi ::= (\alpha)(i) \mid \neg\varphi \mid (\varphi \vee \varphi) \mid (\varphi \wedge \varphi)$$

where for each formula $(\alpha)(i)$ we require α to be a sentence in TMSO(Σ_i). The notation $(\alpha)(i)$ is meant to indicate that the sentence α comes from the logic TMSO(Σ_i). We introduce the operator $\wedge$ as a first class operator in the logic as a matter of convenience.

We note that the formulas in this logic are all sentences – i.e. they have no free variables.

A model for a TMSO$^\otimes(\widetilde{\Sigma})$ sentence is a timed word in $T\widetilde{\Sigma}^\infty$. For a word $\sigma \in T\widetilde{\Sigma}^\infty$ and a sentence $\varphi \in$ TMSO$^\otimes(\widetilde{\Sigma})$, the satisfaction relation $\sigma \models \varphi$ is given inductively as follows.

$$\begin{array}{lll} \sigma \models (\alpha)(i), & \text{iff} & \sigma\restriction i \models \alpha \text{ (as a TMSO}(\Sigma_i)\text{ formula)}, \\ \sigma \models \neg\alpha, & \text{iff} & \sigma \not\models \alpha, \\ \sigma \models (\varphi \vee \varphi'), & \text{iff} & \sigma \models \varphi \text{ or } \sigma \models \varphi', \\ \sigma \models (\varphi \wedge \varphi'), & \text{iff} & \sigma \models \varphi \text{ and } \sigma \models \varphi'. \end{array}$$

For a sentence $\varphi \in$ TMSO$^\otimes(\widetilde{\Sigma})$ we set $L(\varphi) = \{\sigma \in T\widetilde{\Sigma}^\infty \mid \sigma \models \varphi\}$.

Theorem 6. *Let $L \subseteq T\widetilde{\Sigma}^\infty$. Then L is a regular product interval language iff $L = L(\varphi)$ for some sentence $\varphi \in$ TMSO$^\otimes(\widetilde{\Sigma})$.*

To prove this theorem we first observe a straightforward consequence of the semantics of TMSO$^\otimes(\widetilde{\Sigma})$.

PROPOSITION 1

For each i in $\mathcal{P}$, let φ_i be a sentence in TMSO(Σ_i). *Consider the* TMSO$^\otimes(\widetilde{\Sigma})$ *formula* $((\varphi_1)(1) \wedge \cdots \wedge (\varphi_k)(k))$. *Then*

$$L((\varphi_1)(1) \wedge \cdots \wedge (\varphi_k)(k)) = \otimes(L(\varphi_1), \ldots, L(\varphi_k)).$$

□

Now given a regular direct product interval language $L = \otimes(L_1, \ldots, L_k)$ we know from theorem 3 that there exist sentences $\varphi_i \in \mathrm{TMSO}(\Sigma_i)$ such that $L(\varphi_i) = L_i$. Using proposition 1, we have $L((\varphi_1)(1) \wedge \cdots \wedge (\varphi_k)(k)) = L$. Thus regular direct product interval languages can be captured in our logic. Regular product interval languages are finite unions of regular direct product interval languages, and hence can be captured using the $\vee$ operator in our logic.

Conversely, given a sentence φ in $\mathrm{TMSO}^{\otimes}(\widetilde{\Sigma})$, we can write φ in disjunctive normal form by first driving in negation symbols (note that $\neg((\alpha)(i)) \equiv (\neg\alpha)(i)$) and then distributing $\wedge$ over $\vee$:

$$\varphi \equiv \bigvee_{j=1}^{m} (\bigwedge_{i=1}^{l_j} \gamma_i^j).$$

Here each γ_i^j is of the form $(\alpha)(p)$, with $\alpha \in \mathrm{TMSO}(\Sigma_p)$ for some $p \in \mathcal{P}$. Further, for $j = 1, \ldots, m$ and $p = 1, \ldots, k$, let X_p^j be the set of $\mathrm{TMSO}(\Sigma_p)$ sentences α such that $\gamma_i^j = (\alpha)(p)$ for some i. Let β_p^j be the conjunction of formulas in X_p^j, with the convention that $\bigwedge \emptyset = \top$. It is easy to verify that

$$\varphi \equiv \bigvee_{j=1}^{m} (\bigwedge_{p=1}^{k} (\beta_p^j)(p)).$$

From the semantics of $\vee$ it follows that

$$L(\varphi) = \bigcup_{j=1}^{m} (L(\bigwedge_{p=1}^{k} (\beta_p^j)(p))).$$

Once again, using proposition 1 we have:

$$L(\varphi) = \bigcup_{j=1}^{m} (\otimes(L(\beta_1^j), \cdots, L(\beta_k^j))).$$

Since for each $p \in \mathcal{P}$, β_p^j is a $\mathrm{TMSO}(\Sigma_p)$ sentence, we know that $L(\beta_p^j)$ is a regular interval language over Σ_p. Thus, it follows that $L(\varphi)$ is a regular product interval language. □

6. Timed product-LTL

In this section we formulate a timed version of LTL called timed product-LTL and denoted $\mathrm{TLTL}^{\otimes}$. An important motivation for studying this logic is that it is expressively complete (cf. § 7), being expressively equivalent to the first-order fragment of the logic $\mathrm{TMSO}^{\otimes}$.

$\mathrm{TLTL}^{\otimes}$ formulas comprise boolean combinations of assertions over individual components in a timed logic called TLTL. It will be convenient to first study the satisfiability problem for this logic. We will make use of this later to solve the satisfiability and model checking problems for the logic $\mathrm{TLTL}^{\otimes}$.

6.1 *TLTL over a single component*

Let Σ be an alphabet of actions. Then the formulas of $\mathrm{TLTL}(\Sigma)$ (parameterised by the alphabet Σ) are given by:

$$\varphi ::= \top \mid \neg\varphi \mid (\varphi \vee \varphi) \mid \langle a, I\rangle\varphi \mid O\varphi \mid (\varphi U \varphi).$$

Here we require $a \in \Sigma$ and $I \in \mathcal{I}\mathbb{R}$.

