Mathematical Foundations of Computer Science, 1996
Proceedings, Wojtek Penczek (Ed.),
Lecture Notes in Computer Science **1113**, Springer-Verlag (1996) 32–62

Linear Time Temporal Logics over Mazurkiewicz Traces

Madhavan Mukund and P.S. Thiagarajan

School of Mathematics, SPIC Science Foundation,
92 G N Chetty Rd, Madras 600 017, India
E-mail: {madhavan,pst}@ssf.ernet.in

Abstract. Temporal logics are a well-established tool for specifying and reasoning about the computations performed by distributed systems. Although temporal logics are interpreted over sequences, it is often the case that such sequences can be gathered together into equivalence classes where all members of an equivalence class represent the same partially ordered stretch of behaviour of the system. This appears to have important implications for improving the practical efficiency of automated verification methods based on temporal logics. With this as motivation, we study logics that are directly interpreted over partial orders. We survey a number of linear time temporal logics whose underlying frames are Mazurkiewicz traces. We describe automata theoretic methods for solving the satisfiability and model checking problems for these logics. It turns out that we still do not know what the "canonical" linear time temporal logic over Mazurkiewicz traces looks like. We identify here the criteria that should be met by this elusive logic.

Introduction

Propositional Linear time Temporal Logic (LTL) proposed by Pnueli [Pnu] has become a well established tool for specifying and reasoning about complex distributed behaviours [MP]. A central feature of LTL is that its formulas are interpreted over infinite sequences. In applications of LTL, the infinite sequences consist of the runs of a distributed system with each run being an infinite sequence of states assumed by the system or an infinite sequence of actions executed by the system during the course of a computation. Interesting distributed systems consist of a number of autonomous sequential agents that coordinate their behaviour with the help of some communication mechanism. In such systems, substantial portions of a computation will consist of causally independent tasks performed by different agents at separate locations. Consequently a single partially ordered stretch of behaviour of the system will be modelled by many different runs that differ from each other only in the order in which they record causally independent occurrences of actions. This kind of run-based view is often referred to as an interleaved semantics of distributed systems.

The interleaved view of the behaviour of distributed systems has proved to be very successful and popular. However it has been known for some time that the practical effectiveness of LTL and related formalisms can be often enhanced by modelling and analyzing the concerned behaviours in terms of partial orders rather than sequences.

In typical applications, an LTL formula constitutes the specification of the system behaviour and the verification problem consists of checking whether every run of the system is a model of the formula and therefore whether the system meets the specification. The property expressed by the specification is very often of the kind where either *all* the interleaved runs corresponding to a single partially ordered computation have the property or *none* of the interleavings have the property. A typical example of such a property is freedom from deadlock, as pointed out by Valmari [Val]. As a result, it suffices to verify the desired property for just one representative run of each partially ordered computation. The resulting saving in running time and memory usage can be substantial in practice [GW]. This is the background and motivation underlying the so called partial order based verification methods which are a subject of active research [GW, KP, Val].

There is an alternative way to exploit non-sequential behaviours and the attendant partial order based verification methods. It consists of developing temporal logics and related techniques that can be *directly* applied to specify and reason about the properties of partial order based runs of a distributed system. In this paper we survey linear time temporal logics that have arisen from this approach.

In going from sequences to partial orders it is easy to go overboard because so many possibilities are available. Fortunately, in the context of distributed behaviours, Mazurkiewicz has formulated a tractable and yet very fruitful way of passing from sequences to partial orders [Maz]. The resulting restricted partial orders are known as Mazurkiewicz traces, often called—as we shall do here—just traces. The theory of traces is well developed [Die, DR] and is strongly related to the theory of other well known formalisms such as Petri nets and event structures. Further, the classical theory of ω-regular (word) languages in terms of its logical, algebraic and automata-theoretic aspects has been successfully extended to ω-regular trace languages [EM, GP]. Finally, the structures that underlie the partial order based verification methods being developed recently can be almost always be viewed as traces.

Hence there is a good deal of motivation for formulating linear time temporal logics that are to be directly interpreted over traces. Many such logics are now available. In the present survey, we will mainly concentrate on the ones that fulfill two criteria:

(i) The logic should be expressible within the first order theory of traces.
(ii) The satisfiability problem for the logic should admit a treatment in terms of asynchronous Büchi automata.

This seemingly arbitrary choice of criteria can be justified as follows. LTL is *the* linear time temporal logic over sequences in that it is equivalent in expressive power to the first order theory of sequences [Zuc]. We consider the task of identifying the counterpart of LTL for traces to be an important one both from a theoretical and practical standpoint (see the last portion of Section 4). At present we do not know what this counterpart of LTL looks like. However, it seems a good starting point to concentrate on those linear time temporal logics that are at least no more expressive than the first order theory of traces.

As for the second criterion, an appealing feature of LTL is that its satisfiability and model checking problems can be transparently solved using Büchi automata [VW]. This has led to a clean separation of the logical and combinatorial aspects of these problems, thus contributing to the development of automated verification methods and related optimization techniques. The evidence available at present suggests that asynchronous Büchi automata are an appropriate machine model for dealing with ω-regular trace languages. Hence it seems worthwhile to lift the interplay between LTL and Büchi automata to the level of traces.

In the next section we review the basic aspects of traces. In Section 2 we describe asynchronous Büchi automata and present our version of these automata called, for want of a better name, A2-automata. In Section 3, the heart of the paper, we present the logic TrPTL (Trace based Propositional Temporal logic of Linear time) and two of its sublogics TrPTL$^{\mathrm{con}}$ and TrPTL$^{\otimes}$. The logic TrPTL is directly interpreted over traces. We show that the satisfiability and model checking problems for TrPTL can be solved using A2-automata. We then show that the syntactic restrictions imposed to obtain TrPTL$^{\mathrm{con}}$ and TrPTL$^{\otimes}$ lead to corresponding simplifications in the world of automata. After presenting these results we survey a number of other temporal logics that use traces as their underlying frames. In Section 4 we show that TrPTL is expressible within the first order theory of traces. The final section contains concluding remarks.

Most of the results will be presented without proofs. The proofs are either available in the literature or can be easily manufactured using the results available in the literature.

1 Traces

The starting point for trace theory is a trace alphabet (Σ, I), where Σ, the alphabet, is a finite set and $I \subseteq \Sigma \times \Sigma$ is an irreflexive and symmetric independence relation. In most applications, Σ consists of the actions performed by a distributed system while I captures a strong static notion of causal independence between actions. The idea is that contiguous independent actions occur with no causal order between them. Thus, every sequence of actions from Σ corresponds to an interleaved observation of a partially-ordered stretch of system behaviour. This leads to a natural equivalence relation over execution sequences: two sequences are equated iff they correspond to different interleavings of the same partially-ordered stretch of behaviour.

To formulate this equivalence relation precisely, we need some terminology. For the rest of the section we fix a trace alphabet (Σ, I) and let a, b range over Σ. $D = (\Sigma \times \Sigma) - I$ is called the dependency relation. Note that D is reflexive and symmetric. A set $p \subseteq \Sigma$ is called a D-clique iff $p \times p \subseteq D$. We set $\Sigma^{\infty} = \Sigma^{*} \cup \Sigma^{\omega}$

where Σ^* is the set of finite words over Σ and Σ^ω is the set of infinite words over Σ. We let σ, σ' with or without subscripts range over Σ^∞ and τ, τ' with or without subscripts range over Σ^*. The equivalence relation $\sim_I \subseteq \Sigma^\infty \times \Sigma^\infty$ induced by I is given by:

$$\sigma \sim_I \sigma' \text{ iff } \sigma \restriction p = \sigma' \restriction p \text{ for every } D\text{-clique } p.$$

Here and elsewhere, if A is a finite set, $\rho \in A^\infty$ and $B \subseteq A$ then $\rho \restriction B$ is the sequence obtained by erasing from ρ all occurrences of letters in $A - B$.

Clearly $\sim_I$ is an equivalence relation. Notice that if $\sigma = \tau ab\sigma_1$ and $\sigma' = \tau ba\sigma_1$ with $(a,b) \in I$ then $\sigma \sim_I \sigma'$. Thus σ and σ' are identified if they differ only in the order of appearance of a pair of adjacent independent actions. In fact, for finite words, an alternative way to characterize $\sim_I$ is to say that $\sigma \sim_I \sigma'$ iff σ' can be obtained from σ by a finite sequence of permutations of adjacent independent actions. Unfortunately, the definition of $\sim_I$ in terms of permutations is too naïve to be transported to infinite words, which is why we work with the less intuitive definition presented here.

The equivalence classes generated by $\sim_I$ are called *(Mazurkiewicz) traces*. The theory of traces is well developed and documented—see [Die, DR] for basic material as well as a substantial number of references to related work.

Traces have many equivalent representations. We shall view traces as special kinds of labelled partial orders. Since sequences can be viewed as labelled *total* orders, this representation emphasizes that traces are an elegant and non-trivial generalization of sequences.

Recall that a Σ-labelled poset is a structure $F = (E, \leq, \lambda)$ where $\leq$ is a partial order on the set E and $\lambda : E \to \Sigma$ is a labelling function. The *covering relation* $\lessdot \subseteq E \times E$ is given by: $e \lessdot e'$ iff $e < e'$ (i.e., $e \leq e'$ and $e \neq e'$) and for every $e'' \in E$, $e \leq e'' \leq e'$ implies $e = e''$ or $e'' = e'$.

For $X \subseteq E$ we define $\downarrow X$ to be the set $\{y \mid y \leq x \text{ for some } x \in X\}$. If X is a singleton $\{x\}$, we write $\downarrow x$ instead of $\downarrow\{x\}$.

We can now formulate traces in terms of labelled partial orders. A *trace* over (Σ, I) is a Σ-labelled poset $F = (E, \leq, \lambda)$ which satisfies the following conditions.

- E is a countable set.
- For each $e \in E$, $\downarrow e$ is a finite set.
- For all $e, e' \in E$, if $e \lessdot e'$ then $(\lambda(e), \lambda(e')) \in D$.
- For all $e, e' \in E$, if $(\lambda(e), \lambda(e')) \in D$ then $e \leq e'$ or $e' \leq e$.

Let $TR(\Sigma, I)$ denote the set of Σ-labelled posets that satisfy the definition above. We now sketch briefly the proof that $\Sigma^\infty/{\sim_I}$ and $TR(\Sigma, I)$ represent the same class of objects. We construct representation maps $\mathsf{str} : \Sigma^\infty \to TR(\Sigma, I)$ and $\mathsf{trs} : TR(\Sigma, I) \to \Sigma^\infty/\sim_I$ and state some results which show that these maps are "inverses" of each other. We shall not prove these results. The details can be easily obtained using the constructions developed in [WN] for relating traces and event structures.

Henceforth, we will not distinguish between isomorphic elements in $TR(\Sigma, I)$. In other words, whenever we write $F = F'$ for traces $F = (E, \leq, \lambda)$ and $F' =$

$(E', \leq', \lambda')$, we mean that there is a label-preserving isomorphism between F and F'.

For $\sigma \in \Sigma^\infty$, $[\sigma]$ stands for the $\sim_I$-equivalence class containing σ. We use $\preceq$ to describe the usual prefix ordering over sequences. Let $\mathrm{prf}(\sigma)$ denote the set of finite prefixes of σ.

We now define $\mathsf{str} : \Sigma^\infty \to TR(\Sigma, I)$. Let $\sigma \in \Sigma^\infty$. Then $\mathsf{str}(\sigma) = (E, \leq, \lambda)$ where:

- $E = \{\tau a \mid \tau a \in \mathrm{prf}(\sigma)\}$. Recall that $\tau \in \Sigma^*$ and $a \in \Sigma$. Thus $E = \mathrm{prf}(\sigma) - \{\varepsilon\}$, where ε is the null string.
- $\leq \subseteq E \times E$ is the least partial order which satisfies:
  For all $\tau a, \tau' b \in E$, if $\tau a \preceq \tau' b$ and $(a, b) \in D$ then $\tau a \leq \tau' b$.
- For $\tau a \in E$, $\lambda(\tau a) = a$.

The map str induces a natural map str' from $\Sigma^\infty / \sim_I$ to $TR(\Sigma, I)$ defined by $\mathsf{str}'([\sigma]) = \mathsf{str}(\sigma)$. One can show that if $\sigma, \sigma' \in \Sigma^\infty$, then $\sigma \sim_I \sigma'$ iff $\mathsf{str}(\sigma) = \mathsf{str}(\sigma')$. This observation guarantees that str' is well defined. In fact, henceforth we shall write str to denote both str and str'.

To go in the other direction let $F = (E, \leq, \lambda)$ be a trace over (Σ, I). Then $\rho \in E^\infty$ is called a linearization of F iff every $e \in E$ appears exactly once in ρ and, moreover, whenever $e, e' \in E$ and $e < e'$, e appears before e' in ρ.

As usual, we can extend the labelling function $\lambda : E \to \Sigma$ to words over E in a canonical way. If $\rho = e_0 e_1 \ldots$ is a word in E^∞ then $\lambda(\rho)$ denotes the corresponding word $\lambda(e_0)\lambda(e_1)\ldots$ in Σ^∞. We can now define the map $\mathsf{trs} : TR(\Sigma, I) \to \Sigma^\infty / \sim_I$ as follows:

$$\mathsf{trs}(F) = \{\lambda(\rho) \mid \rho \text{ is a linearization of } F\}.$$

Proposition 1.1

(i) *For every* $\sigma \in \Sigma^\infty$, $\mathsf{trs}(\mathsf{str}(\sigma)) = [\sigma]$.
(ii) *For every* $F \in \Sigma^\infty$, $\mathsf{str}(\mathsf{trs}(F)) = F$.

This result justifies our claim that $\Sigma^\infty / \sim_I$ and $TR(\Sigma, I)$ are indeed two equivalent ways of talking about the same class of objects.

In the poset representation of traces, finite configurations play the same role that finite prefixes do in sequences. Let $F = (E, \leq, \lambda)$ be a trace over (Σ, I). Then $c \subseteq E$ is a *configuration* iff c is finite and $\downarrow c = c$. We let $\mathcal{C}_F$ denote the set of configurations of F. Notice that $\emptyset$, the empty set, is a configuration. It is the least configuration under set inclusion. More importantly, $\downarrow e$ is a configuration for every e. These "pointed" configurations associated with the events are also called *prime configurations*. They constitute the building blocks for the Scott domains induced by traces [NPW]. We shall see that they also play a fundamental role in defining linear time temporal logics over traces.

We now turn our attention to distributed alphabets. Distributed alphabets can be viewed as "implementations" of trace alphabets. They form the basis for defining machine models with a built-in notion of independence which recognize trace languages.

Let $\mathcal{P}$ be a finite set of sequential agents called *processes*. A distributed alphabet is a family $\{\Sigma_p\}_{p\in\mathcal{P}}$ where Σ_p is a finite non-empty alphabet for each $p \in \mathcal{P}$. The idea is that whenever an action from Σ_p occurs, the agent p must participate in it. Hence the agents can constrain each other's behaviour, both directly *and* indirectly.

