Product Interval Automata: A Subclass of Timed Automata

Deepak D'Souza* and P. S. Thiagarajan*†

Chennai Mathematical Institute, 92 G. N. Chetty Road, Chennai 600 017, India.
{deepak,pst}@smi.ernet.in

Abstract. We identify a subclass of timed automata and develop its theory. These automata, called product interval automata, consist of a network of timed agents. The key restriction is 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. It turns out that the study of these timed automata can exploit the rich theory of partial orders known as Mazurkiewicz traces. An important consequence is that the partial order reduction techniques being developed for timed automata [4, 10] can be readily applied to the verification tasks associated with our automata. Indeed we expect this to be the case even for the extension of product interval automata called distributed interval automata.

1 Introduction

Timed automata as formulated by Alur and Dill [1] have become a standard model for describing timed behaviours. These automata are very powerful in language-theoretic terms. Their languages are not closed under complementation. Further, their language inclusion problem is undecidable and hence cannot be reduced to the emptiness problem which is decidable. Hence in order to solve verification problems posed as language inclusion problems, one must use deterministic timed automata for specifications (which 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). Roughly speaking, such an automaton will consist of a network of timed agents $\|_{i=1}^{K} \mathcal{A}_i$ where each $\mathcal{A}_i$ will operate over an alphabet Σ_i of events. Further, there will be 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 will

* A part of this work has been supported by BRICS, Computer Science Dept., Aarhus University.

† A part of this work has been supported by the IFCPAR Project 1502-1.

only involve the clocks $\{c_1, c_3, c_4\}$. Moreover, the set of clocks that is reset at the end of each a-execution will be $\{c_1, c_3, c_4\}$. Thus the distribution $\widetilde{\Sigma} = \{\Sigma_i\}_{i=1}^{K}$ of events over the agents will canonically determine 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. Admittedly, the technique of presenting a global timed automaton as a product of component timed automata has been used by many authors starting from [1]. 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. It is worth pointing out that the model considered by Yi and Jonsson [24] in the framework of timed CSP can be easily represented as PI automata. Their main result, in our terms, is that language inclusion problem for PI automata is decidable. We establish a variety of results concerning PI automata which subsume the decidability of the language inclusion problem.

In principle, one could view our automata as a restricted kind of labelled timed Petri net. However the semantics we attach to our automata is the standard one for timed automata whereas the semantics one uses for timed Petri nets – with earliest and latest firing times for the transitions – is somewhat different.

A key feature of our automata is that their timed behaviour can be symbolically represented *without any loss of information* by conventional words. As a consequence, it turns out that their theory can be developed with the help of powerful results available in the theory of Mazurkiewicz traces [6]. We wish to emphasise, our automata will however have a conventional timed semantics with the non-negative reals serving as the time frame. A final aspect of PI automata that we wish to mention is that partial order reduction techniques that are under development [4, 10] can be readily applied to our automata.

In pragmatic terms, it is not clear how much modelling power is lost through the restrictions we place on the usage of clocks. In many multi-agent timed systems it seems sufficient to have just one clock for each agent. For instance, a network of timed automata that communicate through shared variables is used to model and analyse the timed behaviour of asynchronous circuits by Maler and Pnueli [17]. It turns out that product interval automata suffice to represent the same class of timed behaviours.

From a theoretical standpoint, PI automata are strictly less expressive than event clock automata due to Alur, Fix, and Henzinger [2] and their state-based version [19] 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 in [14] for a generalisation of event recording automata called recursive event recording automata. Nevertheless we feel that PI automata are of independent interest due to the reasons sketched earlier. For basic information about timed automata and their logics we cite the surveys [3, 13] and their references. An interesting early instance of timed languages which have nice closure properties and which admit a clean logical characterisation can be found in [22].

In the next section we define PI automata. In section 3 we show that these automata (more precisely their languages) are closed under boolean operations. In section 4 we first present a monadic second order logic denoted $\mathrm{TMSO}^{\otimes}$ to capture the timed regular languages recognised by PI automata. We then formulate a linear time temporal logic denoted $\mathrm{TLTL}^{\otimes}$ and sketch automata-theoretic solutions to the satisfiability and model checking problems for $\mathrm{TLTL}^{\otimes}$ in terms of PI automata.

As we point out in the final section, all our ideas can be extended smoothly to a larger setting in which the underlying "symbolic" automata are asynchronous Büchi automata [11]. The resulting timed automata are called distributed interval automata and they can also be studied using techniques taken from trace theory. It is the case that PI automata are less expressive than distributed interval automata which in turn are less expressive than event recording automata [2]. It turns out that the so called cellular version of distributed interval automata correspond to event recording automata. All these extensions as well detailed proofs of all the results presented below are available in the full paper [7].