The formulas of TLTL(Σ) are interpreted over timed words over Σ. In what follows, $\sigma \in T\Sigma^\infty$, and $\tau \in T\Sigma^*$ with $\tau \in prf(\sigma)$.

$$
\begin{array}{lll}
\sigma, \tau \models \top, & & \\
\sigma, \tau \models \neg\alpha, & \text{iff} & \sigma, \tau \not\models \alpha, \\
\sigma, \tau \models \alpha \vee \beta, & \text{iff} & \sigma, \tau \models \alpha \text{ or } \sigma, \tau \models \beta, \\
\sigma, \tau \models \langle a, I\rangle\alpha, & \text{iff} & \exists t : \tau(a,t) \preceq \sigma \text{ and } t - time(\tau) \in I, \text{ and } \sigma, \tau(a,t) \models \alpha, \\
\sigma, \tau \models O\alpha, & \text{iff} & \exists \tau(a,t) \preceq \sigma \text{ with } \sigma, \tau(a,t) \models \alpha, \\
\sigma, \tau \models \alpha U \beta\,, & \text{iff} & \exists \delta \in prf(\sigma) \text{ with } \tau \preceq \delta, \text{ such that } \sigma, \delta \models \beta, \text{ and} \\
 & & \quad \forall \gamma : \tau \preceq \gamma \prec \delta,\ \sigma, \gamma \models \alpha.
\end{array}
$$

We say $\sigma \models \varphi$ iff $\sigma, \epsilon \models \varphi$. Define $L(\varphi) = \{\sigma \in T\Sigma^\infty \mid \sigma \models \varphi\}$.

We mention here some of the standard abbreviations used in temporal logic. The formula $\Diamond\alpha$ (read as "future α" or "eventually α") is defined as $\Diamond\alpha = \top U \alpha$. $\Box\alpha$ ("globally α") is defined as $\Box\alpha = \neg\Diamond\neg\alpha$.

In the next theorem we give a construction of an interval automaton which accepts the set of models of a given TLTL(Σ) formula. The theorem is phrased so as to facilitate its use in solving the satisfiability problem for TLTL$^\otimes$. The construction follows the classical construction of Vardi *et al* (1983).

We note here that a simpler route to follow would be to translate a given formula of TLTL(Σ) into an equivalent formula which mentions only intervals taken from a *proper* interval set. We can then use the classical construction to associate an appropriate interval automaton with the given formula. However, this method could lead to an exponential blow-up in the size of the translated formula. This blow-up is avoided in the method we adopt below.

Theorem 7. *Let X be a non-empty set of formulas of* TLTL(Σ). *Then we can construct a structure $\mathcal{A}_X = (Q, \longrightarrow, F, G)$ (an interval automaton without start states) such that for each non-empty subset Y of X there exists $(Q_{in})^X_Y \subseteq Q$ such that*

(1) *$L(\bigwedge Y) = L(\mathcal{A}_X, (Q_{in})^X_Y)$, where by $(\mathcal{A}_X, (Q_{in})^X_Y)$ we mean the interval automaton $(Q, \longrightarrow, (Q_{in})^X_Y, F, G)$. Thus, by suitably choosing start states for $\mathcal{A}_X$ we can accept exactly the models of the conjunction of a subset of formulas in X.*
(2) *The number of states in $\mathcal{A}_X$ is at most $2^{O(\sum_{\varphi\in X}|\varphi|)}$.*
(3) *The largest bound mentioned in $\mathcal{A}_X$ is at most the largest bound mentioned in the formulas in X.*

Proof. Let $\mathcal{I}$ be the set of intervals mentioned in the formulas in X. Let $\mathcal{I}'$ be a proper interval set covering $\mathcal{I}$. Using a method similar to the one outlined in § 3 we can construct $\mathcal{I}'$ such that the size of $\mathcal{I}'$ is at most $2 \cdot |\mathcal{I}|$, and the largest integer constant mentioned in $\mathcal{I}'$ is the largest integer constant mentioned in $\mathcal{I}$.

For a TLTL formula φ, let $sfc(\varphi)$ denote the subformula closure of φ. For a set of formulas X we write $sfc(X)$ to denote the set $\bigcup_{\alpha\in X} sfc(\alpha)$.

Define $CL(X)$, the Fisher–Ladner closure of a set of formulas X, to be

$$CL(X) = S \cup \{\neg\alpha \mid \alpha \in S\}$$

where $S = sfc(X) \cup \{O(\alpha U \beta) \mid \alpha U \beta \in sfc(X)\}$.

It is easy to verify that $|CL(X)|$ is linear in the size of $\sum_{\alpha\in X}|\alpha|$.

An *atom* of X is a maximal "propositionally" consistent subset of $CL(X)$. Formally, a subset A of $CL(X)$ is an atom of X iff,

(1) If $\top \in CL(X)$, then $\top \in A$,
(2) $\forall \alpha \in CL(X)$, $\neg\alpha \in A$, iff $\alpha \notin A$ (here we identify $\neg\neg\alpha$ with α),
(3) $\forall(\alpha \vee \beta) \in CL(X)$, $(\alpha \vee \beta) \in A$, iff $\alpha \in A$ or $\beta \in A$,
(4) $\forall(\alpha U \beta) \in CL(X)$, $(\alpha U \beta) \in A$, iff $\beta \in A$, or, both α, $O(\alpha U \beta) \in A$.

We can now define the automaton $\mathcal{A}_X$. Take the set of states Q to be the set of atoms of X. The transition relation $\longrightarrow$ is given by the following rule. We have $A \xrightarrow{(a,I')} B$, iff each of the following is satisfied:

(1) $I' \in \mathcal{I}'$,
(2) if $\langle b, I\rangle\alpha \in A$ then,

  (i) $b = a$,
  (ii) $I \cap I' \neq \emptyset$,
  (iii) $\alpha \in B$,

(3) if $\langle a, I\rangle\alpha \in CL(X)$ with $I \cap I' \neq \emptyset$ and $\alpha \in B$, then $\langle a, I\rangle\alpha \in A$,
(4) for all $O\alpha \in CL(X)$, $O\alpha \in A$ iff $\alpha \in B$.

The set of finitary final states F consists of atoms which have no "next-state" formulas – i.e. formulas of the form $\langle a, I\rangle\alpha$ or $O\alpha$.