Trace alphabets and distributed alphabets are closely related to each other. Let $\widetilde{\Sigma} = \{\Sigma_p\}_{p\in\mathcal{P}}$ be a distributed alphabet. Then $\Sigma_\mathcal{P}$, the global alphabet associated with $\widetilde{\Sigma}$, is the collection $\bigcup_{p\in\mathcal{P}} \Sigma_p$. The distribution of $\Sigma_\mathcal{P}$ over $\mathcal{P}$ can be described using a *location function* $\mathrm{loc}_{\widetilde{\Sigma}} : \Sigma_\mathcal{P} \to 2^\mathcal{P}$ defined as follows:

$$\mathrm{loc}_{\widetilde{\Sigma}}(a) = \{p \mid a \in \Sigma_p\}.$$

This in turn induces the relation $I_{\widetilde{\Sigma}} \subseteq \Sigma_\mathcal{P} \times \Sigma_\mathcal{P}$ given by:

$$(a, b) \in I_{\widetilde{\Sigma}} \text{ iff } \mathrm{loc}_{\widetilde{\Sigma}}(a) \cap \mathrm{loc}_{\widetilde{\Sigma}}(b) = \emptyset.$$

Clearly $I_{\widetilde{\Sigma}}$ is irreflexive and symmetric and hence $(\Sigma_\mathcal{P}, I_{\widetilde{\Sigma}})$ is a trace alphabet. Thus every distributed alphabet canonically induces a trace alphabet. Two actions are independent according to $\widetilde{\Sigma}$ if they are executed by disjoint sets of processes. Henceforth, we write loc for $\mathrm{loc}_{\widetilde{\Sigma}}$ whenever $\widetilde{\Sigma}$ is clear from the context.

Going in the other direction there are, in general, many different ways to implement a trace alphabet as a distributed alphabet. A standard approach is to create a separate agent for each maximal D-clique generated by (Σ, I). Recall that a D-clique of (Σ, I) is a non-empty subset $p \subseteq \Sigma$ such that $p \times p \subseteq D$. Let $\mathcal{P}$ be the set of maximal D-cliques of (Σ, I). This set of processes induces the distributed alphabet $\widetilde{\Sigma} = \{\Sigma_p\}_{p\in\mathcal{P}}$ where $\Sigma_p = p$ for every process p. The alphabet $\widetilde{\Sigma}$ implements (Σ, I) in the sense that the canonical trace alphabet induced by it is exactly (Σ, I). In other words, $\Sigma_\mathcal{P} = \Sigma$ and $I_{\widetilde{\Sigma}} = I$.

For example, consider the trace alphabet (Σ, I) where $\Sigma = \{a, b, d\}$ and $I = \{(a, b), (b, a)\}$. The canonical D-clique implementation of (Σ, I) yields the distributed alphabet $\widetilde{\Sigma} = \{\{a, d\}, \{d, b\}\}$.

As mentioned earlier, distributed alphabets play a crucial role in the automata-theoretic aspects of trace theory. The fundamental result of Zielonka [Zie] says that every regular trace language over (Σ, I) can be recognized by an asynchronous automaton over a distributed alphabet $\widetilde{\Sigma}$ which implements (Σ, I). This result has been extended to ω-regular trace languages in terms of asynchronous Büchi automata by Gastin and Petit [GP].

Distributed alphabets arise naturally in a variety of models of distributed systems. In particular they are associated with the restricted but very useful model of a distributed system consisting of a network of sequential agents that coordinate their behaviour by performing common actions together. The linear time temporal logics that we consider in this paper will be based on distributed alphabets.

We conclude this section with a technical remark. Most of the theory of traces presented in this paper, including the automata-theoretic and logical aspects, constitutes a natural and conservative extension of the existing theory in the sequential setting. The sequential theory can almost always be recovered by setting $I = \emptyset$ when dealing with trace alphabets. Correspondingly, when dealing with distributed alphabets, the sequential case corresponds to having just one agent—i.e., $|\mathcal{P}| = 1$.

2 Automata over Infinite Traces

From now on we shall focus on infinite traces. With a little additional work most of the material we shall present on automata and logics can be extended to handle finite traces as well. Through the rest of this section we fix a distributed alphabet $\widetilde{\Sigma} = \{\Sigma_p\}_{p\in\mathcal{P}}$ with the induced trace alphabet (Σ, I), where $\Sigma = \bigcup_{p\in\mathcal{P}} \Sigma_p$ and $I = \{(a, b) \mid \mathrm{loc}(a) \cap \mathrm{loc}(b) = \emptyset\}$.

The terminology and notational conventions developed in the previous section are assumed here as well. We will be dealing with many $\mathcal{P}$-indexed families. For convenience we shall often write $\{X_p\}$ to denote the $\mathcal{P}$-indexed family $\{X_p\}_{p\in\mathcal{P}}$. A similar convention will be followed in dealing with Σ-indexed families: $\{Y_a\}$ will denote the family $\{Y_a\}_{a\in\Sigma}$.

Asynchronous Büchi automata, due to Gastin and Petit [GP], are the basic class of automata operating over infinite traces. They constitute a common generalization of the asynchronous automata of Zielonka [Zie] operating over finite traces and a mild variant of the classical Büchi automata operating over infinite sequences. We shall consider here a number of variants of asynchronous Büchi automata, each with a slightly different acceptance condition.

We begin with a brief and slightly non-standard presentation of Büchi automata. A word ω-automaton over Σ is a pair $\mathcal{B} = (TS, \mathcal{T})$ where

- $TS = (S, \{\rightarrow_a\}, S_{in})$ is a finite state transition system over Σ. In other words, S is a finite set of states, $\rightarrow_a \subseteq S \times S$ is an a-labelled transition relation for each $a \in \Sigma$ and $S_{in} \subseteq S$ is a set of initial states.
- $\mathcal{T}$ is an acceptance table accompanied by an acceptance condition.

Before considering a number of possibilities for $\mathcal{T}$, let us define the notion of a run. The Σ-indexed family of transition relations $\{\rightarrow_a\}$ induces a global transition relation $\rightarrow_{\mathcal{B}} \subseteq S \times \Sigma \times S$ given by $s \xrightarrow{a}_{\mathcal{B}} s'$ iff $(s, s') \in \rightarrow_a$. Where $\mathcal{B}$ is clear from the context $\rightarrow_{\mathcal{B}}$ will be written as $\rightarrow$.

Let $\sigma \in \Sigma^\omega$ (i.e., $\sigma : \omega \rightarrow \Sigma$ where, as usual, $\omega = \{0, 1, 2, \ldots\}$ is the set of natural numbers). A *run* of TS over σ is a map $\rho : \omega \rightarrow S$ such that $\rho(0) \in S_{in}$ and $\rho(i) \xrightarrow{\sigma(i)} \rho(i{+}1)$ for every $i \geq 0$.

The set of states encountered infinitely along the run ρ is denoted $\inf(\rho)$: $\inf(\rho) = \{s \mid \text{ for infinitely many } i, \rho(i) = s\}$.

Let us now consider just two of the various possibilities for $\mathcal{T}$.

(B0) $\mathcal{T} = F \subseteq S$.

A run ρ over σ is accepting with respect to B0 iff $\inf(\rho) \cap F \neq \emptyset$. We shall say that $\mathcal{A}$ is a B0-automaton if it uses B0 as its acceptance criterion. Of course, we shall also refer to these by their standard name; Büchi automata.

$L(\mathcal{A})$, the language accepted (recognized) by $\mathcal{A}$, is the set of infinite words σ such that there is an accepting run of $\mathcal{A}$ on σ. A language $L \subseteq \Sigma^\omega$ is said to be ω-regular iff there exists a Büchi automaton $\mathcal{A}$ over Σ such that $L(\mathcal{A}) = L$. As is well known, ω-regular languages have equivalent algebraic and logical presentations, as detailed in the excellent survey [Tho].

A second possibility for $\mathcal{T}$ is:

(B1) $\mathcal{T} \subseteq 2^S$.

A run ρ over σ is accepting with respect to B1 iff there exists $F \in \mathcal{T}$ such that $\inf(\rho) \supseteq F$. It is easy to show that $L \subseteq \Sigma^\omega$ is ω-regular iff there exists a B1-automaton $\mathcal{A}$ (i.e., an automaton $\mathcal{A}$ that uses B1 as its acceptance criterion) such that $L = L(\mathcal{A})$. Thus at the level of sequences there is no difference in expressive power between Büchi automata and B1-automata. As we shall see, at the level of traces, B0 is weaker than B1.

For defining automata on infinite traces we need to develop some notation. Let $F = (E, \leq, \lambda) \in TR(\Sigma, I)$. Then F is an infinite trace iff E is an infinite set. Let $TR^\omega(\Sigma, I)$ denote the subclass of infinite traces over (Σ, I). Often, we shall write TR^ω instead of $TR^\omega(\Sigma, I)$.

Let $F \in TR^\omega$ with $F = (E, \leq, \lambda)$ and let $p \in \mathcal{P}$. Then $e \in E$ is a p-event iff $\lambda(e) \in \Sigma_p$. Similarly, e is an a-event iff $\lambda(e) = a$. We let E_p denote the set of p-events and E_a denote the set of a-events.

There are two natural transition relations that one can associate with F. The event based transition relation $\Rightarrow_F \subseteq \mathcal{C}_F \times E \times \mathcal{C}_F$ is defined as $c \stackrel{e}{\Longrightarrow}_F c'$ iff $e \notin c$ and $c \cup \{e\} = c'$. The action-based transition relation $\rightarrow_F \subseteq \mathcal{C}_F \times \Sigma \times \mathcal{C}_F$ is defined as $c \stackrel{a}{\longrightarrow}_F c'$ iff there exists $e \in E$ such that $\lambda(e) = a$ and $c \stackrel{e}{\Longrightarrow}_F c'$.

To define automata on infinite traces, we have to first define a distributed version of transition systems. The distributed transition systems we work with here are essentially the *asynchronous automata* of Zielonka [Zie]. We begin with some notation involving local and global states.

Let $\mathcal{P}$ be a set of processes. We equip each process $p \in \mathcal{P}$ with a finite non-empty set of local p-states, denoted S_p. We set $S = \bigcup_{p \in \mathcal{P}} S_p$ and call S the set of *local states*.

We let P, Q range over non-empty subsets of $\mathcal{P}$ and let p, q range over $\mathcal{P}$. A Q-state is a map $s : Q \rightarrow S$ such that $s(q) \in S_q$ for every $q \in Q$. We let S_Q denote the set Q-states. We call $S_{\mathcal{P}}$ the set of *global states*.

If $Q' \subseteq Q$ and $s \in S_Q$ then $s_{Q'}$ is s restricted to Q'. In other words $s_{Q'}$ is the Q'-state s' which satisfies $s'(q') = s(q')$ for every q' in Q'. We use a to abbreviate $\mathrm{loc}(a)$ when talking about states (recall that $\mathrm{loc}(a) = \{p \mid a \in \Sigma_p\}$). Thus an a-state is just a $\mathrm{loc}(a)$-state and S_a denotes the set of all $\mathrm{loc}(a)$-states. If $\mathrm{loc}(a) \subseteq Q$ and s is a Q-state we shall write s_a to mean $s_{\mathrm{loc}(a)}$.

A *distributed transition system* TS over $\widetilde{\Sigma}$ is a structure $(\{S_p\}, \{\rightarrow_a\}, S_{in})$ where

- S_p is a finite non-empty set of p-states for each process p.
- For $a \in \Sigma$, $\rightarrow_a \subseteq S_a \times S_a$ is a transition relation between a-states.
- $S_{in} \subseteq S_{\mathcal{P}}$ is a set of initial global states.

The idea is that an a-move by TS involves only the local states of the agents which participate in the execution a. This is reflected in the global transition relation $\rightarrow_{TS} \subseteq S_{\mathcal{P}} \times \Sigma \times S_{\mathcal{P}}$ which is defined as:

$$s \xrightarrow{a}_{TS} s' \text{ iff } (s_a, s'_a) \in \rightarrow_a \text{ and } s_{\mathcal{P}-\mathrm{loc}(a)} = s'_{\mathcal{P}-\mathrm{loc}(a)}.$$

From the definition of $\rightarrow_{TS}$, it is clear that actions which are executed by disjoint sets of agents are processed independently by TS.

A trace ω-automaton over $\widetilde{\Sigma} = \{\Sigma_p\}$ is a pair $\mathcal{A} = (TS, \mathcal{T})$ where $TS = (\{S_p\}, \{\rightarrow_a\}, S_{in})$ is a distributed transition system over $\widetilde{\Sigma}$ and $\mathcal{T}$ is an acceptance table (which we will elaborate on later).

A *trace run* of TS over $F \in TR^\omega$ is a map $\rho : \mathcal{C}_F \rightarrow S_{\mathcal{P}}$ such that $\rho(\emptyset) \in S_{in}$ and for every $(c, a, c') \in \rightarrow_F$, $\rho(c) \xrightarrow{a}_{TS} \rho(c')$.

To define acceptance we must now compute $\inf_p(\rho)$, the set of p-states that are encountered infinitely often along ρ. The obvious definition, namely $\inf_p(\rho) = \{s_p \mid \rho(c)(p) = s_p \text{ for infinitely many } c \in \mathcal{C}_F\}$, will not work. The complication arises because some processes may make only finitely many moves, even though the overall trace consists of an infinite number of events.

For instance, consider the distributed alphabet $\widetilde{\Sigma}_0 = \{\{a\}, \{b\}\}$. In the corresponding distributed transition system, there are two processes p and q which execute a's and b's completely independently. Consider the trace $F = (E, \leq, \lambda)$ where $|E_p| = 1$ and E_q is infinite—i.e., all the infinite words in $\mathsf{trs}(F)$ contain one a and infinitely many b's. Let s_p be the state of p after executing a. Then, there will be infinitely many configurations whose p-state is s_p, even though p only moves a finite number of times.

Continuing with the same example, consider another infinite trace $F' = (E', \leq', \lambda')$ over the same alphabet where both E_p and E_q are infinite. Once again, let s_p be the local state of p after reading one a. Further, let us suppose that after reading the second a, p never returns to the state s_p. It will still be the case that there are infinitely many configurations whose p-state is s_p: consider the configurations $c_0, c_1, c_2, \ldots$ where c_j is the finite configuration after one a and j b's have occurred.

So, we have to define $\inf_p(\rho)$ carefully in order to be able to distinguish whether or not process p is making progress. The appropriate formulation is as follows:

Case 1 *E_p is finite:* $\inf_p(\rho) = \{s_p\}$, where $\rho(\downarrow E_p) = s$ and $s_p = s(p)$.
Case 2 *E_p is an infinite set:*
$$\inf_p(\rho) = \{s_p \mid \text{for infinitely many } e \in E_p, s_e(p) = s_p, \text{ where } \rho(\downarrow e) = s_e\}.$$

We can now begin to consider various acceptance tables.

