Axioms for locality as product

Kamal Lodaya and R. Ramanujam

The Institute of Mathematical Sciences, CIT Campus, Chennai 600113, India

Abstract. We consider systems of finite state agents that are sequential in themselves and interact with each other by synchronously performing common actions together. We study reasoning about such systems along two lines. We consider an enrichment of Kleene's regular expressions with a parallel composition operator and offer a sound axiomatization of equality on expressions. We also study a simple linear time product temporal logic which is the standard one at a local level, and boolean combinations of located formulas at the global level. We offer a complete axiomatization of the valid formulas of this logic.

1 Introduction

When we study the dynamics of systems with concurrently evolving components that interact among themselves, a natural question arises: at what level of system description do we explicitly model concurrent dynamics ? This is an old conundrum that has kept philosophers busy. One solution is to record concurrency at the atomic level, so that the basic system events are already comprised of several simultaneously occurring ones, and we study the temporal evolution of such snapshots. In this view, if three events occur simultaneously at an instant, and two in the next, there is no particular 'actor' that provides any continuity from one instant to the next. This model is close to the way physical sciences study phenomena and adopted by Petri nets [Pet].

An alternative viewpoint is to see concurrent action at a much higher level of description: complex processes that evolve concurrently and independently, but interact occasionally. In this view, the component processes are sequential (and hence exhibit no concurrency), and the entire system is merely a fixed parallel composition of these processes. Games in extensive form, especially those of partial information, are examples of such systems: the continuity provided by each player over time, articulated by the notion of strategy, is crucial for game evolution [OR]. In the theory of computation, this notion of actor or agent can be identified with a *location*: the concurrent components notionally describe a distributed system. This was a view pioneered by Edsger Dikstra [Dijk] and developed into a rich theory of *Process algebra* by Hoare [Hoa], Milner [Mil1] and others [BPS]. This is the viewpoint we discuss in this article.

In particular, we consider systems of *finite state* agents that are sequential in themselves and interact with each other by synchronously performing common actions together. These common actions can be thought of as telephone calls: the caller waits for the other to pick up, they exchange information, end the

call and proceed further on their own asynchronous way. This is as opposed to communication by mail where the sender does not wait for the recipient but proceeds asynchronously.

The semantics of such systems can be envisioned as follows: each agent, being sequential non-deterministic, can be seen as a tree and the parallel composition of such trees as generating a set of trees obtained by nondeterministic interleaving of the agent trees, subject to synchronizations. Process algebraists have studied this operator extensively.

Considered as automata, the parallel composition operator above corresponds to *synchronized product*, and on languages, the corresponding operator is that of *synchronized shuffle*. It is easily seen that these operations preserve regularity, in the sense that the product machines are again finite state and that synchronized shuffle of regular languages is regular.

Over such systems, we focus on one particular aspect: that of axiomatizations. A celebrated theorem of Kleene offers a syntax of rational expressions that precisely define regular languages, and a complete axiom system of equality over these expressions was provided by Salomaa [Sal]. In this context it is natural to look for equations over rational expressions that involve an operator for parallel composition as well. We earlier presented a complete axiomatization using a reduction of paralellism to interleaving [Lod] (as is common in process algebra), here we present a sound axiom system which does not adopt this reduction.

Equational reasoning of this kind can be considered *global reasoning*, in the sense that it is carried out by someone who observes the behaviour of the entire system as it evolves. An alternative is *local reasoning*, where we reason about each agent separately (as far as possible) and combine the properties in some systematic way to infer properties of the composite system. Such compositional reasoning is naturally formalized as inference rules in logics. This is the other approach taken in this paper: we study specifications of agent properties in *propositional temporal logic of linear time* PTL [MP] and their global combinations. Once again we present a axiom system, and prove its completeness.

There is an important technical motivation for such local presentations. In general, if we have m agents, each of which is a k-state machine, the global state space has k^m states. Such a blow-up, exponential in the number of agents, is referred to as the *state explosion problem*. On the other hand, if we can reason about each component by itself, we have only km states to navigate. Referred to as partial order based methods, these approaches tend to utilize the idea that the entire set of interleavings may be large, but working with representative interleavings may suffice for many interesting properties. Several tools were developed in the 1990's based on such intuition [GW,Val,Pel]. In recent times, with the advent of multi-core architectures and relaxed memory models, such methods are acquiring renewed importance.

What is interesting about these axiomatizations ? When we consider top level parallelism we speak of parallel composition of sequential nondeterministic behaviours, and the central difficulty is that of determinizing the components separately, since local choices made by agents influences global choices at system

level. This difficulty manifests in both the equational axiomatization and the inference rules for the temporal logic studied here.

A natural but challenging question relates to how far such techniques can be generalized when the number of interacting agents is not a fixed finite number, but unbounded, and hence potentially infinite. [RS] offers some tentative suggestions for such reasoning.

2 Languages and automata

Fix a finite set Σ as an *alphabet*. We will call its elements *actions*. A finite sequence of actions $w : \{1, \ldots, n\} \to \Sigma$, such as *aabab*, is called a *word* over the alphabet. A set of words is called a *language*.

The syntax of **rational expressions** over Σ is given by:

$e \ ::= \ 0 \mid 1 \mid a, \ a \in \Sigma \mid e_1 e_2 \mid e_1 + e_2 \mid e_1^*$

The set of all words over Σ is Σ^* and the empty set is 0. The null word is written 1, we also use the same notation for the language $\{1\}$. In general, given expressions e_1, e_2 for languages L_1, L_2, the concatenation of their words is denoted $e_1 e_2$. The iteration e^* stands for the language formed by repeatedly concatenating words from e to form another word. For instance, given the language $\{aa, ab, b\}$, also written $aa + ab + b$, the word *aabab* is in $\{aa, ab, b\}^*$, but the word *ba* is not. The null word 1 is always in any e^* (by taking words from e zero times).

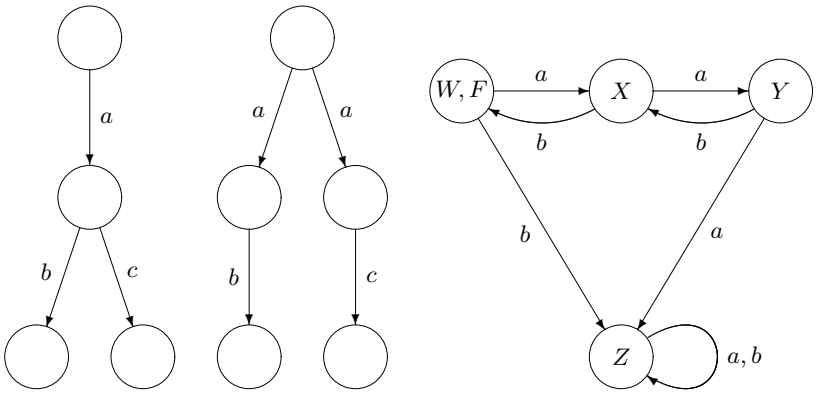

Fig. 1. The transition systems $a(b + c)$, $ab + ac$ and $Buff_2$

Definition 1. *A **labelled transition system** over the alphabet Σ is a directed graph $(Q, \to)$, with* states *Q and* transitions *$\to \subseteq Q \times \Sigma \times Q$. We will only be interested in transition systems which are* rooted *with a state $r \in Q$ from which all vertices are reachable. A **finite automaton** is a finite rooted transition system with a distinguished set of* final states*, let us say, marked by the variable F.*

We usually write $q \xrightarrow{a} q'$ to mean $(q, a, q') \in \rightarrow$, and interpret it to mean that the system can perform an action a at a state q and the resulting state is q'. A *run* of TS is a sequence $q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} \ldots$ — a possible "execution" of the system. Labelled transition systems provide a natural model for the study of system behaviour. In Section 4 onwards we will only consider *maximal* infinite runs of a transition system.