For the infinitary final states it is convenient to make use of a *generalized* Büchi condition. A generalised Büchi condition is a family $\mathcal{G} = \{G_1, \ldots, G_m\}$ of subsets of Q. A run ρ on a word σ is accepting according to this condition iff for every $i \in \{1, \ldots, m\}$, there exist infinitely many prefixes τ of σ such that $\rho(\tau) \in G_i$. Such a condition can easily be converted to a Büchi condition by including a 0 to k counter in the states.

The generalized Büchi condition here is given by $\mathcal{G} = \{G_1, \ldots, G_m\}$ where $m \geq 0$ is the number of until formulas in $CL(X)$, and the G_i's are given as follows. Let $\{\alpha_1 U \beta_1, \ldots, \alpha_m U \beta_m\}$ be the set of until formulas in $CL(X)$. Then for each $i \in \{1, \ldots, m\}$ we define $G_i = \{A \mid \alpha_i U \beta_i \notin A \text{ or } \beta_i \in A\}$.

Now let Y be a non-empty subset of X, $\varphi = \bigwedge Y$, and $(Q_{in})^X_Y = \{A \mid Y \subseteq A\}$. We will show that $L(\varphi) = L((\mathcal{A}_X, (Q_{in})^X_Y))$.

We first show that $L(\varphi) \subseteq L((\mathcal{A}_X, (Q_{in})^X_Y))$. Let σ be a model for φ. Let $\rho : prf(\sigma) \to Q$ be given by $\rho(\tau) = \{\alpha \in CL(X) \mid \sigma, \tau \models \alpha\}$. It is routine to verify that ρ is an accepting run of $(\mathcal{A}_X, (Q_{in})^X_Y)$ on σ.

Conversely, to argue that $L((\mathcal{A}_X, (Q_{in})^X_Y)) \subseteq L(\varphi)$, we use the following claim.

Claim 1. Let $\sigma \in T\Sigma^\infty$ and let ρ be an accepting run of $(\mathcal{A}_X, (Q_{in})^X_Y)$ on σ. Let $\alpha \in CL(X)$. Then for each $\tau \in prf(\sigma)$ we have $\sigma, \tau \models \alpha$ iff $\alpha \in \rho(\tau)$.

This claim can be proved in the standard way by induction on the structure of α. □

Now given a timed word σ in $L(\mathcal{A}_X, (Q_{in})^X_Y)$, we know there is an accepting run ρ of the automaton on σ. By definition of the set of initial states, $\alpha \in \rho(\epsilon)$ for each $\alpha \in Y$. By the above claim, we have that $\sigma, \epsilon \models \alpha$ for each $\alpha \in Y$. Since $\varphi = \bigwedge Y$, it follows that $\sigma, \epsilon \models \varphi$ and $\sigma \in L(\varphi)$. □

6.2 *Product-TLTL*

We now give the syntax and semantics of TLTL$^\otimes$. Let $\widetilde{\Sigma} = \{\Sigma_i\}$ be a distributed alphabet. The syntax of the logic TLTL$^\otimes(\widetilde{\Sigma})$ is given by:

$$\varphi ::= (\varphi)(i) \mid \neg\varphi \mid (\varphi \wedge \varphi') \mid (\varphi \vee \varphi'),$$

where, as in § 5, we require of $(\varphi)(i)$ that $i \in \mathcal{P}$ and $\varphi \in \mathrm{TLTL}(\Sigma_i)$. Thus $(\varphi)(i)$ is an arbitrary i-type formula φ, tagged with the index i. Once again we introduce $\wedge$ as a first class operator in the logic for convenience in working out proofs. The expressiveness of the logic is unaffected even if we drop the $\wedge$ operator from the syntax.

Models for $\mathrm{TLTL}^{\otimes}(\widetilde{\Sigma})$ formulas are timed words in $T\widetilde{\Sigma}^{\infty}$. Let $\varphi \in \mathrm{TLTL}^{\otimes}(\widetilde{\Sigma})$ and let $\sigma \in T\widetilde{\Sigma}^{\infty}$. The satisfaction relation $\sigma \models \varphi$ is defined inductively as follows:

$$\begin{array}{lll}
\sigma \models (\varphi)(i), & \text{iff} & \sigma\restriction i \models \varphi \text{ (as a TLTL}(\Sigma_i) \text{ formula)},\\
\sigma \models \neg\varphi, & \text{iff} & \sigma \not\models \varphi,\\
\sigma \models (\varphi \wedge \varphi'), & \text{iff} & \sigma \models \varphi \text{ and } \sigma \models \varphi',\\
\sigma \models (\varphi \vee \varphi'), & \text{iff} & \sigma \models \varphi \text{ or } \sigma \models \varphi'.
\end{array}$$

Once again, we let $L(\varphi)$ denote the set $\{\sigma \in T\widetilde{\Sigma}^{\infty} \mid \sigma \models \varphi\}$.

Example 4. The following formula over the distributed alphabet $\widetilde{\Sigma} = (\{a, b\}, \{b\})$ describes the language L of example 3.

$$(\langle b, (0,\infty)\rangle\langle a, (0,\infty)\rangle\langle b, (0,\infty)\rangle\top)(1) \ \wedge \ (\langle b, (0,\infty)\rangle\langle b, (1,\infty)\rangle\top)(2).$$

From the semantics of $\mathrm{TLTL}^{\otimes}(\widetilde{\Sigma})$ the following is immediate.

