Locally linear time temporal logic

R. Ramanujam
The Institute of Mathematical Sciences
C. I. T. Campus
MADRAS 600 113
India
jam@imsc.ernet.in

Abstract

We study linear time temporal logics of multiple agents, where the temporal modalities are local. *These modalities not only refer to local next-instants and local eventuality, but also global views of agents at any local instant, which are updated due to communication from other agents. Thus agents also reason about the future, present and past of other agents in the system.*

The models for these logics are simple : runs of networks of synchronizing automata. Problems like gossipping in interconnection networks are naturally described in the logics proposed here. We present solutions to the satisfiability and model checking problems for these logics. Further, since formulas are insensitive to different interleavings of runs, partial order based verification methods become applicable for properties described in these logics.

1. Introduction

[1] The Propositional Temporal Logic of Linear Time (PTL) has proved to be a successful logical tool for specifying and reasoning about the behaviour of distributed programs [MP91]. The logic describes the temporal evolution of global states of the distributed system being specified. Reasoning about safety and liveness properties of systems in PTL is well understood by now.

However PTL formulas are *global* in structure, unlike the systems they specify, which are networks of automata built according to specific composition mechanisms. Global states of systems are typically tuples of local states of agents in the system, and reachability or avoidance of global states depends on possible sequences of local moves by individual agents or groups of synchronizing agents in the system. Such system structure is not reflected in the PTL formulas describing the structure.

On the other hand, when we look at PTL specifications of concurrent systems in the literature, often the formulas are of the form "if agent 1 will eventually reach l_1 then agent 2 will eventually reach l_2" etc. In fact, many interesting properties studied are expressible as boolean combinations of such local formulas.

Is it then simply a matter of structuring the set of atomic propositions by designating some to be local to a specific agent ? While this can be done, we believe that it is insufficient for the following reason: in distributed systems, an agent's view of the system at any point of time is local, and temporal reasoning is dependent on these views. Further, these views get updated with communication. This suggests the need for *local reasoning* in PTL.

In this paper, we study PTL with a view to reflecting the agent structure of distributed systems in the logic. The temporal modalities are local, and at a global level we use boolean combinations of local formulas. Thus temporal assertions of agents refer to their local time and global assertions put them together. Of course, this is expressively weak, so we also study how we can enrich the local modalities to make them more expressive. In the process, we obtain locally linear time temporal logics, where agents refer to the past, present and future of other agents depending on their current view of the system, which in turn changes with communication.

In linear time temporal logic, while past and future tenses are modal in nature, clearly the present tense is not modal. However, when time is only locally linear, we have an interesting *present tense modality*. In the logic studied here, we can have assertions of the form "necessarily now α holds at j" holding locally at an agent i. It turns out that this modality is closely related to *knowledge* modalities which have been studied extensively in the literature ([FHMV95]). The relationship between these knowledge and present tense

[1]Part of this work was done while I was visiting the Institut für Informatik und Praktische Mathematik, Universität Kiel, Germany. I thank Wolfgang Thomas for making the visit possible. I thank P.S.Thiagarajan for comments on the paper.

modalities is discussed in [R96].

The frames for our logics are simple : systems consisting of a fixed finite number of sequential components that synchronize by performing common actions together. Each component is a finite state automaton, and when a common ("handshake") action is performed, it is a 'lock-step' transition involving several automata at once. Despite this simplicity, the logics here are partial-order based, like $TrPTL$ ([T94]). We prefer to use local clock functions, thus leaving the underlying partial orders to be implicit, rather than interpret formulas on infinite traces or on event structures. (For a discussion of temporal logics on partial order models of concurrency like traces and event structures, see [LRT92], [T94] and [P95].)

The choice of 'top level parallelism' in the frames is deliberate to keep the agent structure simple. It is important to study hierarchical composition of such systems, but we do not address that issue here. Choosing synchronization as the basic communication mechanism between agents is again for simplicity. Shared variable mechanisms can be simulated in such models by making shared variables themselves to be processes. However, message based systems will require substantial additional work, and this opens up some interesting questions.

The work closest in spirit to this can be found in [T94], [T95] and [NP95]. In a sense, the logics here can be seen as 'bridging the gap' between $TrPTL^{\otimes}$ in [T95] (also [NP95]) and $TrPTL$ in [T94]. In the process, we obtain simpler proofs of decidability, and complete axiomatizations. Other related work include trace based μ-calculus in [N95], and branching time modalities in [ADP95].

In recent years, a number of researchers ([V90], [GW94]) have noted that there is an equivalence relation on runs of systems which equates different interleavings of the same run, and that this fact can be utilized for efficient verification of certain properties : those which hold for the entire equivalence class whenever they hold for one. Such properties are said to be *equivalence robust* ([P93]) or *trace consistent* ([T95]). Interestingly every formula in the logics proposed here is trace consistent, and hence the partial order based verification methods referred to above can be applied for our logics.

The type of local reasoning based on views described here makes it natural to describe variants of problems like broadcasting and gossipping in interconnection networks. Further, with some modifications, the trace consistent subsets of PTL identified in the papers above are subsumed in the logics here. However, we have not explored the expressiveness of these logics.

In this account, we mention only decidability results ; complete Hilbert-style axiomatizations can be provided for the logics, but we do not go into that issue here.

2. The basic logic L_0

In this section, we define the basic logic L_0 interpreted over systems with n agents. Locally the logic is PTL for each agent, and globally, we have boolean combinations of located formulas.

We consider an action-indexed version of PTL, since we will use synchronization on common actions as the means of expressing communication between agents. For this, we begin with the notion of a **distributed alphabet**. Fix a finite set of *locations* $Loc = \{1, \ldots, n\}, n > 0$. A distributed alphabet is a tuple $\widetilde{\Sigma} = (\Sigma_1, \ldots, \Sigma_n)$, 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. Given a distributed alphabet $\widetilde{\Sigma}$, we often speak of the set $\Sigma \stackrel{\text{def}}{=} \Sigma_1 \cup \ldots \cup \Sigma_n$ as the alphabet of the system. We also make implicit use of the associated function $loc : \Sigma \rightarrow 2^{\{1,\ldots,n\}}$ defined by $loc(a) \stackrel{\text{def}}{=} \{i \mid a \in \Sigma_i\}$. We will keep a fixed n and a fixed distributed alphabet for the discussion from now on.

Let $P = (P_1, \ldots, P_n)$ be a tuple of countable sets of atomic propositions. We will speak of $p \in P_i$ as a proposition of agent i. There is no reason for $P_i \cap P_j$ to be empty for distinct agents i and j, as there can be common meaningful propositions. ("Has terminated" is one such.) The **i-formulas** are defined as follows:

$$\Phi_i \ ::= \ p \ \in \ P_i \mid \neg\, \alpha \mid \alpha \ \vee \ \beta \mid \langle a \rangle\ \alpha, \ a \in \Sigma_i \mid \alpha \mathbf{U} \beta$$

The **global formulas** are defined from these using boolean combinations:

$$L_0 ::= \alpha @ i, \ \alpha \in \Phi_i, \ i \in Loc \mid \neg\, \phi \mid \phi_1 \ \vee \ \phi_2$$

Firstly, there are no modalities at the global level, and secondly, formulas in Φ_i cannot refer to any other agent j in the system. Further note that L_0 is parametrized by the distributed alphabet, and so will be the other logics in the next sections.

The frames for the logic are systems of n transition systems over $\widetilde{\Sigma} = (\Sigma_1, \ldots, \Sigma_n)$. As usual, a transition system over a finite nonempty alphabet Σ is a pair $T = (Q, \rightarrow)$, where Q is a **finite** set of states, $\rightarrow \subseteq Q \times \Sigma \times Q$ is the transition relation. We write $q \xrightarrow{a} q'$ to mean that $(q, a, q') \in \rightarrow$. A (finite or infinite) *run* from $q_0 \in Q$ is a sequence of transitions of the system : $q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} \ldots$. Let R_T denote the set of runs of T.

