Trace consistency and inevitability

R. Ramanujam [1]

The Institute of Mathematical Sciences
C.I.T. Campus
Madras 600 113.
jam@imsc.ernet.in

Abstract. Two runs of a distributed system can be considered equivalent if they represent different interleavings of the same run. Formulas of the propositional temporal logic of linear time (PTL) are said to be trace consistent when they cannot distinguish between equivalent runs. Determining whether a formula is trace consistent is decidable. In this paper, we consider a closely related notion: a PTL formula is said to be *inevitable*, if for every system run, there is at least one equivalent run (that is, one interleaving of that run) which satisfies the formula. We show that determining whether a formula is inevitable is undecidable. However, for a subclass of formulas termed coherent (propositions in them can be affected by either of a pair of independent operations but not both), inevitability becomes decidable.

1 A synopsis

The Propositional Temporal Logic of Linear Time (PTL) is used to describe the temporal evolution of global states of distributed systems. Reasoning in PTL about safety, liveness and fairness properties of systems is well understood by now. With the development of both automata-theoretic model checking techniques [VW86] and theorem proving techniques [MP91], automatic or semi-automatic verification of temporal properties has seen considerable progress.

A number of researchers ([GW94], [KP92a], [V90]) have noted that there is a natural equivalence relation on runs of systems which equates runs obtained by different interleavings of the same run. For instance, if a_1 and a_2 happen to be causally independent operations and a run δ is of the form $s_0 \xrightarrow{a_1} s_1 \xrightarrow{a_2} s_2$, we would consider another run of the form $s_0 \xrightarrow{a_2} s_1' \xrightarrow{a_1} s_2$ to be equivalent to δ. These constitute two different ways of interleaving the concurrent operations a_1 and a_2. If a_1 and a_2 are actions on distinct variables, the sequence of state changes that each variable is subject to is identical in both runs. Often, a system is defined to be truly distributed (or in some contexts, fair) only when it is closed with respect to this equivalence, that is, given any system run, all other interleavings of concurrent operations along that run are also admitted as runs.

[1] Part of this work was done while I was visiting Institut für Informatik und Praktische Mathematik, Christian - Albrechts - Universität Kiel, Germany. I thank Wolfgang Thomas for making the visit possible. I also thank the referees for comments.

The systems typically studied in PTL model checking, namely synchronized products of automata, or other parallel programs, are of this kind: their behaviour consists of all possible interleavings. This fact can be utilized for efficient verification of those properties, which hold for an entire equivalence class of runs whenever they hold for one run in the class. Such properties are said to be *equivalence robust* ([P93]) or *trace consistent* ([T95]). Identifying such properties is important, since partial-order based techniques developed in [GW94], [P93], [V90] and other papers become available for the verification of such properties.

Since PTL formulas are interpreted on system runs, when runs are treated as representatives of their equivalence classes, it becomes necessary to revisit the notions of satisfiability of PTL formulas in this context. The standard notions of satisfiability and model checking in PTL are phrased as follows:

– **Satisfiability**: given a formula, is there a system and a run of that system which satisfies the formula?
– **Model checking**: given a system and a formula, does every run of the system satisfy the formula?

The dual notion to satisfiability is that of **validity**: a valid formula is satisfied in every run of every system. When we begin to consider system behaviour as represented by an equivalence class of runs rather than by a single run, two new notions become interesting:

– **Strong satisfiability**: given a formula, is there a system and a run of that system such that *every* run equivalent to it satisfies the formula?
– **Weak model checking**: given a system and a formula, is it the case that for every run of the system, there is *at least one* equivalent run (some interleaving) which satisfies the formula?

The dual notion to strong satisfiability is that of **weak validity**: a weakly valid formula holds in some interleaving of every run of every system. (Weak validity is closely related to **inevitability**, a notion introduced by [MOP89]. In this paper, we simply refer to weak validity as inevitability.) Seen in this light, we can term the classical notions of satisfiability and model checking listed above respectively as **weak satisfiability** (with strong validity as its dual notion) and **strong model checking**.

In terms of language theory, we can see these notions in the following terms: the classical notion of satisfiability corresponds to the question whether a given ω-regular language is nonempty. Given in addition an equivalence on infinite strings, strong satisfiability asks whether there exists a nonempty equivalence-closed subset. Given two such languages L_S and L_α, classical model checking asks whether $L_S \subseteq L_\alpha$, whereas weak model checking only requires to know whether for every string in L_S there exists an equivalent one (a representative) in L_α. We can see the latter relation as a weaker form of inclusion.