PROPOSITION 2

Let $\varphi_1, \ldots, \varphi_k$ *be formulas in* $\mathrm{TLTL}(\Sigma_1), \ldots, \mathrm{TLTL}(\Sigma_k)$ *respectively. Then,*

$$L((\varphi_1)(1) \wedge \cdots \wedge (\varphi_k)(k)) = \otimes(L(\varphi_1), \ldots, L(\varphi_k)).$$

□

We now show how we can associate with a given $\mathrm{TLTL}^{\otimes}(\widetilde{\Sigma})$ formula a product interval automaton which recognises exactly the models of the formula. Let $\varphi \in \mathrm{TLTL}^{\otimes}(\widetilde{\Sigma})$. As done for $\mathrm{TMSO}^{\otimes}(\widetilde{\Sigma})$ in theorem 6, we can write φ as

$$\varphi \equiv \bigvee_{j=1}^{m} \Big(\bigwedge_{i=1}^{l_j} \alpha_i^j\Big).$$

with each α_i^j being of the form $(\gamma)(p)$ for some $p \in \mathcal{P}$ and $\gamma \in \mathrm{TLTL}(\Sigma_p)$. Further, for $j = 1, \ldots, m$ and $p = 1, \ldots, k$, let X_p^j be the set of p-type formulas γ such that $\alpha_i^j = (\gamma)(p)$ for some i. Let β_p^j be the conjunction of formulas in X_p^j, with the convention that $\bigwedge \emptyset = \top$. Proceeding as in theorem 6 and making use of proposition 2, it follows that,

$$L(\varphi) = \bigcup_{j=1}^{m} (\otimes(L(\beta_1^j), \cdots, L(\beta_k^j))).$$

Now using theorem 7 we can construct for each $p = 1, \ldots, k$ an interval automaton $\mathcal{A}_{X_p}$ (where $X_p = \bigcup_{j=1}^{m} X_p^j$), such that for each $j = 1, \ldots, m$ the interval automaton $(\mathcal{A}_{X_p}, (Q_{in})_{X_p^j}^{X_p})$ accepts $L(\beta_p^j)$. Using lemma 5 it follows that

$$L(\varphi) = \bigcup_{j=1}^{m} L((\{\mathcal{A}_{X_p}\}_{p=1}^{k}, ((Q_{in})_{X_1^j}^{X_1} \times \cdots \times (Q_{in})_{X_k^j}^{X_k}))).$$

Now note that the product interval automata

$$(\{\mathcal{A}_{X_p}\}_{p=1}^{k}, ((Q_{in})_{X_1^j}^{X_1} \times \cdots \times (Q_{in})_{X_k^j}^{X_k})) \text{ for } j = 1, \ldots, m$$

are the same except for the start states. It then follows easily that

$$L(\varphi) = L(\mathcal{A}_\varphi),$$

where

$$\mathcal{A}_\varphi = (\{\mathcal{A}_{X_p}\}_{p=1}^{k}, \bigcup_{j=1}^{m}((Q_{in})_{X_1^j}^{X_1} \times \cdots \times (Q_{in})_{X_k^j}^{X_k})).$$

Thus the satisfiability problem for TLTL$^{\otimes}$ can be solved as follows. Given a formula $\varphi \in \text{TLTL}^{\otimes}(\widetilde{\Sigma})$ we can generate the product interval automaton $\mathcal{A}_\varphi$ and then check the emptiness of $\mathcal{A}_\varphi$ as outlined in § 4.1.

To analyse the time complexity of checking satisfiability of φ, note that each component $\mathcal{A}_{X_p}$ of $\mathcal{A}_\varphi$ can be generated in time $2^{O(\sum_{\alpha \in X_p} |\alpha|)}$ using theorem 7. Further, it is not difficult to see that $\prod_{p=1}^{k} 2^{O(\sum_{\alpha \in X_p} |\alpha|)} = 2^{O(|\varphi|)}$. Using these values in the time bounds obtained in § 4.1 we see that the satisfiability of φ can be decided in time

$$|\Sigma| \cdot 2^{O(|\varphi|)} \cdot 2^{O(k)} \cdot k!.$$

As regards the space complexity of satisfiability we have the following result.

Theorem 8. *Given $\widetilde{\Sigma}$ and a formula $\varphi \in \text{TLTL}^{\otimes}(\widetilde{\Sigma})$, the problem of checking whether φ is satisfiable is* PSPACE*-complete.*

Proof. The satisfiability problem for LTL is known to be PSPACE-complete (Sistla & Clarke 1985). PSPACE-hardness for TLTL$^{\otimes}$ follows easily by reducing the satisfiability problem for LTL to the one-component case of TLTL$^{\otimes}$.

To show that the satisfiability check can be done in PSPACE, we argue equivalently that it can be done non-deterministic PSPACE. Though the number of states in the region graph $R(\mathcal{A}_\varphi)$ is exponential in $|\varphi|$ and k, it is an implicitly defined graph whose adjacency relation can be checked in space polynomial in $|\varphi| + |\widetilde{\Sigma}|$ (see Alur & Dill 1994). Further, the emptiness check boils down to a reachability check on the region graph, which can be done non-deterministically in space polynomial in $|\varphi|$ and $|\widetilde{\Sigma}|$. □

Next suppose we consider a real-time program Pr modelled by a product interval automaton $\mathcal{A}_{Pr}$, and a formula φ of TLTL$^{\otimes}(\widetilde{\Sigma})$. Then Pr is said to meet the specification φ iff $L(\mathcal{A}_{Pr}) \subseteq L(\varphi)$. The model checking problem for TLTL$^{\otimes}(\widetilde{\Sigma})$ is to determine whether Pr meets the specification φ.

Theorem 9. *The model checking problem for* TLTL$^{\otimes}(\widetilde{\Sigma})$ *is* PSPACE*-complete.*

Proof. PSPACE-hardness follows from the fact that the satisfiability problem for TLTL$^{\otimes}$ can be reduced to the model checking problem. This is because the question of whether φ is satisfiable can be reduced to whether $\mathcal{A}_{univ} \not\models \neg\varphi$, where $\mathcal{A}_{univ}$ is a product interval automaton which recognises the language $T\widetilde{\Sigma}^{\infty}$.

To see that the problem can be solved in PSPACE, we must check the emptiness of the intersection of $\mathcal{A}_{Pr}$ and $\mathcal{A}_{\neg\varphi}$ in space polynomial in $|\mathcal{A}_{Pr}| + |\varphi|$. This is a similar argument to the one we have sketched for theorem 8 above. □

7. Expressive completeness of TLTL$^{\otimes}$

The aim of this section is to show that TLTL$^{\otimes}(\widetilde{\Sigma})$ is expressively equivalent to the first-order fragment of TMSO$^{\otimes}(\widetilde{\Sigma})$. This is a standard way to measure the expressive power of a logic, and the result we obtain here is along the lines of classical results concerning LTL (Kamp 1980; Gabbay *et al* 1980; Henriksen & Thiagarajan 1997).