Let $\widetilde{\Sigma} = (\Sigma_1, \ldots, \Sigma_n)$ be a distributed alphabet. A **System** over $\widetilde{\Sigma}$ is a tuple $\mathcal{T} = (T_1, \ldots, T_n)$, where $T_i = (Q_i, \rightarrow_i)$ is a transition system over Σ_i, for $i \in Loc$. The **product transition** relation of the system $\rightarrow \subseteq (Q_1 \times \ldots \times Q_n) \times \Sigma \times (Q_1 \times \ldots \times Q_n)$ is defined as follows: $(q_1, \ldots, q_n) \stackrel{a}{\Rightarrow} (q'_1, \ldots, q'_n)$ iff $\forall i \in loc(a) : q_i \xrightarrow{a}_i q'_i$, and

$\forall j \notin loc(a) : q_j = q'_j$. We can use the product transitions to define the *global runs* of $\mathcal{T}$, and the set of all runs is denoted $R_{\mathcal{T}}$.

Given $\delta \in R_{\mathcal{T}}$, we can speak of the *i-projection* of δ, given by projecting down to the i^{th} components of states and actions in Σ_i. (Essentially, erase all states not in Q_i, and all actions not in Σ_i.) It is easy to check that this gives us an element of R_{T_i}, and we will denote this by $\delta\lceil i$. Note that $\delta\lceil i$ can be finite even if δ is itself infinite.

We can now define an equivalence relation $\sim$ on runs of $\mathcal{T}$: $\delta \sim \delta'$ iff $\forall i, \delta\lceil i = \delta'\lceil i$. This equates different interleavings of the same run. (Naturally, we can define $\lceil$ on strings in Σ^* and Σ^ω exactly as above, and this again gives an equivalence relation on strings ; by abuse of notation we use $\sim$ for this relation as well).

Further, for a fixed $\delta = x_0 \xrightarrow{a_0} x_1 \xrightarrow{a_1} \ldots$ (finite or infinite), we can define the **local clock functions** $\tau_{\delta,i} : \{0, 1, \ldots\} \rightarrow \{0, 1, \ldots\}$ by: $\tau_{\delta,i}(0) \stackrel{\text{def}}{=} 0$ and $\tau_{\delta,i}(k+1) \stackrel{\text{def}}{=} \tau_{\delta,i}(k) + 1$, if $a_k \in \Sigma_i$, and $\tau_{\delta,i}(k+1) \stackrel{\text{def}}{=} \tau_{\delta,i}(k)$, otherwise. When the context is clear, we will often omit δ from the subscript.

A **frame** for the logic is a pair $F = (\mathcal{T}, \delta)$, where $\mathcal{T} = (T_1, \ldots, T_n)$ is a system over $\widetilde{\Sigma}$ and $\delta \in R_{\mathcal{T}}$ is an infinite run of system $\mathcal{T}$. A **model** is a pair $M = (F, V)$, where F is a frame, and $V = (V_1, \ldots, V_n), V_i : Q_i \rightarrow 2^{P_i}$ give the local valuations. Given a model M, we define its i^{th} projection to be the tuple $M_i = ((T_i, \delta\lceil i), V_i)$.

Note that the equivalence relation $\sim$ on runs of $\mathcal{T}$ induces an equivalence relation on models based on the same system $\mathcal{T}$ and the same valuation.

We now define the semantics of formulas. This is first done for local formulas : $M_i, k \models_i \alpha$ denotes that the i-formula α is satisfied at (local) instant k for agent i. Below let $\delta\lceil i = q_0 \xrightarrow{a_0} q_1 \xrightarrow{a_1} \ldots$.

- $M_i, k \models_i p$ iff $p \in V_i(q_k)$.
- $M_i, k \models_i \neg\alpha$ iff $M_i, k \not\models_i \alpha$.
- $M_i, k \models_i \alpha \vee \beta$ iff $M_i, k \models_i \alpha$ or $M_i, k \models_i \beta$.
- $M_i, k \models_i \langle a\rangle\alpha$ iff q_{k+1} exists, $a_k = a$ and $M_i, k+1 \models_i \alpha$.
- $M_i, k \models_i \alpha\mathbf{U}\beta$ iff $\exists m$ such that $k \leq m, M_i, m \models_i \beta$ and for every $l : k \leq l < m : M_i, l \models_i \alpha$.

We now define the semantics of L_0 formulas. Since there are no global modalities, the formulas are not defined at every instant, but only at the initial state of the run.

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

As usual, we say that a formula ϕ is satisfiable if there exists a model M such that $M, 0 \models \phi$. A formula is valid if and only if its negation is not satisfiable. Derived connectives and dual modalities $(\alpha \wedge \beta, \alpha \supset \beta, [a]\alpha, \Diamond\alpha, \Box\alpha)$ are standard. We use abbreviations $\langle a\rangle_i\alpha$ for $(\langle a\rangle\alpha)@i$, $\Diamond_i\alpha$ for $(\Diamond\alpha)@i$ etc. Note that this simple logic already can express nontrivial global reachability. For instance if $a \in \Sigma_1 \cap \Sigma_2$, $(\langle a\rangle True)@1 \wedge (\Box[a]False)@2$ is not satisfiable.

Call a formula ϕ *trace consistent* if whenever a model M satisfies ϕ, then for every model $M' \sim M$, M' satisfies ϕ. It is trivial to see that every L_0-formula is trace consistent. This is a desirable property from the point of view of verification of properties.

In fact, L_0 is easily seen to be a subset of PTL, and [T95] characterizes L_0 as $PTL^{\otimes}$, the smallest subset of PTL containing i-consistent formulas for all $i \in Loc$, and closed under negation, disjunction and preserving equivalence. (An i-consistent formula is one, which if satisfied along δ also holds along every other δ' such that $\delta\lceil i = \delta'\lceil i$.)

The model checking problem for L_0 is defined as follows: given a finite system $\mathcal{T}$, a valuation V and a formula ϕ we are asked whether there is a model based on $\mathcal{T}$ and V satisfies ϕ. Both the satisfiability problem and the model checking problem for L_0 are decidable. (We can also define the model checking problem to ask whether *every* model based on the given system and given valuation satisfies the formula. In terms of decidability, both questions are equivalent, in the sense that a decision procedure for one version of the problem gives a decision procedure for the other version.) We will discuss decidability issues in Section 6.

3. Learning by synchronization

Though L_0 is a simple subset of PTL, we can already express interesting properties like deadlock freedom and liveness of actions in it. If we have local propositions of the form en_a to mean 'a is enabled at this local state', for $a \in \Sigma_i$, the fomula $(\Box\Diamond en_a)@i \supset (\Box\Diamond\langle a\rangle True)@i$ specifies fairness with respect to action a.

However, the fact that agents in the system *learn* about others' states on synchronization is not reflected in L_0-formulas. Indeed, the whole purpose of synchronizing is to exchange information, and hence we should be able to utilize this in reasoning. Below we propose a logic, where agents can "see" the local states of their partners in synchronization at the time they synchronize. This means that they can also see the other's future at that instant, and hence this gives a good deal of expressive power.

Formally, the logic L_1 is defined as follows:

$$\Psi_i ::= p \in P_i \mid \neg\alpha \mid \alpha \vee \beta$$

$$\mid \langle a\rangle\, \phi,\ a \in \Sigma_i,\ AV(\phi) \subseteq loc(a) \mid \alpha\mathbf{U}\beta$$

$$L_1 ::= \alpha@i,\ \alpha \in \mathbf{\Psi}_i,\ i \in Loc \mid \neg\,\phi \mid \phi_1\ \vee\ \phi_2$$

$AV(\phi)$, the *agent vocabulary* of ϕ is inductively defined in the obvious manner :

$$AV(\alpha@i) \stackrel{\text{def}}{=} \{i\};\ AV(\neg\phi) \stackrel{\text{def}}{=} AV(\phi);$$

$$AV(\phi_1 \vee \phi_2) \stackrel{\text{def}}{=} AV(\phi_1) \cup AV(\phi_2)$$

The semantics is suitably changed to reflect the restricted global view that an agent has access to. Below we give only the parts of the definition that require change. Fix a model $M = (F, V)$, where $F = (\mathcal{T}, \delta)$ and $\delta = x_0 \xrightarrow{a_0} x_1 \ldots$.