2 Product Interval Automata

We fix a finite set of agents $\mathcal{P} = \{1, 2, \ldots, K\}$ and let i, j range over $\mathcal{P}$. 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 $\widetilde{\Sigma}$. We let a, b range over Σ. The set of agents that participate in each occurrence of the action a is denoted by $loc(a)$ and is given by: $loc(a) = \{i \mid a \in \Sigma_i\}$. Through the rest of the paper we fix such a $\mathcal{P}$-distributed alphabet $\widetilde{\Sigma}$.

We let $\mathbb{R}^{>0}$ and $\mathbb{R}^{\geq 0}$ denote the set of positive and non-negative reals respectively. Without loss of generality we will use intervals with rational bounds to specify timing constraints (and use ∞ as the upper bound to capture unbounded intervals). If an interval is of the form $[x, y)$ or $[x, y]$ we require x to be a positive rational and if an interval is of the form $(x, y]$ or $[x, y]$ we require y to be a positive rational. An interval defines a subset of reals in the obvious way. Let INT be the set of all intervals of $\mathbb{R}^{>0}$.

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 $(Q_i, \longrightarrow_i, F_i, G_i)$ such that Q_i is a finite set of states, $\longrightarrow_i$, the transition relation, is a finite subset of $Q_i \times (\Sigma_i \times INT) \times Q_i$, and $F_i, G_i \subseteq Q_i$ are, respectively, finitary and infinitary acceptance state sets. $Q_{in} \subseteq Q = Q_1 \times \cdots \times Q_K$ is a set of global start states.

For convenience we will be interested only in infinite runs of a PI automaton. However in such a run some of the component automata may execute only a finite number of actions. It is for this reason we have two types of accepting states for each component automaton.

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

For a word σ in A^∞, we let $prf(\sigma)$ be the set of finite prefixes of σ. A timed word σ over $\widetilde{\Sigma}$ is an element of $(\Sigma \times \mathbb{R}^{>0})^\infty$ such that:

(i) σ is non-decreasing: if $\tau(a,t)(b,t')$ is a prefix of σ then $t \leq t'$.
(ii) if $\tau(a,t)\tau'(b,t')$ is a prefix of σ with $t = t'$, then we must have $loc(a) \cap loc(b) = \emptyset$. Thus, simultaneous actions are allowed but only if they are independent – i.e. their locations are disjoint.
(iii) if σ is infinite, then it must be progressive: for each $t \in \mathbb{R}^{>0}$ there exists a prefix $\tau(a,t')$ of σ such that $t' > t$.

We let $T\widetilde{\Sigma}^\omega$ denote the set of infinite timed words over $\widetilde{\Sigma}$.

Now suppose $K = 1$ so that $\mathcal{P} = \Sigma_1$. Then the definition above (condition (ii) will not apply) will yield the usual notion of a timed word over the alphabet Σ_i. With this in mind, for an alphabet A, we let TA^* and TA^ω be the set of finite and infinite (ordinary) timed words over A, respectively, and set $TA^\infty = TA^* \cup TA^\omega$.

Let $\sigma \in T\widetilde{\Sigma}^\omega$. 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 $\sigma \restriction i$ belongs to $T{\Sigma_i}^\infty$.

Finally, let τ be a finite timed word over $\widetilde{\Sigma}$. Then $time_i(\tau)$ is given inductively by: $time_i(\epsilon) = 0$. Further, $time_i(\tau'(a,t)) = t$ if $a \in \Sigma_i$, and equals $time_i(\tau')$ otherwise. Thus $time_i(\tau)$ is the time at which the last i-action took place during the execution of τ. This notion will play an important role in the rest of the paper. We can now define the timed language accepted by a PI automaton.

Let $\mathcal{A} = (\{\mathcal{A}_i\}_{i \in \mathcal{P}}, Q_{in})$ be a PI automaton over $\widetilde{\Sigma}$ and let $\sigma \in T\widetilde{\Sigma}^\omega$. 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. Suppose $\tau(a,t)$ is a prefix of σ. Then for each $i \in loc(a)$, there exists a transition $\rho(\tau)[i] \xrightarrow{(a,I)}_i \rho(\tau(a,t))[i]$ with $(t - time_i(\tau)) \in I$. Further, 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 $i \in \mathcal{P}$:

(i) If $\sigma \restriction i$ is finite then $\rho(\tau)[i] \in F_i$ where τ is a prefix of σ such that $\tau \restriction i = \sigma \restriction i$.
(ii) If $\sigma \restriction i$ is infinite then $\rho(\tau)[i] \in G_i$ for infinitely many $\tau \in prf(\sigma)$.

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

We can solve the emptiness problem for PI automata by reducing it to the emptiness problem for timed Büchi automaton (TBA) as formulated by Alur and Dill [1]. The concerned timed simulation is the obvious one and the details can be found in [7]. In going from a PI automaton to a timed automaton, there will be a $2^{O(K)}$ blow-up in the size of the automaton.