Let TFO(A) denote the first-order fragment of the logic TMSO(A). TFO(A) is obtained from TMSO(A) by disallowing the use of quantification over set variables. The first-order fragment of TMSO$^{\otimes}(\widetilde{\Sigma})$, denoted TFO$^{\otimes}(\widetilde{\Sigma})$ is obtained by taking boolean combinations of the first-order fragment of TMSO(Σ_i) for each $i \in \mathcal{P}$. Thus the syntax of TFO$^{\otimes}(\widetilde{\Sigma})$ is given by

$$\varphi ::= (\alpha)(i) \mid \neg\varphi \mid (\varphi \vee \varphi) \mid (\varphi \wedge \varphi)$$

where in each formula $(\alpha)(i)$, α is a sentence in TFO(Σ_i).

We will first establish the result that TLTL(Σ) corresponds to TFO(Σ).

Theorem 10. *For any alphabet* Σ, TLTL(Σ) *is expressively equivalent to* TFO(Σ).

The method of proof will be to translate TLTL formulas into classical LTL over an appropriate interval alphabet. The method is similar to the proof of theorem 3 and we also make use of the translation used there.

It is useful to first recall the result concerning the expressive completeness of LTL. Let A be an alphabet of actions. Let FO(A) denote the first-order fragment of the logic MSO(A). As before, FO(A) is obtained from the logic MSO(A) defined in § 3, by disallowing the use of set variables. Then a well known result due to the work of Kamp (1968), and Gabbay *et al* (1980) is:

Theorem 11. LTL(A) *is expressively equivalent to* FO(A).

Looking back at the syntax of TLTL(Σ) formulas, we see that they are simply LTL(Γ) formulas for some interval alphabet Γ based on Σ. Of course, we must bear in mind that TLTL(Σ) formulas are interpreted over *timed* words over Σ. Thus, a formula $\varphi \in$ LTL(Γ) defines a language $L_{sym}(\varphi) \subseteq \Gamma^{\infty}$ when interpreted as an LTL(Γ) formula, and it defines a timed language $L(\varphi) \subseteq T\Sigma^{\infty}$ when interpreted as an TLTL(Σ) formula.

The following lemma describes the relationship between these two languages.

Lemma 6. *Let* Γ *be a proper interval alphabet based on* Σ. *Let* φ *be a formula in* LTL(Γ). *Then* $L(\varphi) = tw(L_{sym}(\varphi))$.

Proof. The proof of this is very similar to our earlier arguments which make use of the properties of proper interval sets. □

Returning now to the proof of theorem 10, let $\varphi_0 \in$ TLTL(Σ). Then it is not difficult to see that we can construct a proper interval alphabet Γ based on Σ such that Γ_a covers $voc(\varphi_0)$ for each $a \in \Sigma$, and a formula $\varphi_1 \in$ LTL(Γ) such that $L(\varphi_0) = L(\varphi_1)$. From lemma 6, we know that $L(\varphi_1) = tw(L_{sym}(\varphi_1))$. Now, by theorem 11, we know that there exists a sentence φ_2 in FO(Γ) such that $L(\varphi_2) = L_{sym}(\varphi_1)$. Now consider the sentence $\varphi_3 = s\text{-}t(\varphi_2)$ w.r.t. the proper interval alphabet Γ (cf. § 3). The translations *s-t* and *t-s* are such that if the given formula is first-order, then so is the translated formula. Thus φ_3 is a TFO(Σ) sentence. Further, since Γ

is proper, by lemma 3 we know that $L(\varphi_3) = tw(L(\varphi_2))$. Thus φ_3 is the required TFO(Σ) sentence with $L(\varphi_0) = L(\varphi_3)$.

Conversely, let φ_0 be a sentence in TFO(Σ). Then, once again, there exists a proper interval alphabet Γ based on Σ such that Γ_a covers $voc(\varphi_0)$ for each $a \in \Sigma$. Consider the MSO(Γ) sentence $\varphi_1 = t\text{-}s(\varphi_0)$ with respect to the interval alphabet Γ (cf. § 3). By lemma 2, $L(\varphi_0) = tw(L(\varphi_1))$. Further, φ_1 is a sentence in FO(Γ). Now, again appealing to theorem 11, we know that there exists an LTL(Γ) formula φ_2 such that $L_{sym}(\varphi_2) = L(\varphi_1)$. By lemma 6, we know that $L(\varphi_2) = tw(L_{sym}(\varphi_2))$. Thus φ_2 is the required formula in TLTL(Σ) such that $L(\varphi_2) = L(\varphi_0)$. □

Using theorem 10 above, we can now prove:

Theorem 12. TLTL$^{\otimes}(\widetilde{\Sigma})$ *is expressively equivalent to* TFO$^{\otimes}(\widetilde{\Sigma})$.

Proof. Let $\varphi \in$ TLTL$^{\otimes}(\widetilde{\Sigma})$. We define a sentence $l\text{-}m(\varphi)$ in TMSO$^{\otimes}(\widetilde{\Sigma})$ such that $L(\varphi) = L(l\text{-}m(\varphi))$. The sentence $l\text{-}m(\varphi)$ is obtained by replacing each subformula of the form $(\alpha)(i)$ in φ by the sentence $(\alpha')(i)$ where α' is a sentence in TMSO(Σ_i) which is equivalent to the TLTL(Σ_i) formula α. Note that the existence of such an α' is guaranteed by Theorem 10. The fact that $L(\varphi) = L(l\text{-}m(\varphi))$ now follows easily by inductive argument on the structure of φ.

The converse direction is proved in a very similar way. □

8. Modelling asynchronous circuits

The aim of this section is to show that product interval automata are expressive enough to model an important class of timed behaviours, namely that of asynchronous digital circuits. In Maler & Pnueli (1995) model the timing behaviour of circuits using a network of timed automata that communicate via shared variables. Their model is based on the non-deterministic inertial delay model for gates Brzozowski & Seger (1994). With this model as our starting point, we describe the behaviour of a circuit using timed words (in contrast to the *signals* used by Maler & Pnueli 1995). We then show that for a given circuit, we can define a product interval automaton which recognises the language of timed words generated by the circuit.