- $M_i, k \models_i \langle a\rangle\phi$ iff $\exists l : \tau_i(l) = k$, $\tau_i(l+1) = k+1$, $a_l = a$ and $M, l+1 \models \phi$.
- $M, k \models \alpha@i$ iff $M_i, \tau_i(k) \models_i \alpha$.
- $M, k \models \neg\phi$ iff $M, k \not\models \phi$.
- $M, k \models \phi_1 \vee \phi_2$ iff $M, k \models \phi_1$ or $M, k \models \phi_2$.

It requires a little bit of work now to show that every formula of L_1 is also trace consistent, but it can be done. This is indeed another motivation for the vocabulary restriction placed above.

Note that global dependencies can be expressed nicely in this logic. For instance, suppose the distributed alphabet $\widetilde{\Sigma}$ is such that $a \in \Sigma_1 \cap \Sigma_2$, $b \in \Sigma_2 \cap \Sigma_3$, $c \in \Sigma_3 \cap \Sigma_4$ and $p \in P_4$. Then a formula like $\langle a\rangle_1 \langle b\rangle_2 \langle c\rangle_3 p@4$ describes a sequence of dependent communications. In this manner one can 'walk down' the global run as in $TrPTL$ [T94]. Suppose in the situation above, we also have $d \in \Sigma_1 \cap \Sigma_3$. Then the formula $[a]_1((\Diamond\langle b\rangle True)@2 \wedge (\Diamond\langle d\rangle True)@3)$ can be read as a promise from 2 to 1 in a telephone conversation : "I will call 3, and if she has not talked to you by then, I'll ensure that she calls you afterwards"!

4. The present tense modality

Formulas in L_1 can express some aspects of learning due to synchronization. However, there are more varied ways of reasoning in distributed systems. An agent's view is necessarily partial, and at any instant of local time, the system can be in one of a number of global states. The agent can reason about properties invariant on these.

In particular, assertions like "no matter what the current global state is, agent j maintains p till next synchronization with me" are quite meaningful in the context of distributed protocols. This kind of **knowledge based reasoning** in distributed systems has been studied extensively in the literature. The logic we propose below is indeed a temporal logic of knowledge, where time is locally linear but globally partial. (We have studied these issues in more detail elsewhere [R 96]).

Interestingly, present tense can be a modality only in partial order based logics as the one studied here. Branching time logics reason about alternative tense, but in general, the present is only an instant, not a modality.

Formally, the logic L_2 is defined as follows:

$$\mathbf{\Omega}_i\ ::=\ p\ \in\ P_i \mid \neg\,\alpha \mid \alpha\ \vee\ \beta$$

$$\mid \langle a\rangle\ \phi,\ a \in \Sigma_i,\ AV(\phi) \subseteq loc(a)$$

$$\mid \alpha \mathbf{U} \beta \mid [now](\alpha@j), \alpha \in \mathbf{\Omega}_j$$

$$L_2 ::= \alpha@i,\ \alpha \in \mathbf{\Omega}_i,\ i \in Loc \mid \neg\,\phi \mid \phi_1\ \vee\ \phi_2$$

The semantics of the new modality is quite straightforward : at a local state an agent i asserts that $\alpha@j$ holds *necessarily now*, if in all the global states consistent with i's local state, the j-formula α holds. Since the agent i's view is the same in all runs in $[\delta]$, the invariance is with respect to all of them. This is easily expressed by :

- $M_i, k \models_i [now]\alpha@j$ iff $\forall\delta' \in [\delta], \forall l$, if $\tau_{\delta',i}(l) = k$, then $M', l \models \alpha@j$, where $M' = ((\mathcal{T}, \delta'), V)$.

The rest of the definitions are as before.

Note that there is a syntactic restriction on the form of the $[now]$ modality : while agent i can assert that right now a local formula holds at j, there is no global assertion of the form $[now]\phi$. This could have been included with the obvious semantics without violating trace consistency ; however, this makes the logic technically harder to analyse, so we content ourselves with the more restrictive definition.

5. The latest information about other agents

We started in L_0 with agents who cannot refer to other agents at all, and added the ability to refer to others' future in L_1, and their present in L_2. The obvious extension to look for is the ability to describe the others' past. This is (partly) done in the logic L_3 presented below.

We add assertions of the form $\ominus\alpha@j$ to i-formulas, where α is a j-formula. This is intended as an assertion by agent i that α held in the *latest* j-local state visible to i. In some sense this is the most reliable information about 'the current state' of j that i can have, and can make only indefinite assertions about the following j-states. In the language of event structures this is the local configuration of j contained in the local configuration of i at which the formula is specified. Note that i and j need not have actually synchronized at this past state; the j-state could be visible to i due to a sequence of communications with others, for instance, $j\&k$, $k\&l$ and then $l\&i$.

With this facility, 'gossip'-like situations can be described in the logic. For instance we can have scenarios where i sees that j has arrived at a certain state, and that k is unaware of it. Formally, the logic L_3 is defined as follows:

$$\Theta_i ::= p \in P_i \mid \neg\, \alpha \mid \alpha \vee \beta$$

$$\mid \langle a \rangle\, \phi,\ a \in \mathbf{\Sigma}_i,\ AV(\phi) \subseteq loc(a)$$

$$\mid \alpha \mathbf{U} \beta \mid [now]\alpha@j,\ \alpha \in \Theta_j \mid \ominus\, \alpha@j,\ \alpha \in \Theta_j$$

$$L_3 ::= \alpha@i,\ \alpha \in \Theta_i,\ i \in Loc \mid \neg\, \phi \mid \phi_1 \vee \phi_2$$

The semantics of the new modality is a little difficult to define. At a local state of an agent i, we need a way of identifying the latest j-state visible to i.

We accomplish this as follows: fix a run δ of the system $\mathcal{T}$ and $i \in Loc$. Let $k \geq 0$. We say that (δ', m) is an *i-minimal view at k in δ* if and only if $\delta' \in [\delta]$, $\tau_{\delta',i}(m) = k$, and $\forall \delta'' \sim \delta, \forall l$, if $\tau_{\delta'',i}(l) = k$, then $m \leq l$.

Proposition 5.1 *Given δ and $i \in Loc$, for every k, there exists an i-minimal view at k in δ. Further, if (δ_1, m_1) and (δ_2, m_2) are both i-minimal views at k in δ, then the following hold :*

- *$m_1 = m_2$ ($= m$, say).*
- *Let $\delta_1 = u_0 \xrightarrow{a_0} u_1 \ldots u_m \xrightarrow{a_m} \ldots$, and let $\delta_2 = v_0 \xrightarrow{b_0} v_1 \ldots v_m \xrightarrow{b_m} \ldots$. Then $u_m = v_m$, $(u_0 \xrightarrow{a_0} \ldots u_m) \sim (v_0 \xrightarrow{b_0} \ldots v_m)$, and $(u_m \xrightarrow{a_m} \ldots) \sim (v_m \xrightarrow{b_m} \ldots)$.*

The proposition assures us that at any instant of local time agent i sees a unique minimal past (upto interleaving equivalence), resulting in a unique global state. Let the j^{th} local state at that global state be s. Then s is the latest j-local state visible to i at the local instant k.