3 Closure under boolean operations

Here we wish to show that the class of languages accepted by PI automata (over $\widetilde{\Sigma}$) is closed under boolean operations. Using the notion of proper interval sets, we will transform the relevant problems to the domain of ordinary trace languages. We can then apply known constructions involving trace languages to obtain the desired results.

It will be useful to first study a single component of a PI automaton. To this end, let A be a finite alphabet of actions. Then an interval alphabet based on A is a *finite* subset of $A \times INT$. Let Γ be an interval alphabet based on A. Then an interval automaton over Γ is a structure $\mathcal{B} = (Q, \longrightarrow, Q_{in}, F, G)$ where Q is a finite set of states, $\longrightarrow \subseteq Q \times \Gamma \times Q$ is a transition relation, $Q_{in} \subseteq Q$ is a set of initial states, $F \subseteq Q$ is a set of finitary accepting states and $G \subseteq Q$ is a set of infinitary accepting states. $L(\mathcal{B}) \subseteq TA^{\infty}$, the language of timed words accepted by $\mathcal{B}$ is now defined in the obvious way. (To be pedantic, one can set $\mathcal{P} = \{1\}$ and $\Sigma_1 = A$ and apply the definitions from the previous section.) There is also a symbolic language $L_{sym}(\mathcal{B}) \subseteq \Gamma^{\infty}$ that one can associate with $\mathcal{B}$ in the classical manner. The finite words in $L_{sym}(\mathcal{B})$ are accepted using members of F and the infinite words are accepted by viewing G as a Büchi condition (see [21] for a standard treatment). It is also the case that there is a natural map $tw : \Gamma^{\infty} \to TA^{\infty}$ such that $tw(L_{sym}(\mathcal{B})) = L(\mathcal{B})$. The map tw can be defined as follows:

Let $\widehat{\sigma} \in \Gamma^{\infty}$ and $\sigma \in TA^{\infty}$. Then $\sigma \in tw(\widehat{\sigma})$ iff $|\sigma| = |\widehat{\sigma}|$ and for each prefix $\tau(a,t)$ of σ and each prefix $\widehat{\tau}(b,I)$ of $\widehat{\sigma}$ with $|\tau| = |\widehat{\tau}|$ we have $a = b$ and $t - time(\tau) \in I$. The function $time$ is of course given inductively by $time(\epsilon) = 0$ and $time(\tau'(a',t')) = t'$. As usual, for $L \subseteq \Gamma^{\infty}$ we set $tw(L) = \bigcup_{\widehat{\sigma} \in L} tw(\widehat{\sigma})$. To sum up, the simple observation we wish to start with is:

Proposition 1. *Let $\mathcal{B}$ be an interval automaton over the interval alphabet Γ (based on A). Then $L(\mathcal{B}) = tw(L_{sym}(\mathcal{B}))$.* □

Now we turn to proper interval sets. We say that $\mathcal{I} \subseteq INT$ is proper iff $\mathcal{I}$ is a *finite* partition of $(0, \infty)$. An interval alphabet Γ based on A is said to be proper iff for each a in A, the set of intervals Γ_a is proper where $\Gamma_a = \{I \mid (a, I) \in \Gamma\}$. The next observation which is easy to verify is crucial for our purposes.

Proposition 2. *Let Γ be a proper interval alphabet based on A and let $\sigma \in TA^{\infty}$. Then there exists a* unique *word $\widehat{\sigma} \in \Gamma^{\infty}$ such that $\sigma \in tw(\widehat{\sigma})$.* □

The uniqueness of $\widehat{\sigma}$ as stated in the above result is the key to reducing the study of timed languages to the study of trace languages in the present setting. This can be achieved by embedding interval alphabets into proper interval alphabets. To formalise this idea, let $\mathcal{I}, \mathcal{I}' \subseteq INT$. Then $\mathcal{I}'$ is said to *cover* $\mathcal{I}$ iff each interval in $\mathcal{I}$ can be expressed as the union of a set of intervals taken from $\mathcal{I}'$. Clearly the choice of such a set of intervals (for a given member of $\mathcal{I}$) will be unique in case $\mathcal{I}'$ is a proper interval set.

Let Γ and Γ' be two interval alphabets based on the alphabet A. Then Γ' covers Γ iff Γ'_a covers Γ_a for each $a \in A$.

We will say that $L \subseteq TA^{\infty}$ is a regular interval language over A iff there exists an interval alphabet Γ based on A and an interval automaton $\mathcal{B}$ over Γ such that $L = L(\mathcal{B})$. For convenience we will often say "interval language" to mean "regular interval language."

Proposition 3. *Let A be a finite alphabet of actions.*

(i) Suppose Γ is an interval alphabet based on A. Then one can effectively construct a proper *interval alphabet Γ' based on A such that Γ' covers Γ.*
(ii) The class of interval languages over A is closed under boolean operations.