A *run* of an automaton operating on a word begins in the root state. On each action, it takes the corresponding transition from the current state into a (possibly) new state. At the end of the word, if the automaton is in a final state, the run *accepts* the word. The *language accepted by the automaton* is all words for which there is a run from the root state to a final state. Two finite automata accepting the same language are said to be *equivalent*. The examples in Figure 1 show that they may be non-isomorphic as transition systems.

Suppose variables are used to denote states of a transition system. A state can be described in terms of the transitions going out to the other states. For example, the transition system $Buff_2$, which describes a language where the number of a's seen at any point in an accepting run can exceed the number of b's by at most 2, is given by the equations

$W = aX + bZ + F,\ X = aY + bW,\ Y = aZ + bX,\ Z = aZ + bZ$

For an expression e such as $aX + bZ$, its *initial actions* $Init(e) = \{a, b\}$, and X and Z are the a- and b-*derivative* of e respectively. The notation goes back to Brzozowski [Brz]. We say e has the *no empty word property* (NEWP) if the empty word is not in its language. This can be syntactically checked using derivatives.

2.1 Axiomatization

The Aanderaa-Salomaa axiomatization [And,Sal] for language equivalence of rational expressions is given below.

AXIOMATIZATION **RAX** FOR EQUIVALENCE OF RATIONAL EXPRESSIONS

(Assoc)	$(e + f) + g = e + (f + g)$
(Comm)	$e + f = f + e$
(Ident)	$e + 0 = e$
(Idemp)	$e + e = e$
(Assoc)	$(ef)g = e(fg)$
(Ident)	$e1 = 1e = e$
(Absorp)	$0 = 0e = e0$
(Distr)	$(e + f)g = eg + fg$
(Distr)	$e(f + g) = ef + eg$
(Guard)	$e^* = (1 + e)^*$
(Fixpt)	$e^* = 1 + ee^*$
(Fixpt)	$e^* = 1 + e^*e$
(GuardInd)	Let e have the NEWP. Then: $\dfrac{x = ex + f}{x = e^*f}$; $\dfrac{x = xe + f}{x = fe^*}$

Theorem 1 (Salomaa). *The proof rules above are sound and complete for language equivalence of rational expressions.*

Proof. We only sketch the completeness. An inductive construction produces for any rational expression e a finite automaton accepting the language defined by e (for example, see [Koz]). Suppose e and f denote equivalent systems $TS(e)$ and $TS(f)$. By applying the axioms, each rational expression is reduced to a guarded sum of prefix form $\sum a_i e_i$ and $\sum b_i f_i$ respectively. Using left-distributivity, we need at most one derivative for each letter of the alphabet. Now equivalence between the roots guarantees equivalence between their successors, and equality among the derivatives of e and f guarantees derivability of $e = f$. So the task is reduced to proving completeness for nodes which are a distance 1 away from the roots. This can be repeated since all nodes of the transition systems are a finite distance away from the roots. □

2.2 Solutions in rational expressions

Consider again the equations

$W = aX + bZ,\ X = aY + bW,\ Y = aZ + bX,\ Z = aZ + bZ.$

Using right-distributivity and introducing star, we get $Z = (a+b)^*$. Substituting for Z and then for Y, we get

$W = aX + b(a + b)^*,\ X = bW + a(a(a + b)^* + bX).$

Now we crucially need to apply left-distributivity. Following that up with another introduction of star, we have

$X = abX + bW + aa(a + b)^* = (ab)^*(bW + aa(a + b)^*).$

Applying the same medicine again,

$W = a(ab)^*bW + a(ab)^*aa(a + b)^* + b(a + b)^*$ and

$W = (a(ab)^*b)^*(a(ab)^*aa(a + b)^* + b(a + b)^*).$

This way of finding solutions is reminiscent of performing Gaussian elimination in linear arithmetic equations and was first used for regular languages by McNaughton and Yamada [MY].

Theorem 2 (Kleene). *The regular languages, those defined by rational expressions, are exactly those accepted by finite automata.*

Proof. We already referred to the forward direction in Theorem 1. Conversely, given a finite automaton TS, we apply the McNaughton-Yamada technique outlined above to end up with a solution which gives a rational expression for the root state. □

3 Product words and product systems

Fix a nonempty finite set of *locations* $Loc = \{1, \ldots, n\}$. We now view the system alphabet $\Sigma \stackrel{\text{def}}{=} \Sigma_1 \cup \ldots \cup \Sigma_n$ as a *distributed* alphabet $\widetilde{\Sigma} = (\Sigma_1, \ldots, \Sigma_n)$, where each Σ_i is a finite nonempty set of *actions of agent* i. When an action a

is in $\Sigma_i \cap \Sigma_j, i \neq j$, we think of it as a *synchronization* action between i and j. (There can be k-way synchronizations also.) On the other hand, a *local* action is one in $\Sigma_i \setminus (\Sigma \setminus \Sigma_i)$, for some i.

We also make use of the associated implicit function $loc : \Sigma \to \wp(Loc) \setminus \{\emptyset\}$ which maps each action to the locations it is executed in, $loc(a) \stackrel{\text{def}}{=} \{a \mid a \in \Sigma_i\}$. For a set of actions A, $loc(A) \stackrel{\text{def}}{=} \bigcup\{loc(a) \mid a \in A\}$.

Given a distributed alphabet (Σ, loc), a *product word* is an element $(w_1, \ldots, w_n)$ of $(\Sigma^*)^{Loc}$ such that for some $w \in \Sigma^*$, every w_i is the restriction of w to actions in Σ_i. Thus $(w_1, \ldots, w_n)a$ is defined to be $(w'_1, \ldots, w'_n)$, where $w'_i = w_i a$ if $i \in loc(e)$ and $w'_i = w_i$ otherwise. If $loc(a) = \{i_1, \ldots, i_k\}$, we will call the expression $1||\ldots||a||\ldots||a||\ldots||1$, where a appears in positions $i_1, \ldots, i_k$ and 1 in the remaining positions, the distributed representation of a.

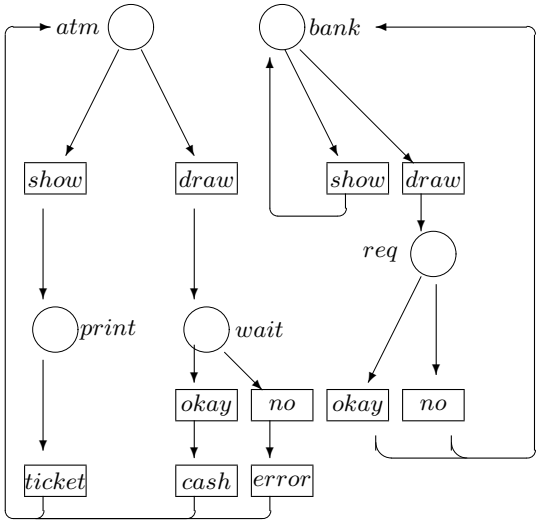

Fig. 2. A product system, the transition labels are boxed

We are interested in interactions between systems. A natural way to represent interactions among n agents is by having n transition systems, each working on its own alphabet of actions, except that the system undergoes "joint" transitions when common actions are encountered. Figure 2 models an ATM and a bank as a product system. A client at an ATM can ask for the balance to be shown and get a ticket printed. Alternately, the client asks to withdraw money and gets cash, or gets an error message, depending on whether the bank okays the transaction or not.

Definition 2. *Let $\widetilde{\Sigma} = (\Sigma_1, \ldots, \Sigma_n)$ be a distributed alphabet. A **parallel program** over $\widetilde{\Sigma}$ is a tuple $\mathcal{T} = (TS_1, \ldots, TS_n)$, where $TS_i = (Q_i, \to_i)$is a labelled*

transition system over Σ_i, for $i \in Loc$. When the individual transition systems are finite automata, we get a finite ***product automaton*** *with final states* $\widetilde{F} \stackrel{\text{def}}{=} F_1 \times \cdots \times F_n$.

The global run is extended by an action a if and only if for every agent i participating in that action, a is enabled at the current local state of i. The obvious way to define global runs is to take products of transition systems.