In some sense, this view is available to i by updating its information about other agents' local states with every synchronization. Such updating is described by a predicate: at a global instant m, among a pair of agents i and j, it asserts that i has at least as much information as j (and possibly better) about another agent k. This is denoted $latest_\delta(m, i, j, k)$ and defined as follows : (the δ subscript is left implicit below.)

- $latest(0, i, j, k)$ holds for all i, j, k.
- Suppose $x_m \xrightarrow{a} x_{m+1}$. Then $latest(m+1, i, j, k)$ holds only in the following cases :
  1. $\{i, k\} \subseteq loc(a)$, or
  2. $i \in loc(a)$, $k \notin loc(a)$ and there exists $l \in loc(a)$ such that $latest(m, l, j, k)$ holds, or
  3. $\{i, k\} \cap loc(a) = \emptyset$, $j \in loc(a)$, and for every $l \in loc(a)$, $latest(m, i, l, k)$ holds, or
  4. $\{i, j\} \cap loc(a) = \emptyset$, and $latest(m, i, j, k)$ holds.

We can check that $latest(m, k, j, k)$ holds for all k at any instant m. Further, if $i \notin loc(a)$ and $\{j, k\} \subseteq loc(a)$, then $latest(m+1, i, j, k)$ cannot hold.

We are now ready to define the meaning of $\ominus\alpha@j$, when asserted by agent i at the local k^{th} instant.

- $M_i, k \models_i \ominus\alpha@j$ iff $M', m \models \alpha@j$, where $M' = ((\mathcal{T}, \delta'), V)$ and (δ', m) is an i-minimal view at k in δ.

Note that the proposition above ensures that this definition is equivalent to one where we insist that $\alpha@j$ holds at *every* i-minimal view.

$\ominus$ refers only to an instant, and is not a modality. For referring to the past of agents, we need a local past modality. A logic L_4 with past operators added is easily defined. However, we have no technical results to report on this 'full' logic.

6. Decidability

Having introduced the logics, the obvious technical question is to ask whether the satisfiability problem for these logics is decidable. This is answered below, in the main result of this paper.

Theorem 6.1 *1. Every formula in L_3 (and hence in L_0, L_1, L_2) is trace consistent.*

2. *The satisfiability problem for L_2 (and hence for L_1 and L_0) is decidable in time ($2^{O(m)}$), where m is the length of the formula to be decided.*
3. *The satisfiability problem for L_3 is decidable in time ($2^{O(m_1 m_2)}$), where m_1 is the length of the formula to be decided, and m_2 is the number of agents mentioned in it.*
4. *The model checking problem for $L_i, i \in \{0, 1, 2, 3\}$, given a system of size k and a formula $\phi \in L_i$, can be solved in time $O(k.t_i)$ where t_i is the time taken by the decision procedure above for satisfiability of ϕ.*

In fact each of the logics also admits a complete axiomatization. What is particularly interesting about the axiom systems is that we can present them in two layers : at the local level, we have a system for each agent, and at the global level, a system to put them together. This requires a separate detailed account, so we omit it from this presentation.

The decidability proofs follow standard automata theoretic techniques developed by [VW86]. Instead of using Product Büchi automata or Asynchronous Büchi automata as done in [T94, T95], we keep the automata structure simple, but use product construction and partial order methods in the analysis.

Recall that a (nondeterministic) Büchi automaton over a finite nonempty alphabet Σ is given as a tuple $A = (S, \rightarrow, I, D)$, where S is a finite set of states, $\rightarrow \subseteq (S \times \Sigma \times S)$ is the transition relation, $I \subseteq S$ are the initial states and $D \subseteq S$ is the set of designated states to be visited infinitely often.

For our purposes, we need a slight variation on these automata, which we continue to call Büchi automata, for convenience. Such an automaton is a tuple $A = (S, \rightarrow, I, (D_1, \ldots, D_k))$, where $S, \rightarrow$ and I are as before and for all $1 \leq l \leq k, D_l \subseteq S$. A run of A over a word $w = a_1 a_2 \ldots \in \Sigma^\omega$ is an infinite sequence $s_0, s_1, \ldots$, where $s_0 \in I, \forall i \geq 1, (s_{i-1}, a_i, s_i) \in \rightarrow$. A run is *accepting* if for every $l, 1 \leq l \leq k$, there exist infinitely many j such that $s_j \in D_l$. $w \in \Sigma^\omega$ is accepted by A if there is an accepting run of A over w. The set of accepted words of A is denoted $L(A)$.

The technique requires the construction of a Büchi automaton A_α, where α is the formula whose satisfiability is in question, such that $L(A_\alpha) \neq \emptyset$ if and only if α is satisfiable. The nonemptiness problem for (our version of) Büchi automata is solvable in time linear in the size of the automaton, hence we get a decision procedure for the satisfiability of α whose complexity is determined by the size of A_α.

We first give the proof for L_2, and mention the complications that arise in the context of L_3. For technical convenience, assume that Θ_i is enriched with $\bigcirc\alpha, \alpha \in \Theta_i$, where $\bigcirc\alpha$ is semantically equivalent to $\bigvee_{a \in \Sigma_i} \langle a \rangle \alpha$. Recall the notation $M_i\alpha$, for the formula $(M\alpha)@i$, where $M\alpha$ is a modal i-formula.

We can define, for any L_2-formula ϕ and any $\alpha \in \Theta_i$, the sets of their subformulas $CL(\phi)$ and $CL_i(\alpha)$ by simultaneous induction in such a way that :

- if $\alpha@i \in CL(\phi)$ then $\{\beta@i | \beta \in CL_i(\alpha)\} \subseteq CL(\phi)$.
- if $\langle a \rangle_i \phi' \in CL(\phi)$ then $CL(\phi') \subseteq CL(\phi)$ and $\forall j \in loc(a), \langle a \rangle_j True \in CL(\phi)$.
- if $\langle a \rangle \phi' \in CL_i(\alpha)$ then $\{\langle a \rangle \beta@i,\ \beta |\ \beta@i \in CL(\phi')\} \subseteq CL_i(\alpha)$.
- if $[now]_i \beta@j \in CL(\phi)$ then $\beta@j \in CL(\phi)$.
- if $\langle now \rangle_i \beta@j \in CL(\phi)$ then for every $k \in Loc$, $\langle now \rangle_k \beta@j \in CL(\phi)$.
- if $\phi' \in CL(\phi)$ then $\neg\phi' \in CL(\phi)$; a similar condition holds for $CL_i(\alpha)$ and here $\neg\neg$ is treated as identity.
- if $\beta_1 \mathbf{U} \beta_2 \in CL_i(\alpha)$ then $\{\beta_1, \beta_2, \bigcirc(\beta_1 \mathbf{U} \beta_2)\} \subseteq CL_i(\alpha)$.
- for all $i \in Loc, True@i, \bigcirc_i True \in CL(\phi)$.

It can be checked that $CL(\phi)$ is linear in the size of ϕ. Define U_ϕ to be the set of all $(\alpha \mathbf{U} \beta)@i$ formulas in $CL(\phi)$, $i \in Loc$. Similarly let V_ϕ be the set of all $\langle now \rangle_i \alpha@j$ formulas in $CL(\phi)$, $i, j \in Loc$. Below we will be interested in the set $V' = V_\phi \times 2^{Loc} \times \{0, 1\}$. In particular, we use $v \subseteq V'$ such that whenever $(\gamma, z, d) \in v$, if $z = \emptyset$ then $d = 0$, and whenever both $(\gamma, z_1, d_1) \in v$ and $(\gamma, z_2, d_2) \in v$, then if $z_1 = z_2$ then $d_1 = d_2$ and if $z_1 \neq z_2$, then either $z_1 = \emptyset$ or $z_2 = \emptyset$. Let $\mathcal{V}$ denote the set of such subsets of V'. For $v \in \mathcal{V}$, we use the notation v_{ij} to denote the set of all triples of the form $(\langle now \rangle_j \alpha@i, z, d)$ in v. Further we use the notation $v\lceil 1$ to mean the set of all formulas γ such that for some $z, d, (\gamma, z, d) \in v$. Similar remarks apply for the notation $v\lceil 2$ and for $v\lceil 3$.