(A0) $\mathcal{T} = \{F_p\}$ with $F_p \subseteq S_p$ for each p.

A run ρ over F is accepting with respect to A0 iff $\inf_p(\rho) \cap F_p \neq \emptyset$ for every p. The trace language accepted by the A0-automaton $\mathcal{A}$ (i.e., where $\mathcal{T}$ is of the form A0) is the set $L_{Tr}(\mathcal{A}) = \{F \mid \exists$ an accepting run of TS over $F\}$. A0-automata are the obvious common generalization of asynchronous automata and Büchi automata. It turns out that A0-automata are not expressive enough: the acceptance criterion cannot distinguish whether or not an agent executes infinitely many actions.

To bring this out and to motivate the acceptance condition we are after, we will put down a crude definition of ω-regular trace languages.

A trace language over $\widetilde{\Sigma}$ is just a subset of TR^ω. To define ω-regular trace languages, we exploit the result from the previous section linking $\Sigma^\infty / \sim_I$ and $TR(\Sigma, I)$ which permits us to associate a language of infinite words with each trace language. We can then transport the definition of ω-regularity from subsets of Σ^ω to infinite traces.

Let $L \subseteq \Sigma^\omega$. Then L is I-consistent iff for every $\sigma \in \Sigma^\omega$, if $\sigma \in L$ then $[\sigma] \subseteq L$. Thus if L is I-consistent either all members of the $\sim_I$-equivalence class $[\sigma]$ are in L or none of them are in L.

Let $L' \subseteq TR^\omega$. We say that L' is an ω-regular trace language iff there exists an I-consistent ω-regular language $L \subseteq \Sigma^\omega$ such that $L' = \{\mathsf{str}(\sigma) \mid \sigma \in L\}$. Stated differently, $L' \subseteq TR^\omega$ is a ω-regular trace language iff $L = \bigcup\{\mathsf{trs}(F) \mid F \in L'\}$ is a ω-regular subset of Σ^ω. As in the word case, algebraic and logical presentations of ω-regular trace languages have been worked out [EM, GP]. These presentations have a flavour which is pleasingly similar to the classical algebraic and logical characterisations of ω-regular subsets of Σ^ω.

Returning to the distributed alphabet $\widetilde{\Sigma}_0 = (\{a\}, \{b\})$, let (Σ_0, I_0) denote the corresponding trace alphabet. Consider $L \subseteq TR^\omega(\Sigma_0, I_0)$ consisting of the single trace $F = (E, \leq, \lambda)$ such that E_a and E_b are both infinite sets. It is easy to check that L is a ω-regular trace language but, as argued in [GP], no A0-automaton over $\widetilde{\Sigma}$ can recognize L.

It is worth noting that having multiple entries in the acceptance table does not help. In other words, one might consider the following acceptance criterion.

(A0′) $\mathcal{T} = \{\mathcal{T}_0, \mathcal{T}_1, \ldots, \mathcal{T}_n\}$ with $\mathcal{T}_i = \{F_p^i\}_{p \in \mathcal{P}}$ and $F_p^i \subseteq S_p$ for each $i \in \{1, 2, \ldots, n\}$ and each $p \in \mathcal{P}$. A run ρ of TS over $F \in TR^\omega$ is accepting with respect to A0′ iff there exists i such that $\inf_p(\rho) \cap F_p^i \neq \emptyset$ for each p.

The reason why A0′ does not help is that the class of languages accepted by A0-automata is closed under union, thanks to the presence of multiple global initial states. We can construct an A0-automaton $\mathcal{A}_i = (TS, \mathcal{T}_i)$ for each entry $\mathcal{T}_i$ from the table of an A0′-automaton $\mathcal{A} = (TS, \mathcal{T})$. If $\mathcal{T} = \{\mathcal{T}_0, \mathcal{T}_1, \ldots, \mathcal{T}_n\}$, it is clear that $L(\mathcal{A}) = \bigcup_{i \in \{1,2,\ldots,n\}} L(\mathcal{A}_i)$. Thus, every A0′-automaton can be simulated by an A0-automaton.

Gastin and Petit showed that the following acceptance condition provides a suitable generalization of classical Büchi automata to the setting of infinite traces.

(A1) $\mathcal{T} = \{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_n\}$ with $\mathcal{T}_i = \{F_p^i\}_{p \in \mathcal{P}}$ and $F_p^i \subseteq S_p$ for each $i \in \{1, 2, \ldots, n\}$ and each $p \in \mathcal{P}$. A run ρ of TS over $F \in TR^\omega$ is accepting with respect to A1 iff there exists i such that $\inf_p(\rho) \supseteq F_p^i$ for each p.

The condition A1 is an extension of the sequential condition B1 in a distributed setting. Notice that A1 "couples" together final sets of the components in each entry $\mathcal{T}_i \in \mathcal{T}$.

Theorem 2.1 ([GP]) *$L \subseteq TR^\omega$ is a ω-regular trace language iff there exists an A1-automaton $\mathcal{A}$ such that $L_{Tr}(\mathcal{A}) = L$.*

Subsequently, Niebert has shown that the A1 condition can be modified to avoid coupling final sets across processes [Nie]. In effect, it is possible to have a local B1 table for each process and define a run ρ to be accepting if for each process p, $\inf_p(\rho)$ satisfies p's B1 table. Going one step further, we arrive at the acceptance criterion A2, which is the one we will use in connection with the logics to be studied in the next section.

(A2) $\mathcal{T} = \{(F_p^\omega, F_p)\}_{p \in \mathcal{P}}$ with F_p^ω, $F_p \subseteq S_p$ for each p.

A run ρ over $F = (E, \leq, \lambda)$ is accepting with respect to A2 iff for each process p the following conditions are met.

Case 1 *E_p is finite*: Then $\inf_p(\rho) \cap F_p \neq \emptyset$.

Case 2 *E_p is an infinite set*: Then $\inf_p(\rho) \cap F_p^\omega \neq \emptyset$.

Thus, on an input F, the decision as to whether a process p uses F_p or F_p^ω to determine acceptance depends on whether or not p executes infinitely many actions in F.

Theorem 2.2

(i) *The class of languages accepted by A2-automata is closed under union.*

(ii) *The class of languages accepted by A1-automata is identical to the class of languages accepted by A2-automata.*

Proof Sketch.

(i) Suppose $\mathcal{A}_1$ and $\mathcal{A}_2$ are two A2-automata. Then we construct an A2-automaton $\mathcal{A}$ which is the *disjoint* union of $\mathcal{A}_1$ and $\mathcal{A}_2$. The global initial states of $\mathcal{A}$ will determine for each run whether $\mathcal{A}_1$ or $\mathcal{A}_2$ (but not both!) is going to be explored. It is easy to check that $L(\mathcal{A}) = L(\mathcal{A}_1) \cup L(\mathcal{A}_2)$.

(ii) Let $\mathcal{A} = (TS, \mathcal{T})$ be an A1-automaton. From part (i), it suffices to consider the case where $\mathcal{T}$ has just one entry. So assume that $\mathcal{T} = \{\mathcal{T}_1\}$ and $\mathcal{T}_1 = \{F_p\}$. Let $TS = (\{S_p\}, \{\rightarrow_a\}, S_{in})$. Define the A2-automaton $\mathcal{A}' = (TS', \mathcal{T}')$ as follows. $TS' = (\{S'_p\}, \{\Rightarrow_a\}, S'_{in})$ where:

- $S'_p = S_p \times 2^{F_p} \times \{\mathsf{on}, \mathsf{off}\}$ for each p.
- Let s'_a, t'_a be a-states in TS' such that $s'_a(p) = (s_p, X_p, u_p)$ and $t'_a(p) = (t_p, Y_p, v_p)$ for each $p \in \mathcal{P}$. Then $(s'_a, t'_a) \in \Rightarrow_a$ iff there exists $(s_a, t_a) \in \rightarrow_a$ such that the following conditions are satisfied for each $p \in \text{loc}(a)$.

  (1) $u_p = \mathsf{on}$, $s_p = s_a(p)$ and $t_p = t_a(p)$.

(2) If $X_p = \emptyset$ then $Y_p = F_p$. Otherwise, $Y_p = X_p - \{t_p\}$.

- $S'_{in} = \{s' \in S'_{\mathcal{P}} \mid \exists s \in S_{in}.\ \forall p \in \mathcal{P}.\ \exists u_p \in \{\mathsf{on}, \mathsf{off}\}.\ s'(p) = (s(p), \emptyset, u_p)\}$
- $\mathcal{T}' = \{(G_p^\omega, G_p)\}$ where for each p,

$$\begin{aligned} G_p^\omega &= S_p \times \{\emptyset\} \times \{\mathsf{on}\} \\ G_p &= F_p \times 2^{F_p} \times \{\mathsf{off}\} \end{aligned}$$

It is easy to check that $L_{Tr}(\mathcal{A}) = L_{Tr}(\mathcal{A}')$.

Conversely, let $\mathcal{A} = (TS, \mathcal{T})$ be an A2-automaton with $TS = (\{S_p\}, \{\longrightarrow_a\}, S_{in})$ and $\mathcal{T} = \{(F_p^\omega, F_p)\}$. We say that $\mathcal{A}$ is in standard form if it satisfies:

- $F_p^\omega \cap F_p = \emptyset$ for each p.
- If $(s_a, t_a) \in \longrightarrow_a$ and $p \in \mathrm{loc}(a)$, then $s_a(p) \notin F_p$.

Thus, if $\mathcal{A}$ is in standard form, the p-states in F_p are "dead" and are disjoint from F_p^ω. It is a simple exercise to verify that every A2-automaton $\mathcal{A}$ can be converted to an A2-automaton $\mathcal{A}'$ in standard form such that $L_{Tr}(\mathcal{A}) = L_{Tr}(\mathcal{A}')$.

So, let $\mathcal{A} = (TS, \mathcal{T})$ be an A2-automaton in standard form with $TS = (\{S_p\}, \{\longrightarrow_a\}, S_{in})$ and $\mathcal{T} = \{(F_p^\omega, F_p)\}$. Let G be the set of functions of the form $g : \mathcal{P} \longrightarrow S$ such that $g(p) \in F_p^\omega \cup F_p$ for each p. Define the A1-automaton $\mathcal{A}' = (TS', \mathcal{T}')$ where $TS' = TS$ and $\mathcal{T}' = \{\mathcal{T}'_g\}_{g \in G}$, such that for each $g \in G$, $\mathcal{T}'_g = \{\{g(p)\}\}_{p \in \mathcal{P}}$. It is easy to verify that $L_{Tr}(\mathcal{A}) = L_{Tr}(\mathcal{A}')$. $\square$

We now argue that the emptiness problem for A2-automata is decidable. This will be required to settle the satisfiability problem for the logics considered in the next section. Let $\mathcal{A} = (TS, \mathcal{T})$ be an A2-automaton be with $TS = (\{S_p\}, \{\longrightarrow_a\}, S_{in})$ and $\mathcal{T} = \{(F_p^\omega, F_p)\}$. Though it is not strictly necessary, it will be illuminating to first associate a language of infinite words with $\mathcal{A}$.

Let $\sigma \in \Sigma^\omega$. Then a (word) run of TS over σ is a map $\rho : \omega \longrightarrow S_{\mathcal{P}}$ such that $\rho(0) \in S_{in}$ and $\rho(i) \xrightarrow{\sigma(i)}_{TS} \rho(i{+}1)$ for each $i \geq 0$. The run ρ over σ is accepting iff the following conditions are satisfied for each p.

(i) If $i \in \omega$ such that $\sigma(j) \notin \Sigma_p$ for every $j \geq i$ then $s_i(p) \in F_p$, where $s_i = \rho(i)$.
(ii) If $\sigma(j) \in \Sigma_p$ for infinitely many j then for infinitely many i it is the case that $s_i(p) \in F_p^\omega$, where $s_i = \rho(i)$.

We define $L_{seq}(\mathcal{A})$, the language of infinite words accepted by $\mathcal{A}$ to be the set of all words σ such that there exists an accepting run of $\mathcal{A}$ over σ. The distributed nature of TS together with the basic properties of the maps str and trs defined earlier lead to the next result.

Theorem 2.3 *For any A2-automaton $\mathcal{A}$, $L_{Tr}(\mathcal{A}) = \{\mathsf{str}(\sigma) \mid \sigma \in L_{seq}(\mathcal{A})\}$. Consequently $L_{Tr}(\mathcal{A}) \neq \emptyset$ iff $L_{seq}(\mathcal{A}) \neq \emptyset$.*

Similar statements hold, of course, for A0-automata and A1-automata.

All the A2-automata that we construct in the next section will be in standard form. So assume that $\mathcal{A} = (TS, \mathcal{T})$ is an A2-automaton in standard form with $TS = (\{S_p\}, \{\rightarrow_a\}, S_{in})$ and $\mathcal{T} = \{(F_p^\omega, F_p)\}$. Construct the directed graph $G_\mathcal{A} = (S_\mathcal{P}, E_\mathcal{A})$ where $S_\mathcal{P}$ is the set of global states of TS and $(s, s') \in E_\mathcal{A}$ if there exists $a \in \Sigma$ such that $s \xrightarrow{a}_{TS} s'$. We also label each edge in $G_\mathcal{A}$ with a set of processes. Let $\pi : E_\mathcal{A} \rightarrow 2^\mathcal{P}$ be given by $\pi((s, s')) = \bigcup\{\text{loc}(a) \mid s \xrightarrow{a}_{TS} s'\}$.

We call $X \subseteq S_\mathcal{P}$ a *good component* iff X is a maximal strongly connected component in $G_\mathcal{A}$ which meets one the following conditions for each p.

(i) There exists $s \in X$ such that $s(p) \in F_p$. (Because $\mathcal{A}$ is in standard form this implies that $s'(p) = s''(p) \in F_p$ for every $s', s'' \in X$).

(ii) There exists $s \in X$ such that $s(p) \in F_p^\omega$ and for some $s' \in X$, $(s', s) \in E_\mathcal{A}$ and $p \in \pi((s', s))$.

From Theorem 2.3 we know that $L_{Tr}(\mathcal{A})$ is non-empty iff $L_{seq}(\mathcal{A})$ is. It is not difficult to prove that $L_{seq}(\mathcal{A})$ is non-empty iff $G_\mathcal{A}$ has a good component. It is known that the maximal strongly connected components of a digraph can be computed in time which is linear in the size of the digraph [AHU]. Clearly, the size of $G_\mathcal{A}$ is bounded by the number of global states of $\mathcal{A}$. As a consequence it is easy to derive the next result.