Proof sketch. (i) follows from the observation that given an interval set $\mathcal{I} = \{I_1, \ldots, I_m\}$ we can form the set $X \cup \{0, \infty\} = X'$ where X is the set of rational numbers that appear as bounds of the intervals in $\mathcal{I}$. Let $X' = \{0, x_1, \ldots, x_n, \infty\}$ with $0 < x_1 < x_2 < \cdots < x_n < \infty$. Then

$$\mathcal{I}' = \{(0, x_1), [x_1, x_1], (x_1, x_2), [x_2, x_2], \ldots, (x_n, \infty)\}$$

is a proper interval set which covers $\mathcal{I}$.

To prove (ii), one first observes that closure under union is obvious. So let $\mathcal{B}$ be an interval automaton over the interval alphabet Γ. Then we can construct Γ' such that Γ' is proper and covers Γ. Let $\mathcal{B} = (Q, \longrightarrow, Q_{in}, F, G)$. Then we construct the interval automaton $\mathcal{B}' = (Q, \Longrightarrow, Q_{in}, F, G)$ over Γ' where $\Longrightarrow \subseteq Q \times \Gamma' \times Q$ is given by: $q \overset{(a,I')}{\Longrightarrow} q'$ iff there exists $q \overset{(a,I)}{\longrightarrow} q'$ in $\mathcal{B}$ such that $I' \subseteq I$. It is easy to verify that $L(\mathcal{B}) = L(\mathcal{B}')$. Using classical techniques we can now find an interval automaton $\mathcal{B}''$ over Γ' such that $L_{sym}(\mathcal{B}'') = (\Gamma')^{\infty} - L_{sym}(\mathcal{B}')$. Since Γ' is proper we can now use proposition 2 to conclude that $L(\mathcal{B}'') = TA^{\infty} - L(\mathcal{B}')$. $\square$

The technique used in the second part of the proof is deployed in different contexts to establish many of our results. It is for this reason we have presented some of the key details in a very simple setting.

We can now address the main concerns of this section. To show closure under boolean operations of the languages accepted by PI automata, we will first characterise these languages using interval languages. Let $\{L_i\}$ be a family of languages such that $L_i \subseteq T\Sigma_i{}^{\infty}$ for each i. (Recall $\widetilde{\Sigma} = \{\Sigma_i\}_{i \in \mathcal{P}}$.) Then $\otimes(L_1, \ldots, L_K)$ denotes the language $L \subseteq T\widetilde{\Sigma}^{\omega}$ defined as:

$$\sigma \in L \text{ iff } \sigma \restriction i \in L_i \text{ for each } i.$$

We will say that $L \subseteq T\widetilde{\Sigma}^{\omega}$ is a regular direct product interval language iff there exists a family $\{L_i\}$ such that each L_i is a regular interval language over Σ_i and $L = \otimes(L_1, \ldots, L_K)$. A regular product interval language over $\widetilde{\Sigma}$ is a finite union of regular direct product interval languages over $\widetilde{\Sigma}$. Again, for convenience, we will often say "product interval language" to mean "regular product interval language" etc..

Theorem 1. *Let $L \subseteq T\widetilde{\Sigma}^{\omega}$. Then L is a product interval language iff there exists a PI automaton $\mathcal{B}$ over $\widetilde{\Sigma}$ such that $L = L(\mathcal{B})$.* $\square$

Using the theory of product languages [15], which is a natural subclass of Mazurkiewicz trace languages and the technique used in the proof of Proposition 3 we can now show:

Theorem 2. *The class of product interval languages over $\widetilde{\Sigma}$ is closed under boolean operations.*

4 A logical characterisation of product interval languages

Our goal here is to first develop a logical characterisation of product interval languages in terms of a monadic second order logic. We shall then present a related linear time temporal logic for reasoning about timed behaviours captured by PI automata. As done in the previous section it will be convenient to first concentrate on the one component case.

Let A be a finite alphabet of actions. Here and in the logics to follow, we assume a supply of individual variables $x, y, \ldots$ and set variables $X, Y, \ldots$. For each $a \in A$ we have a unary predicate Q_a. There are also predicates of the form $\Delta(x, I)$ in which x is an individual variable, and I is a member of INT.