Definition 3. *Let $\mathcal{T}$ be a parallel program over $\widetilde{\Sigma}$. The* ***product system*** *for $\mathcal{T}$ is the Σ-labelled transition system $TS = (Q, \Rightarrow)$, where*

- $\widetilde{Q} \stackrel{\text{def}}{=} Q_1 \times \cdots \times Q_n$, *and*
- $\Rightarrow \subseteq Q \times \Sigma \times Q$ *is the* ***global*** *transition function defined as follows:* $(q_1, \ldots, q_n) \stackrel{a}{\Rightarrow} (q'_1, \ldots, q'_n)$ *iff* $\forall i \in loc(a), q_i \stackrel{a}{\rightarrow}_i q'_i$, *and* $\forall j \notin loc(a), q_j = q'_j$.

We will use $q_1, q_2, \ldots$ to denote local states and $s_1, s_2, \ldots$ to denote global states. $s[i]$ will refer to the i^{th} component of the global state s. For the purposes of presentation we will assume a fixed distributed alphabet $\widetilde{\Sigma} = (\Sigma_1, \ldots, \Sigma_n)$ and the entire discussion will be relative to $\widetilde{\Sigma}$.

A product automaton operates on a product word $(w_1, \ldots, w_n)$ distributedly. Suppose it has inductively processed a prefix of this word reaching the global state $s = (s[1], \ldots, s[n])$, and $(u_1, \ldots, u_n)$ is the remaining suffix. Suppose that for all $i \in loc(a)$, all the u_i are of the form au'_i and for the remaining i, let $u'_i = u_i$. Then the corresponding transition is taken and the automaton moves to the new state tuple $s' = (s'[1], \ldots, s'[n])$, where for $j \notin loc(a)$, $s'[j] = s[j]$. The distributed representation of a is concatenated to the prefix and $(u'_1, \ldots, u'_n)$ is the remaining suffix. At the end of the word, if the automaton is in a final state in each component agent, the product word is accepted.

Suppose $\delta = s_0 \stackrel{a_1}{\Longrightarrow} s_1 \stackrel{a_2}{\Longrightarrow} \ldots$; by $\delta(k)$, we mean the global state s_k, whereas we use δ_k to denote the suffix of δ starting at s_k. We can meaningfully define a map $\lceil$, which, given a global run δ, and $i \in Loc$, retains only the i^{th} components of global states and erases all actions not in Σ_i. Clearly, $\delta \lceil i$ is a run of TS_i.

Further, suppose $\doteq$ is a binary relation on runs of $\mathcal{T}$, defined by: $\delta \doteq \delta'$ if and only if $\delta = \delta_1 s \stackrel{a}{\Rightarrow} s_1 \stackrel{b}{\Rightarrow} s_2 \delta_2$ and $\delta' = \delta_1 s \stackrel{b}{\Rightarrow} s_3 \stackrel{a}{\Rightarrow} s_2 \delta_2$, for some a, b such that $loc(a) \cap loc(b) = \emptyset$. Thus, δ and δ' are permutations of independent actions a and b. Let $\approx \stackrel{\text{def}}{=} (\doteq)^*$. The following assertion is easy to prove:

Proposition 1. *$\delta \approx \delta'$ iff for every $i \in Loc, \delta \lceil i = \delta' \lceil i$.*

Thus, we can think of δ as a representative of the equivalence class of δ under $\approx$ (denoted $[\delta]$), a non-sequential run of $\mathcal{T}$. This representation of product words is known in the literature as *Mazurkiewicz traces* [Maz]. Each trace over the distributed alphabet $\widetilde{\Sigma}$ can be thought of as the set of linearizations of the product word (the possible w's whose restrictions are w_i in the explanation above).

3.1 Parallel products of rational expressions

To model product systems over a fixed $\widetilde{\Sigma}$, we now introduce one outermost level for parallel product of rational expressions. This is much simpler than a language like Hoare's CSP [Hoa] which can have nested occurrences of the parallel operator.

$r^i \ ::= \ 0 \mid 1 \mid a,\ a \in \Sigma_i \mid r_1^i r_2^i \mid r_1^i + r_2^i \mid (r_1^i)^*$

$e \ ::= \ r^1 || \ldots || r^n$

We generalize the Brzozowski derivative of a rational expression [Brz] to a distributed alphabet.

Definition 4. *$Der_a^{\widetilde{\Sigma}}(e_1|| \ldots ||e_n) \stackrel{\text{def}}{=} f_1|| \ldots ||f_n$, where for $1 \leq i \leq n$, if i in $loc(a)$ then $f_i = Der_a^{\Sigma_i}(e_i)$, otherwise $f_i = e_i$.*

This is an "expansion law" [Mil1], a global analysis seen as a product of actions on local components. Note that 1 appears in the derivative precisely when the local component can terminate, so termination of a product system is modelled by the expression $1|| \ldots ||1$.

In reasoning about a parallel product we may have to identify the initial actions which will never make progress. Here is a sufficient condition.

Definition 5. *Let $\widetilde{\Sigma} = (\Sigma_1, \ldots, \Sigma_n)$ and consider $e = e_1 + f_1|| \ldots ||e_n + f_n$. Suppose that for every i we have that $Init(e_i + f_i) \subseteq \Sigma_j$ for some $j \neq i$ and $Init(f_i)$ and $Init(f_j)$ are disjoint. We say that the actions in $Init(f_1), \ldots, Init(f_n)$ (also the expressions $f_1, \ldots, f_n$) are **useless in the sum** e.*

We now give a sound axiomatization for parallel products of rational expressions and an example of its use. The rules are parameterized by the arity of the parallel product, that is, the number of agents (n below) and depend on the distributed alphabet $\widetilde{\Sigma}$.

AXIOMS FOR EQUIVALENCE OF PRODUCTS **PAX**=**RAX**+

(Absorp) $0 = 0||e = e||0$

(Use) $e_1 + f_1|| \ldots ||e_n + f_n = e_1|| \ldots ||e_n$, if $f_1, \ldots, f_n$ are useless in the sum.

(Deriv) $$\frac{Der_a^{\widetilde{\Sigma}}(e_1|| \ldots ||e_n) = Der_a^{\widetilde{\Sigma}}(f_1|| \ldots ||f_n), \text{ for all } a \in \Sigma}{e_1|| \ldots ||e_n = f_1|| \ldots ||f_n}$$

(ProdInd) Let $f_1, \ldots, f_n$ have the NEWP. Then:

$$\frac{x_1|| \ldots ||x_n = (e_1 x_1 + f_1)|| \ldots ||(e_n x_n + f_n)}{x_1|| \ldots ||x_n = e_1^* f_1|| \ldots ||e_n^* f_n}$$

The (Absorp) axiom models the fact that a deadlock in some part of the system means that the entire system is deadlocked. The (Use) axiom eliminates a useless chain of waiting. (By adding dummy sums like $\cdots + a_i 0$ we may be able to eliminate useless chains of waiting through a subset of the n agents.) The (Deriv) rule was explained above. The (ProdInd) rule is a straightforward generalization of the (GuardInd) rule of **RAX** to the case of product systems.

3.2 Seeking solutions

To attempt a proof of completeness, we can apply the **RAX** axioms of Section 2.1 and reduce each product to guarded sum forms (from these the derivatives can be computed):

$(\sum_{i_1} a_{i_1,1}e_{i_1,1})||\ldots||(\sum_{i_n} a_{i_n,n}e_{i_n,n})$ and $(\sum_{j_1} b_{j_1,1}f_{j_1,1})||\ldots||(\sum_{j_n} b_{j_n,n}f_{j_n,n})$.

We can further assume from the (Use) axiom that none of the initial actions is useless.

If one can proceed ahead using an action, we suppose that equality at the level of successors is derivable and use the derivative rule to conclude that $e = f$. Since overall there are finitely many possible derivatives for the expressions, either this strategy must succeed or we must come back to a situation seen earlier and the product induction rule can be used.