Fix a formula $\phi_0 \in L_2$. We will refer to $CL(\phi_0)$ simply as CL. Call $X \subseteq CL$ an *atom* if it is propositionally consistent both globally and locally for every i, and satisfies the conditions :

- if both $\langle a \rangle_i \phi_1$ and $\langle b \rangle_i \phi_2$ are in A, then $a = b$.
- $(\alpha \mathbf{U} \beta)@i \in A$ iff $\beta@i \in A$ or both $\alpha@i$ and $\bigcirc_i \alpha \mathbf{U} \beta$ are in A.
- if $[now]_i \alpha@j \in A$ for some i, then $\alpha@j \in A$.
- if $\langle a \rangle_i \phi \in A$, then $\bigcirc_i True \in A$.

We can now define the automaton for formula ϕ_0 by $A_{\phi_0} \stackrel{\text{def}}{=} (S, \rightarrow, I, (F_1, F_2, < F_3^{ij} | i, j \in Loc, i \neq j >))$, where

- $S = \{(A, e, u, v) \mid A$ is an atom, $e \subseteq Loc$ such that if $i \in e$, then $\bigcirc_i True \in A, u \subseteq U_{\phi_0} \cap A, v \in \mathcal{V}$ such that $v\lceil 1 \subseteq A$ and if $(\langle now \rangle_j \alpha@i, z, d) \in v$ then $\alpha@i \notin A\}$.
- $(A, e, u, v) \xrightarrow{a} (A', e', u', v')$ iff :
  1. $\forall i \in loc(a), (\forall \bigcirc_i \alpha \in CL,\ \bigcirc_i \alpha \in A$ iff $\alpha@i \in A')$ and $(\forall \langle a \rangle_i \phi \in CL,\ \langle a \rangle_i \phi \in A$ iff $\phi \in A')$.
  2. $\forall j \notin loc(a), \alpha@j \in A$ iff $\alpha@j \in A'$.
  3. if $e = \emptyset$, then $e' = \{i | \bigcirc_i True \in A'\}$, and otherwise, $e' = e - loc(a)$.
  4. if $u = \emptyset$, then $u' = \{(\alpha \mathbf{U} \beta)@j \in A' \mid \beta@j \notin A',\ j \in Loc\}$ and otherwise, $u' = \{(\alpha \mathbf{U} \beta)@j \in u \mid \beta@j \notin A, j \in Loc\}$.
  5. $\forall i, j \in loc(a)$, $v_{ij} = \emptyset$.
  6. If $\exists j \notin loc(a)$: $(\langle now \rangle_j \alpha@i, z, 1) \in v$ and $z \cap loc(a) = \emptyset$, then $\alpha@i \notin A'$.
  7. Suppose that $i \notin loc(a)$. Then $(\forall k \in loc(a), \langle now \rangle_k \alpha@i \in A)$ iff $(\forall k \in loc(a), \langle now \rangle_k \alpha@i \in A')$; Further, $\forall j \in loc(a)$, if

$(\langle now\rangle_j \alpha@i, z, d) \in v$, then $z = \emptyset$, $d = 0$ and for all $k \in loc(a)$, $(\langle now\rangle_k \alpha@i, \emptyset, 0) \in v$, and $\langle now\rangle_k \alpha@i \in A'$.

8. For $\{i, j\} \subseteq loc(a)$, $v'_{ij}\lceil 1 = \{\langle now\rangle_j \alpha@i \in A' \mid \alpha@i \notin A'\}$, and $v'_{ij}\lceil 2 = \{loc(a), \emptyset\}$.
9. For $j \in loc(a)$ and $i \notin loc(a)$, $v'_{ij} = v_{ij} \cup \{(\langle now\rangle_j \alpha@i, loc(a), d) | \langle now\rangle_j \alpha@i \in A'$, $\alpha@i \notin A'$, $\exists k \in loc(a)$ such that $\langle now\rangle_k \alpha@i \notin A$ and if $d = 1$ then $\langle now\rangle_j \alpha@i \notin A\}$.
10. Let $j \notin loc(a)$. If $i \notin loc(a)$ as well, then $v'_{ij} = v_{ij}$. If $i \in loc(a)$, $v'_{ij} = (v_{ij} - \{(\langle now\rangle_j \alpha@i, z, d) \in v \mid (\alpha@i \in A'$ and $z = \emptyset)$ or $z \cap loc(a) \neq \emptyset\}) \cup \{(\langle now\rangle_j \alpha@i, z \cup loc(a), d) \mid \alpha@i \notin A', (\langle now\rangle_j \alpha@i, z, d) \in v$ and $z \cap loc(a) \neq \emptyset\}$.

- $I = \{(A, \emptyset, \emptyset, v) \in S | \phi_0 \in A,\ v = \{(\gamma, \emptyset, 0) | \gamma \in ((V_{\phi_0} \cap A) - \{\langle now\rangle_i \alpha@j | \alpha@j \in A\})\}$.
- $F_1 = \{(A, e, u, v) \in S | e = \emptyset\}$.
- $F_2 = \{(A, e, u, v) \in S | u = \emptyset\}$.
- $F_3^{ij} = \{(A, e, u, v) \in S | v_{ij} = \emptyset\}$.

We need to show that $L(A_{\phi_0}) \neq \emptyset$ if and only if ϕ_0 is satisfiable. We describe only the argument relating to the $\langle now\rangle$ modality.

The main problem with the $\langle now\rangle$ modality is that it refers to other runs which are equivalent to δ, the run on which the model is based. Note that the automaton associated with the formula in itself works only with δ, and has no access to other runs in the equivalence class $[\delta]$. Suppose the run begins with an a-move followed by a b-move, and $loc(a) \cap loc(b) = \emptyset$, where $a \in \Sigma_1$ and $b \in \Sigma_2$. Both 1 and 2 can make $\langle now\rangle$ assertions about the local states of each other. However, every sequence of local states that they go through in $\delta' \in [\delta]$ is available in δ itself. This is the property crucially used for checking that $\langle now\rangle$ properties are satisfied.

However, checking for $\langle now\rangle$ formulas within runs is problematic, and quite unlike checking eventuality requirements. The latter requires checking only the future instants, whereas at any instant in the run, the 'present' is an interval bracketing it. When a $\langle now\rangle$ requirement comes up, the automaton may already have passed the state at which it was satisfied. Moreover, when an agent i makes a move, the $\langle now\rangle$ requirements before the move and those after the move may refer to different intervals. Indeed, the same formula $\langle now\rangle_i \alpha@j$ may be asserted both before and after the move referring to very different requirements : this happens when the assertion before the move refers to a j-state which is concurrent but not encountered yet within the run, and the assertion after refers to a j-state which is causally dependent on the current i-state. This means that it is not sufficient to simply carry along a set of $\langle now\rangle$ formulas as we do for $\mathbf{U}$ formulas. It is to distinguish these two kinds of requirements that we carry along a second componet in the set v. Checking dependency is achieved by taking along the set of agents synchronizing at the instant when the future dependence requirement comes up, and adding further synchronizing agents to the set later on. A synchronization chain of the form $(i, k)(k, l)(l, j)$ generates the sequence of sets $\{i, k\}, \{i, k, l\}, \{i, k, l, j\}$. We thus find that the j-move seen later must be dependent on the i-move which gave rise to the requirement. Further, consider an i-move, where before the move, agent i asserts that $[now]\neg\alpha@j$ and after the move, asserts that $\langle now\rangle \alpha@j$. In the run, after this move, eventually a dependent j-move will be encountered when α holds at j, but before that there may be independent j-moves encountered, but in all these j-local states $\alpha@j$ cannot hold, since these moves are in the present of agent i. It is to capture this that we carry along the third component in v.