Theorem 2.4 *Let $\mathcal{A}$ be an A2-automaton in standard form. Then $L_{Tr}(\mathcal{A}) \neq \emptyset$ iff $G_\mathcal{A}$ has a good component. For $p \in \mathcal{P}$, let $n_p = |S_p|$ denote the number of p-states. Let $n = \max\{n_p\}_{p \in \mathcal{P}}$ and $m = |\mathcal{P}|$. Then checking that $G_\mathcal{A}$ has a good component can be done in time $O(m^{2n})$.*

We conclude this section with a few remarks on deterministic automata over infinite traces. As with automata on infinite words, non-deterministic A2-automata on infinite traces are strictly more expressive than deterministic A2-automata. In the absence of determinacy, complementation is difficult. When applying these automata to settle questions in logic, complementation is often required to handle negation in formulas. (Fortunately, the automata-theoretic treatment of linear time temporal logic on traces which we will describe here does *not* require complementation.)

To obtain determinacy without loss of expressive power one must use a more sophisticated acceptance criterion corresponding to the Muller, Rabin or Streett acceptance conditions for infinite words. Here, we will look only at the Muller acceptance condition.

(M) $\mathcal{T} = \{\mathcal{T}_1, \ldots, \mathcal{T}_n\}$ with $\mathcal{T}_i = \{F_p^i\}$ and $F_p^i \subseteq S_p$ for each i and each p. A run ρ over $F \in TR^\omega$ is accepting with respect to M iff there exists $\mathcal{T}_i \in \mathcal{T}$ such that $\inf_p(\rho) = F_p^i$ for each p.

Diekert and Muscholl [DM] showed that *deterministic* M-automata are as expressive as non-deterministic A1-automata. Their proof however does not lead to a determinization construction for A1-automata.

There are two independent solutions available in the literature for the difficult problem of complementing A1-automata. Muscholl first showed how to directly construct a non-deterministic A1-automaton which is the complement of the given automaton [Mus]—this approach does not yield a determinization construction for A1-automata. In [Mus] the complementation is carried out for asynchronous *cellular* Büchi automata, in which there is one agent for each letter. To transport this complementation result to A1-automata, one has to resort to a simulation which carries non-trivial overheads in the size of the alphabet. The second solution due to Klarlund, Mukund and Sohoni [KMS] is a direct determinization construction for A1-automata which then easily leads to the complementation result. In both cases, the blow-up in the local state space of each process is exponential in the global state space of the original automaton, which is essentially optimal. Surprisingly in both [Mus] and [KMS], the A1 acceptance condition must be first transformed into an equivalent one which describes in considerable detail the communication patterns established by the infinite trace that is being examined for acceptance.

3 Linear Time Temporal Logics over Traces

A variety of linear time temporal logics to be interpreted over traces have been proposed in the literature. As mentioned in the Introduction, our focus here will be on those logics which meet the following criteria:

(i) The logic should be expressible within the first order theory of traces.
(ii) The logic should admit a treatment in terms of asynchronous Büchi automata of one kind or the other.

We begin with the logic TrPTL (Trace based Propositional Temporal logic of Linear time). This is the earliest and—to date—the most expressive linear time logic of the chosen kind. For a detailed treatment of this logic the reader is referred to [Thi1]. After presenting TrPTL we will consider two subsystems denoted $\mathrm{TrPTL}^{\mathrm{con}}$ (connected TrPTL) and $\mathrm{TrPTL}^{\otimes}$ (product TrPTL). These subsystems are obtained by placing suitable syntactic restrictions on the formulas. The interesting point is that these restrictions result in proportionate simplification of the automata theoretic constructions associated with the logics. Towards the end of the section we will take a quick look at other temporal logics that have been proposed with traces as the underlying frames.

Henceforth, it will be notationally convenient to deal with distributed alphabets in which the names of the processes are positive integers. Through this section and the next, we fix a distributed alphabet $\widetilde{\Sigma} = \{\Sigma_i\}_{i \in \mathcal{P}}$ with $\mathcal{P} = \{1, 2, \ldots, K\}$ and $K \geq 1$. We let i, j and k range over $\mathcal{P}$. As before, let P, Q range over non-empty subsets of $\mathcal{P}$. The trace alphabet induced by $\widetilde{\Sigma}$ is denoted (Σ, I). We assume the terminology and notations developed in the previous sections. In particular when dealing with a $\mathcal{P}$-indexed family $\{X_i\}_{i \in \mathcal{P}}$, we will often write just $\{X_i\}$.

The logic TrPTL is parameterized by the class of distributed alphabets. Having fixed $\widetilde{\Sigma}$ we shall often almost always write TrPTL to mean TrPTL($\widetilde{\Sigma}$), the logic associated with $\widetilde{\Sigma}$. Fix a set of atomic propositions AP with p, q ranging over AP. Then $\Phi_{\mathrm{TrPTL}(\widetilde{\Sigma})}$, the set of formulas of TrPTL($\widetilde{\Sigma}$), is defined inductively via:

- For $p \in AP$ and $i \in \mathcal{P}$, $p(i)$ is a formula (which is to be read "p at i").
- If α and β are formulas, so are $\neg\alpha$ and $\alpha \vee \beta$.
- If α is a formula and $a \in \Sigma_i$ then $\langle a \rangle_i \alpha$ is a formula.
- If α and β are formulas so is $\alpha\, \mathcal{U}_i \beta$.

From now on, we denote $\Phi_{\mathrm{TrPTL}(\widetilde{\Sigma})}$ as just Φ. In the semantics of the logic which will be based on infinite traces, the i-view of a configuration will play a crucial role. Let $F \in TR^\omega$ with $F = (E, \leq, \lambda)$. Recall that $E_i = \{e \mid e \in E$ and $\lambda(e) \in \Sigma_i\}$. Let $c \in \mathcal{C}_F$ and $i \in \mathcal{P}$. Then $\downarrow^i(c)$ is the i-view of c and it is defined as:

$$\downarrow^i(c) = \downarrow(c \cap E_i).$$

We note that $\downarrow^i(c)$ is also a configuration. It is the "best" configuration that the agent i is aware of at c. We say that $\downarrow^i(c)$ is an *i-local configuration*. Let $\mathcal{C}_F^i = \{\downarrow^i(c) \mid c \in \mathcal{C}_F\}$ be the set of i-local configurations. For $Q \subseteq \mathcal{P}$ and $c \in \mathcal{C}_F$, we let $\downarrow^Q(c)$ denote the set $\bigcup\{\downarrow^i(c) \mid i \in Q\}$. Once again, $\downarrow^Q(c)$ is a configuration. It represents the collective knowledge of the processes in Q about the configuration c.

The following basic properties of traces follow directly from the definitions.

Proposition 3.1 *Let $F = (E, \leq, \lambda)$ be an infinite trace. The following statements hold.*

(i) *Let $\leq_i = \leq \cap (E_i \times E_i)$. Then $(E_i, \leq_i)$ is a linear order isomorphic to ω if E_i is infinite and isomorphic to a finite initial segment of ω if E_i is finite.*
(ii) *$(\mathcal{C}_F^i, \subseteq)$ is a linear order. In fact $(\mathcal{C}_F^i - \{\emptyset\}, \subseteq)$ is isomorphic to $(E_i, \leq_i)$.*
(iii) *Suppose $\downarrow^i(c) \neq \emptyset$ where $c \in \mathcal{C}_F$. Then there exists $e \in E_i$ such that $\downarrow^i(c) = \downarrow e$. In fact e is the $\leq_i$-maximum event in $(c \cap E_i)$.*
(iv) *Suppose $Q \subseteq Q' \subseteq \mathcal{P}$ and $c \in \mathcal{C}_F$. Then $\downarrow^Q(c) = \downarrow^Q(\downarrow^{Q'}(c))$. In particular, for a single process i, $\downarrow^i(c) = \downarrow^i(\downarrow^i(c))$.*

We can now present the semantics of TrPTL. A model is a pair $M = (F, \{V_i\}_{i \in p})$ where $F = (E, \leq, \lambda) \in TR^\omega$ and $V_i : \mathcal{C}_F^i \rightarrow 2^{AP}$ is a valuation function which assigns a set of atomic propositions to i-local configurations for each process i. Let $c \in \mathcal{C}_F$ and $\alpha \in \Phi$. Then $M, c \models \alpha$ denotes that α is satisfied at c in M and it is defined inductively as follows:

- $M, c \models p(i)$ for $p \in AP$ iff $p \in V_i\ (\downarrow^i(c))$.
- $M, c \models \neg\alpha$ iff $M, c \not\models \alpha$.
- $M, c \models \alpha \vee \beta$ iff $M, c \models \alpha$ or $M, c \models \beta$

- $M, c \models \langle a \rangle_i \alpha$ iff there exists $e \in E_i - c$ such that $\lambda(e) = a$ and $M, \downarrow e \models \alpha$. Moreover, for every $e' \in E_i$, $e' < e$ iff $e' \in c$.
- $M, c \models \alpha\, \mathcal{U}_i \beta$ iff there exists $c' \in \mathcal{C}_F$ such that $c \subseteq c'$ and $M, \downarrow^i(c') \models \beta$. Moreover, for every $c'' \in \mathcal{C}_F$, if $\downarrow^i(c) \subseteq \downarrow^i(c'') \subset \downarrow^i(c')$ then $M, \downarrow^i(c'') \models \alpha$.

Thus TrPTL is an action based agent-wise generalization of LTL. Indeed both in terms of its syntax and semantics, LTL corresponds to the case where there is only one agent and where this agent can execute only one action at any time. With $\mathcal{P} = \{1\}$ and $\Sigma_1 = \{a_0\}$ one then writes p instead of $p(1)$, $O\alpha$ instead of $\langle a_0 \rangle \alpha$ and $\alpha\, \mathcal{U} \beta$ instead of $\alpha\, \mathcal{U}_i \beta$. The semantics of TrPTL when specialized down to this case yields the usual LTL semantics. In the next section we will say more about the relationship between TrPTL and LTL.

Returning to TrPTL, the assertion $p(i)$ says that the i-view of c satisfies the atomic proposition p. Observe that we could well have $p(i)$ satisfied at c but not $p(j)$ (with $i \neq j$). It is interesting to note that all atomic assertions (that we know of) concerning distributed behaviours are local in nature. Indeed, it is well-known that global atomic propositions will at once lead to an undecidable logic in the current setting [LPRT, Pen].

Suppose $M = (F, \{V_i\})$ is a model and $c \xrightarrow{a}_F c'$ with $j \notin \mathrm{loc}(a)$. Then $M, c \models p(j)$ iff $M, c' \models p(j)$. In this sense the valuation functions are local. There are, of course, a number of equivalent ways of formulating this idea which we will not get into here.

The assertion $\langle a \rangle_i \alpha$ says that the agent i will next participate in an a-event. Moreover, at the resulting i-view, the assertion α will hold. The assertion $\alpha\, \mathcal{U}_i \beta$ says that there is a future i-view (including the present i-view) at which β will hold and for all the intermediate i-views (if any) starting from the current i-view, the assertion α will hold.

Before considering examples of TrPTL specifications, we will introduce some notation. We let α, β with or without subscripts range over Φ. Abusing notation, we will use loc to denote the map which associates a set of locations with each formula.

- $\mathrm{loc}(p(i)) = \mathrm{loc}(\langle a \rangle_i \alpha) = \mathrm{loc}(\alpha\, \mathcal{U}_i \beta) = \{i\}$.
- $\mathrm{loc}(\neg \alpha) = \mathrm{loc}(\alpha)$.
- $\mathrm{loc}(\alpha \vee \beta) = \mathrm{loc}(\alpha) \cup \mathrm{loc}(\beta)$.

In what follows, $\Phi^i = \{\alpha \mid \mathrm{loc}(\alpha) = \{i\}\}$ is the set of i-type formulas. A basic observation concerning the semantics of TrPTL can be phrased as follows:

Proposition 3.2 *Let $M = (F, \{V_i\})$ be a model, $c \in \mathcal{C}_F$ and α a formula such that* $\mathrm{loc}(\alpha) \subseteq Q$. *Then $M, c \models \alpha$ iff $M, \downarrow^Q(c) \models \alpha$.*

A corollary to this result is that in case $\alpha \in \Phi^i$ then $M, c \models \alpha$ iff $M, \downarrow^i(c) \models \alpha$. As a result, the formulas in Φ^i can be used in exactly the same manner as one would use LTL (in the setting of sequences) to express properties of the agent i. Boolean combinations of such local assertion can be used to capture various interaction patterns between the agents implied by the logical connectives as well as the coordination enforced by the distributed alphabet $\widetilde{\Sigma}$.

For writing specifications, apart from the usual derived connectives of propositional calculus such as $\wedge$, $\Rightarrow$ and $\equiv$, the following operators are also available

- $\top \triangleq p_1(1) \vee \neg p_1(1)$ denotes the constant "True", where $AP = \{p_1, p_2, \ldots\}$. We use $\bot = \neg\top$ to denote "False". .
- $\Diamond_i \alpha \triangleq \top\, \mathcal{U}_i \alpha$ is a local version of the $\Diamond$ modality of LTL.
- $\Box_i \alpha \triangleq \neg\Diamond_i \neg\alpha$ is a local version of the $\Box$ modality of LTL..
- Let $X \subseteq \Sigma_i$ and $\overline{X} = \Sigma_i - X$. Then $\alpha\, \mathcal{U}_i^X \beta \triangleq (\alpha \wedge \bigwedge_{a\in\overline{X}} [a]_i \bot)\, \mathcal{U}_i \beta$. In other words $\alpha\, \mathcal{U}_i^X \beta$ is fulfilled using (at most) actions taken from X. We set $\Diamond_i^X \alpha \triangleq \top\, \mathcal{U}_i^X \alpha$ and $\Box_i^X \alpha \triangleq \neg\Diamond_i^X \neg\alpha$.
- $\alpha(i) \triangleq \alpha\, \mathcal{U}_i \alpha$ (or equivalently $\bot\, \mathcal{U}_i \alpha$). $\alpha(i)$ is to be read as "α at i". If $M = (F, \{V_i\})$ is a model and $c \in \mathcal{C}_F$ then $M, c \models \alpha(i)$ iff $M, \downarrow^i(c) \models \alpha$. It could of course be the case that $\mathrm{loc}(\alpha) \neq \{i\}$.

A simple but important observation is that every formula is a boolean combination of formulas taken from $\bigcup_{i\in\mathcal{P}} \Phi^i$. In TrPTL we can say that a specific global configuration is reachable from the initial configuration. Let $\{\alpha_i\}_{i\in\mathcal{P}}$ be a family with $\alpha_i \in \Phi^i$ for each i. Then we can define a derived connective $\Diamond(\alpha_1, \alpha_2, \ldots, \alpha_K)$ which has the following semantics at the empty configuration. Let $M = (F, \{V_i\})$ be a model. Then $M, \emptyset \models \Diamond(\alpha_1, \alpha_2, \ldots, \alpha_k)$ iff there exists $c \in \mathcal{C}_F$ such that $M, c \models \alpha_1 \wedge \alpha_2 \wedge \cdots \wedge \alpha_K$.