Here is a worked-out example. Using **RAX**, we get:
$X||Y = (a+ba)^*||(ab)^* = 1+a(1+(a+ba)(a+ba)^*)+ba(a+ba)^*||1+ab(ab)^*$.

Distributing and eliminating useless actions we have that:
$X||Y = 1 + aba(a + ba)^*||1 + ab(ab)^* = 1 + abaX||1 + abY = (aba)^*||(ab)^*$ by product induction.

To prove $X||Y = (aba)^*||(ab)^* = 1+aba(aba)^*||1+ab(ab)^*$ using derivatives, we will need to eventually show that:
$W||Z = aba(aba)^*||ab(ab)^* = ae||bf$ for some e, f. Eliminating useless actions, $W||Z = 0$. Hence $X||Y = 1||1$.

However, the weakness of the axiomatization is that it lacks a full analysis of all the cases which arise. In the next section we will see a temporal logic that uses another induction to solve this problem.

One can add a further axiom, Milner's expansion law, which reduces parallel product to interleaving (for example, $a||b = ab + ba$) and then directly uses the completeness of **RAX**. This route to establish completeness is explored in [Lod].

4 Temporal logic

Let $\mathcal{T} = (TS_1, \ldots, TS_n)$ be a parallel program, and let L_i denote the runs of TS_i, for $i \in Loc$, the local runs of agent i. The frames for our logic will be global runs, which represent arbitrary interleavings of actions of different agents. In accordance with verification literature on temporal logic (for example, [MP]) we will henceforth be concerned with infinite runs. Let $R_{\mathcal{T}}$ denote the set of all *maximal* runs of the product system for $\mathcal{T}$. We restrict our attention to only those programs which have at least one infinite run.

Definition 6. *A **frame** is a pair $F = (\mathcal{T}, \delta)$, where δ is an infinite run in $R_{\mathcal{T}}$.*

We now present the logical language which we will call $PrPTL$. Let $AP = \{p_0, p_1, \ldots\}$ be a countable set of atomic propositions with p ranging over AP. We use α, β, γ etc. (with or without subscripts) to denote local formulas. The syntax of **i-local formulas** is given by:

$\Phi_i \ ::= \ p \mid \neg\alpha \mid \alpha \vee \beta \mid \langle a \rangle_i \alpha \mid \alpha \mathbf{U}_i \beta$

where, $a \in \Sigma_i$. This is basically PTL, where the next state modality has been indexed by actions.

We let $\phi_1, \phi_2, \ldots$ range over global formulas, whose syntax is given by:

$\Phi \ ::= \ \alpha@i, \alpha \in \Phi_i \mid \neg\phi \mid \phi_1 \vee \phi_2$

A *model* is a pair $M = (F, V)$, where $F = (\mathcal{T}, \delta)$ is a frame, and $V : Q \to \wp(AP)$ is the valuation function over Q, the set of all local states of the system. Thus, atomic propositions are evaluated at local states.

The formula ϕ being satisfied in a model M at a temporal instant is defined below. We first define the notion for i-local formulas over local runs in L_i. Let $M_i \stackrel{\text{def}}{=} ((\mathcal{T}, \delta\lceil i), V_i)$, where V_i is the restriction of V to Q_i. Let $\rho = \delta\lceil i \in L_i$.

- $M_i, 0 \models p$ iff $p \in V_i(\rho(0))$.
- $M_i, k \models \neg\alpha$ iff $M_i, k \not\models \alpha$.
- $M_i, k \models \alpha \vee \beta$ iff $M_i, k \models \alpha$ or $M_i, k \models \beta$.
- $M_i, k \models \langle a \rangle_i \alpha$ iff $\rho(k+1)$ exists, $\rho(k) \stackrel{a}{\to}_i \rho(k+1)$ and $M_i, k+1 \models \alpha$.
- $M_i, k \models \alpha \mathbf{U}_i \beta$ iff $\exists m \geq k$ such that $M_i, m \models \beta$, and for all $l : k \leq l < m, M_i, l \models \alpha$.

The derived connectives of propositional calculus such as $\wedge$, $\Longrightarrow$ and $\equiv$ are defined in terms of $\neg$ and $\vee$ in the usual way. Let $True$ abbreviate the Φ_i-formula $p_0 \vee \neg p_0$ and let $False$ stand for $\neg True_i$.

The derived modalities $\Diamond$, $\Box$, $\bigcirc$ and $[a]_i$ are given by:

$\Diamond\alpha \stackrel{\text{def}}{=} True \mathbf{U}_i \alpha$; $\bigcirc_i \alpha \stackrel{\text{def}}{=} \bigvee_{a \in \Sigma_i} \langle a \rangle_i \alpha$.

$\Box\alpha \stackrel{\text{def}}{=} \neg\Diamond\neg\alpha$; $[a]_i\alpha \stackrel{\text{def}}{=} \neg\langle a \rangle_i \neg\alpha$; $\odot_i \alpha \stackrel{\text{def}}{=} \neg\bigcirc_i\neg\alpha$.

We now define the semantics of global formulas.

- $M \models \alpha@i$ iff $M_i, 0 \models \alpha$.
- $M \models \neg\phi$ iff $M \not\models \phi$.
- $M \models \phi_1 \vee \phi_2$ iff $M \models \phi_1$ or $M \models \phi_2$.

We will use the notation $\widehat{a}$ to abbreviate the global formula $\bigwedge_{i \in loc(a)} (\langle a \rangle_i True)@i$. We can use $\widehat{a}$ to denote enabling of action a. Note that for any model M, if $M \models \widehat{a}$, and $\delta(0) = s$, then there exists a global state s' such that $s \stackrel{a}{\Rightarrow} s'$.

The formula ϕ is **satisfiable** if $M \models \phi$ for some model $M = ((\mathcal{T}, \delta), V)$. ϕ is **valid** (denoted $\models \phi$) if ϕ is satisfied in every model M. On the other hand, for a formula $\alpha \in \Phi_i$, we say that α is *i-valid*, if for every model M, we have $M_i, 0 \models \alpha$.

The following proposition is trivial to prove, and is the basis for expecting procedures for verification of properties in PrPTL:

Proposition 2. *Let $M = ((\mathcal{T}, \delta), V)$ be a model and let $\delta \approx \delta'$. Let $M' = ((\mathcal{T}, \delta'), V)$. Then for every PrPTL formula ϕ, $M \models \phi$ iff $M' \models \phi$.*

5 The axiom system

We now present an axiomatization of the valid formulas. We have one axiom system for each agent in the system, and in addition a global axiom system to reason about synchronization. In some sense, this helps us isolate how much global reasoning is required.

AXIOMATIZATION **LAXi** FOR AGENT i

(A0i) All the substitutional instances of the tautologies of PC

(A1i) $[a]_i(\alpha \implies \beta) \implies ([a]_i\alpha \implies [a]_i\beta)$

(A2i) $\langle a\rangle_i True \implies [b]_i False, \quad a \neq b$

(A3i) $\langle a\rangle_i\alpha \implies [a]_i\alpha$

(A4i) $\alpha \mathbf{U}_i\beta \implies (\beta \vee (\alpha \wedge \bigodot_i \alpha \mathbf{U}_i\beta))$

(MPi) $\dfrac{\alpha, \quad \alpha \implies \beta}{\beta}$

(TGi) $\dfrac{\alpha}{[a]_i\alpha}$

The axioms are quite standard. (A2i) expresses the fact that in any run, the next move made by agent i is unique.

We use the notation $\vdash_i \alpha$ to mean that the formula $\alpha \in \Phi_i$ is a theorem of system **LAXi**. We will call α *i-consistent* when $\neg\alpha$ is not a theorem of **LAXi**.

Proposition 3. *Every theorem of **LAXi** is i-valid.*

Some remarks are in order, before we proceed to present the global axiom system. We haven't included any axioms (or rules) in the local axiom systems for eventuality. This is so because information about local reachability is quite useless in PrPTL. Even an apparently local specification as $\langle a\rangle_i\alpha$ is in reality a global eventuality specification when there are other agents $j \neq i, j \in loc(a)$.