In general, when the automaton proceeds down a run and a new requirement of the form $\langle now\rangle_j \alpha@i$ appears, there is no way to tell whether this must be satisfied using dependent or independent moves of i, and the automaton simply guesses one. This explains the complexity of the update conditions for v' in terms of v in the automaton construction.

Consider a string $w = a_1 a_2 \dots a_k$ in Σ^*, $k > 1$. We say that a_1 and a_k are *concurrent* in w if $loc(a_1) \cap loc(a_k) = \emptyset$, $w \sim w' \sim w''$ where $w' = a_k a_1 a_2 \dots a_{k-1}$, and $w'' = a_2 \dots a_k a_1$. Given this, in a run $\delta = x_0 \xrightarrow{a_1} x_1 \xrightarrow{a_2} \dots$ we say that k and l ($l > k$) are concurrent in δ, if a_{k+1} and a_l are concurrent in the string $a_{k+1} \dots a_l$. Let C_δ denote this relation. Suppose $l > k$ and $(k, l) \notin C_\delta$. We say that l is *dependent* on k, and we can check that there exist $k = j_1 < j_2 < \dots j_m = l$ such that $loc(a_{j_p}) \cap loc(a_{j_{p+1}}) \neq \emptyset$, for $1 \leq p < m$. In this case, we will say that there is a dependency path from k to l.

Proposition 6.2 *Suppose $\delta \sim \delta'$, and $i \in Loc$ such that for some k, l, $\tau_{\delta,i}(k) = \tau_{\delta',i}(l)$. Then for every $j \in Loc$, there exists m_j such that $\tau_{\delta,i}(k) = \tau_{\delta,i}(m_j)$ or $(k, m_j) \in C_\delta$, or $(m_j, k) \in C_\delta$, and the j^{th} local state is the same in $\delta'(l)$ and $\delta(m_j)$.*

Proposition 6.3 *Let M be a model based on a run δ and $k \geq 0$. $M, k \models \langle now\rangle_i \alpha@j$ if and only if $\exists l$ such that $M, l \models \alpha@j$, where, either $\tau_{\delta,i}(k) = \tau_{\delta,i}(l)$ or $(k, l) \in C_\delta$ or $(l, k) \in C_\delta$.*

Call l (in the proposition) above a *witness* for the formula $\langle now\rangle_i \alpha@j$ at the instant k. In a similar fashion, we can define witnesses for formulas of the form $\langle a\rangle_i True$ and

$(\alpha \mathbf{U} \beta)@i$. In reducing satisfiability to emptiness checking for the automaton, the crucial step is showing that a run is accepting if and only if witnesses exist for all formulas of these three types. When we show that every satisfiable formula generates an accepting run, the given model for the formula includes witnesses and we will refer to these as model witnesses. On the other hand, when the automaton has accepting runs, we will argue that it creates witnesses for every requirement, and refer to these as automaton witnesses.

Lemma 6.4 *If ϕ_0 has a model then $L(A_{\phi_0}) \neq \emptyset$.*

Proof: Let $M = ((\mathcal{T}, \delta), V)$, where $\mathcal{T} = (T_1, \ldots, T_n)$ is a system over $\widetilde{\Sigma}$, and $M, 0 \models \phi_0$. Let the given run δ be over the word $a_1 a_2 \ldots$. Define the sequence $(A_0, e_0, u_0, v_0) \xrightarrow{a_1} (A_1, e_1, u_1, v_1) \xrightarrow{a_2} \ldots$ as follows : for all k, $A_k = \{\phi' \in CL \mid M, k \models \phi'\}$; $e_0 = \emptyset$, and e_{k+1} is obtained from e_k in the same way e' is defined in terms of e in condition (3) on automaton transitions above. Similarly, $u_0 = \emptyset$ and u_{k+1} is defined from u_k using condition (4).

The definition of v_k is semantic. Let $\gamma = \langle now \rangle_j \alpha@i$ and $wit = \{l | l \text{ is a model witness for } \gamma \text{ at } k\}$. $(\gamma, z, d) \in v_k$ if and only if: for every $l \in wit$, $l > k$, if $z \neq \emptyset$, then ($\exists l \in wit : (k, l) \notin C_\delta$ and $m \in z$ iff there is a dependency path from j to m after the last j-action before k) and if $d = 1$ and l is the earliest dependent witness in wit, then for every $p < l$, $M, p \not\models \alpha@i$.

It can be easily checked that each of the quadruples is a state in our automaton, and the fact that each A_k is an atom follows from the semantics of formulas. We use the notation $x_k.A$, $x_k.v$ etc to denote the sets A_k, v_k etc respectively. (Below we often abuse notation to write $\gamma \in x_k.v$ when we mean that for some z, d, $(\gamma, z, d) \in x_k.v$.)

We need to show that every move in the sequence defines a legal transition in the automaton. Consider the move $x_k \xrightarrow{a_{k+1}} x_{k+1}$ in the sequence. (For convenience, we write a for a_{k+1}.) We discuss the verification of conditions 5 through 10. Condition 5 follows immediately from the following proposition and the definition of v_k above.

Proposition 6.5 *Let M be a model based on a run δ, $k \geq 0$, $x_k \xrightarrow{a_{k+1}} x_{k+1}$ and $\{i, j\} \subseteq loc(a_{k+1})$. Let γ be the formula $\langle now \rangle_j \alpha@i$. $M, k \models \gamma$ if and only if there exists a witness $l \leq k$ for γ at k.*

Condition 6 is easily seen to follow from the definition of v_k. To verify condition 7, suppose that $i \notin loc(a)$ and that for every $m \in loc(a)$, $M, k \models \gamma_m$, where $\gamma_m = \langle now \rangle_m \alpha@i$. Now suppose that for some m, $M, k+1 \not\models \gamma_m$. Clearly $M, k+1 \not\models \alpha@i$, and since $i \notin loc(a)$, $M, k \not\models \alpha@i$ as well. Now the latest witness l for γ_m at k must be before k, and must be one such that $(l, k) \notin C_\delta$. Then there must be a last instant after l when i moves, say k', $M, k' \models \neg\alpha@i$, and m moves (dependently) after k'. But then $M, k \models [now]_m \neg\alpha@i$, a contradiction. Hence, for every $m \in loc(a)$, $M, k+1 \models \gamma_m$ as required. The other direction is proved similarly.

Now further suppose that $j \in loc(a)$, and let $\gamma = \langle now \rangle_j \alpha@i$, $(\gamma, z, d) \in v_k$. Now if l is the earliest witness for γ at k, then $l > k$. Indeed, $l > k+1$ since $k+1$ is not a witness either (because γ is in v, $\alpha@i \notin x_k.A$ and since $i \notin loc(a)$, $\alpha@i \notin x_{k+1}.A$). But then z must be empty; otherwise, l must be a dependent witness on k and $\tau_{\delta,j}(k) = \tau_{\delta,j}(l)$, which is impossible, since $\tau_{\delta,j}(k) < \tau_{\delta,j}(k+1) \leq \tau_{\delta,j}(l)$. If $z = \emptyset$, then $d = 0$ as well, by definition. Now, l is a concurrent witness for k. But then it is a concurrent witness for γ_m, for every $m \in loc(a)$, $\gamma_m = \langle now \rangle_m \alpha@i$. Hence $(\gamma_m, \emptyset, 0)$ must be in v for every such m, and since l is a witness for $k+1$ as well, these formulas must also be in $x_{k+1}.A$.