To define this derived connective set $\Sigma_1' = \Sigma_1$ and, for $1 < i \leq K$, set $\Sigma_i' = \Sigma_i - \cup\{\Sigma_j \mid 1 \leq j < i\}$. Then $\Diamond(\alpha_1, \alpha_2, \ldots, \alpha_K)$ is the formula:

$$\Diamond_1^{\Sigma_1'}(\alpha_1 \wedge \Diamond_2^{\Sigma_2'}(\alpha_2 \wedge \Diamond_3^{\Sigma_3'}(\alpha_3 \wedge \cdots \Diamond_K^{\Sigma_K'} \alpha_K)) \cdots).$$

The idea is that the sequence of actions leading up to the required configuration can be reordered so that one first performs all the actions in Σ_1, then all the actions in $\Sigma_2 - \Sigma_1$ etc. Hence, if now is an atomic proposition, the formula $\Diamond(\mathsf{now}(1), \mathsf{now}(2), \ldots, \mathsf{now}(K))$ is satisfied at the empty configuration iff there is a reachable configuration at which all the agents assert now.

Dually, safety properties that hold at the initial configuration can also be expressed. For example, let $\mathsf{crt}(i)$ be the atomic assertion declaring that the agent i is currently in its critical section. Then it is possible to write a formula φ_{ME} which asserts that at all reachable configurations at most one agent is in its critical section, thereby guaranteeing that the system satisfies the mutual exclusion property. We omit the details of how to specify φ_{ME}.

On the other hand, it seems difficult to express nested global and safety properties in TrPTL. This is mainly due to the local nature of the modalities which results in information about the past sneaking into the semantics even though there are no explicit past operators in the logic. In particular, TrPTL admits formulas that are satisfiable but not root-satisfiable.

A formula α is said to be root-satisfiable iff there exists a model M such that $M, \emptyset \models \alpha$. On the other hand, α is said to be satisfiable iff there exists a model

$M = (F, \{V_i\})$ and $c \in \mathcal{C}_F$ such that $M, c \models \alpha$. It turns out that these two notions are not equivalent. Consider the distributed alphabet $\widetilde{\Sigma}_0 = \{\Sigma_1, \Sigma_2\}$ with $\Sigma_1 = \{a, d\}$ and $\Sigma_2 = \{b, d\}$. Then it is not difficult to verify that the formula $p(2)(1) \wedge \Box_2 \neg p(2)$ is satisfiable but not root-satisfiable. (Recall that $p(2)(1)$ abbreviates $\bot\, \mathcal{U}_1 p(2)$). One can however transform every formula α into a formula α' such that α is satisfiable iff α' is root satisfiable.

This follows from the observation that every α can be expressed as a boolean combination of formulas taken from the set $\bigcup_{i \in \mathcal{P}} \Phi^i$. Hence the given formula α can be assumed to be of the form $\alpha = \bigvee_{j=1}^{m} (\alpha_{j1} \wedge \alpha_{j2} \wedge \cdots \wedge \alpha_{jK})$ where $\alpha_{ji} \in \Phi^i$ for each $j \in \{1, 2, \ldots, m\}$ and each $i \in \mathcal{P}$. Now convert α to the formula α' where $\alpha' = \bigvee_{j=1}^{m} \Diamond(\alpha_{j1}, \alpha_{j2}, \cdots, \alpha_{jK})$. (Recall the derived modality $\Diamond(\alpha_1, \alpha_2, \ldots, \alpha_K)$ introduced earlier.) From the semantics of $\Diamond(\alpha_1, \alpha_2, \ldots, \alpha_K)$ it follows that α is satisfiable iff α' is root-satisfiable.

Hence, in principle, it suffices to consider only root-satisfiability in developing a decision procedure for TrPTL. There is of course a blow-up involved in converting satisfiable formulas to root-satisfiable formulas. If one wants to avoid this blow-up then the decision procedure for checking root-satisfiability can be suitably modified to yield a direct decision procedure for checking satisfiability as done in [Thi1]. In any case, it is root satisfiability which is of importance from the standpoint of model checking. Hence here we shall only develop a procedure for deciding if a given formula of TrPTL is root-satisfiable.

As a first step we augment the syntax of our logic by one more construct.

- If α is a formula, so is $O_i\alpha$. In the model $M = (F, \{V_i\})$, at the configuration $c \in \mathcal{C}_F$, $M, c \models O_i\alpha$ iff $M, c \models \langle a \rangle_i \alpha$ for some $a \in \Sigma_i$. We also define $\mathrm{loc}(O_i\alpha) = \{i\}$.

Thus $O_i\alpha \equiv \bigvee_{a \in \Sigma_i} \langle a \rangle_i \alpha$ is a valid formula and O_i is expressible in the former syntax. It will be however more efficient to admit O_i as a first class modality.

Fix a formula α_0. Our aim is to effectively associate an A2-automaton $\mathcal{A}_{\alpha_0}$ with α_0 such that α_0 is root-satisfiable iff $L_{Tr}(\mathcal{A}_{\alpha_0}) \neq \emptyset$. Since the emptiness problem for A2-automata is decidable (Theorem 2.4), this will yield the desired decision procedure. Let $CL'(\alpha_0)$ be the least set of formulas containing α_0 which satisfies:

- $\neg\beta \in CL'(\alpha_0)$ implies $\beta \in CL'(\alpha_0)$.
- $\alpha \vee \beta \in CL'(\alpha_0)$ implies $\alpha, \beta \in CL'(\alpha_0)$.
- $\langle a \rangle_i \alpha \in CL'(\alpha_0)$ implies $\alpha \in CL'(\alpha_0)$.
- $O_i\alpha \in CL'(\alpha_0)$ implies $\alpha \in CL'(\alpha_0)$.
- $\alpha\, \mathcal{U}_i \beta \in CL'(\alpha_0)$ implies $\alpha, \beta \in CL'(\alpha_0)$. In addition, $O_i(\alpha\, \mathcal{U}_i \beta) \in CL'(\alpha_0)$.

We then define $CL(\alpha_0)$ to be the set $CL'(\alpha_0) \cup \{\neg\beta \mid \beta \in CL'(\alpha_0)\}$.

Thus $CL(\alpha_0)$, sometimes called the Fisher-Ladner closure of α_0, is closed under negation with the convention that $\neg\neg\beta$ is identified with β. From now we shall write CL instead of $CL(\alpha_0)$.

$A \subseteq CL$ is called an *i-type atom* iff it satisfies:

- $\forall \alpha \in CL.\ \alpha \in A$ iff $\neg\alpha \notin A$.
- $\forall \alpha \vee \beta \in CL.\ \alpha \vee \beta \in A$ iff $\alpha \in A$ or $\beta \in A$.
- $\forall \alpha\, \mathcal{U}_i \beta \in CL.\ \alpha\, \mathcal{U}_i \beta \in A$ iff $\beta \in A$ or ($\alpha \in A$ and $O_i(\alpha\, \mathcal{U}_i \beta) \in A$).
- If $\langle a \rangle_i \alpha,\ \langle b \rangle_i \beta \in A_i$ then $a = b$.

AT_i denotes the set of i-type atoms. We now need to define the notion of a formula in CL being a member of a collection of atoms. Let $\alpha \in CL$ and $\{A_i\}_{i \in Q}$ be a family of atoms with $\mathrm{loc}(\alpha) \subseteq Q$ and $A_i \in AT_i$ for each $i \in Q$. Then the predicate $\alpha \in \{A_i\}_{i \in Q}$ is defined inductively as:

- If $\mathrm{loc}(\alpha) = \{j\}$ then $\alpha \in \{A_i\}_{i \in Q}$ iff $\alpha \in A_j$.
- If $\alpha = \neg\beta$ then $\alpha \in \{A_i\}_{i \in Q}$ iff $\beta \notin \{A_i\}_{i \in Q}$.
- If $\alpha = \alpha_1 \vee \alpha_2$ then $\alpha_1 \vee \alpha_2 \in \{A_i\}_{i \in Q}$ iff $\alpha_1 \in \{A_i\}_{i \in Q}$ or $\alpha_2 \in \{A_i\}_{i \in Q}$.

The construction of the A2-automaton $\mathcal{A}_{\alpha_0}$ is guided by the construction due to Vardi and Wolper for LTL [VW]. However in the much richer setting of traces it turns out that one must make crucial use of the latest information that the agents have about each other when defining the transitions of $\mathcal{A}_{\alpha_0}$. It has been shown by Mukund and Sohoni [MS] that this information can be kept track of by a deterministic A2-automaton whose size depends only on $\widetilde{\Sigma}$. (Actually the automaton described in [MS] operates over finite traces but it is a trivial task to convert it into A2-automaton having the desired properties). To bring out the relevant properties of this automaton, let $F \in TR^\omega$ with $F = (E, \leq, \lambda)$. For each subset Q of processes, the function $\mathsf{latest}_{F,Q} : \mathcal{C}_F \times \mathcal{P} \to Q$ is given by $\mathsf{latest}_{F,Q}(c, j) = \ell$ iff ℓ is the least member of Q (under the usual ordering over the integers) with the property $\downarrow^j(\downarrow^q(c)) \subseteq \downarrow^j(\downarrow^\ell(c))$ for every $q \in Q$. In other words, among the agents in Q, ℓ has the best information about j at c, with ties being broken by the usual ordering over integers.

Theorem 3.3 ([MS]) *There exists an effectively constructible deterministic A2-automaton $\mathcal{A}_\Gamma = (TS, \mathcal{T})$ with $TS = (\{\Gamma_i\}, \{\Rightarrow_a\}, \Gamma_{in})$ such that:*

(i) $L_{Tr}(\mathcal{A}_\Gamma) = TR^\omega$.

(ii) *For each $Q = \{i_1, i_2, \ldots, i_n\}$, there exists an effectively computable function $\mathsf{gossip}_Q : \Gamma_{i_1} \times \Gamma_{i_2} \times \cdots \times \Gamma_{i_n} \times \mathcal{P} \to Q$ such that for every $F \in TR^\omega$, every $c \in \mathcal{C}_F$ and every $j \in \mathcal{P}$, $\mathsf{latest}_{F,Q}(c, j) = \mathsf{gossip}_Q(\gamma(i_1), \ldots, \gamma(i_n), j)$ where $\rho_F(c) = \gamma$ and ρ_F is the unique (accepting) run of $\mathcal{A}_\Gamma$ over F.*

Henceforth, we refer to $\mathcal{A}_\Gamma$ as the *gossip automaton.* Each process in the gossip automaton has $2^{O(K^2 \log K)}$ local states, where $K = |\mathcal{P}|$. Moreover the function g_Q can be computed in time which is polynomial in the size of K.

Each i-state of the automaton $\mathcal{A}_{\alpha_0}$ will consist of an i-type atom together with an appropriate i-state of the gossip automaton. Two additional component will be used to check for liveness requirements. One component will take values from the set $N_i = \{0, 1, 2, \ldots, |U_i|\}$ where $U_i = \{\alpha\, \mathcal{U}_i \beta \mid \alpha\, \mathcal{U}_i \beta \in CL\}$. This component will be used to ensure that all "until" requirements are met. The other component will take values from the set {on,off}. This will be used to detect when an agent has quit.

The automaton $\mathcal{A}_{\alpha_0}$ can now be defined.

Definition 3.4 $\mathcal{A}_{\alpha_0} = (TS, \mathcal{T})$, *where* $TS = (\{S_i\}, \{\rightarrow_a\}, S_{in})$ *and* $\mathcal{T} = \{(F_i^\omega, F_i)\}$ *are defined as follows:*

(i) *For each* i, $S_i = AT_i \times \Gamma_i \times N_i \times$ {on,off}. *Recall that* Γ_i *is the set of* i*-states of the gossip automaton and* $N_i = \{0, 1, 2, \ldots, |U_i|\}$ *with* $U_i = \{\alpha\, \mathcal{U}_i \beta \mid \alpha\, \mathcal{U}_i \beta \in CL\}$.

(ii) *Let* $s_a, s'_a \in S_a$ *with* $s_a(i) = (A_i, \gamma_i, n_i, v_i)$ *and* $s'_a(i) = (A'_i, \gamma'_i, n'_i, v'_i)$ *for each* $i \in \mathrm{loc}(a)$. *Then* $(s_a, s'_a) \in \rightarrow_a$ *iff the following conditions are met.*

  (1) $(\gamma_a, \gamma'_a) \in \Rightarrow_a$ *(recall that* $\{\Rightarrow_a\}$ *is the family of transition relations of the gossip automaton) where* $\gamma_a, \gamma'_a \in \Gamma_a$ *such that* $\gamma_a(i) = \gamma_i$ *and* $\gamma'_a(i) = \gamma'_i$ *for each* $i \in \mathrm{loc}(a)$.

  (2) $\forall i, j \in \mathrm{loc}(a)$, $A'_i = A'_j$.

  (3) $\forall i \in \mathrm{loc}(a)\ \forall \langle a \rangle_i \alpha \in CL.\ \langle a \rangle_i \alpha \in A_i$ *iff* $\alpha \in A'_i$.

  (4) $\forall i \in \mathrm{loc}(a)\ \forall O_i \alpha \in CL.\ O_i \alpha \in A$ *iff* $\alpha \in A'_i$.

  (5) $\forall i \in \mathrm{loc}(a) \forall \langle b \rangle_i \beta \in CL.$ *If* $\langle b \rangle_i \beta \in A_i$ *then* $b = a$.

  (6) *Suppose* $j \notin \mathrm{loc}(a)$ *and* $\beta \in CL$ *with* $\mathrm{loc}(\beta) = \{j\}$. *Further suppose that* $\mathrm{loc}(a) = \{i_1, i_2, \ldots, i_n\}$. *Then* $\beta \in A'_i$ *iff* $\beta \in A_\ell$ *where* $\ell = \mathsf{gossip}_{\mathrm{loc}(a)}(\gamma_{i_1}, \gamma_{i_2}, \ldots, \gamma_{i_n}, j)$.

  (7) *Let* $i \in \mathrm{loc}(a)$, $U_i = \{\alpha_1\, \mathcal{U}_i \beta_1, \alpha_2\, \mathcal{U}_i \beta_2, \ldots, \alpha_{n_i}\, \mathcal{U}_i \beta_{n_i}\}$. *Then* u'_i *and* u_i *are related to each other via:*

$$u'_i = \begin{cases} (u_i{+}1) \bmod (n_i{+}1), & \text{if } u_i = 0 \text{ or } \beta_{u_i} \in A_i \text{ or } \alpha_{u_i}\, \mathcal{U}_i \beta_{u_i} \notin A_i \\ u_i, & \text{otherwise} \end{cases}$$

  (8) *For each* $i \in \mathrm{loc}(a)$, $v_i =$ on. *Moreover, if* $v'_i =$ off *then* $\langle a \rangle_i \alpha \notin A'_i$ *for every* $i \in \mathrm{loc}(a)$ *and every* $\langle a \rangle_i \alpha \in CL$.