This suggests that we need some reasoning about eventuality at the global level. In temporal logic, this is typically achieved by an induction axiom or rule. Unfortunately since we have only boolean formulas at the global level, we cannot expect an axiom. The standard form of temporal induction for reachability looks like this:

$$\frac{\text{Global Invariant} \implies \alpha \wedge \bigodot(\text{Global Invariant})}{\text{Global Invariant} \implies \Box\alpha}$$

Since no global next state modality is available, we can only hope for something like this in PrPTL:

$$\frac{\bigwedge_k \text{Local Invariant@}k \implies \alpha@i \wedge \bigodot_j \text{Local Invariant@}j}{\bigwedge_k \text{Local Invariant@}k \implies (\Box_i\alpha)@i}$$

We can in fact write sound rules in this form, but they are too weak to express global reachability. Note that the global invariant is to specify "being one of several reachable global states". Now consider two global states characterized by formulas α and β respectively. We can assume that they are of the form $\bigwedge_{i\in Loc} \alpha_i@i$ and $\bigwedge_{i\in Loc} \beta_i@i$. Now notice that the formula $\bigwedge_{i\in Loc} (\alpha_i \vee \beta_i)@i$ is only

implied by $\alpha \vee \beta$, but does not imply it. Thus, combination of local invariants can in general specify global states which are not reachable, and we need to somehow specify the following:

$$\widehat{b} \wedge \bigwedge_k \text{Pre-move@}k \implies ([b]_j \text{ Post-move})@j, \text{ for } j \in loc(b),$$

$$\bigwedge_{k \notin loc(b)} \text{Pre-move@}k \wedge \bigwedge_{j \in loc(b)} \text{Post-move@}j \implies \text{Global Invariant}$$

and relate the global invariant to the local properties. Unfortunately, this turns out to necessitate an infinitary scheme.

GLOBAL AXIOMATIZATION **GAX**

(A0) $(\neg\alpha)@i \equiv \neg\alpha@i$

(A1) $(\alpha \vee \beta)@i \equiv (\alpha@i \vee \beta@i)$

(A2) $\bigvee_{a \in \Sigma} \widehat{a}$

(MP) $\dfrac{\alpha, \quad \alpha \implies \beta}{\beta}$

(GG) $\dfrac{\vdash_i \alpha}{\alpha@i}$

(GM) $\dfrac{\bigwedge_{i \in loc(a)} \alpha_i@i \implies \bigvee_{j \notin loc(a)} \alpha_j@j}{\bigwedge_{i \in loc(a)} (\langle a \rangle_i \alpha_i)@i \implies \bigvee_{j \notin loc(a)} \alpha_j@j}$

Let $m > 0$ and $\alpha_1, \ldots, \alpha_m$ be formulas such that for all $l \in \{1, \ldots, m\}$, α_l is of the form $\bigwedge_{k \in Loc} \alpha_l(k)@k$. Let $\gamma \stackrel{\text{def}}{=} \bigvee_{l=1}^{m} \alpha_l$.

(Sy$_m$) $\gamma \implies \neg\widehat{a}$

$$\bigwedge_{l \in \{1,\ldots,m\}} (\alpha_l \implies (\bigwedge_{b \notin \Sigma_i} (\widehat{b} \implies \bigwedge_{j \in loc(b)} ([b]_j \beta(l,b,j))@j)))$$

$$\bigwedge_{l \in \{1,\ldots,m\}} \bigwedge_{b \notin \Sigma_i} ((\bigwedge_{k \notin loc(b)} \alpha_l(k)@k \wedge \bigwedge_{j \in loc(b)} \beta(l,b,j)@j) \implies \gamma)$$

$\gamma \implies ([a]_i False)@i$, for $i \in loc(a)$

(Un$_m$) $\gamma \implies \neg\gamma_2@i$

$$\bigwedge_{l \in \{1,\ldots,m\}} (\alpha_l \implies (\bigwedge_{b \in \Sigma} (\widehat{b} \implies \bigwedge_{j \in loc(b)} ([b]_j \beta(l,b,j))@j)))$$

$$\bigwedge_{l \in \{1,\ldots,m\}} \bigwedge_{b \in \Sigma} ((\bigwedge_{k \notin loc(b)} \alpha_l(k)@k \wedge \bigwedge_{j \in loc(b)} \beta(l,b,j)@j) \implies \gamma)$$

$\gamma \implies \neg(\gamma_1 \mathbf{U}_i \gamma_2)@i$

The axiom (A2) ensures that some move is always enabled during the run. The rule (GG) allows us to globally infer theorems about agent i from those

which have been proved in **LAXi**. (For instance, this rule, alongwith (A0) and (A1) allows us to infer "@-versions" of tautologies.) The rule (GM) specifies that when a global move labelled a is made, the local states of agents not involved in a remain unchanged. The (Sy) and (Un) rules describe eventual synchronization and the semantics of *until* formulas respectively.

$\vdash \phi$ is the notation used to denote theoremhood in **GAX**. ϕ is said to be *consistent* when $\neg\phi$ is not a theorem of **GAX**.

Proposition 4. *$\vdash \phi$ implies $\models \phi$.*

Proof. The axioms are obviously valid formulas. To show that the inference rules preserve validity, consider the rule (GM). Suppose that the premise is valid, but not the conculsion. Then there exists a model $M = ((\mathcal{T}, \delta), V)$ and $M \models \widehat{a}$, for every $i \in loc(a), M_i, 0 \models \langle a \rangle_i \alpha_i$, and for every $j \notin loc(a), M_j, 0 \models \neg\alpha_j$. We thus have $\delta(0) = s \stackrel{a}{\Rightarrow} s' = \delta(1)$, and for every $j \notin loc(a), s[j] = s'[j]$. It is easy to check that for every $i \in loc(a), M_i, 1 \models \alpha_i$, and for every $j \notin loc(a), M_j, 1 \models \neg\alpha_j$. Now consider the model $M' = ((\mathcal{T}, \delta'), V)$, where $\delta' = \delta_1$. Clearly, $M' \models \bigwedge_{i \in loc(a)} \alpha_i @ i \wedge \bigwedge_{j \notin loc(a)} \neg\alpha_j @ j$, contradicting validity of the premise.

Similarly, suppose that the premises of (Sy) are valid but that the negation of its conclusion is satisfiable. Then we have a model $M = ((\mathcal{T}, \delta), V)$ and $M \models \alpha_k @ k$ for every $k \in Loc$. Further, $M_i, 0 \models \langle a \rangle_i True$, for some $i \in loc(a), a \in \Sigma$. then clearly there exist $b_1, \ldots, b_m$ such that $\{b_1, \ldots, b_m\} \cap \Sigma_i = \emptyset, \delta(0) \stackrel{b_1}{\Rightarrow} \delta(1) \ldots \stackrel{b_m}{\Rightarrow} \delta(m) \stackrel{a}{\Rightarrow} \delta(m+1)$. Now consider the models $M^l = ((\mathcal{T}, \delta_l), V)$, $l \in \{1, \ldots, m\}$. Obviously, since a is enabled at $\delta(m)$, we find that $M^m \models \widehat{a}$. Since the first premise says that $\gamma \implies \neg\widehat{a}$ is valid, it suffices to prove that $M^m \models \gamma$ to obtain a contradiction.

In fact, we argue that $M^l \models \gamma$ for every $l \in \{1, \ldots, m\}$. Firstly, since $M \models \gamma$ by assumption, for some $l, M \models \alpha_l$. Since $\delta(0) \stackrel{b_1}{\Rightarrow} \delta(1)$, we can show that $M \models \widehat{b_1}$, where $b_1 \notin \Sigma_i$. By validity of the second premise, for every $j \in loc(b), M \models ([b]_j \beta(l, b, j)) @ j$. therefore, we can show that for every $k \notin loc(b), M^1 \models \alpha_l(k) @ k$ and for every $j \in loc(b), M^1 \models \beta(l, b, j) @ j$. Thus, by validity of the third premise, we find that $M^1 \models \gamma$. We can repeat this argument to show that $M^l \models \gamma$, for $l > 1$ as well, and we are done.