Trace consistent formulas are exactly those for which the notions of strong and weak satisfiability coincide. (Correspondingly, for such formulas weak and strong model checking coincide.)

When we study decidability questions for PTL, we now have several questions to answer. The state-of-the-art can be summarised as follows:

- Satisfiability and model checking are decidable for PTL, and in fact PSPACE-complete. The frames for the logic studied in this context are simply labelled transition systems without explicit concurrency information. On the other hand, runs can be equated only when there is some concurrency specified explicitly. Once such concurrency is introduced, the frames include commuting of independent transitions, and then it is unclear whether the original results still obtain. This is important, particularly in light of the strong negative results shown in [LPRT95] [P92] and other work, which suggest that logics interpreted over global states of systems with commuting transitions tend to be undecidable.
- In [DGP95], it is shown that checking whether an ω-regular language is trace equivalence closed is decidable, and this implies that determining whether a PTL formula is trace consistent is also decidable. (See the recent paper [PWW96] giving an algorithmic proof of this, with lower bound results on deciding closure under trace and other equivalences.) Therefore, once we show that the standard notions of (weak) satisfiability and (strong) model checking are decidable, it follows that for trace consistent formulas, *all* the notions mentioned above are decidable.

In this paper we show that: (a) In the context of concurrent systems, despite the presence of commuting transitions, (weak) satisfiability and (strong) model checking are decidable. (b) Strong satisfiability is undecidable, when there are both local and common propositions in the language, whose truth or falsity can be affected by independent operations.

Once we find that strong satisfiability is undecidable, we ask: can we at least find subclasses of formulas for which strong satisfiability is decidable? (Obviously, this class cannot be pinned down effectively; if it were, we would get a decision procedure for strong satisfiability contradicting our result.) We already know of one subclass, namely the trace consistent ones. (In this context, it would be nice to have a syntactic characterization of trace consistent formulas. [T95] provides a semantic characterization.) We offer here a different subclass of PTL formulas for which strong satisfiability is decidable, and which is obtained by a syntactic restriction. We call such formulas *coherent*, and these closely correspond to the version of PTL studied in [T95]. For coherent formulas, weak model checking is seen to be decidable as well.

Apart from formal interest, is weak model checking relevant in the context of specifying and reasoning about distributed systems? Rather interestingly, the "p causes q" modality discussed by [R95] has the same flavour as asserting reachability in some interleaving. [PP94] studies a logic which is more expressive than PTL where these situations are easily described. However, we have no detailed examples of system verification necessitating weak model checking.

A word about the syntax of logic and the definition of frames for the logic studied here: PTL modalities refer only to successor states and eventuality, whereas state transitions in systems are labelled by actions. When we define equivalence on runs, it is by studying concurrency implicit in the dependency alphabet of actions. Therefore, we must have some way of referring to such dependence in the syntax of formulas as well. (Otherwise formulas can never force concurrency.) We do this by defining a dependence structure on propositions: for each proposition, this specifies which system variables are involved in determining its truth value. This also helps us in defining the class of coherent formulas: a proposition is coherent, if no two independent actions may both affect its truth value; a formula is coherent when every proposition mentioned in it is coherent.

This study could equally well be carried out with different definitions of frames. For instance, we could have action-indexed modalities in the logic. Ensuring that systems generate all possible interleavings can be done with different (and weaker) conditions on the systems than the one given here, with the same technical results as in this paper.

2 Preliminaries

2.1 Programs

A program is given by a tuple $Pr = (Q, \Sigma, \rightarrow)$, where Q is a finite set of states, Σ is a finite nonempty set of operations and $\rightarrow \subseteq (Q \times \Sigma \times Q)$ is the transition relation. An operation a is said to be *enabled* at a state s if there exists s' such that $(s, a, s') \in \rightarrow$, denoted $s \xrightarrow{a} s'$.

Partial order based verification methods involve an *independence relation* on transitions, so that whenever independent transitions occur in sequence, they can be 'commuted', resulting in the same state (they can be 'summed' up in any order). A good deal of effort is required to determine which transitions in a program are independent. Obviously it is impractical to enumerate reachable states and check commutation for all states and pairs of transitions. More messy is the question of when two transitions within a run are independent. (See [KP92b] for details on discovering and refining independence information.)