(iii) *Let* $s \in S_{\mathcal{P}}$ *with* $s(i) = (A_i, \gamma_i, u_i, v_i)$ *for every* i. *Then* $s \in S_{in}$ *iff* $\alpha_0 \in \{A_i\}_{i \in \mathcal{P}}$ *and* $\gamma \in \Gamma_{in}$ *where* $\gamma \in \Gamma_{\mathcal{P}}$ *satisfies* $\gamma(i) = \gamma_i$ *for every* i. *Furthermore,* $u_i = 0$ *for every* i. *Finally, for every* i, $v_i =$ off *implies that* $\langle a \rangle_i \alpha \notin A_i$ *for every* $\langle a \rangle_i \alpha \in CL$.

(iv) *For each* i, $F_i^\omega \subseteq S_i$ *is given by* $F_i^\omega = \{(A_i, \gamma_i, u_i, v_i) \mid u_i = 0$ *and* $v_i =$ on$\}$ *and* $F_i \subseteq S_i$ *is given by* $F_i = \{(A_i, \gamma_i, u_i, v_i) \mid v_i =$ off$\}$.

The automaton $\mathcal{A}_{\alpha_0}$ extends the automata theoretic construction for LTL described in [VW] to the setting of TrPTL. The main new feature is the use of the gossip automaton in step (ii)(6) when dealing with formulas located at agents not taking part in the current action. A detailed explanation of $\mathcal{A}_{\alpha_0}$ can be found in [Thi1].

This construction differs from the original construction for TrPTL presented in [Thi1] in a number of ways. Each S_i in [Thi1] was defined to be $AT_1 \times AT_2 \times \cdots \times AT_K \times \widetilde{U}_i \times \{\mathsf{act}_i^+, \mathsf{act}_i^-, \mathsf{stop}_i\}$ with $\widetilde{U}_i$ as the set of subsets of U_i. The acceptance condition used was $A1$. Using A2, we need just two elements

$\{\mathsf{on}, \mathsf{off}\}$ to record when an agent has quit. Using the counter N_i instead of $\widetilde{U}_i$ leads to a more compact description of $\mathcal{A}_{\alpha_0}$. The significant improvement, namely, replacing $AT_1 \times AT_2 \times \cdots \times AT_K$ by just AT_i is due to Narayan Kumar [Nar]. The arguments described in [Thi1] go through in the present setting with minor modifications. These arguments lead to the next set of results.

Theorem 3.5

(i) *α_0 is root-satisfiable iff $L_{Tr}(\mathcal{A}_{\alpha_0}) \neq \emptyset$.*

(ii) *The number of local states of $\mathcal{A}_{\alpha_0}$ is bounded by $2^{O(\max(n, m^2 \log m))}$ where $n = |\alpha_0|$ and m is the number of agents mentioned in α_0. Clearly, $m \leq n$. It follows that the root-satisfiability problem (and in fact the satisfiability problem) for TrPTL is solvable in time $2^{O(\max(n, m^2 \log m) \cdot m)}$.*

The number of local states of each process in $\mathcal{A}_{\alpha_0}$ is determined by two quantities: the length of α_0 and the size of the gossip automaton $\mathcal{A}_\Gamma$. As far as the size of $\mathcal{A}_\Gamma$ is concerned, it is easy to verify that we need to consider only those agents in $\mathcal{P}$ that are mentioned in $\mathrm{loc}(\alpha_0)$, rather than all agents in the system.

The model checking problem for TrPTL can be phrased as follows. A finite state distributed program over $\widetilde{\Sigma}$ is a pair $Pr = (\mathcal{A}_{Pr}, V_{Pr})$ where $\mathcal{A}_{Pr} = ((\{S_i^{Pr}\}, \{\Rightarrow_a^{Pr}\}, S_{in}^{Pr}), \{(S_i, S_i)\})$ is an A2-automaton modelling the state space of Pr and $V_{Pr} : S \rightarrow 2^{AP}$ is an interpretation of the atomic propositions over the local states of the program. (In this context, one assumes AP to be a finite set.)

Let ρ be a run of $\mathcal{A}_{Pr}$ over $F = (E, \leq, \lambda)$. Then ρ induces the model M_ρ via V_{Pr} as follows: $M_\rho = (F, \{V_i^\rho\})$ where for each i and each $c \in \mathcal{C}_F$, $V_i^\rho(\downarrow^i(c)) = V_{Pr}(s_i) \cap P$, where $s = \rho(c)$ and $s_i = s(i)$. Viewing a formula α_0 as a specification, we say that Pr meets the specification α_0—denoted $Pr \models \alpha_0$—if for every $F \in Tr^\omega$ and for every run ρ of $\mathcal{A}_{Pr}$ over F, it is the case that $M_\rho, \emptyset \models \alpha_0$.

The *model checking problem* is to determine whether $Pr \models \alpha_0$. This problem can be solved by "intersecting" the program automaton $\mathcal{A}_{Pr}$ with the formula automaton $\mathcal{A}_{\neg\alpha_0}$ to yield an automaton $\mathcal{A}$ such that $L_{Tr}(\mathcal{A}) = L_{Tr}(\mathcal{A}_{Pr}) \cap L_{Tr}(\mathcal{A}_{\neg\alpha_0})$. It turns out that $L_{Tr}(\mathcal{A}) = \emptyset$ iff $Pr \models \alpha_0$. It is easy to construct $\mathcal{A}$. The only point to care of is that the i-local states of $\mathcal{A}$ should consist of only those pairs (s_i, s_i') (where s_i is an i-local state of $\mathcal{A}_{Pr}$ and $s_i' = (A_i', \gamma_i', n_i', v_i')$ is an i-local state of $\mathcal{A}_{\neg\alpha_0}$) such that $V_{Pr}(s_i) \cap AP = A_i \cap AP$. The details can be found in [Thi1].

It turns out that this model checking problem has time complexity $O(|\mathcal{A}_{Pr}| \cdot 2^{O(\max(n, m^2 \log m) \cdot m)})$ where $|\mathcal{A}_{Pr}|$ is the size of the global state space of the A2-automaton modelling the behaviour of the given program Pr and, as before, $n = |\alpha_0|$ and m is the number of agents mentioned in α_0, where α_0 is the specification formula.

We now turn to two interesting sublogics of TrPTL. The first is the sublogic TrPTL$^{\mathrm{con}}$, which consists of the so called connected formulas of TrPTL. We define $\Phi^{\mathrm{con}}_{\mathrm{TrPTL}}$ (from now on written as Φ^{con}) to be the least subset of Φ satisfying the following conditions:

(i) $p(i) \in \Phi^{\mathrm{con}}$ for every $p \in P$ and every $i \in \mathcal{P}$.
(ii) If $\alpha, \beta \in \Phi^{\mathrm{con}}$, so are $\neg\alpha$ and $\alpha \vee \beta$.
(iii) If $\alpha \in \Phi^{\mathrm{con}}$ and $a \in \Sigma_i$ such that $\mathrm{loc}(\alpha) \subseteq \mathrm{loc}(a)$ then $\langle a \rangle_i \alpha \in \Phi^{\mathrm{con}}$.
(iv) If $\alpha, \beta \in \Phi^{\mathrm{con}}$ with $\mathrm{loc}(\alpha) = \mathrm{loc}(\beta) = \{i\}$ then $\alpha\, \mathcal{U}_i \beta \in \Phi^{\mathrm{con}}$. Actually one need only demand that $\mathrm{loc}(\alpha), \mathrm{loc}(\beta) \subseteq \bigcap\{\mathrm{loc}(a) \mid a \in \Sigma_i\}$ but this leads to notational complications that we wish to avoid here.
(v) If $\alpha \in \Phi^{\mathrm{con}}$ and $\mathrm{loc}(\alpha) = \{i\}$ then $O_i \alpha \in \Phi^{\mathrm{con}}$. (Once again one needs to just demand that $\alpha \subseteq \bigcap\{\mathrm{loc}(a) \mid a \in \Sigma_i\}$.)

Connected formulas were first identified by Niebert and used by Huhn [Huh]. They have also been independently identified by Ramanujam [Ram]. Thanks to the syntactic restrictions imposed on the next state and until formulas, past information is not allowed to creep in. Indeed one can prove the following:

Proposition 3.6 *Let $\alpha \in \Phi^{\mathrm{con}}$. Then α is satisfiable iff α is root-satisfiable.*

Yet another pleasing feature of $\mathrm{TrPTL}^{\mathrm{con}}$ is that the gossip automaton can be eliminated in the construction of the automaton $\mathcal{A}_{\alpha_0}$ whenever $\alpha_0 \in \Phi^{\mathrm{con}}$. In fact one can do a bit more.

Let $\alpha_0 \in \Phi^{\mathrm{con}}$ and let $CL_i = CL \cap \Phi^i$ for each i (recall that CL is an abbreviation for $CL(\alpha_0)$). We redefine an i-type atom to be a subset A of CL_i such that:

- $\forall \beta \in CL_i\ \beta \in A$ iff $\neg\beta \notin A$.
- $\forall \alpha \vee \beta \in CL_i.\ \alpha \vee \beta \in A$ iff $\alpha \in A$ or $\beta \in A$.
- $\forall \alpha\, \mathcal{U}_i \beta \in CL_i\ \alpha\, \mathcal{U}_i \beta \in A$ iff $\beta \in A$ or $\alpha \in A$ and $O_i(\alpha\, \mathcal{U}_i \beta) \in A$.

As before (but with the new definition in operation!), AT_i is the set of i-type atoms.

Let $\alpha \in CL$ with $\mathrm{loc}(\alpha) \subseteq Q$. The notion of α belonging to a family of atoms $\{A_i\}_{i \in Q}$, with $A_i \in AT_i$ for each $i \in Q$, is defined inductively in the obvious way—if $\mathrm{loc}(\alpha) = \{i\}$ then $\alpha \in \{A_i\}_{i \in Q}$ iff $\alpha \in A_i$ etc. etc. The construction of $\mathcal{A}_{\alpha_0}$ is as specified in Definition 3.4 with the following modifications:

(i) $S_i = AT_i \times N_i \times \{\mathsf{on}, \mathsf{off}\}$ for each $i \in \mathcal{P}$. Thus the gossip automaton is eliminated and AT_i is the set of i-type atoms of the new kind.
(ii) (1) This condition is obviously dropped.
  (2) Interestingly enough, this condition is also dropped.
  (3) This condition is modified to $\forall \langle a \rangle_i \alpha \in CL_i. \langle a \rangle_i \alpha \in A_i$ iff $\alpha \in \{A'_j\}_{j \in \mathrm{loc}(a)}$.

In addition, condition (ii)(6) is dropped, while conditions (ii)(4), (ii)(5), (ii)(7) and (ii)(8) remain unchanged. Parts (iii) and (iv) are modified to eliminate all references to the gossip automaton. After these alterations, it is not difficult to prove the following result.

Theorem 3.7 *Let* $\alpha_0 \in \Phi^{\mathrm{con}}$ *and* $\mathcal{A}_{\alpha_0}$ *be constructed as detailed above.*

(i) α_0 *is satisfiable iff* $L_{Tr}(\mathcal{A}_{\alpha_0}) \neq \emptyset$.
(ii) *The satisfiability problem for* TrPTL$^{\mathrm{con}}$ *is solvable in time* $2^{O(|\alpha_0|)}$.

Once again, a suitably modified statement can be made about the associated model checking problem. At present we do not know whether or not TrPTL is strictly more expressive than TrPTL$^{\mathrm{con}}$. We shall formulate this question more rigorously in the next section.

Yet another sublogic of TrPTL is called product TrPTL and is denoted as TrPTL$^{\otimes}$. Let $\Phi^{\otimes}$, the set of formulas of TrPTL$^{\otimes}$, be the least subset of Φ which satisfies:

(i) $p(i) \in \Phi^{\otimes}$ for every $p \in P$ and every $i \in \mathcal{P}$.
(ii) If $\alpha, \beta \in \Phi^{\otimes}$ then so are $\neg\alpha$ and $\alpha \vee \beta$.
(iii) If $\alpha \in \Phi^{\otimes}$ with $\mathrm{loc}(\alpha) = \{i\}$ and $a \in \Sigma_i$ then $\langle a \rangle_i \alpha \in \Phi^{\otimes}$.
(iv) If $\alpha, \beta \in \Phi^{\otimes}$ with $\mathrm{loc}(\alpha) = \mathrm{loc}(\beta) = \{i\}$ then $\alpha\, \mathcal{U}_i \beta \in \Phi^{\otimes}$.

Clearly $\Phi^{\otimes} \subseteq \Phi^{\mathrm{con}} \subseteq \Phi$. In case $\alpha_0 \in \Phi^{\otimes}$, the automaton $\mathcal{A}_{\alpha_0}$ can be simplified even further (than the case when $\alpha_0 \in \Phi^{\mathrm{con}}$). $\mathcal{A}_{\alpha_0}$ essentially consists of a synchronized product of Büchi automata. A detailed treatment of TrPTL$^{\otimes}$ is provided in [Thi2]. The interest in this subsystem lies in the fact that the accompanying program model is particularly simple and commonplace. Namely, it consists of a fixed set of finite state transition systems that coordinate their behaviour by performing common actions together. Here we shall just sketch the construction for $\mathcal{A}_{\alpha_0}$.

A product Büchi automaton over $\widetilde{\Sigma}$ is a structure $\mathcal{A} = (\{TS_i\}_{i \in \mathcal{P}}, S_{in}, \mathcal{T})$ where $TS_i = (S_i, \rightarrow_i)$ for each i with $\rightarrow_i \subseteq S_i \times \Sigma_i \times S_i$ as the local transition relation of the agent i. Everything else is as in the definition of an A2-automaton. Thus the key difference is that each agent comes with its own local transition relation. From these agent transition relations, one can derive the action indexed transition relations $\{\rightarrow_a\}$ as follows: $(s_a, s'_a) \in \rightarrow_a$ iff $s_a(i) \xrightarrow{a} s'_a(i)$ for every $i \in \mathrm{loc}(a)$. Thus product Büchi automata are a (strict) subclass of the class of A2-automata.

Given $\alpha_0 \in \Phi^{\otimes}$, the construction of $\mathcal{A}_{\alpha_0}$ proceeds as in the case where $\alpha_0 \in \Phi^{\mathrm{con}}$. The only difference is, we must define the transition relations $\{\rightarrow_i\}_{i \in \mathcal{P}}$ instead of the transition relations $\{\rightarrow_a\}_{a \in \Sigma}$. This can be done as follows:

Let $s_i, s'_i \in S_i$ with $s_i = (A_i, u_i, v_i)$ and $s'_i = (A'_i, u'_i, v'_i)$. Let $a \in \Sigma_i$. Then $s_i \xrightarrow{a_i} s'_i$ iff the following conditions are satisfied:

(i) $\forall \langle a \rangle_i \alpha \in CL.\ \langle a \rangle_i \alpha \in A_i$ iff $\alpha \in A'_i$.
(ii) $\forall O_i \alpha \in CL.\ O_i \alpha \in A$ iff $\alpha \in A'_i$.
(iii) If $\langle b \rangle_i \alpha \in A_i$ then $b = a$.
(iv) u_i and u'_i are related to each other just as in part (ii)(7) of Definition 3.4.
(v) v_i and v'_i satisfy part (ii)(8) of Definition 3.4.