The proof that the (Un) rule preserves validity is similar. □

Thus, we have soundness. We now proceed to show that **GAX** is indeed a complete axiomatization as well.

6 Completeness and decidability

In this section, we show that every consistent formula ϕ is satisfiable in a finite model whose size is bounded by $2^{c|\phi|}$. This at once shows that the logic is also decidable in nondeterministic exponential time.

We will find the following notation useful: when X is a finite set of formulas, by $\widehat{X}$ we mean the conjunction of all formulas in X (this is itself a formula).

Given Γ, a finite set, whose members are themselves finite sets of formulas, by $\widetilde{\Gamma}$, we mean the formula $\bigvee_{X \in \Gamma} \widehat{X}$.

We first define the notion of *subformula closure* of a formula.

Definition 7. *Let α be a formula in Φ_i.*

1. *$CL'_i(\alpha)$ is the least set of formulas containing α and satisfying the conditions*
   (a) $\neg\beta \in CL'_i(\alpha)$ implies $\beta \in CL'_i(\alpha)$.
   (b) $\beta \vee \gamma \in CL'_i(\alpha)$ implies $\{\beta, \gamma\} \subseteq CL'_i(\alpha)$.
   (c) $\langle a \rangle_i \beta \in CL'_i(\alpha)$ implies $\{[a]_i\beta, \beta\} \subseteq CL'_i(\alpha)$.
   (d) $\langle a \rangle_i True \in CL'_i(\alpha)$, for every $a \in \Sigma_i$.
   (e) $\alpha \mathbf{U}_i \beta \in CL'_i(\alpha)$ implies $\{\alpha, \beta, \bigodot_i(\alpha \mathbf{U}_i \beta)\} \subseteq CL'_i(\alpha)$.
2. *$CL_i(\alpha) \stackrel{\text{def}}{=} CL'_i(\alpha) \cup \{\neg\beta \mid \beta \in CL'_i(\alpha)\}$.*

For any α, $CL_i(\alpha)$ is finite and linear in the size of α. Similarly, we can define the subformula closure of global formulas.

Definition 8. *Let ϕ be a formula in Φ.*

1. *$CL'(\phi)$ is the least set of formulas containing ϕ and satisfying the conditions*
   (a) $\alpha@i \in CL'(\phi)$ implies $\{\beta@i | \beta \in CL_i(\alpha)\} \subseteq CL'(\phi)$.
   (b) $\neg\phi_1 \in CL'(\phi)$ implies $\phi_1 \in CL'(\phi)$.
   (c) $\phi_1 \vee \phi_2 \in CL'(\phi)$ implies $\{\phi_1, \phi_2\} \subseteq CL'(\phi)$.
2. *$CL(\phi) \stackrel{\text{def}}{=} CL'(\phi) \cup \{\neg\phi' \mid \phi' \in CL'(\phi)\}$.*

Once again, for any ϕ, $CL(\phi)$ is finite and linear in the size of ϕ. For convenience, we will abuse notation to also define $CL_i(\phi) \stackrel{\text{def}}{=} \{\beta | \beta@i \in CL(\phi)\}$, the set of i-subformulas of the global formula ϕ.

Fix a formula ϕ, and let $X \subseteq CL_i(\phi)$. We say that X is an *i-atom* of ϕ iff it satisfies the following conditions:

- for every $\beta \in CL_i(\phi), \neg\beta \in X$ iff $\beta \notin X$,
- for every $\beta \vee \gamma \in CL_i(\phi), \beta \vee \gamma \in X$ iff $\beta \in X$ or $\gamma \in X$,
- for every $a \in \Sigma_i$, if $\langle a \rangle_i True \in X$, then for every $b \in \Sigma_i, b \neq a, \langle b \rangle_i True \notin X$,
- for every $a \in \Sigma_i$, if $\langle a \rangle_i \alpha \in X$, then $[a]_i\alpha \in X$, and
- for every $\alpha \mathbf{U}_i \beta \in CL_i(\phi)$,
  $\alpha \mathbf{U}_i \beta \in X$ iff $(\beta \in X)$ or $(\alpha \in X$ and $\bigodot_i \alpha \mathbf{U}_i \beta \in X)$.

On the other hand, for a formula ϕ, we say that $A \subseteq CL(\phi)$ is a **atom** for ϕ iff for every $i, \{\alpha | \alpha@i \in A\}$ is an i-atom for ϕ. It can be easily checked that for every formula $\alpha@i \in CL(\phi)$, either that formula or its negation will be found in an atom (but not both). Further a formula of the form $\alpha@i \vee \beta@i$ (in $CL(\phi)$) is in an atom if and only if either of the disjuncts is in it. For an atom A, let $A[i]$ denote the associated i-atom, i.e. $\{\alpha | \alpha@i \in A\}$.

Let $AT_i(\phi)$ denote the set of all i-atoms for ϕ. Define $\rightarrow_i \subseteq AT_i \times \Sigma_i \times AT_i$ as follows: $X \stackrel{a}{\rightarrow}_i Y$ iff $([a]_i\alpha \in X$ implies $\alpha \in Y)$.

The global atom graph for a formula ϕ is defined as $G(\phi) \stackrel{\text{def}}{=} (AT(\phi), \Rightarrow')$, where $AT(\phi)$ is the set of all atoms for ϕ and $\Rightarrow'$ is defined by:

$$A \stackrel{a}{\Rightarrow}' B \text{ iff } \forall i \in loc(a), A[i] \stackrel{a}{\rightarrow}_i B[i] \text{ and } \forall j \notin loc(a), A[j] = B[j].$$

Suppose that $(W, \Rightarrow)$ is a subgraph of $(AT(\phi), \Rightarrow')$ in the sense that $W \subseteq AT$ and $\Rightarrow' \subseteq \Rightarrow$. Then we say that $(W, \Rightarrow)$ is *ϕ-good* if it satisfies the following conditions:

- there exists $A \in W$ such that $\phi \in A$, and
- for every $A \in W$,
  - A has a successor, i.e. B such that $A \stackrel{a}{\Longrightarrow} B$ for some a.
  - for every $a \in \Sigma$, if $\langle a \rangle_i True \in A$ for every $i \in loc(a)$, then there exists B such that $A \stackrel{a}{\Longrightarrow} B$.
  - for every $a \in \Sigma$, if $\langle a \rangle_i True \in A$ for some $i \in loc(a)$, then there exist $B_0, B_1, \ldots, B_k$, $k \geq 0$ and $b_1, \ldots, b_k$ such that $\{b_1, \ldots, b_k\} \cap \Sigma_i = \emptyset$, $A = B_0 \stackrel{b_1}{\Longrightarrow} \ldots \stackrel{b_k}{\Longrightarrow} B_k \stackrel{a}{\Longrightarrow} C$ for some C.
  - if $\alpha \mathbf{U}_i \beta \in A$ for some $i \in Loc$, then either $\beta @ i \in A$ or there exists B reachable from A such that $\beta @ i \in B$.

As it turns out, checking satisfiability of a formula ϕ amounts to checking the existence of such a ϕ-good subgraph in the syntactic graph $G(\phi)$. We will first show that every consistent formula does guarantee the existence of such a subgraph.

Lemma 1. *If ϕ_0 is a consistent formula, then $G(\phi_0)$ has a ϕ_0-good subgraph.*

Proof. Define W to be the set of maximal consistent subsets of $CL(\phi_0)$. (From now on, we fix ϕ_0 and use CL to mean $CL(\phi_0)$.) It is easy to check that each element of W is indeed an atom, using the local axiom systems, and (A0), (A1) and rule (GG). Thus $W \subseteq AT$. Simply define $\Rightarrow \stackrel{\text{def}}{=} \Rightarrow' \cap (W \times W)$. We claim that $(W, \Rightarrow)$ is ϕ_0-good.