Below we take the approach of determining independence syntactically. Let us assume that a program operates on n variables, for some fixed $n > 0$. Since states describe values of program variables, we can think of Q as a subset of $Q_1 \times Q_2 \times \ldots \times Q_n$, where Q_i represents possible (finitely many!) values of the i^{th} variable. Further assume that we are given a function $\theta : \Sigma \rightarrow 2^{\{1,\ldots,n\}}$; $\theta(a)$ specifies which variables are affected when an a operation is performed. This automatically gives an irreflexive, symmetric independence relation I_θ on Σ : $(a, b) \in I_\theta$ iff $\theta(a) \cap \theta(b) = \emptyset$. We will refer to (Σ, θ) as a *dependence alphabet*. (An action a can be thought of as a simultaneous assignment of values to variables in $\theta(a)$. In this sense, updates of distinct sets of variables are independent.)

Definition 2.1 *Let $n > 0$. A* finite state n-variable program *over a dependence alphabet (Σ, θ) is a tuple $Pr = (\widetilde{Q}, \rightarrow)$, where $\widetilde{Q} \subseteq (Q_1 \times \ldots \times Q_n)$, for all i, $1 \le i \le n$, Q_i is a finite set of states and $\rightarrow \subseteq (\widetilde{Q} \times \Sigma \times \widetilde{Q})$, such that*

1. *if* $(q_1,\ldots,q_n)\xrightarrow{a}(q'_1,\ldots,q'_n)$ *then* $\forall i \notin \theta(a), q_i = q'_i$*, and*
2. $\forall x_0, x_1, x_2 \in \widetilde{Q}$*, if* $x_0 \xrightarrow{a} x_1, x_1 \xrightarrow{b} x_2, (a,b) \in I_\theta$*, then* $\exists x_3 \in \widetilde{Q}$ *such that* $x_0 \xrightarrow{b} x_3$ *and* $x_3 \xrightarrow{a} x_2$*.*

The latter condition is called *the forward diamond condition* and is typical in concurrency theory (for instance, asynchronous automata exhibit this property).

In order to have nontrivial independence relations, we will confine our interest to programs where $n > 1$, and alphabets (Σ, θ) such that I_θ is nonempty. This will be an unstated assumption throughout the discussion in the paper. Further, we fix a dependence alphabet (Σ, θ) and refer to programs (and later formulas too) over this alphabet, unless otherwise stated.

A *run* of a program is a finite or infinite sequence $x_0 \xrightarrow{a_0} x_1 \xrightarrow{a_1} \ldots$ generated by the transition relation of the program. We denote runs by δ, δ' etc. By $\delta\lceil i$, we mean the sequence generated by projecting down to the i^{th} components of states and erasing all a such that $i \notin \theta(a)$. This gives a finite or infinite sequence $q_0 \xrightarrow{a_0} q_1 \xrightarrow{a_1} \ldots$ such that for every j, $q_j \in Q_i, i \in \theta(a_j)$.

We can now define an equivalence relation on runs as follows : $\delta \sim \delta'$ iff $\forall i \in \{1,\ldots,n\}$, $\delta\lceil i = \delta'\lceil i$. It can be checked that the induced equivalence relation on Σ^ω is the same as the infinite trace equivalence (with respect to the concurrency alphabet (Σ, I_θ)) obtained from Mazurkiewicz trace theory and used by [P93] and other work on partial order based verification.

2.2 The logic

The logic we study is standard linear time temporal logic, except that we include some information to reflect the dependency structure of programs in the logic. A *propositional structure* is a pair (P, η), where P is a countable set of atomic propositions, and $\eta : P \rightarrow 2^{\{1,\ldots,n\}}$ is a map which, for any $p \in P$ specifies which variables **affect** the truth or falsity of p ([P93]). In a sense, η delimits the vocabulary of the statements denoted by atomic propositions in the logic.

Let $p \in P$ be a proposition. We say that p is **coherent** with respect to (Σ, θ) if and only if the following condition holds:

$$\forall a, b \in \Sigma, \text{ if } (a,b) \in I_\theta, \text{ then } (\eta(p) \cap \theta(a) = \emptyset) \text{ or } (\eta(p) \cap \theta(b) = \emptyset).$$

A formula (defined below) is coherent if and only if every proposition mentioned in it is coherent. (Since we have assumed a fixed dependence alphabet for discussion in the paper, we will simply talk of coherent formulas, where coherence is with respect to the fixed alphabet.) Thus, a coherent formula is distributable in the sense that if X is the set of propositions mentioned in it, then *independent operations affect disjoint subsets of propositions in* X.

It is worth noting what kind of formulas are declared incoherent by this condition. When variables r and s are acted on by independent operations, statements of the form "$r = 1$ or $s = 0$" or "the value of r is updated if and only if that of s is also updated" or "eventually ($r = 1$ and $s = 0$)" can always be expressed in the logic using separate propositions involving r and s, logical connectives and modalities. What we `cannot` express are statements of the form "the values of r and s are the same" when these variables can take on unboundedly many values. Since this needs infinitary expressions not part of the logic, such 'nonlogical' properties require atomic propositions and these are incoherent (in case they involve variables affected by independent operations).