We now check conditions relating $x_{k+1}.v_{ij}$ and $x_k.v_{ij}$. If $\{i, j\} \cap loc(a) = \emptyset$, these two sets are obviously the same, so there are three cases. For the first one, assume that $\{i, j\} \subseteq loc(a)$. Since i and j have just synchronized, the 'present' of j begins at $k+1$. Clearly, for every triple $(\gamma, z, d) \in x_{k+1}.v$, a witness l can be only after $k+1$ (hence $l > k+1$) because $\alpha@i \notin x_{k+1}.A$. Conversely, for every such formula whose witness comes after $k+1$, $\alpha@i$ cannot hold now, and hence would be present in $x_{k+1}.v$. If $(\gamma, z, d) \in x_{k+1}.v$ and $z \neq \emptyset$, the only dependent path now is from agents in $loc(a)$ to themselves and hence $z = loc(a)$.

Now consider the case when $j \in loc(a)$, and $i \notin loc(a)$. Suppose $M, k+1 \models \gamma$ and every witness for γ at $k+1$ is after $k+1$. Now there are two possibilities : first assume that $M, k \not\models \gamma$. The earliest witness for $k+1$ must be a dependent one, and further at every instant after $k+1$ if i makes an independent move, $\alpha@i$ cannot hold at that instant. Thus $(\gamma, loc(a), 1) \in x_{k+1}.v$ in this case. Now consider the other possibility : $M, k \models \gamma$. There are again two cases : there is a witness $l \leq k$ for γ at k, or every witness for γ at k is after k (and hence after $k+1$). In the latter case, $\gamma \in x_k.v \subseteq x_{k+1}.v$. In the former, we must have k dependent on some l' when i made a move, and after which $\alpha@i$ never held. But then, due to dependency, there must be some $m \in loc(a)$ such that $M, k \models [now]_m \neg\alpha@i$. In this situation the earliest witness for γ at $k+1$ must be a dependent one and we see that $(\gamma, loc(a), 0) \in x_{k+1}.v$. We have covered all the cases, and formulas of this kind exactly make up $x_{k+1}.v$, verifying condition 9.

Let $j \notin loc(a)$ and $i \in loc(a)$. Firstly, if $M, k+1 \models \gamma$, we also have $M, k \models \gamma$. Suppose every witness for γ at $k+1$ comes after $k+1$. Clearly, $\alpha@i \notin x_{k+1}.A$, and hence there must be a triple $(\gamma, z, d) \in x_k.v$, and by construction, it gets into $x_{k+1}.v$ as well. On the other hand, suppose that $(\gamma, z, d) \in x_k.v$ such that $z \cap loc(a) = \emptyset$ and either $\alpha@i \notin x_{k+1}.A$ or $z \neq \emptyset$. We argue that $k+1$ is not a witness for γ at k. If it were, then either it is an independent witness, in which case we would have $\alpha@i \in x_{k+1}$ and $z = \emptyset$, a contradiction, or it is a dependent witness, in which case $\alpha@i \in x_{k+1}$ and $z \cap loc(a) \neq \emptyset$, again a contradiction. Thus every witness for γ at k comes after $k+1$, and therefore, every witness for γ at $k+1$ comes after $k+1$ as well. Now suppose $(\gamma, z, d) \in x_k.v$ such that $z \cap loc(a) \neq \emptyset$ and $\alpha@i \notin x_{k+1}.A$. In this case, a dependent witness comes after $k+1$, and since the dependency path is extended now, (γ, z, d) cannot be in $x_{k+1}.v$, but $(\gamma, z \cup loc(a), d)$ must be, and this verifies condition 10.

Since we have shown that every move in the constructed sequence is a transition of the automaton, the sequence constitutes a run of the automaton. We need to show that this is an accepting run and hence that $L(A_{\phi_0}) \neq \emptyset$. The crucial step consists in showing that some state in F^{ij} is visited infinitely often, for each i and j. For this, define the map, for each $i, j \in Loc$, $i \neq j$, $\chi_{ij}: \mathcal{N} \to \mathcal{N} \times (\mathcal{N} \cup \{*\})$ by $\chi_{ij}(k) \stackrel{\text{def}}{=} (p, q)$, where $p = 0$ or p is the latest instant such that $p \leq k$, $i \in loc(a_p)$, and for some $r \leq k$, $j \in loc(a_r)$, and r is dependent on p. Similarly, $q = *$ or q is the earliest instant such that $q \geq k$, $i \in loc(a_q)$, and for some $r \geq k$, $j \in loc(a_r)$ and q is dependent on r. It can be easily checked that if l is a witness for a formula $\langle now \rangle_j \alpha@i$ at k, and $\chi_{ij}(k) = (p, q)$ then $p \leq l$ and if $q \neq *$, then $l \leq q$.

Now consider the set $v' = x_k.v_{ij}$. Note that new formulas are added to this set only when i and j synchronize, or when a future j action is dependent on an i action. If $\chi_{ij}(k) = (p, *)$, then in this case the agents i and j proceed independently for ever after k and we can show that v' decreases monotonically. Otherwise, if $\chi_{ij}(k) = (p, q)$, we show that v' becomes empty before q. For this, note that between the instants k and q, once a formula is removed from v', it never enters again, that every formula is eventually removed because there is a witness before q and that the size of v' is bounded by the size of V_{ϕ_0}. This completes the proof of the lemma. □

Lemma 6.6 *If $L(A_{\phi_0}) \neq \emptyset$, then ϕ_0 is satisfiable.*

Proof: Suppose that the automaton accepts some string. Below fix an accepting run $\delta = x_0 \xrightarrow{a_1} x_1 \xrightarrow{a_2} \ldots$ of A_{ϕ_0}. If $x_j = (A_j, e_j, u_j, v_j)$, we again use the notation $x_j.A$ to denote A_j etc.

Note that we can define the map τ and C_δ for the automaton run δ as well, and hence we can speak of automaton witnesses : if $\gamma = \langle now \rangle_j \alpha@i \in x_k.A$, we say that l is an automaton witness for γ at k if $\alpha@i \in x_l.A$, and $\tau_{\delta,j}(k) = \tau_{\delta,i}(l)$ or $(k, l) \in C_\delta$ or $(l, k) \in C_\delta$. The central observation is the following :

Claim: Let γ be the formula $\langle now \rangle_j \alpha@i$. For $k \geq 0$, $\gamma \in x_k.A$ if and only if there exists an automaton witness for γ at k.

We omit the proof that when such a witness exists, γ is indeed in $x_k.A$; the crucial property used is this : whenever $i \notin loc(a)$, $(\forall m \in loc(a) \langle now \rangle_m \alpha@i \in x_k.A)$ iff $(\forall m \in loc(a) \langle now \rangle_m \alpha@i \in x_{k+1}.A)$. The other direction is proved in two steps.

- For $k \geq 0$, if $\gamma \in (x_k.A - x_k.v)$ then there exists $l \leq k$ such that l is an automaton witness for γ at k.
- For $k \geq 0$, if $(\gamma, z, d) \in x_k.v$, then there exists a witness $l > k$ for γ at k.

The first step is proved by induction on k. The base case, when $k = 0$, is easily seen to hold. For the induction step, suppose $\gamma \in (x_{k+1}.A - x_{k+1}.v)$. If $\alpha@i \in x_{k+1}.A$, the statement is trivial, so suppose not. We first consider the case when $j \notin loc(a)$. Clearly, $\gamma \in (x_k.A - x_k.v)$ as well, by construction (due to condition 10). By inducion hypothesis, there exists a latest witness $l \leq k$ for γ at k. If j makes no moves between l and k, $\tau_{\delta,j}(l) = \tau_{\delta,j}(k) = \tau_{\delta,j}(k+1)$ and we are done. If j made any move between l and k, it must be an independent move (otherwise, if m were the latest such instant when j moved, there is no witness for γ at $m+1$ and by induction hypothesis, either $\gamma \notin x_{m+1}.A$ or $\gamma \in x_{m+1}.A \cap x_{m+1}.v$; in either case we can show a contradiction). But then it is easy to see that l must be a witness for γ at $k+1$ as well.