Firstly, since ϕ_0 is consistent, there exists a maximal consistent set $A_0 \in W$ such that $\phi_0 \in A_0$. Now we have to prove that every element of W has an a-successor, and further that whenever $\{(\langle a \rangle_i True) @ i | i \in loc(a)\} \subseteq A$, for $A \in W$, then A has an a-successor. Once we prove the second condition of these two, the first one follows, thanks to axiom (A2). Now, fix $A \in W$ and $a \in \Sigma$ such that $\{(\langle a \rangle_i True) @ i | i \in loc(a)\} \subseteq A$. It can be easily checked that $\vdash \widehat{A} \implies \bigwedge_{i \in Loc} \widehat{A[i]} @ i$. Further, let $X_i \stackrel{\text{def}}{=} \{\alpha | [a]_i \alpha \in A[i]\}$, for $i \in loc(a)$. Since $\langle a \rangle_i True \in A[i]$, we can show that $\vdash_i \widehat{A[i]} \implies \langle a \rangle_i \widehat{X_i}$. Thus, we find that $\vdash \widehat{A} \implies \bigwedge_{i \in loc(a)} (\langle a \rangle_i \widehat{X_i}) @ i \wedge \bigwedge_{j \notin loc(a)} \widehat{A[j]}$. But since $\widehat{A}$ is a consistent formula, so is the formula implied by it. then, by rule (GM), we find that $\bigwedge_{i \in loc(a)} \widehat{X_i} @ i \wedge \bigwedge_{j \notin loc(a)} \widehat{A[j]}$ is consistent too. That is, the set $X \stackrel{\text{def}}{=} \bigcup_{i \in loc(a)} X_i \cup \bigcup_{j \notin loc(a)} A[j]$

is consistent. Hence, there exists a maximal consistent set $B \in W$ such that $X \subseteq B$. It can be easily checked that $A \stackrel{a}{\Rightarrow} B$ as required.

We now show that the eventual synchronization condition is also satisfied in the graph $(W, \Rightarrow)$. Fix $A \in W$ such that $(\langle a \rangle_i True)@i \in A$ for some $i \in loc(a)$. Let Γ be the least subset of W which satisfies the following conditions:

- $A \in \Gamma$, and
- whenever $B \in \Gamma$ and $B \stackrel{b}{\Rightarrow} C$ for any $b \notin \Sigma_i, C \in \Gamma$.

Note that every element in Γ is reachable from A via a path in $(W, \Rightarrow)$ which goes through actions outside Σ_i. Clearly, if there exists $B \in \Gamma$ such that B has an a-successor at all, then we are through.

Now suppose that there is no such B in Γ. We show that this assumption leads to a contradiction. It can be checked that $\vdash \widehat{B} \implies \neg\widehat{a}$ for every $B \in \Gamma$. Therefore, $\vdash \widetilde{\Gamma} \implies \neg\widehat{a}$.

Let $B \in \Gamma$. For every $b \notin \Sigma_i$, and for every $j \in loc(b)$, define

$$\Delta(B, b, j) \stackrel{\text{def}}{=} \begin{array}{ll} \{C[j] | B \stackrel{b}{\Rightarrow} C\}, & if \vdash \widehat{B} \implies \widehat{b}, \ and \\ \{B[j]\}, & otherwise. \end{array}$$

Claim (1). $\vdash \bigwedge_{B \in \Gamma} (\widehat{B} \implies (\bigwedge_{b \notin \Sigma_i} (\widehat{b} \implies \bigwedge_{j \in loc(b)} ([b]_j \widetilde{\Delta(B, b, j)})@j))))$.

Claim (2). $\vdash \bigwedge_{B \in \Gamma} \bigwedge_{b \notin \Sigma_i} (\bigwedge_{k \notin loc(b)} (\widehat{B[k]})@k \wedge \bigwedge_{j \in loc(b)} (\widetilde{\Delta(B, b, j)})@j) \implies \widetilde{\Gamma})$.

Suppose the claims are true. Then we have derived every premise in rule (Sy) in our axiom system. Hence, by its conclusion, $\widetilde{\Gamma} \implies ([a]_i False)@i$. But then $\widehat{A} \implies \widetilde{\Gamma}$ (since $A \in \Gamma$) and hence $\widehat{A} \implies ([a]_i False)@i$, clearly contradicting the fact that $(\langle a \rangle_i True)@i \in A$.

The only condition left to be proved (for $(W, \Rightarrow)$ to be ϕ_0-good) is the "until" condition. The proof of this proceeds in a fashion quite similar to the one for eventual synchronization, and hence is omitted here. □

Proof (of Claim (1)). Suppose that the formula in the Claim is not a theorem of the system. Then its negation is consistent, and we show that this leads to a contradiction. Skipping a few obvious steps, we can see that for some $B \in \Gamma, b \notin \Sigma_i$ and some $j \in loc(b)$, $\widehat{B} \wedge \widehat{b} \wedge (\langle b \rangle_j \neg \widetilde{\Delta(B, b, j)})@j$ is consistent. By the consistency of $\widehat{B} \wedge \widehat{b}$, we can expect B to have a b-successor. In addition, by a reasoning similar to what we employed earlier, we can find a b-successor, say C, such that $(\widehat{C[j]})@j \wedge (\neg \widetilde{\Delta(B, b, j)})@j$ is consistent. But, by construction of $\Delta(B, b, j)$, every such $C[j] \in \Delta(B, b, j)$. Then we get $\vdash (\widehat{C[j]})@j \implies (\widetilde{\Delta(B, b, j)})@j$, clearly a contradiction. □

Proof (of Claim (2)). This claim obviously follows from the fact that if $B \in \Gamma, b \notin \Sigma_i$ and $B \stackrel{b}{\Rightarrow} C$, then by construction, $C \in \Gamma$ and the fact that for $k \notin$

$loc(b), B[k] = C[k]$ by definition of $\Rightarrow$, whereas for $j \in loc(b), C[j] \in \Delta(B, b, j)$, again by construction. When B has no such b-successor, we get the required thesis by observing that $\widehat{B} \implies \Gamma$. □

Lemma 2. *If $G(\phi_0)$ has a ϕ_0-good subgraph, then ϕ_0 is satisfiable.*

Proof. Suppose that $(W, \Rightarrow)$ is a ϕ_0-good subgraph of $G(\phi_0)$. Let $A_0 \in W$ such that $\phi_0 \in A_0$. We claim that there is a maximal run δ of $(W, \Rightarrow)$ of the form $A_0 \stackrel{a_1}{\Rightarrow} A_1 \stackrel{a_2}{\Rightarrow} \ldots$ which satisfies the following conditions: (let $k \geq 0$)

- If $(\langle a \rangle_i True)@i \in A_k$ then there exists $m \geq 0$ such that $A_{k+m} \stackrel{a}{\Rightarrow} A_{k+m+1}$ and for every l such that $0 \leq l < m$, $A_{k+l} \stackrel{b_l}{\Rightarrow} A_{k+l+1}$ implies that $b_l \notin \Sigma_i$.
- If $(\alpha \mathbf{U}_i \beta)@i \in A_k$, then there exists $m \geq 0$ such that $\beta@i \in A_{k+m}$.

The details of construction of δ are straightforward, though not trivial: we consider each of the n agents in a round-robin fashion, and keep fulfilling eventuality (until) requirements. Note that when an until-requirement is met for an agent, next-action requirements are also fulfilled upto the last action.

Now consider the parallel program $\mathcal{T} = (TS_1, \ldots, TS_n)$ with $TS_i = (AT_i, \rightarrow_i)$, for $i \in Loc$. Let $TS \stackrel{\text{def}}{=} (Q, \Rightarrow')$ be the product system for $\mathcal{T}$. It can be checked that $\{(A[1], \ldots, A[n]) | A \in W\} \subseteq Q$ and that $(A[1], \ldots, A[n]) \stackrel{a}{\Rightarrow}' (B[1], \ldots, B[n])$ iff $A \stackrel{a}{\Rightarrow} B$ in the given ϕ_0-good subgraph. Thus δ induces a maximal run δ' of the product system as well.