The syntax of formulas is defined below. We assume special propositions which allow us to speak of actions being enabled or not. (These propositions are not important for the decidability results, but merely convenient.)

$$\mathcal{L} ::= p \in P \mid \lambda_a,\ a \in \Sigma \mid \neg\, \alpha \mid \alpha \vee \beta \mid \bigcirc\, \alpha \mid \alpha \mathbf{U} \beta$$

The derived connectives and modalities $\wedge$, $\supset$, $\equiv$, $\Diamond$, $\Box$ are all defined as usual. (In particular, $\Diamond\alpha \stackrel{\text{def}}{=} True\mathbf{U}\beta$ where $True$ is defined to be $p_0 \vee \neg p_0$, for some $p_0 \in P$. $\Box\alpha \stackrel{\text{def}}{=} \neg\Diamond\neg\alpha$.)

A *system* is a pair $S = (Pr, V)$ where Pr is an n-variable program and $V : \widetilde{Q} \rightarrow 2^P$ is a partial valuation map on program states such that for all $p \in P$, whenever $x \xrightarrow{a} x'$, $\eta(p) \cap \theta(a) = \emptyset$ and both $V(x)$ and $V(x')$ are defined, $p \in V(x)$ iff $p \in V(x')$. Since the operation a can only update values of variables in $\theta(a)$ and these do not affect the truth of p, the valuation must respect this. Note that even with this condition, we have more general models than those having only 'local' valuations, namely, $V_i : Q_i \rightarrow 2^P$. (Having partial valuation maps is a matter of technical convenience: we can choose to ignore the truth or falsity of propositions in program states which are unreachable in the runs of interest.)

Given a system $S = (Pr, V)$, a *model* is a pair $M = (S, \delta)$, where $\delta = x_0 \xrightarrow{a_0} x_1 \xrightarrow{a_1} \ldots$ is an infinite run of Pr such that for all k, $V(x_k)$ is defined. (Excluding finite runs is technically convenient, though not critical.) The notion $S, \delta, k \models \alpha$ denoting that the formula α is satisfied at instant k along run δ of the system S is defined in the usual manner. We leave S and δ implicit below.

- $k \models p$ iff $p \in V(x_k)$.
- $k \models \lambda_a$ iff $\exists x' \in \widetilde{Q}$ such that $x_k \xrightarrow{a} x'$ in Pr.
- $k \models \neg\alpha$ iff $k \not\models \alpha$.
- $k \models \alpha \vee \beta$ iff $k \models \alpha$ or $k \models \beta$.
- $k \models \bigcirc\alpha$ iff $k + 1 \models \alpha$.
- $k \models \alpha\mathbf{U}\beta$ iff $\exists m : k \leq m, m \models \beta$ and $\forall l : k \leq l < m : l \models \alpha$.

We say $S, \delta \models \alpha$ if $S, \delta, 0 \models \alpha$.

A formula α is said to be **trace consistent** if and only if for every system $S = (Pr, V)$, for every δ of Pr, if $S, \delta \models \alpha$ then for all runs $\delta' \sim \delta$, we have $S, \delta' \models \alpha$ as well.

We say that a given system $S = (Pr, V)$ *weakly satisfies* a formula α if there exists a run δ of Pr such that $S, \delta \models \alpha$. We say that a given system $S = (Pr, V)$ *strongly satisfies* a formula α if there exists a run δ of Pr such that for every $\delta' \sim \delta$, $S, \delta' \models \alpha$. A formula α is said to be **weakly (strongly) satisfiable** if there exists a system S which weakly (strongly) satisfies it.

It is easily seen that for a trace consistent formula α, any system S weakly satisfies α if and only if it strongly satisfies α. Thus the two notions of satisfiability coincide for trace consistent formulas.

The dual notions are interesting for model checking. Given a system $S = (Pr, V)$, call a formula α *strongly S-valid*, if $S, \delta \models \alpha$, for every run δ of Pr. Similarly, call α *weakly S-valid*, if for every run δ of Pr, there exists $\delta' \sim \delta$ such that $S, \delta' \models \alpha$. A *strongly valid* formula is one which is strongly S-valid for every system S, and a *weakly valid* or **inevitable** formula is one which is weakly S-valid for every system S.