As shown in [Thi2] one can establish the following result for TrPTL$^{\otimes}$.

Theorem 3.8 *Let $\alpha_0 \in \Phi^{\otimes}$ and $\mathcal{A}_{\alpha_0}$ be constructed as above.*

(i) *α_0 is satisfiable iff $L_{Tr}(\mathcal{A}_{\alpha_0}) \neq \emptyset$.*
(ii) *The satisfiability problem for TrPTL$^{\otimes}$ can be solved in time $2^{O(|\alpha_0|)}$.*

Once again, one can make suitably modified statements about the accompanying model checking problem. As mentioned earlier, the program model in this setting consists of a fixed set (one for each i) finite state transition systems.

We conclude this section with a quick look at some related logics. Katz and Peled introduced the logic ISTL [KP] which can be easily viewed as a temporal logic over traces. However, it has branching time modalities which permit quantification over the so called observations of a trace. ISTL uses global atomic propositions rather than local atomic propositions. Penczek has also studied a number of temporal logics (including a version of ISTL) with branching time modalities and global atomic propositions [Pen]. His logics are interpreted directly over the space of configurations of a trace resulting in a variety of axiomatizations and undecidability results. We feel that local atomic propositions (as used in TrPTL) are crucial for obtaining tractable partial order based temporal logics. Niebert has considered a μ-calculus version of TrPTL [Nie] and has obtained a decidability result using a variant of asynchronous Büchi automata. Since this logic uses “local” fixed points, it is not clear at present what is the expressive power of this logic. The four linear time temporal logics studied by Ramanujam in a closely related setting [Ram] can be easily captured as four sublogics of TrPTL through purely syntactic restrictions. Two of the resulting sublogics are TrPTL$^{\otimes}$ and TrPTL^{con}. It is not clear at present whether the other two logics admit a simpler treatment in terms of asynchronous Büchi automata (than the one for TrPTL).

The temporal logic of causality (TLC) proposed by Alur, Peled and Penczek is basically a temporal logic over traces [APP]. The concurrent structures used in [APP] as frames for TLC can be easily represented as traces over an appropriately chosen trace alphabet. The interesting feature of TLC is that its branching time modalities are interpreted over causal paths. In a trace $(E, \leq, \lambda)$, the sequence $e_0 e_1 \cdots \in E^{\infty}$ is a causal path if $e_0 \lessdot e_1 \lessdot e_2 \cdots$. This logic is almost certainly not expressible within the first order theory of traces although it admits an elementary time (in fact essentially exponential time) decision procedure.

Finally, Ebinger has also proposed a linear time temporal logic to be interpreted over traces [Ebi]. An interesting property of this logic is that when its frames are restricted to be *finite* traces then it is exactly equivalent to the first order theory of *finite* traces. Unfortunately the decidability of this logic is settled using a translation into the first order theory of infinite traces. Hence the decision procedure has non-elementary time complexity.

4 Expressiveness Issues

Our main aim here is to show that TrPTL is expressible within the first order theory of traces. In order to simplify the presentation, we shall eliminate atomic propositions and instead use the single constant $\top$ standing for "True" (and $\bot = \neg\top$ standing for "False"). The resulting logic will also be called TrPTL accompanied by the notations and terminology developed in the previous section. The function loc which assigns a set of processes to a formula works exactly as before except that we start with $\mathrm{loc}(\top) = \emptyset$. As will be seen later, this will entail minor changes in the definition of the syntax of $\mathrm{TrPTL}^{\mathrm{con}}$ and $\mathrm{TrPTL}^{\otimes}$. For now, we repeat that the syntax of Φ, the set of formulas of TrPTL is now given by:

$$\Phi ::= \top \mid \neg\alpha \mid \alpha \vee \beta \mid \langle a \rangle_i \alpha \mid \alpha\, \mathcal{U}_i \beta.$$

As before, for $\langle a \rangle_i \alpha$ to be a formula we require $a \in \Sigma_i$. Local atomic propositions can be coded up into the actions and hence their elimination does not result in loss of expressive power.

A model is just an infinite trace $F \in TR^\omega$. We set $F, c \models \top$ for every $c \in \mathcal{C}_F$. The rest of the semantics is as before. L_α, the ω-trace language defined by the formula α is given by, $L_\alpha = \{F \mid F \in TR^\omega \text{ and } F, \emptyset \models \alpha\}$. We say that $L \subseteq TR^\omega$ is TrPTL-definable iff there exists $\alpha \in \Phi$ such that $L = L_\alpha$.

First we shall compare the expressive powers of TrPTL, $\mathrm{TrPTL}^{\mathrm{con}}$ and $\mathrm{TrPTL}^{\otimes}$. In order to do so, we must define the syntax of the two sublogics in the present setting. For $\mathrm{TrPTL}^{\mathrm{con}}$ the only changes that are required are:

- $\top \in \Phi^{\mathrm{con}}$.
- If $\alpha, \beta \in \Phi^{\mathrm{con}}$ such that $\mathrm{loc}(\alpha), \mathrm{loc}(\beta) \subseteq \{i\}$ then $\alpha\, \mathcal{U}_i \beta \in \Phi^{\mathrm{con}}$.

For $\mathrm{TrPTL}^{\otimes}$, the only changes that are required are

- $\top \in \Phi^{\otimes}$.
- If $\alpha \in \Phi^{\otimes}$ such that $\mathrm{loc}(\alpha) \subseteq \{i\}$ and if $a \in \Sigma_i$ then $\langle a \rangle_i \alpha \in \Phi^{\otimes}$.
- If $\alpha, \beta \in \Phi^{\otimes}$ with $\mathrm{loc}(\alpha), \mathrm{loc}(\beta) \subseteq \{i\}$ then $\alpha\, \mathcal{U}_i \beta \in \Phi^{\otimes}$.

The notion of $L \subseteq TR^\omega$ being $\mathrm{TrPTL}^{\mathrm{con}}$-definable or $\mathrm{TrPTL}^{\otimes}$-definable is formulated in the obvious way. Since $\Phi^{\otimes} \subseteq \Phi^{\mathrm{con}} \subseteq \Phi$ it is clear that TrPTL is at least as expressive as $TrPTL^{\mathrm{con}}$ which in turn is at least as expressive as $TrPTL^{\otimes}$. As mentioned earlier we do not know at present if TrPTL is strictly more expressive than $TrPTL^{\mathrm{con}}$, though we conjecture that this the case.

We do know however that $\mathrm{TrPTL}^{\mathrm{con}}$ is strictly more expressive than $\mathrm{TrPTL}^{\otimes}$. To illustrate this it will be convenient to extend the notion of definability to subsets of Σ^ω. We say that $L \subseteq \Sigma^\omega$ is TrPTL-definable iff L is I-consistent and $\{\mathsf{str}(\sigma) \mid \sigma \in L\}$ is TrPTL-definable. This notion is defined for $\mathrm{TrPTL}^{\mathrm{con}}$ and $\mathrm{TrPTL}^{\otimes}$ in the obvious way. Hence in order to show that $\mathrm{TrPTL}^{\mathrm{con}}$ is more expressive than $\mathrm{TrPTL}^{\otimes}$ it suffices to exhibit some $L \subseteq \Sigma^\omega$ which is I-consistent and is $\mathrm{TrPTL}^{\mathrm{con}}$-definable but not $\mathrm{TrPTL}^{\otimes}$-definable.

Let $\widetilde{\Gamma} = \{\Gamma_1, \Gamma_2\}$ with $\Gamma_1 = \{a, a', d\}$ and $\Gamma_2 = \{b, b', d\}$. Let $\Gamma = \{a, a', b, b', d\}$. Consider $L \subseteq \Gamma^\omega$ given by:

$$L = (d(ab + ba + a'b' + b'a'))^\omega.$$

It turns out that L is *not* $\mathrm{TrPTL}^{\otimes}$-definable. Clearly L is I-consistent. As shown in [Thi2], for L to be $\mathrm{TrPTL}^{\otimes}$-definable, it must be a so-called (synchronized) product language. As a result, it would have to possess the following property:

(PR) Suppose $\sigma \in \Gamma^\omega$. Then $\sigma \in L$ iff there exist $\sigma_1, \sigma_2 \in L$ such that $\sigma\lceil\Gamma_1 = \sigma_1\lceil\Gamma_1$ and $\sigma\lceil\Gamma_2 = \sigma_2\lceil\Gamma_2$.

Now let $\sigma = (dab')^\omega$, $\sigma_1 = (dab)^\omega$ and $\sigma_2 = (da'b')^\omega$. Clearly $\sigma\lceil\Gamma_1 = \sigma_1\lceil\Gamma_1$ and $\sigma\lceil\Gamma_2 = \sigma_2\lceil\Gamma_2$. Since $\sigma_1, \sigma_2 \in L$, this implies that $\sigma \in L$ which it is not. Hence L cannot be a product language and therefore is not TrPTL$^\otimes$-definable. On the other hand, it is a simple exercise to come up with a formula $\alpha \in \Phi^{\text{con}}$ such that $\{\mathsf{str}(\sigma) \mid \sigma \in L\} = L_\alpha$.

We now turn to $FO(\widetilde{\Sigma})$, the first order theory of infinite traces over $\widetilde{\Sigma}$. One starts with a countable set of individual variables $X = \{x_0, x_1, \ldots\}$ with x, y, z with or without subscripts ranging over X. For each $a \in \Sigma$ there is a unary predicate symbol R_a. There is also a binary predicate symbol $\leq$.

$R_a(x)$ and $x \leq y$ are atomic formulas. If φ and φ' are formulas, so are $\neg\varphi$, $\varphi \vee \varphi'$ and $(\exists x)\varphi$. The structures for this first order theory are elements of TR^ω. Let $F \in TR^\omega$ with $F = (E, \leq, \lambda)$ and let $\mathcal{I} : X \to E$ be an interpretation. Then $F \models_{\mathcal{I}}^{FO} R_a(x)$ iff $\lambda(\mathcal{I}(x)) = a$ and $F \models_{\mathcal{I}}^{FO} x \leq y$ iff $\mathcal{I}(x) \leq \mathcal{I}(y)$. The remaining semantic definitions go along the expected lines. Each sentence φ (i.e., a formula with no free occurrences of variables) defines the ω-trace language $L_\varphi = \{F \mid F \models^{FO} \varphi\}$.

We say that $L \subseteq TR^\omega$ is FO-definable iff there exists a sentence φ in $FO(\widetilde{\Sigma})$ such that $L = L_\varphi$. As before we will say that $L \subseteq \Sigma^\omega$ is FO-definable iff L is I-consistent and $\{\mathsf{str}(\sigma) \mid \sigma \in L\}$ is FO-definable.

Using the fact that LTL has the same expressive power as the first order theory of sequences, one can show that $L \subseteq \Sigma^\omega$ is FO-definable iff it is I-consistent and LTL-definable [EM]. It will be worthwhile to pin down the notion of LTL-definability. In the current setting, remembering that (Σ, I) is the trace alphabet induced by $\widetilde{\Sigma}$, we define the syntax of the logic LTL(Σ) as follows:

$$\text{LTL}(\Sigma) ::= \top \mid \neg\alpha \mid \alpha \vee \beta \mid \langle a \rangle\alpha \mid \alpha\, \mathcal{U} \beta.$$

A model is a infinite word σ. For $\sigma \in \Sigma^\omega$ and $n \in \omega$, the notion of $\alpha \in \text{LTL}(\Sigma)$ being satisfied at stage n is denoted by $\sigma, n \models \alpha$. This satisfaction relation is defined in the usual manner. The only point of interest might be that $\sigma, n \models \langle a \rangle\alpha$ iff $\sigma(n{+}1) = a$ and $\sigma, n{+}1 \models \alpha$. We say that $L \subseteq \Sigma^\omega$ is LTL-definable iff there exists $\alpha \in \text{LTL}(\Sigma)$ such $L = L_\alpha$ where $L_\alpha = \{\sigma \in \Sigma^\omega \mid \sigma, 0 \models \alpha\}$.

The result in [EM] relating FO-definable subsets of TR^ω and LTL-definable subsets of Σ^ω can now be phrased as follows.

Proposition 4.1 *Let $L \subseteq \Sigma^\omega$. Then, the following statements are equivalent.*

(i) *L is I-consistent and LTL-definable.*
(ii) *$\{\mathsf{str}(\sigma) \mid \sigma \in L\}$ is an FO-definable subset of TR^ω.*

We now wish to concentrate on showing that TrPTL is expressible within the first order theory of infinite traces.

To show this, we will freely use the standard derived connectives of Propositional Calculus, together with universal quantification and abbreviations such as $x = y$ for $(x \leq y) \wedge (y \leq x)$, $x \leq y \leq z$ for $(x \leq y) \wedge (y \leq z)$ etc.

An event e is an i-event iff $\lambda(e) \in \Sigma_i$. With this in mind, we let $x \in E_i$ stand for the formula $\bigvee_{a \in \Sigma_i} R_a(x)$. The key to the result we are after is the observation that configurations of a trace can be described using predicates of *bounded* dimension. In what follows we let Q, Q', Q'' range over the non-empty subsets of $\mathcal{P}$. For $Q = \{i_1, i_2, \ldots, i_n\}$, the formula $\mathsf{config}(\{x_i\}_{i \in Q})$ is defined as:

$$\begin{aligned}\mathsf{config}(\{x_i\}_{i\in Q}) = (\varphi_1 \wedge \varphi_2 \wedge \varphi_3),\ &\text{where}\\ &\varphi_1 = \textstyle\bigwedge_{i \in Q} x_i \in E_i,\\ &\varphi_2 = \textstyle\bigwedge_{i,j} \bigwedge_a (R_a(x_i) \wedge R_a(x_j)) \Rightarrow x_i = x_j,\\ &\varphi_3 = \textstyle\bigwedge_{i,j} (\forall y)\ (y \in E_j \wedge y \leq x_i) \Rightarrow y \leq x_j.\end{aligned}$$

We can now write down a formula describing prime configurations—recall that a prime configuration is one of the form $\downarrow\! e$, where $e \in E$. Let $\mathrm{loc}(a) \subseteq Q$. Then the formula $\mathsf{prime}_a(\{x_i\}_{i \in Q})$ is defined as

$$\mathsf{config}(\{x_i\}_{i\in Q}) \wedge \bigwedge_{i \in \mathrm{loc}(a)} \bigwedge_{j \in Q - \mathrm{loc}(a)} R_a(x_i) \wedge (x_j \leq x_i).$$

A careful examination of this formula along with the basic properties of traces at once leads to the next result.