The syntax of the logic TMSO(A) is given by:

$$\begin{aligned}\mathrm{TMSO}(A) ::= {} & x \in X \mid x < y \mid Q_a(x) \mid \Delta(x, I) \mid \varphi \vee \varphi' \mid \sim \varphi \mid \\ & (\exists x)\varphi \mid (\exists X)\varphi.\end{aligned}$$

A structure for this logic is a pair $(\sigma, \mathbb{I})$ where $\sigma \in TA^{\infty}$ and $\mathbb{I}$ is an interpretation which assigns to each individual and set variable, an element and subset, respectively, of the set $pos(\sigma)$. We define $pos(\sigma)$ to be the set $\{1, 2, \ldots, |\sigma|\}$ in case σ is finite, and $\{1, 2, \ldots\}$ if σ is infinite. As is customary, σ can be also viewed as a map from $pos(\sigma)$ into A with $\sigma(n)$ for $n \in pos(\sigma)$ having the obvious meaning. We will use this view of σ whenever convenient. Turning now to the semantics, let $(\sigma, \mathbb{I})$ be a structure. Then $\sigma \models_{\mathbb{I}} (x < y)$ iff $\mathbb{I}(x) <' \mathbb{I}(y)$. (Here $<'$ is the usual ordering over the integers). Further, $\sigma \models_{\mathbb{I}} Q_a(x)$ iff $\sigma(\mathbb{I}(x)) = (a, t)$ for some t. As for the predicate $\Delta(x, I)$, we have

$$\begin{aligned}\sigma \models_{\mathbb{I}} \Delta(x, I) \text{ iff } & t - time(\tau) \in I, \text{ where } \tau(a,t) \text{ is the prefix of } \sigma \text{ s.t.} \\ & |\tau(a, t)| = \mathbb{I}(x).\end{aligned}$$

The semantics of the logical connectives $\sim$, and $\vee$, and the existential quantifiers $\exists x$ and $\exists X$ are given in the expected manner. For a sentence $\varphi \in$ TMSO(A) we set $L(\varphi) = \{\sigma \in TA^{\infty} \mid \sigma \models \varphi\}$.

We can now use the classical logical characterisation of ω-regular languages and the notion of proper interval alphabets to establish:

Theorem 3. *Let $L \subseteq TA^{\infty}$. Then L is an interval language iff there exists a sentence $\varphi \in$ TMSO(A) such that $L = L(\varphi)$.*

Next we wish to present the logic TMSO$^{\otimes}(\widetilde{\Sigma})$ which will characterise product interval languages. The syntax of the logic is given by:

$$\mathrm{TMSO}^{\otimes}(\widetilde{\Sigma}) ::= \varphi(i) \mid \ \sim \alpha \mid \alpha \vee \beta \mid \alpha \wedge \beta$$

where for each formula $\varphi(i)$ we require $\varphi \in$ TMSO(Σ_i).

A structure for the logic is a pair $(\sigma, \{\mathbb{I}_i\}_{i \in \mathcal{P}})$, where $\sigma \in T\widetilde{\Sigma}^{\omega}$ and for each $i \in \mathcal{P}$, $\mathbb{I}_i$ is an interpretation for individual and set variables over the set of positions $pos(\sigma \restriction i)$. As for the semantics,

$$\sigma \models_{\{\mathbb{I}_i\}_{i \in \mathcal{P}}} \varphi(i) \text{ iff } \sigma \restriction i \models_{\mathbb{I}_i} \varphi.$$

The boolean operators $\sim$, $\vee$ and $\wedge$ are interpreted in the usual manner. Once again, for a sentence $\varphi \in \mathrm{TMSO}^{\otimes}(\widetilde{\Sigma})$, we set $L(\varphi) = \{\sigma \in T\widetilde{\Sigma}^{\omega} \mid \sigma \models \varphi\}$.

Theorem 4. *Let $L \subseteq T\widetilde{\Sigma}^{\omega}$. Then L is a product interval language iff $L = L(\varphi)$ for some sentence $\varphi \in \mathrm{TMSO}^{\otimes}(\widetilde{\Sigma})$.*

This result is established by exploiting Theorem 1 and yet again the notion of proper interval alphabets.

As a natural temporal logic counterpart of $\mathrm{TMSO}^{\otimes}(\widetilde{\Sigma})$, we present here the logic $\mathrm{TLTL}^{\otimes}(\widetilde{\Sigma})$. Again, we will build up the formulas in two steps. For ease of presentation, we will not deal with atomic propositions here. Where necessary, they can be introduced component-wise and dealt with easily.

$$\mathrm{TLTL}(\Sigma_i) ::= \top \mid \ \sim\varphi \mid \varphi \vee \varphi' \mid \langle a, I\rangle\varphi \mid O\varphi \mid \varphi U \varphi' \,.$$

For each formula of the form $\langle a, I\rangle\varphi$, we require that $a \in \Sigma_i$.

The formulas of $\mathrm{TLTL}^{\otimes}(\widetilde{\Sigma})$ are simply boolean combinations of $\mathrm{TLTL}(\Sigma_i)$-formulas:

$$\mathrm{TLTL}^{\otimes}(\widetilde{\Sigma}) ::= \varphi(i) \mid \ \sim\alpha \mid \alpha \vee \beta.$$

Here we require from each formula of the form $\varphi(i)$ that $\varphi \in \mathrm{TLTL}(\Sigma_i)$.