Consider a propositional structure which has propositions p and q such that $\eta(p) = \{1\}$ and $\eta(q) = \{2\}$. Further suppose that there is no action a such that $\{1, 2\} \subseteq \theta(a)$. Then consider the formula $\alpha = p \wedge q \wedge \bigcirc(\neg p \wedge \bigcirc \neg q)$. α is coherent but not trace consistent, and weakly satisfiable but not strongly satisfiable. On the other hand, if $\eta(r) = \{1, 2\}, \theta(a) = \{1\}$ and $\theta(b) = \{2\}$, the formula $r \vee \neg r$ is obviously valid and trace consistent, yet not coherent.

3 Satisfiability

We first show that the standard notions of weak satisfiability and strong model checking remain decidable, despite the presence of diamond conditions on frames. (We will simply call these notions satisfiability and model checking in this section). This problem is nontrivial, since the standard proof for deciding PTL formulas does not yield a commuting frame by itself. That such a commuting frame can be extracted is the central claim in the following argument.

As usual, we need the notion of *subformula closure* of a formula. $CL(\alpha)$ is defined to be the smallest set of formulas containing the formula α and satisfying the following conditions:

- $\neg\beta \in CL(\alpha)$ iff $\beta \in CL(\alpha)$, where $\neg\neg\beta$ is considered as the same as β.
- if $\beta_1 \vee \beta_2 \in CL(\alpha)$ then $\{\beta_1, \beta_2\} \subseteq CL(\alpha)$.
- if $\bigcirc\beta \in CL(\alpha)$ then $\beta \in CL(\alpha)$.
- if $\beta_1 \mathbf{U} \beta_2 \in CL(\alpha)$ then $\{\beta_1, \beta_2, \bigcirc(\beta_1 \mathbf{U} \beta_2)\} \subseteq CL(\alpha)$.

The size of $CL(\alpha)$ is linear in the length of α.

Fix a formula α_0, whose satisfiability is to be decided. For the rest of this section, we will call $CL(\alpha_0)$, simply CL. An *atom* A is a subset of CL which is propositionally consistent, and satisfies the additional condition : $\alpha \mathbf{U} \beta \in A$ iff (either $\beta \in A$ or both α and $\bigcirc(\alpha \mathbf{U} \beta)$ are in A). Let AT denote the set of all atoms. We use A, B, A_1 etc to denote atoms.

We define a *successor* relation $Succ \subseteq AT \times \Sigma \times AT$ as follows: $(A, a, B) \in Succ$ iff $(\forall \bigcirc\alpha \in CL$, $\bigcirc\alpha \in A$ iff $\alpha \in B)$ and $(\forall p : \eta(p) \cap \theta(a) = \emptyset : p \in A$ iff $p \in B)$. In this case, we say that B is an a-successor of A.

The standard decision procedure for PTL ([VW86]) consists of building a nondeterministic Büchi automaton whose states are atoms and transitions are given by $Succ$ above, with suitable modifications for keeping track of $\mathbf{U}$ requirements. It is then shown that α_0 is satisfiable if and only if the ω-language accepted by the associated automaton is nonempty. The proof below can be presented in terms of such an automaton; we have chosen a different presentation mainly as a preparation for the proof in Section 5.

Consider the finite graph $G = (AT, Succ)$. $H = (W, \Rightarrow)$ is a subgraph of G if $W \subseteq AT$ and $\Rightarrow \subseteq Succ$. H is said to be *good* if and only if it satisfies the following conditions :

- There exists $A_0 \in W$ such that $\alpha_0 \in A_0$.
- $\forall A \in W, \exists a \in \Sigma$ such that A has an a-successor in H.
- $\forall \alpha \mathbf{U} \beta \in CL$, if $A \in W$ such that $\alpha \mathbf{U} \beta \in A$, then there exists a path $A = A_0 \stackrel{a_1}{\Rightarrow} A_1 \ldots \stackrel{a_m}{\Rightarrow} A_m$ in H such that $\beta \in A_m$.

Lemma 3.1 *α_0 is satisfiable if and only if the graph G defined above contains a good subgraph.*

Note that the lemma at once yields the theorem below as a corollary :

Theorem 3.2 *Satisfiability of a formula α_0 is decidable in time $2^{O(m)}$, where m is the length of α_0.*

One direction of the lemma is straightforward to prove. Suppose that the formula α_0 is satisfiable. Let $S = (Pr, V)$ be a system and δ a run of Pr such that $S, \delta, 0 \models \alpha_0$, where $\delta = x_0 \stackrel{a_1}{\rightarrow} x_1 \stackrel{a_2}{\rightarrow} \ldots$. Define $A_k \stackrel{\text{def}}{=} \{\alpha \in CL | M, k \models \alpha\}$, $W \stackrel{\text{def}}{=} \{A_k | k \geq 0\}$, and $X \stackrel{a}{\Rightarrow} Y$ iff there exists k such that $X = A_k$, $Y = A_{k+1}$ and $a_{k+1} = a$. It can be easily checked that each member of W is in fact an atom, that $\Rightarrow \subseteq Succ$ and that $H = (W, \Rightarrow)$ is good.