We now consider the case when $j \in loc(a)$. If $i \in loc(a)$ as well, by construction, $\gamma \in x_{k+1}.A - x_{k+1}.v$ iff $\alpha@i \in x_{k+1}.A$ and we are done. Hence suppose $i \notin loc(a)$. For nontriviality, again assume that $\alpha@i \notin x_{k+1}.A$. By construction (condition 9), we see that $\gamma \notin x_k.v$. Again if $\gamma \notin x_k.A$, condition 9 ensures that $\gamma \in x_{k+1}.v$ (with $d = 1$), so we have $\gamma \in x_k.A - x_k.v$. By conditions 7 and 9, we further see that for all $m \in loc(a)$, $\gamma_m \in x_k.A - x_k.v$, where $\gamma_m = \langle now \rangle_m \alpha@i$. There exists $p \in loc(a)$ such that the latest i-state visible to p is the latest among i-states visible to agents in $loc(a)$. Let l_p be the witness for γ_p given by the induction hypothesis. It is easily seen that l_p is also a witness for γ at $k+1$. This completes the induction.

We now proceed to the second step; that is, we now show that for $k \geq 0$, if $(\gamma, z, d) \in x_k.v$, then there exists a witness $l > k$ for γ at k, and if $z \neq \emptyset$, l is a dependent witness. Now suppose that $(\gamma, z, d) \in x_k.v$. First consider the case when $z = \emptyset$, which implies that $d = 0$. Since v_{ij} eventually becomes empty, there exists an earliest $l > k$ such that $\gamma \in x_{l-1}.v$, but $\gamma \notin x_l.v$. Clearly $i \in loc(a_l)$ and $j \notin loc(a_l)$ (due to condition 5). If $\tau_{\delta,j}(k) = \tau_{\delta,j}(l)$, or if $(k, l) \in C_\delta$, we are done, so assume otherwise. Then there is a j-move at p between k and l, such that l is dependent on p. Note that $\gamma \in x_p.v$. Let the dependency chain between p and l be at instants $p = q_1, q_2, ..., q_m = l$. Consider $j' \in loc(a_p) \cap loc(a_{q_2})$: $\langle now \rangle_{j'} \alpha@i \in x_p.v$. Thanks to condition 7, we can check that $\langle now \rangle_{j'} \alpha@i \in x_{q_2}.v$, and that $\langle now \rangle_{j''} \alpha@i \in x_{q_2}.v$, where $j'' \in loc(a_{q_2}) \cap loc(a_{q_3})$ and so on. (Note that $\neg\alpha@i$ holds throughout.) Finally we arrive at $l-1$, where we have some i' such that $\langle now \rangle_{i'} \alpha@i \in x_{l-1}.v$, and $\{i', i\} \subseteq loc(a_l)$. But then $v_{i'i} \neq \emptyset$, contradicting condition 5. Thus, l is the required concurrent witness.

Now consider the situation when $z \neq \emptyset$. Again, as v_{ij} becomes empty eventually, let l be the earliest instant when any triple of the form (γ, z', d) is removed from $x_l.v$. From condition 9 we can check that $z' \cap loc(a) \neq \emptyset$ and $\alpha@i \in x_l.A$. (If the latter were not true, this triple would be merely replaced by another.) Let k' be the instant $(k' \leq k)$ that any triple of the form (γ, z'', d) entered the set. Clearly, $z'' = loc(a_{k'})$. We can then construct a dependency chain as above, such that at each stage $loc(a_q)$ is added, ending with z'. It is easy to show then that l is a dependent witness for γ at k.

This establishes the claim, and only model construction remains. Define transition systems $T_i = (Q_i, \rightarrow_i)$ by $Q_i = \{x_k.A \cap CL_i | k \geq 0\}$ and for $a \in \Sigma_i$, $X \xrightarrow{a}_i Y$ if and only if $\exists k$ such that $x_k \xrightarrow{a} x_{k+1}$, $x_k.A \cap CL_i = X$ and $x_{k+1}.A \cap CL_i = Y$. Now consider the product system $\mathcal{T} = (T_1, \ldots T_n)$. Thanks to conditions 3 and 4, we can easily check that the automaton run induces a run δ_T on this transition system: the k^{th} element of δ_T is the tuple $(x_k.A \cap CL_1, \ldots, x_k.A \cap CL_n)$. The valuation is obvious: $V_i(X) = X \cap P_i$. We now have a model M. We show by induction that for all k, for every $\phi' \in CL$, $M, k \models \phi'$ iff $\phi' \in x_k.A$. The important case of the $\langle now \rangle$ modality in the induction step is proved using the claim above. $\Box$

This completes the proof of decidability for L_2. The proof for L_3 requires similar analysis for the $\ominus\alpha@j$ modality. Firstly, we cannot work with CL above but need to extend it to another 'level': whenever $\alpha@i \in CL$, put in $(\ominus\alpha@i)@j$ for every j mentioned in the formula in the extended CL. This at once increases complexity by a factor k, where k is the number of agents mentioned in the formula being decided. Further, we ensure that when the automaton (A_{ϕ_0}) makes an a-move, in the resulting state, for all $i, j \in loc(a)$, $\alpha@i$ is present iff $(\ominus\alpha@i)@j$ is also present. Deciding about formulas of the form $(\ominus\alpha@k)@i$, where $k \notin loc(a)$ and $i \in loc(a)$ is much harder : for this we carry along sets of triples of the form (i, j, k) which represents the situation that i has at least as recent information as j about k. Whenever an a-move is made, for $k \notin loc(a)$, we need to find an agent $i \in loc(a)$ such that for every $j \in loc(a)$, (i, j, k) is in this set : that is, i has the latest information about k among all agents in $loc(a)$. This information is used to update the $\ominus$ requirements appropriately.

References

[ADP95] Alur, R., Peled, D. and Penczek, W., "Model checking of causality properties", *Proc LICS*, IEEE, 1995.

[FHMV95] Fagin, R., Halpern, J., Moses, Y. and Vardi, M., *Reasoning about knowledge*, M.I.T. Press, 1995.

[GW94] Godefroid, P. and Wolper, P., "A partial approach to model checking", *Information and Computation*, vol 110, 1994, 305-326.

[LRT92] Lodaya, K., Ramanujam, R. and Thiagarajan, P.S., "Temporal Logics for Communicating Sequential Agents: I", *Int. J. Found. Comp. Sci.*, vol. 3, #2, 1992, 117-159.

[MP91] Manna, Z. and Pnueli, A., *The temporal logic of reactive and concurrent systems (Specification)*, Springer-Verlag, 1991.

[N95] Niebert, P., "A μ-calculus with local views for systems of sequential agents", *Springer LNCS 969*, 1995.

[NP95] Niebert, P. and Penczek, W., "On the connection of partial order logics and partial order reduction methods", *Rep 95-15, Tech. Univ. of Eindhoven*, 1995.

[P93] Peled, D., "All from one and one from all: on model checking using representatives", *Springer LNCS 697*, 1993, 409-423.

[P95] Penczek, W., "Branching time and partial order in temporal logics", in L. Bolc and A. Szalas (ed), *Time and logic : a computational approach*, UCL Press, 1995.

[R96] Ramanujam, R., "Local knowledge assertions in a changing world", *Proc TARK VI*, Morgan Kaufmann, 1996, 1-17.

[T94] Thiagarajan, P.S., "A trace based extension of propositional linear time temporal logic", *Proc LICS*, IEEE, 1994, 438-447.

[T95] Thiagarajan, P.S., "A trace consistent subset of PTL", *Springer LNCS 962*, 1995.

[V90] Valmari, A., "A stubborn attack on state explosion", *Springer LNCS 531*, 1990, 156-165.

[VW86] Vardi, M.Y. and Wolper, P., "An automata theoretic approach to program verification", *Proc LICS*, IEEE, 1986, 332-345.