The semantics of this logic is given by defining the relation $\sigma \models \alpha$ with $\sigma \in T\widetilde{\Sigma}^{\omega}$. To spell out the details, $\sigma \models \varphi(i)$ iff $\sigma \restriction i, \epsilon \models_i \varphi$. The relation $\models_i$ is defined in the obvious way. For instance,

$$\begin{aligned} \sigma, \tau \models_i \langle a, I\rangle\varphi \ \text{ iff } \ & \text{there exists } t \text{ such that } \tau(a,t) \in \mathit{prf}(\sigma) \text{ and} \\ & \sigma, \tau(a,t) \models_i \varphi \text{ and } t - \mathit{time}(\tau) \in I. \end{aligned}$$

On the other hand $\sigma, \tau \models_i O\varphi$ iff there exists $\tau(a,t) \in \mathit{prf}(\sigma)$ such that $\sigma, \tau(a,t) \models_i \varphi$.

For each α in $\mathrm{TLTL}^{\otimes}(\widetilde{\Sigma})$ we define $L(\alpha) = \{\sigma \in T\widetilde{\Sigma}^{\omega} \mid \sigma \models \alpha\}$. Clearly, α is satisfiable iff $L(\alpha)$ is non-empty.

Theorem 5. *For each α in $\mathrm{TLTL}^{\otimes}(\widetilde{\Sigma})$ we can effectively construct a PI automaton $\mathcal{A}_\alpha$ such that $L(\alpha) = L(\mathcal{A}_\alpha)$ and the number of global states of $\mathcal{A}_\alpha$ is $2^{O(|\alpha|)}$. Further, the largest constant mentioned in the i^{th} component of $\mathcal{A}_\alpha$ is at most the largest i-type constant mentioned in α.*

Proof sketch. The proof follows the usual automata-theoretic techniques. To account for negation correctly, the local transition relations of $\mathcal{A}_\alpha$ must be over proper alphabets. The obvious idea of converting α into a formula which mentions only intervals taken from a proper interval set will cause an exponential blow-up. To avoid this, we define the (Fisher-Ladner) closure of α to be smallest set of formulas containing the subformulas of α and satisfying: If $(\varphi U \varphi')(i)$ is in the closure then so is $O((\varphi U \varphi')(i))$. Let CL be the closure of α and $\mathrm{CL}_i = \mathrm{CL} \cap \mathrm{TLTL}(\Sigma_i)$ for each i. We now define an i-type atom to be a propositionally consistent subset of CL_i which satisfies: If $\langle a, I\rangle\varphi$ and $\langle b, I'\rangle\varphi'$

are in the atom then $a = b$ and $I \cap I' \neq \emptyset$. We can now use i-type atoms to manufacture the i-local states. The i-local transition relation can then be chosen to be proper. The remaining details are guided by the way the logic $\mathrm{LTL}^{\otimes}(\Sigma)$ is dealt with in [20]. $\square$

Using theorem 5, and the emptiness check for PI automata in section 2, we can decide satisfiability of α in time $2^{O(|\alpha|+K \log K)}$. In fact, we can show:

Corollary 1. *The satisfiability problem for* $\mathrm{TLTL}^{\otimes}(\widetilde{\Sigma})$ *is* PSPACE-*complete.*

Given a formula α in $\mathrm{TLTL}^{\otimes}(\widetilde{\Sigma})$ we can check the emptiness of the automaton $\mathcal{A}_\alpha$ of theorem 5 non-deterministically while using space polynomial in the size of α. The argument is similar to the one in [1]. The fact that the problem is PSPACE-hard follows from the observation that the satisfiability problem for LTL can be reduced to the satisfiability problem for the one-agent case of $\mathrm{TLTL}^{\otimes}(\widetilde{\Sigma})$.

Next suppose we consider a real-time program Pr modelled by a PI automaton $\mathcal{A}_{Pr}$, and a formula α of $\mathrm{TLTL}^{\otimes}(\widetilde{\Sigma})$. Then Pr is said to meet the specification α iff $L(\mathcal{A}_{Pr}) \subseteq L(\alpha)$. The model checking problem for $\mathrm{TLTL}^{\otimes}(\widetilde{\Sigma})$ is to determine whether Pr meets the specification α. It is not difficult to show:

Theorem 6. *The model checking problem for* $\mathrm{TLTL}^{\otimes}(\widetilde{\Sigma})$ *is* PSPACE *complete.*

We have not studied in detail the power and limitations of $\mathrm{TLTL}^{\otimes}(\widetilde{\Sigma})$ from a pragmatic standpoint. We have explored $\mathrm{TLTL}^{\otimes}(\widetilde{\Sigma})$ here mainly because it is the natural temporal logic suggested by $\mathrm{TMSO}^{\otimes}(\widetilde{\Sigma})$, the characteristic logic for PI automata. In fact, it is not difficult to show that $\mathrm{TLTL}^{\otimes}(\widetilde{\Sigma})$ is expressively equivalent to the first-order fragment of $\mathrm{TMSO}^{\otimes}(\widetilde{\Sigma})$.