On the other hand, suppose G contains a good subgraph $H = (W, \Rightarrow)$. Building an n-variable program from H is nontrivial. We proceed in two steps: we first unfold H to get an infinite sequence $\sigma = x_0 \stackrel{a_1}{\Rightarrow} x_1 \stackrel{a_2}{\Rightarrow} \ldots$ along which all eventuality requirements are met, and then fold σ into an n-variable program. This is done in such a way that σ has the crucial property (*) that whenever for any $k \geq 0$, $\alpha \mathbf{U} \beta \in x_k$, there exists $l \geq k$ such that $\beta \in x_l$. Further, by definition of atoms and $Succ$, for every j such that $k \leq j < l$, $\alpha \in x_j$. Also, for all k, when $\bigcirc \alpha \in x_k$, we have that $\alpha \in x_{k+1}$.

Based on σ, we now define another sequence $\rho = \chi_0 \stackrel{a_1}{\Rightarrow} \chi_1 \stackrel{a_2}{\Rightarrow} \ldots$ where for all k, χ_k is an n-tuple of the form $(X_1^k, X_2^k, \ldots, X_n^k)$, where for $i \in \{1, \ldots, n\}$, $X_i^k \in 2^{AT}$, and the action sequence is the same as in σ.

X_i^k is defined to be the set of all atoms in σ after the last i-action before k and before the next i-action (if any) after k. Formally, $X_i^k \stackrel{\text{def}}{=} \{x_j | (j \leq k$ and $\forall l, j < l \leq k, i \notin \theta(a_l))$ or $(j > k$ and $\forall l, k < l \leq j, i \notin \theta(a_l))\}$. Clearly, for all k we have $X_i^k = X_i^{k+1}$ if $i \notin \theta(a_k)$.

We can now construct the model : let $Pr_i = (Q_i, \rightarrow_i)$ be defined by : $Q_i = \{X_i^k | \; k \geq 0\}$; for $i \in \theta(a)$, $q \stackrel{a}{\rightarrow}_i q'$ iff $\exists k$: $X_i^k = q, X_i^{k+1} = q'$ and $a_{k+1} = a$. Consider $Pr = (Q_1 \times \ldots \times Q_n, \rightarrow)$ be the n-variable program, where $\rightarrow$ is the product transition relation: $(q_1, \ldots, q_n) \stackrel{a}{\rightarrow} (q_1', \ldots, q_n')$ iff $\forall i \in \theta(a), q_i \stackrel{a}{\rightarrow}_i q_i'$ and $\forall i \notin \theta(a), q_i = q_i'$. It is easy to check that the frame conditions are satisfied and that ρ constructed above is a run of Pr. The valuation needs to be defined only for χ_k: $V(\chi_k) \stackrel{\text{def}}{=} x_k \cap P$. We now have $M = (Pr, \rho, V)$, it remains to show that M is a model for α_0.

Claim: $\forall \gamma \in CL$, $\forall k \geq 0$, $M, k \models \gamma$ if and only if $\gamma \in x_k$.

The claim is proved by induction on the complexity of γ. The proof is routine and follows from the assertion (*) above.

We can easily adapt the decision procedure for satisfiability to obtain a decision procedure for model checking as well: given a system $S = (Pr, V)$ and α, we need a procedure which checks whether every model based on S satisfies α. For this, it is sufficient to build a procedure which checks whether *there exists* a model based on S which satisfies α (running the latter on $\neg\alpha$ answers the former question). For this, we proceed as above, but instead of working with AT, we work with $\widetilde{Q} \times AT$, where $\widetilde{Q}$ is the state space of the given program Pr and whenever we consider $(\widetilde{q}, A)$, $p \in A$ iff $p \in V(\widetilde{q})$. The rest of the proof is similar. Thus we have the following theorem :

Theorem 3.3 *Model checking for a system S and formula α can be done in time $k.2^{O(m)}$, where k is the size of S and m is the length of α.*

4 Inevitability

We now show that strong satisfiability, and hence inevitability, is undecidable, when there are at least two groups of agents in the system with propositions local to each group as well as common propositions. For this we recursively reduce every instance of the Finite Colouring Problem (FCP) [LPRT95] to the strong satisfiability of a formula. The result then follows from the undecidability of FCP (which is just a variant of standard tiling problems). We merely outline the construction below; the proof follows easily from [LPRT95].