8.1 *The non-deterministic delay model*

A k-wire circuit (see figure 3) is modelled as a tuple

$$\chi = (X, F, D, b_0),$$

where $X = \{x_1, \ldots, x_k\}$ is a set of wires, $F = \{f_1, \ldots, f_k\}$ is a set of gates modelled as functions from $\{0,1\}^k$ to $\{0,1\}$, and $D = \{(l_1, u_1), \ldots, (l_k, u_k)\}$ is a set of pairs of positive integers (u_i could be ∞ as a special case) with $l_i \leq u_i$ for each i. The pair (l_i, u_i) is meant to model both the *delay* and the *latency* of the gate f_i. Delay and latency are often assumed to be modelled by the same pair of values, and we follow the same assumption here. Roughly speaking, a change in the input signal must hold for at least l_i units of time for it to be reflected in the output signal of the gate, and if a change holds for u_i units of time, it *must* be reflected in the output of the gate. This models the latency of the gate. Further, the amount of delay for the gate to switch must again lie in the interval $[l_i, u_i]$. These notions will be formalised below. The component b_0 is an element of $\{0,1\}^k$, and represents the initial values on the wires of the circuit.

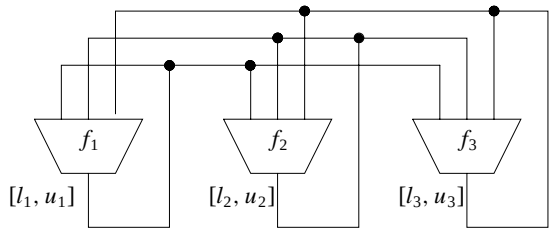

Figure 3. A 3-wire circuit

Some simplifying assumptions have been made in this model. All gates are assumed to be of fan-in k and all wires are fed into a gate as input. Further, inputs to the circuit need to be modelled using the initial state b_0 of the circuit, or as NOT-gates within the circuit, with suitable delay values.

Maler & Pnueli (1995) used infinite, $\{0,1\}^k$-valued "signals" to describe the behaviour of a circuit. In our framework, we use finite and infinite timed words over the alphabet $\Sigma = \{0,1\}^k$. A signal with finitely many discontinuities can be represented as a finite timed word (these are the so called "stable" signals). A signal with infinitely many discontinuities can be represented as an infinite timed word.

To begin with we will need the following notions.

- Let $\tau \in T\Sigma^*$. We define $action(\tau)$ (w.r.t. χ) to be a if $\tau = \tau'(a,t)$ for some τ' and t, and we set $action(\epsilon) = b_0$. This is the analogue of the $time$ function used earlier in the paper.
- Let $s \in \{0,1\}^k$. The "hidden" value vector of s will be denoted by $h(s)$ and is given by $h(s)(i) = f_i(s)$. For each i, $h(s)(i)$ will represent the value computed by gate i, which may or may not be propagated to the output of the gate.
- Let $s \in \{0,1\}^k$. Then $excited(s)$ is a subset of $\{1,\ldots,k\}$ given by $i \in excited(s)$ iff $s(i) \neq h(s)(i)$. We will say that s is i-excited if $i \in excited(s)$, and we will say that s is excited if $excited(s) \neq \emptyset$, Finally, we will say $\tau \in T\Sigma^*$ is excited (i-excited) if $action(\tau)$ is excited (i-excited).
- Let $s, s' \in \{0,1\}^k$. Then we define

$$switches(s,s') = \{i \in \{1,\ldots,k\} \mid s'(i) \neq s(i)\}.$$

  These are the gates which have switched in going from s to s'.
- For $s, s' \in \{0,1\}^k$ we define $qtoe(s,s') \subseteq \{1,\ldots,k\}$ given by $i \in qtoe(s,s')$ iff s' is i-excited and s is not i-excited. $qtoe$ is a mnemonic for "quiescent to excited."
  Similarly, $i \in etoq(s,s')$ iff s is i-excited and s' is not i-excited.
- Let $\tau \in T\Sigma^*$ and i be such that τ is i-excited. Let τ' be the smallest prefix of τ such that

  (1) τ' is i-excited, and,
  (2) For each τ'' such that $\tau' \preceq \tau'' \preceq \tau$ we have that τ'' is i-excited and further that $action(\tau'')(i) = action(\tau)(i)$.

  Then we define $etime_i(\tau) = time(\tau')$. Thus $etime_i(\tau)$ is defined when gate-i is excited in τ, and it is the time at which gate-i last became excited, without having switched in between.

We define when a finite timed word $\tau \in T\Sigma^*$ is *valid* (w.r.t. χ), inductively on the length of τ. The empty word ϵ is valid. A word of the form $\tau(a,t)$ is valid iff the following conditions hold. Let $action(\tau) = b$. Then

(1) τ must be valid.
(2) For each $i \in \{1, \ldots, k\}$, if $i \in switches(b, a)$ then we must have $i \in excited(b)$. Thus, for a gate to switch, it must be in an excited state.
(3) For each $i \in switches(b, a)$ we must have $(t - etime_i(\tau)) \in [l_i, u_i]$.
(4) Suppose $i \in excited(b)$ and $i \notin switches(b, a)$. Then
   (a) if $i \notin excited(a)$ (i.e. gate i becomes quiescent) then we must have $(t - etime_i(\tau)) < u_i$.
   (b) if $i \in excited(a)$ (i.e. gate i remains excited) then also we must have $(t - etime_i(\tau)) < u_i$.

We say an infinite timed word in $T\Sigma^\omega$ is valid, iff every finite prefix of it is valid.

We now define $L(\chi)$, the behaviour of the circuit χ. For a finite word τ in $T\Sigma^*$ we have $\tau \in L(\chi)$ iff τ is valid w.r.t. χ and τ is not excited. For an infinite word $\sigma \in T\Sigma^\omega$, we have $\sigma \in L(\chi)$ iff it is valid w.r.t. to χ.

8.2 *Modelling a circuit as a PIA*

We now show how we can model the circuit χ as a product interval automaton $\mathcal{A}_\chi$ over the distributed alphabet $\widetilde{\Sigma}'$ with $\Sigma' \subseteq \Sigma \times \Sigma$. This automaton has the property that $L(\mathcal{A}_\chi) = L(\chi)$, modulo a (geometric) projection applied to the alphabet. Thus we have $L(\chi) = \varsigma(L(\mathcal{A}_\chi))$ where ς is a substitution which maps actions of the form (s, s') to s'.