In order to provide some example properties that can be stated in this timed temporal logic, we first sketch briefly how product interval automata can be used to model asynchronous digital circuits. (The full details will appear elsewhere.) Our starting point is the circuit model developed in [17] using inertial delays. The behaviour of a k-gate circuit χ, denoted $L(\chi)$, can be defined as a subset of $T\Sigma^\infty$, where Σ is the alphabet $\{0,1\}^k \times \{0,1\}^k$. An element of $\{0,1\}^k$ will represent the visible values of the gate outputs. The occurrence of an action (s, s') will indicate the visible values of the gate outputs going from s to s' due to the simultaneous switching of an appropriate set of gates. We can associate with χ, a $(k+1)$-component product interval automaton $\mathcal{A}_\chi$ such that $L(\mathcal{A}_\chi) = L(\chi)$. We use one component for each gate in the circuit. This component will keep track of the switching status of the gate (whether excited or quiescent); and, in case it is in an excited state, the time elapsed since its excitement was initiated last. The $(k+1)$-th component keeps track of the current vector of visible values. Suppose its state is s and it permits an action (t, t') at s, then it will be the case that $s = t$. It will also be the case that the circuit allows the transition from t to t' through the switching of a subset of gates that are excited at t. Furthermore, the distribution of actions across the components is such that the action (t, t') will involve – apart from the component $k+1$ – all those components whose switching status is affected by the switches. As for timing constraints, the component i will have timing constraints that enforce the inertial delay corresponding to the output wire of gate-i, while the $(k+1)$-th component will be free of any timing constraints.

We can use the logic $\mathrm{TLTL}^{\otimes}(\widetilde{\Sigma})$ to express (and hence model-check) a variety of properties of χ. Two examples of such properties are:

- If the input signals to χ stabilise then the behaviour of χ eventually stabilises.
- gate i is d-persistent; whenever it becomes excited it turns quiescent only through its switching and further more this switching occurs within d time units since it got excited.

5 Distributed Interval Automata

Here we consider a more expressive class of distributed timed automata but having the same flavour as PI automata. The extension we have in mind will parallel the extension of product automata to asynchronous automata in the setting of traces as detailed in [15].

We shall continue to work with the $\mathcal{P}$-distributed alphabet $\widetilde{\Sigma}$. An a-interval for us will be a map $J : loc(a) \rightarrow INT$. Let $INT^{(a)}$ be the set of a-intervals. An a-interval, viewed as a rectangular region, will constitute the guard constraining the timed occurrence of an a-action.

In the definition of our automata and elsewhere we will often write $\{X_i\}$ to denote the family $\{X_i\}_{i\in\mathcal{P}}$. A distributed interval automaton over $\widetilde{\Sigma}$ is a structure

$$\mathcal{A} = (\{Q_i\}, \{\longrightarrow_a\}_{a\in\Sigma}, Q_{in}, \{F_i, G_i\})$$

where the various components of $\mathcal{A}$ are defined as follows. While doing so we will also develop some notations.

(i) Each Q_i is a finite non-empty set called the set of i-local states. Let P be a non-empty subset of $\mathcal{P}$. Then a P-state is a map $q : P \rightarrow \bigcup_{i\in\mathcal{P}} Q_i$ such that $q(i) \in Q_i$ for each $i \in P$. An a-state is just a $loc(a)$-state and a $\mathcal{P}$-state will be called a global state. We let Q_a denote the set of a-states and $Q_{\mathcal{P}}$ the set of global states.
(ii) $\longrightarrow_a$ is a finite subset of $Q_a \times INT^{(a)} \times Q_a$ for each $a \in \Sigma$.
(iii) $Q_{in} \subseteq Q_{\mathcal{P}}$ is the set of global initial states.
(iv) $F_i, G_i \subseteq Q_i$ for each i.

Thus a distributed interval automaton is a timed version of an asynchronous Büchi automaton as formulated in [18] which in turn is a minor variant of the original formulation due to Gastin and Petit [11]. If one ignores the acceptance conditions then an asynchronous automaton can be viewed as a labelled 1-safe Petri net. Consequently a distributed interval automaton can also be viewed as a timed 1-safe Petri net. However, as pointed out in the introduction, the timed semantics we assign to our automata is *not* the semantics one encounters in the literature concerning timed Petri nets [5].

We can now systematically extend the theory of PI automata – as sketched in the previous sections – to distributed interval automata. The details can be found in [7].

It turns out that the so called cellular version of distributed interval automata, has the same expressive power as the event recording automata [2]. In this version of asynchronous automata there will be one component for each letter in Σ and one clock associated with each component. The interesting fact is that in the untimed setting both versions of asynchronous automata have the same expressive power. Once again we refer the reader to [7] for the details.