In the finite colouring problem, one is given a finite set of colours, two functions 'right' and 'up', and asked whether there is a finite rectangle that can be coloured at lattice points such that the colouring is consistent with the right and up functions. Formally, an *instance* of FCP is a tuple $\Delta = (\mathcal{C}, R, U, c_0, c_1)$, where $\mathcal{C} = \{c_0, c_1, \ldots, c_k\}, k > 0$ is a finite set of **colours**, and $R, U : \mathcal{C} \rightarrow 2^{\mathcal{C}}$ are the **right** and **up** functions respectively. A *solution* to Δ is a triple (u, v, Col), where $u \geq 0, v \geq 0$ and $Col : \{0, \ldots, u\} \times \{0, \ldots, v\} \rightarrow \mathcal{C}$ is a **colouring function** which satisfies the conditions:

- $Col(0,0) = c_0$ and $Col(u,v) = c_1$,
- $\forall i, j : 0 \leq i < u, 0 \leq j \leq v, Col(i+1, j) \in R(Col(i,j))$, and
- $\forall i, j : 0 \leq i \leq u, 0 \leq j < v, Col(i, j+1) \in U(Col(i,j))$.

Given an instance Δ, we construct a formula α_Δ as follows. Firstly, associate a proposition with each colour in $\mathcal{C}$. Without loss of generality we will use these colours themselves as propositions. Secondly, by nontriviality of the alphabet, $I_\theta \neq \emptyset$, and hence there exist actions a and b such that $(a, b) \in I_\theta$. Without loss of generality, let $1 \in \theta(a)$, $2 \in \theta(b)$. Assume that for $c \in \mathcal{C}, \eta(c) = \{1, 2\}$. Further let $\{p,q\} \subset P$ such that $\eta(p) = \{1\}$ and $\eta(q) = \{2\}$. Consider now the formula $\alpha_\Delta \stackrel{\text{def}}{=} \bigwedge_{i=1}^{5} \alpha_i$, where ($\otimes$ denotes exclusive disjunction below):

- $\alpha_1 \stackrel{\text{def}}{=} c_0 \wedge p \wedge q \wedge \Diamond c_1 \wedge ((\bigwedge_{d \in \Sigma - \{a,b\}} \neg\lambda_d)\mathbf{U} c_1)$.
- $\alpha_2 \stackrel{\text{def}}{=} \bigwedge_{\beta \in \{p,\neg p\}} \bigwedge_{\gamma \in \{q,\neg q\}} \Box((\beta \wedge \gamma) \supset (c_1 \vee (\bigcirc(\neg\beta \wedge \gamma) \otimes \bigcirc(\beta \wedge \neg\gamma))))$.
- $\alpha_3 \stackrel{\text{def}}{=} \Box \bigwedge_{i=0}^{k} (c_i \equiv \bigwedge_{j \neq i} \neg c_j)$.
- $\alpha_4 \stackrel{\text{def}}{=} \bigwedge_{\beta \in \{p,\neg p\}} \Box \bigwedge_{i=0}^{k} (c_i \supset ((\beta \wedge \bigcirc\neg\beta) \supset (c_1 \vee \bigcirc \bigvee_{c \in R(c_i)} c))$.
- $\alpha_5 \stackrel{\text{def}}{=} \bigwedge_{\beta \in \{q,\neg q\}} \Box \bigwedge_{i=0}^{k} (c_i \supset ((\beta \wedge \bigcirc\neg\beta) \supset (c_1 \vee \bigcirc \bigvee_{c \in U(c_i)} c))$.

Lemma 4.1 *Δ has a solution if and only if α_Δ is strongly satisfiable.*

Theorem 4.2 *Strong satisfiability and hence inevitability is undecidable when L contains incoherent propositions.*

This raises the possibility that weak model checking may also be undecidable, but we have not been able to establish either the decidability or undecidability of weak model checking.

The use of the enabling propositions above is mainly a technical convenience. These can be eliminated but the proof becomes more complicated: we need to reduce the strong satisfiability of a formula to that over alphabets whose actions are 'necessitated' by the formula.

5 Coherent formulas

We now turn our attention to showing decidability of strong satisfiability for coherent formulas. Recall that for a coherent formula, independent operations affect disjoint subsets of propositions mentioned in it.

The notion of subformula closure of a formula is as before. Again fix a formula α_0 whose strong satisfiability is to be decided, and let CL denote $CL(\alpha_0)$. The definition of atoms, AT, the graph $G = (AT, Succ)$ and good subgraphs of G are as before.