Now consider the frame $F = (\mathcal{T}, \delta')$, and the model $M = (F, V)$, where $V(X) \stackrel{\text{def}}{=} X \cap AP$, for $X \in AT_i$, for some i. Let $\rho = \delta' \lceil i$ and let $M_i \stackrel{\text{def}}{=} (\rho, V_i)$, where V_i is the restriction of V to AT_i.

Claim (3). For every $\alpha \in CL_i(\phi_0), k \geq 0, M_i, k \models \alpha$ iff $\alpha \in \rho(k)$.

Assuming the claim, we can go on to show that for all $\phi_1 \in CL(\phi_0), M \models \phi_1$ iff $\phi_1 \in A_0$. This is proved by an easy induction argument. But then, since $\phi_0 \in A_0$, it follows that $M \models \phi_0$, and we have demonstrated the satisfiability of ϕ_0. □

Proof (of Claim (3)). The proof proceeds by induction on the structure of α. The base case, when $\alpha \in AP$ is trivial and follows by the definition of V above.

The induction step proceeds by cases: when α is of the form $\neg\beta$ or of the form $\beta_1 \vee \beta_2$, the proof is by routine applications of the induction hypothesis. Now suppose that α of the form $\langle a \rangle_i \beta \in \rho(k)$. By construction of δ above (and hence of δ'), $\rho(k+1)$ exists, and $\rho(k) \stackrel{a}{\rightarrow}_i \rho(k+1)$. Further, since $\rho(k)$ is an i-atom, we find that $[a]_i \alpha \in \rho(k)$ as well, and by definition of $\Rightarrow$ above, $\alpha \in \rho(k+1)$. By induction hypothesis, $M_i, k+1 \models \alpha$ and hence $M_i, k \models \langle a \rangle_i \alpha$, as required.

On the other hand, when $M_i, k \models \langle a \rangle_i \alpha$, we are given that $\rho(k+1)$ exists, that $\rho(k) \stackrel{a}{\rightarrow}_i \rho(k+1)$ and that $M_i, k+1 \models \alpha$. By induction hypothesis, $\alpha \in \rho(k+1)$, and by definition of $\Rightarrow$, $\langle a \rangle_i \alpha \in \rho(k)$.

Now suppose α of the form $\beta \mathbf{U}_i \gamma \in \rho(k)$. If $\gamma \in \rho(k)$, we have (by induction hypothesis $M_i, k \models \gamma$ and hence $M_i, k \models \beta \mathbf{U}_i \gamma$. Otherwise, by construction

of δ above, we find that there exists $m > k$ such that $\gamma \in \rho(m)$ and for all $l : k \leq l < m, \gamma \notin \rho(l)$. Now consider $\rho(k)$: since $\{\beta \mathbf{U}_i \gamma, \neg\gamma\} \subseteq \rho(k)$, being an i-atom, $\{\beta, \bigodot_i(\beta \mathbf{U}_i \gamma)\} \subseteq \rho(k)$. Hence $\beta \mathbf{U}_i \gamma \in \rho(k+1)$ as well. By the same reasoning $\beta \in \rho(k+1)$. Thus, we can show that for every $l : k \leq l < m, \beta \in \rho(l)$. The result follows by the induction hypothesis. The converse is proved similarly.

This completes the induction and the claim is proved. □

The above two lemmas together lead us at once to the main results of the paper:

Theorem 3. *$\models \phi$ implies $\vdash \phi$; that is, **GAX** provides a complete axiomatization of the valid formulas of PrPTL.*

Theorem 4. *The satisfiability of a PrPTL formula ϕ can be decided by an algorithm taking $NTIME(2^{O(|\phi|)})$.*

We expect that the time complexity can be shown to be deterministic (singly) exponential time, using a more careful argument than the one presented above.

We can also consider the model checking problem for PrPTL: given a parallel program $\mathcal{T} = (TS_1, \ldots, TS_n)$, a PrPTL formula ϕ, and a valuation $V : Q \to 2^X$ (where X is the set of atomic propositions mentioned in ϕ), the problem is to determine whether every model based on $\mathcal{T}$ and V satisfies ϕ. By a standard argument, we can consider the product of the syntactic graph of ϕ above with the product system and check for connected components generating ϕ-good subgraphs. Here, we mention only the result:

Theorem 5. *Model checking a PrPTL formula ϕ against a parallel program with m global states is decidable in $NTIME(m.2^{O(|\phi|)})$.*

References

[And] Anderaa, S., On the algebra of regular expressions, in *Appl. Math. (course notes)*, Harvard, Jan 1965, 1–18.

[BPS] Bergstra, J., Ponse, A. and Smolka, S.A., eds., *Handbook of process algebra* Elsevier, 2001, 333–389.

[Brz] Brzozowski, J., Derivatives of regular expressions, *JACM* **11**,4, 1964, 481–494.

[Dijk] Dijkstra, E.W., *A discipline of programming* Prentice-Hall, 1976.

[GV] van Glabeek, R.J. and Vaandrager, F.W., Petri net models for algebraic theories of concurrency, *Proc. PARLE* **2**, Eindhoven (J.W. de Bakker, A.J. Nijman and P.C. Treleaven, eds.), *LNCS* **259**, 1987, 224–242.

[GW] Godefroid, P. and Wolper, P., A partial approach to model checking, *Inf. Comput.* **110**, 1994, 305–326.

[Hoa] Hoare, C.A.R., *Communicating sequential processes*, Prentice-Hall, 1985.

[Kle] Kleene, S.C., Representation of events in nerve nets and finite automata, in *Automata studies* (C.E. Shannon and J. McCarthy, eds.). Princeton, 1956, 3–41.

[Koz] Kozen, D., *Automata and computability*, Springer, 1997.

[LPRT] Lodaya, K., Parikh, R., Ramanujam, R. and Thiagarajan, P.S., A logical study of distributed transition systems, *Inf. Comput.* **119**,1, 1995, 91–115.

[Lod] Lodaya, K., Product automata and process algebra, *Proc 4th SEFM*, Pune (P.K. Pandya and D.v.Hung, eds.), IEEE, 2006, 128–136.

[MP] Manna, Z. and Pnueli, A., *The temporal logic of reactive and concurrent systems (Specification)*, Springer, 1991.

[Maz] Mazurkiewicz, A., Concurrent program schemes and their interpretations, Report DAIMI PB-78, Aarhus University, 1977.

[MY] McNaughton, R. and Yamada, H., Regular expressions and state graphs for automata, *IEEE Trans. Electr. Comp.* **9**, 1960, 39–47.

[Mil1] Milner, R., *A calculus of communicating systems*, *LNCS* **92**, 1980.

[Mil2] Milner, R., A complete inference system for a class of regular behaviours, *JCSS* **28**,3, 1984, 439–466.

[OR] Osborne, M.J. and Rubinstein, A., *A course in game theory* MIT Press, 1994.

[Pel] Peled, D., All from one and one from all: on model checking using representatives, *Proc CAV*, *LNCS* **697**, 1993, 409–423.

[Pet] Petri, C.-A., Fundamentals of a theory of asynchronous information flow, *Proc. IFIP*, Munich (C.M. Popplewell, ed.), North-Holland, 1962, 386–390.

[PW] Pinter, S., and Wolper, P., A temporal logic for reasoning about partially ordered computations, *Proc 3rd ACM PODC* , 1984, 28–37.

[Pnu] Pnueli, A., The temporal logic of programs, *Proc IEEE FOCS*, 1977, 46–57.

[RS] Ramanujam, R. and Sheerazuddin, S., A counting temporal logic for services with unboundedly many clients, *Journal of Applied Logics*, 2011, to appear.

[Sal] Salomaa, A., Two complete axiom systems for the algebra of regular events, *JACM* **13**,1, 1966, 158–169.

[Thi] Thiagarajan, P.S., A trace based extension of propositional linear time temporal logic, *Proc IEEE LICS*, 1994, 438–447.

[Val] Valmari, A., A stubborn attack on state explosion, *LNCS* **531**, 1990, 156–165.