References

1. R. Alur, D. L. Dill: A theory of timed automata, *Theoretical Computer Science* 126: 183–235 (1994).
2. R. Alur, L. Fix, T. A. Henzinger: Event-clock automata: a determinizable class of timed automata, *Proc. 6th International Conference on Computer-aided Verification, LNCS 818,* 1–13, Springer-Verlag (1994).
3. R. Alur, T. A. Henzinger: Logics and Models of Real Time: A Survey, in *Real-Time: Theory in Practice,* J. W. de Bakker, H. Huizing, W. -P. de Roever, G. Rozenberg (Eds.), LNCS 600, 74–106, (1992).
4. J. Bengtsson, B. Jonsson, J. Lilius, W Yi: Partial Order Reductions for Timed Systems, *Proc. CONCUR '98, LNCS 1466* (1998).
5. B. Berthomieu, M. Diaz: Modelling and Verification of Time Dependent Systems Using Time Petri Nets, *IEEE trans. on Soft. Engg.* Vol 17, No. 3, March 1991
6. V. Diekert, G. Rozenberg: *The Book of Traces*, World Scientific, Singapore (1995).
7. D. D'Souza, P. S. Thiagarajan: Distributed Interval Automata, Internal Report TCS-98-3, Chennai Mathematical Institute (1998). (available at `http://www.smi.ernet.in/techreps/`)
8. W. Ebinger, A. Muscholl: Logical definability on infinite traces, *Theoretical Computer Science* 154: 67–84 (1996).
9. T. A. Henzinger, P. W. Kopke, A. Puri, P. Varaiya: What's decidable about hybrid automata?, *Proc. 27th Annual Symposium on Theory of Computing,* 373–382, ACM Press (1995).
10. M. Minea: Partial Order Reduction for Model Checking of Timed Automata. To appear in the Proceedings of CONCUR'99.
11. P. Gastin, A. Petit: Asynchronous cellular automaton for infinite traces, *Proceedings of ICALP '92, LNCS 623*, 583–594 (1992).
12. R. Gerth, D. Peled, M. Vardi, P. Wolper: Simple On-the-fly Automatic Verification of Linear Temporal Logic. *Proc. 15th IFIP WG 6.1 Int. Workshop on Protocol Specification, Testing, and Verification.* North-Holland Publ. (1995).
13. T. A. Henzinger: It's About Time: Real-Time Logics Reviewed, *Proc. CONCUR '98, LNCS 1466,* 366–372 (1998).
14. T. A. Henzinger, J.-F. Raskin, and P.-Y. Schobbens: The regular real-time languages, *Proc. 25th International Colloquium on Automata, Languages, and Programming 1998, LNCS 1443,* 580–591 (1998).
15. J. G. Henriksen, P. S. Thiagarajan: A Product Version of Dynamic Linear Time Temporal Logic, *Proc. CONCUR '97, LNCS 1243,* 45–58, (1997).
16. N. Klarlund, M. Mukund, M. Sohoni: Determinizing Büchi Asynchronous Automata, *Proceedings of FSTTCS 15, LNCS 1026,* 456–470 (1995).

17. O. Maler, A. Pnueli: Timing Analysis of Asynchronous Circuits using Timed Automata, in *Proc. CHARME '95, LNCS 987,* 189–205 (1995).
18. M. Mukund, P. S. Thiagarajan: Linear Time Temporal Logics over Mazurkiewicz Traces, *Proc. MFCS 96, LNCS 1113,* 62–92 (1996).
19. J. -F. Raskin, P. -Y. Schobbens: State-clock Logic: A Decidable Real-Time Logic, *Proc. HART '97: Hybrid and Real-Time Systems, LNCS 1201,* 33–47 (1997).
20. P. S. Thiagarajan: A Trace Consistent Subset of PTL, *Proc. CONCUR '95,* LNCS 962, 438–452 (1995).
21. W. Thomas: Automata on Infinite Objects, in J. V. Leeuwen (Ed.), *Handbook of Theoretical Computer Science*, Vol. B, 133–191, Elsevier Science Publ., Amsterdam (1990).
22. Th. Wilke: Specifying Timed State Sequences in Powerful Decidable Logics and Timed Automata, in *Formal Techniques in Real-Time and Fault-Tolerant Systems, LNCS 863,* 694–715 (1994).
23. G. Winskel, M. Nielsen: Models for Concurrency, in S. Abramsky, D. Gabbay (Eds.) *Handbook of Logic in Computer Sc., Vol. 3,* Oxford Univ Press (1994).
24. W. Yi, B. Jonsson: Decidability of Timed Language-Inclusion for Networks of Real-Time Communicating Sequential Processes, in *Proc. FST&TCS 94, LNCS 880* (1994).