We define our distributed alphabet $\widetilde{\Sigma}'$ as follows. The set of actions Σ' is the set of all $(s, s') \in \Sigma \times \Sigma$ which satisfy:

(1) $switches(s, s') \neq \emptyset$,
(2) $i \in switches(s, s')$ implies $i \in excited(s)$.

We have $k + 1$ locations and the distribution of Σ' over the locations is given by:

$$loc((s, s')) = \{k + 1\} \cup switches(s, s') \cup etoq(s, s') \cup qtoe(s, s').$$

(Note that we have no independent actions here as $k + 1 \in loc(a)$ for all $a \in \Sigma'$. Hence $T\widetilde{\Sigma}'^\infty = T\Sigma'^\infty$.)

The components $\mathcal{A}_i$, for $i = 1, \ldots, k$, corresponds to the ith gate. The $(k + 1)$-th component, $\mathcal{A}_{k+1}$, is free of any timing constraints and keeps track of the current vector of output values of the circuit.

We define $\mathcal{A}_\chi = (\{\mathcal{A}_i\}_{i=1}^{k+1}, Q_{in})$ with each $\mathcal{A}_i = (Q_i, \longrightarrow_i, F_i, G_i)$ given as follows:

- For $i \in \{1, \ldots, k\}$, we have:
  - $Q_i = \{e, q\}$,
  - $\longrightarrow_i$ is given by:
    - $e \xrightarrow{((s,s'),[l_i,u_i])}_i e$ provided $i \in switches(s, s')$ and $i \in excited(s')$,
    - $e \xrightarrow{((s,s'),[l_i,u_i])}_i q$ provided $i \in switches(s, s')$ and $i \notin excited(s')$,
    - $e \xrightarrow{((s,s'),(0,u_i))}_i q$ provided $i \notin switches(s, s')$ and $i \in etoq(s, s')$,
    - $q \xrightarrow{((s,s'),(0,\infty))}_i e$ provided $i \in qtoe(s, s')$,
  - $F_i = \{q\}$,

- $G_i = \{e, q\}$,

- For $\mathcal{A}_{k+1}$ we have:
  - $Q_{k+1} = \{0, 1\}^k$,
  - $\longrightarrow_{k+1}$ is given by:

$$s \xrightarrow{((t,t'),I)}_{k+1} s' \text{ iff } s = t,\ s' = t',\ \text{and } I = (0, \infty).$$

  - $F_{k+1} = \{s \in Q_{k+1} \mid excited(s) = \emptyset\}$.
  - $G_{k+1} = Q_{k+1}$.

Finally, the set of initial states Q_{in} is a singleton $\{q_{in}\}$ with $q_{in} = \{(d_1, \ldots, d_k, b_0)\}$ where for each $i \in \{1, \ldots, k\}$ $d_i = e$, if b_0 is i-excited, and q otherwise.

A couple of comments about the choice of components and their structure may be in order here. The reader may ask why the $(k+1)$-th is required at all. The answer is that without this component, each of the gate components would have to have access to the states of the other components, and this would mean that they must take part in *every* action of the circuit. This would destroy the role played by the clocks since they would all be reset with every action. Once the $(k+1)$-th component is there to take care of the valid consecution of actions in the circuit, the gate components need only synchronise with the actions in which they switch, or which affect their excited state. It is for this reason that we need the actions to be pairs which tell us the state from which, and to which, the circuit switches.

Finally, one may wonder why the "excited" and "quiescent" states are needed for each gate component. This is so that we can use the finitary accepting states F_i to reject behaviours in which a gate is excited continuously without ever switching.

It is not difficult to prove the correctness of the construction and the interested reader can find the details in the work by D'Souza (2000a).

8.3 *Properties expressed in TLTL$^{\otimes}$*

We list below some properties of circuits that we can specify in our logic TLTL$^{\otimes}(\widetilde{\Sigma}')$. Using the model-checking algorithm of § 6, one can automatically check whether a given circuit satisfies these properties.

- The formula φ below specifies that every possible behaviour of a circuit is eventually stable.

$$\varphi = (\bigvee_{excited(s')=\emptyset} \Diamond\langle(s, s'), (0, \infty)\rangle\top)\ (k+1).$$

- We can specify that gate i always switches within d time units of becoming excited. This is specified by the formula

$$(\Box(\varphi_1 \Rightarrow \varphi_2))(i),$$

if $i \notin excited(b_0)$, and by the formula

$$(\Box(\varphi_1 \Rightarrow \varphi_2))(i) \wedge (\varphi_3)(i),$$

if $i \in excited(b_0)$, where

$$\begin{aligned}
\varphi_1 &= \textstyle\bigvee_{i \in excited(s')} \langle(s, s'), (0, \infty)\rangle\top, \\
\varphi_2 &= \textstyle\bigvee_{a \in \Sigma'} \langle a, (0, \infty)\rangle(\bigvee_{i \in switches(t,t')} \langle(t, t'), (0, d]\rangle\top)\ (i), \\
\varphi_3 &= \textstyle\bigvee_{i \in switches(s,s')} \langle(s, s'), (0, d]\rangle\top.
\end{aligned}$$

9. Conclusion

We have studied product interval automata, a subclass of timed automata which admit a clean logical theory. Product interval automata are closed under boolean operations and admit a logical characterisation via the monadic second order logic TMSO$^{\otimes}$.

We have also formulated a timed temporal logic called TLTL$^{\otimes}$ to reason about timed behaviours captured by product interval automata. We have solved the satisfiability and model-checking problems by automata-theoretic means while establishing tight space complexity bounds for these problems. This temporal logic turns out to be a natural one in the sense that it is expressively complete; it has exactly the expressive power of the first order fragment of TMSO$^{\otimes}$. These results parallel the results in the classical setting and lay the basis for a similar verification methodology in a timed framework.