Proposition 4.2 *Let $F = (E, \leq, \lambda) \in TR^\omega$ and let $\mathcal{I} : X \to E$ be an interpretation. Then $F \models^{FO}_{\mathcal{I}} \mathsf{prime}_a(\{x_i\}_{i\in Q})$ iff there exists an a-event e such that for each $j \in Q$, $\mathcal{I}(x_j)$ is the $\leq_j$-maximum event in $\downarrow\! e \cap E_j$ and for each $j \notin Q$, $\downarrow\! e \cap E_j = \emptyset$.*

For each $\alpha \in \Phi$ we now define the sentence $\mathrm{SAT}(\emptyset, \alpha)$ and the set of formulas $\{\mathrm{SAT}(\{x_i\}_{i\in Q}, \alpha) \mid \{x_i\}_{i \in Q} \subseteq X \text{ and } \emptyset \neq Q \subseteq \mathcal{P}\}$ through simultaneous induction as follows:

- $\mathrm{SAT}(\emptyset, \top) = \mathrm{SAT}(\{x_i\}_{i\in Q}, \top) = (\exists x)\ x = x$.

- $\mathrm{SAT}(\emptyset, \neg\alpha) = \neg\mathrm{SAT}(\emptyset, \alpha)$.
  $\mathrm{SAT}(\{x_i\}_{i\in Q}, \neg\alpha) = \neg\mathrm{SAT}(\{x_i\}_{i\in Q}, \alpha)$.

- $\mathrm{SAT}(\emptyset, \alpha \vee \beta) = \mathrm{SAT}(\emptyset, \alpha) \vee \mathrm{SAT}(\emptyset, \beta)$.
  $\mathrm{SAT}(\{x_i\}_{i\in Q}, \alpha \vee \beta) = \mathrm{SAT}(\{x_i\}_{i\in Q}, \alpha) \vee \mathrm{SAT}(\{x_i\}_{i\in Q}, \beta)$.

- $\mathrm{SAT}(\emptyset, \langle a \rangle_j \alpha) = \bigvee_{Q \supseteq \mathrm{loc}(a} (\exists x_{i_1}, \exists x_{i_2}, \ldots, \exists x_{i_n})\ \varphi_1 \wedge \varphi_2 \wedge \varphi_3$
  where $Q = \{i_1, i_2, \ldots, i_n\}$ and

$$\begin{aligned}\varphi_1 &= \mathsf{prime}_a(\{x_i\}_{i\in Q}),\\ \varphi_2 &= \mathrm{SAT}(\{x_i\}_{i\in Q}, \alpha),\\ \varphi_3 &= (\forall y)\ (y \in E_j \wedge y \leq x_j) \Rightarrow y = x_j.\end{aligned}$$

$\textsc{Sat}(\{x_i\}_{i\in Q}, \langle a\rangle_j \alpha)$ is defined according to two cases.

Case 1 $j \notin Q$: $\textsc{Sat}(\{x_i\}_{i\in Q}, \langle a\rangle_j \alpha) = \textsc{Sat}(\emptyset, \langle a\rangle_j \alpha)$.

Case 2 $j \in Q$
$\textsc{Sat}(\{x_i\}_{i\in Q}, \langle a\rangle_j \alpha) = \bigvee_{Q'\supseteq \mathrm{loc}(a)} (\exists y_{k_1}, \exists y_{k_2}, \ldots \exists y_{k_n})\ \varphi_1 \wedge \varphi_2 \wedge \varphi_3$
where $Q' = \{k_1, k_2, \ldots, k_n\}$ and $\{y_k\}_{k\in Q'}$ is disjoint from $\{x_i\}_{i\in Q}$ and

$$\begin{aligned}
\varphi_1 &= \mathsf{prime}_a(\{y_k\}_{k\in Q'}),\\
\varphi_2 &= \textsc{Sat}(\{y_k\}_{k\in Q'}, \alpha),\\
\varphi_3 &= \forall y\ (y \in E_j \Rightarrow (y < y_j \Leftrightarrow y \le x_j)).
\end{aligned}$$

- $\textsc{Sat}(\emptyset, \alpha\, \mathcal{U}_j \beta) = \textsc{Sat}(\emptyset, \beta) \vee (\textsc{Sat}(\emptyset, \alpha) \wedge \textsc{Sat}(\emptyset, \bigvee_{a\in\Sigma_j} \langle a\rangle_j \alpha\, \mathcal{U}_j \beta)$.

$\textsc{Sat}(\{x_i\}_{i\in Q}, \alpha\, \mathcal{U}_j \beta)$ is defined according to two cases.

Case 1 $j \notin Q$: $\textsc{Sat}(\{x_i\}_{i\in Q}, \alpha\, \mathcal{U}_j \beta) = \textsc{Sat}(\emptyset, \alpha\, \mathcal{U}_j \beta)$.

Case 2 $j \in Q$:
$\textsc{Sat}(\{x_i\}_{i\in Q}, \alpha\, \mathcal{U}_j \beta) = \bigvee_{a\in\Sigma_j} \bigvee_{Q'\supseteq \mathrm{loc}(a)} (\exists y_{k_1}, \exists y_{k_2}, \ldots \exists y_{k_n})\ \varphi_1 \wedge \varphi_2 \wedge \varphi_3 \wedge \varphi_4$
where $Q' = \{k_1, k_2, \ldots, k_n\}$ and $\{y_k\}_{k\in Q'}$ is disjoint from $\{x_i\}_{i\in Q}$ and

$$\begin{aligned}
\varphi_1 &= \mathsf{prime}_a(\{y_k\}_{k\in Q'}),\\
\varphi_2 &= x_j \le y_j,\\
\varphi_3 &= \textsc{Sat}(\{y_k\}_{k\in Q'}, \beta),\\
\varphi_4 &= \forall z (z \in E_j \wedge x_j \le z < y_j) \Rightarrow \varphi_4'.
\end{aligned}$$

where $\varphi_4' = \bigvee_{a\in\Sigma_j} \bigvee_{Q''\supseteq \mathrm{loc}(a)} (\exists z_{\ell_1}, \exists z_{\ell_2}, \ldots, \exists z_{\ell_m})\ \varphi_{41}' \wedge \varphi_{42}' \wedge \varphi_{43}'$
with $Q'' = \{\ell_1, \ell_2, \ldots, \ell_m\}$ and $\{z_\ell\}_{\ell\in Q''}$ disjoint from both $\{x_i\}_{i\in Q}$ and $\{y_k\}_{k\in Q'}$ and

$$\begin{aligned}
\varphi_{41}' &= \mathsf{prime}_a(\{z_\ell\}_{\ell\in Q''}),\\
\varphi_{42}' &= (z = z_j),\\
\varphi_{43}' &= \textsc{Sat}(\{z_\ell\}_{\ell\in Q''}, \alpha).
\end{aligned}$$

Let f be the map which sends each formula in Φ to a sentence in $FO(\widetilde{\Sigma})$ via $f(\alpha) = \textsc{Sat}(\emptyset, \alpha)$. Using the previous proposition and the semantics of TrPTL, it is not difficult to prove the following:

Theorem 4.3
(i) *For every $F \in TR^\omega$, $F, \emptyset \models \alpha$ iff $F \models^{FO} f(\alpha)$.*
(ii) *If $L \subseteq TR^\omega$ is TrPTL definable then it is also $FO(\widetilde{\Sigma})$-definable.*

As mentioned earlier we do not know at present if TrPTL is expressively complete—i.e., whether every $L \subseteq TR^\omega$ which is $\mathrm{FO}(\widetilde{\Sigma})$-definable is also TrPTL-definable. Clearly from Proposition 4.1 it follows that the expressive completeness of TrPTL can be characterized as follows:

Corollary 4.4 *The following statements are equivalent:*

(i) TrPTL *is expressively complete.*
(ii) *For every* $L \subseteq \Sigma^\omega$, *if* L *is* I*-consistent and* L *is LTL-definable then* L *is TrPTL definable.*

We believe that TrPTL is *not* expressively complete. This leads to the following question: What is *the* linear time temporal logic of infinite traces? Such a logic should possess the following properties:

(TR1) It should be expressively complete.
(TR2) It should admit a decision procedure (preferably in terms of asynchronous Büchi automata) whose time complexity is $2^{p(n,m)}$ where n is the size of the input formula, $m = |\Sigma|$ and p is a (low degree) polynomial in n and m.
(TR3) It should be possible to *transparently* express global liveness and safety properties in the logic.

It is worth noting that TrPTL and most of the decidable temporal logics over traces mentioned earlier such as [Nie] and [APP] cannot express all global invariant properties. The somewhat awkward semantics of the logic in [Ebi] also makes it event-based and hence not suitable for expressing invariant properties. However we believe that it should be possible to define a logic with a variant of the until operator defined in [Ebi] which will be able to capture global liveness and safety properties in a straightforward manner.

Any linear time temporal logic over traces which fulfills the properties (PR1)–(PR3) will be a very useful specification tool. In particular it will exactly capture properties that are expressible by I-consistent formulas in LTL—($\alpha \in \mathrm{LTL}(\Sigma)$ is I-consistent iff L_α is I-consistent). This is important because it is such properties which can be verified efficiently using partial order based verification methods [GW, Val].

5 Conclusion

In this paper we have considered linear time temporal logics over traces. Our emphasis has been on TrPTL and its two sublogics $\mathrm{TrPTL}^{\mathrm{con}}$, $\mathrm{TrPTL}^{\otimes}$. The choice of these logics has been mainly motivated by the fact that they are expressible within the first order theory of traces and the fact that they can be studied using asynchronous Büchi automata.

Our formulation of asynchronous Büchi automata in terms of the acceptance condition A2 appears to be particularly suited for logical studies. The present constructions are much more compact and transparent than the ones in [Thi1] which used A1 as the acceptance condition. We feel that, in the future, alternating versions of our automata will play an important role in the study of temporal logics over traces.

As we have mentioned a number of times, an important open problem is to pin down a linear time temporal logic for traces (assuming it exists!) which will fulfill the properties set out in the previous section. A solution to this problem will at once open up the possibility of investigating branching time temporal logics where path quantification is over traces.

References

[AHU] A.V. Aho, J.E. Hopcroft and J.D. Ullman: *The Design and Analysis of Algorithms*, Addison-Wesley, Reading (1974).

[APP] R. Alur, D. Peled and W. Penczek: Model-Checking of Causality Properties, *Proc. 10th IEEE LICS* (1994).

[Die] V. Diekert: *Combinatorics on traces, LNCS* **454** (1990).

[DM] V. Diekert and A. Muscholl: Deterministic Asynchronous Automata for Infinite Traces, *Acta Inf.*, **31** (1993) 379-397.

[DR] V. Diekert, G. Rozenberg (Eds.): *The Book of Traces*, World Scientific, Singapore (1995).

[Ebi] W. Ebinger: *Charakterisierunǵ von Sprachklassenunendlicher Spuren durch Logiken*, Ph.D. Thesis, Institut für Informatik, Universität Stuttgart, Stuttgart, Germany (1994).

[EM] W. Ebinger and A. Muscholl: Logical Definability on Infinite Traces, *Proc. ICALP '93, LNCS* **700** (1993) 335–346.

[GP] P. Gastin and A. Petit: Asynchronous Cellular Automata for Infinite Traces, *Proc. ICALP '92, LNCS* **623** (1992) 583–594.

[GW] P. Godefroid and P. Wolper: A Partial Approach to Model Checking, *Inform. and Comput.*, **110** (1994) 305–326.

[Huh] M. Huhn: On Semantic and Logical Refinement of Actions, Technical Report, Institut für Informatik, Universität Hildesheim, Germany (1996).

[KP] S. Katz and D. Peled: Interleaving Set Temporal Logic, *Theor. Comput. Sci.*, **75** (3) (1992) 21–43.

[KMS] N. Klarlund, M. Mukund and M. Sohoni: Determinizing Büchi asynchronous automata, *Proc. FST&TCS 1995, LNCS* **1026** (1995) 456–470.

[LPRT] K. Lodaya, R. Parikh, R. Ramanujam and P.S. Thiagarajan: A logical study of distributed transition systems, *Inform. and Comput.*, **119** (1995) 91–118.

[MP] Z. Manna and A. Pnueli: *The Temporal Logic of Reactive and Concurrent Systems (Specification)*, Springer-Verlag, Berlin (1991).

[Maz] A. Mazurkiewicz: Concurrent Program Schemes and their Interpretations, *Report DAIMI-PB-78*, Computer Science Department, Aarhus University, Denmark (1978).

[MS] M. Mukund and M. Sohoni: Keeping track of the latest gossip: Bounded time-stamps suffice, *Proc. FST&TCS '93, LNCS* **761** (1993) 388–399.

[Mus] A. Muscholl: On the complementation of Büchi asynchronous cellular automata, *Proc. ICALP '94, LNCS* **820** (1994) 142–153.

[Nar] K. Narayan Kumar: An Improved Decision Procedure for TrPTL, Unpublished Manuscript, Tata Institute of Fundamental Research, Bombay, India (1994).

[Nie] P. Niebert: A ν-Calculus with Local Views for Systems of Sequential Agents, *Proc. MFCS'95, LNCS* **969** (1995) 563–573.

[NPW] M. Nielsen, G.D. Plotkin and G. Winskel: Petri Nets, Event Structures and Domains I, *Theor. Comput. Sci.*, **13** (1980) 86–108.

[Pen] W. Penczek: Temporal Logics for Trace Systems: On Automated Verification, *Int. J. Found. of Comput. Sci.*, **4**(1) (1993) 31-68.

[Pnu] A. Pnueli: The Temporal Logic of Programs, *Proc. 18th IEEE FOCS* (1977) 46–57.

[Ram] R. Ramanujam: Locally Linear Time Temporal Logic, To appear in *Proc. 11th IEEE LICS* (1996).

[Tho] W. Thomas: Automata on infinite objects, in J. van Leeuwen (ed.), *Handbook of Theoretical Computer Science, Volume B*, North-Holland, Amsterdam (1990) 133–191.

[Thi1] P.S. Thiagarajan: A Trace Based Extension of Linear Time Temporal Logic, *Proc. 9th IEEE LICS* (1994) 438–447. Full version available as: TrPTL: A Trace Based Extension of Linear Time Temporal Logic, *Report TCS-93-6*, School of Mathematics, SPIC Science Foundation, Madras, India (1993).

[Thi2] P.S. Thiagarajan: A Trace Consistent Subset of PTL, *Proc. CONCUR'95, LNCS* **962** (1995) 438-452. Full version available as: PTL over Product State Spaces, *Report TCS-95-4*, School of Mathematics, SPIC Science Foundation, Madras, India (1995).

[Val] A. Valmari: Stubborn Sets for Reduced State Space Generation, *LNCS* **483** (1990) 491–515.

[VW] M. Vardi, P. Wolper: An automata theoretic approach to automatic program verification, *Proc. 1st IEEE LICS* (1986) 332–345.

[WN] G. Winskel and M. Nielsen: Models for Concurrency, In: S. Abramsky and D. Gabbay (Eds.), *Handbook of Logic in Computer Science*, Vol 3, Oxford University Press, Oxford (1994).

[Zie] W. Zielonka: Notes on finite asynchronous automata, *R.A.I.R.O.—Inf. Théor. et Appl.*, **21** (1987) 99–135.

[Zuc] L. Zuck: *Past Temporal Logic*, Ph.D. Thesis, Weizmann Institute, Rehovot, Israel (1986).