We have shown that despite their simple structure, product interval automata are expressive enough to model an important class of timed behaviours, namely that of asynchronous circuits. Coupled with the fact that the simple distributed nature of product interval automata make them amenable to efficient application of partial order methods for timed systems (Minea 1999), we expect that our methods will lead to efficient methods for analysing these circuits.

The expressive power of product interval automata can be increased by considering a timed variant of asynchronous automata (Gastin & Petit 1992), called distributed interval automata. A interesting fact is that the natural timed extension of "cellular" asynchronous automata (Zielonka 1987) (which in the untimed setting are equal in expressive power to asynchronous automata) are more powerful than distributed interval automata. In fact, cellular interval automata are as expressive as the event recording automata of Alur *et al* (1994). These results are detailed elsewhere (D'Souza & Thiagarajan 1999; D'Souza 2000a). Finally, the techniques used here lead naturally to an unrestricted logical characterisation of event recording automata (D'Souza 2000b).

References

Alur R, Dill D L 1994 A theory of timed automata. *Theor. Comput. Sci.* 126: 183–235

Alur R, Henzinger T A 1992 Logics and models of real time: A survey. In *Real-time: Theory in practice* (eds.) J W de Bakker *et al, LNCS 600* (Berlin: Springer-Verlag) pp 74–106

Alur R, Fix L, Henzinger T A 1994 Event-clock automata: a determinizable class of timed automata. *Proc. 6th Int. Conf. on Computer-aided Verification, LNCS 818* (Berlin: Springer-Verlag) pp 1–13

Bengtsson J, Jonsson B, Lilius J, Yi W 1998 Partial order reductions for timed systems. *Proc. CONCUR '98, LNCS 1466* (Berlin: Springer–Verlag)

Brzozowski J A, Seger C-J H 1994 *Asynchronous circuits* (Berlin: Springer-Verlag)

Büchi J R 1960 Weak second-order arithmetic and finite automata. *Z. Math. Logik Grundlagen Math.* 6: 66–92

Diekert V, Rozenberg G 1995 *The book of traces* (Singapore: World Scientific)

D'Souza D 2000a *A logical study of distributed timed automata*. Ph D thesis, Chennai Mathematical Institute (available at http://www.cmi.ac.in/~deepak)

D'Souza D 2000b A logical characterisation of event recording automata. *Proc. 6th Intl. Symp. on Formal Techniques in Real-Time and Fault Tolerant Systems (FTRTFT) LNCS 1926* (Berlin: Springer-Verlag)

D'Souza D, Thiagarajan P S 1998 Distributed interval automata. Internal Report TCS-98-3, Chennai Mathematical Institute, Chennai

D'Souza D, Thiagarajan P S 1999 Product interval automata: A subclass of timed automata. *Proc. 19th Foundations of Software Technology and Theoretical Computer Science (FSTTCS), LNCS 1732* (Berlin: Springer-Verlag)

Gabbay D, Pnueli A, Shelah S, Stavi J 1980 The temporal analysis of fairness. *Seventh ACM Symposium on Principles of Programming Languages*, pp 163–173

Gastin P, Petit A 1992 Asynchronous cellular automaton for infinite traces. *Proceedings of ICALP '92, LNCS 623* (Berlin: Springer-Verlag) pp 583–594

Henriksen J G, Thiagarajan P S 1997 A product version of dynamic linear time temporal logic. *Proc. CONCUR '97, LNCS 1243* (Berlin: Springer-Verlag) pp 45–58

Henzinger T A 1998 It's about time: Real-time logics reviewed. *Proc. CONCUR '98, LNCS 1466* (Berlin: Springer-Verlag) pp 366–372

Henzinger T A, Kopke P W, Puri A, Varaiya P 1995 What's decidable about hybrid automata? *Proc. 27th Annual Symposium on Theory of Computing* (ACM Press) pp 373–382

Henzinger T A, Raskin J-F, Schobbens P-Y 1998 The regular real-time languages. *Proc. 25th Int. Colloquium on Automata, Languages, and Programming 1998, LNCS 1443* (Berlin: Springer-Verlag) pp 580–591

Kamp H 1968 *Tense logic and the theory of linear order*. Ph D thesis, University of California

Maler O, Pnueli A 1995 Timing analysis of asynchronous circuits using timed automata. In *Proc. CHARME '95, LNCS 987* (Berlin: Springer-Verlag) pp 189–205

Merlin P, Faber D J 1976 Recoverability of communication protocols. *IEEE Trans. Commun.* 24: 431–446

Minea M 1999 Partial order reduction for model checking of timed automata. In *Proc. CONCUR '99, LNCS 1664* (Berlin: Springer Verlag)

Pnueli A 1977 The temporal logic of programs. *Proc. 18th IEEE Symp. on Foundations of Computer Science* (IEEE Comput. Soc. Press) pp 46–57

Raskin J-F, Schobbens P-Y 1997 State-clock logic: A decidable real-time logic. *Proc. HART '97: Hybrid and Real-Time Systems, LNCS 1201* (Berlin: Springer-Verlag) pp 33–47

Sistla A P, Clarke E M 1985 The complexity of propositional linear temporal logic. *J. Assoc. Comput. Mach.* 32: 733–749

Thiagarajan P S 1995 A trace consistent subset of PTL. *Proc. CONCUR '95, LNCS 962* (Berlin: Springer-Verlag) pp 438–452

Thomas W 1990 Automata on infinite objects. In *Handbook of theoretical computer science* (ed.) J V Leeuwen (Elsevier) vol. B, pp 133–191

Vardi M Y, Wolper P, Sistla A P 1983 Reasoning about infinite computation paths. *Proc. 24th IEEE Symposium on Foundations of Computer Science* (IEEE Comput. Soc. Press) pp 185–194

Wilke Th 1994 Specifying timed state sequences in powerful decidable logics and timed automata. In *Formal techniques in real-time and fault-tolerant systems, LNCS 863* (Berlin: Springer-Verlag) pp 694–715

Yi W, Jonsson B 1994 Decidability of timed language-inclusion for networks of real-time communicating sequential processes. *Proc. Foundations of Software Technology and Theoretical Computer Science 94, LNCS 880* (Berlin: Springer-Verlag)

Zielonka W 1987 Notes on finite asynchronous automata. *RAIRO – Inf. Theor. Appl.* 21: 99–135