Let $H = (W, \Rightarrow)$ be a subgraph of G. By $H\lceil Prop$, we mean the graph $(U, \rightarrow)$, where $U = \{A \cap P | A \in W\}$ and $Q_1 \xrightarrow{a} Q_2$ iff $\exists A, B \in W$ such that $A \cap P = Q_1$, $B \cap P = Q_2$ and $A \stackrel{a}{\Rightarrow} B$ in H.

Let $J_1 = (U_1, \rightarrow_1)$ and $J_2 = (U_2, \rightarrow_2)$ be graphs such that $U_1, U_2 \subseteq 2^P$ and $\rightarrow_j \subseteq (U_j \times \Sigma \times U_j)$, $j \in \{1,2\}$. We say $J_1 \doteq_I J_2$ iff there exist $Q \in U_1, Q' \in U_2$

such that $U_2 = (U_1 - \{Q\}) \cup \{Q'\}$ and there exist $Q_1, Q_2 \in U_1 \cap U_2$ such that for some $(a,b) \in I_\theta$, $Q_1 \xrightarrow{a}_1 Q \xrightarrow{b}_1 Q_2$ and $\rightarrow_2 = (\rightarrow_1 - \{(Q_1, a, Q), (Q, b, Q_2)\}) \cup \{(Q_1, b, Q'), (Q', a, Q_2)\}$. Let $=_I$ denote the reflexive transitive closure of $\doteq_I$.

Call a subgraph $H = (W, \Rightarrow)$ of $G = (AT, Succ)$ *strongly good* iff it contains a good subgraph, and for every good subgraph $H_1 = (W_1, \Rightarrow_1)$ of H, if $J_1 = H_1 \lceil Prop$, and there exists $J_2 {=_I} J_1$, then there exists $H_2 = (W_2, \Rightarrow_2)$ a good subgraph of H such that $H_2 \lceil Prop = J_2$.

Given any good subgraph $H = (W, \Rightarrow)$ of G, it is easy to see that there is an infinite path δ in H of the form $A_0 \xrightarrow{a_0} A_1 \xrightarrow{a_1} \ldots$, where $\alpha_0 \in A_0$ and for every j, if $\alpha \mathbf{U} \beta \in A_j$, there exists $k \geq j$ such that $\beta \in A_k$. Call δ a good path induced by H, and we say δ is *on* $\rho = a_0 a_1 \ldots \in \Sigma^\omega$.

We can then show the following: suppose that H is strongly good. Then for every good subgraph $H_1 = (W_1, \Rightarrow_1)$ of H and $\rho_1 \in \Sigma^*$, if there exists a good path induced by H_1 on $\rho_1 \rho \in \Sigma^\omega$, and $\rho_1 \sim \rho_2$, then there exists a good subgraph $H_2 = (W_2, \Rightarrow_2)$ of H such that there is a good path induced by H_2 on the string $\rho_2 \rho$.

From this we can prove the following stronger statement, which will be needed later: suppose that H is strongly good. Then for every good subgraph $H_1 = (W_1, \Rightarrow_1)$ of H, if there exists a good path induced by H_1 on $\rho_1 \rho_2 \rho_3 \ldots \in \Sigma^\omega$, where for all $j, \rho_j \in \Sigma^*$, and for all j, $\rho_j \sim \chi_j$, then there exists a good subgraph $H_2 = (W_2, \Rightarrow_2)$ of H such that there is a good path induced by H_2 on the string $\chi_1 \chi_2 \chi_3 \ldots \in \Sigma^\omega$.

Lemma 5.1 *α is strongly satisfiable if and only if G above has a strongly good subgraph.*

When α_0 is strongly satisfiable, constructing a strongly good subgraph of G is done as follows. Let (Pr, δ, V) be the given model satisfying α_0, where $\delta = x_0 \xrightarrow{a_1} x_1 \xrightarrow{a_2} \ldots$. For $k \geq 0$ and $\delta' \in [\delta]$, let $\mu(\delta', k)$ denote the set of formulas $\{\phi \in CL | \delta', k \models \phi\}$. It is easily seen that every such set is an atom. Collect all such sets to form W. Transitions between them are induced by the transitions in the model: suppose $A, B \in W$. Define $A \stackrel{a}{\Rightarrow} B$ iff there exists $\delta' \in [\delta]$ and $k \geq 0$ such that $A = \mu(\delta', k)$, $B = \mu(\delta', k+1)$ and a is the action in the k^{th} transition in δ'. It can be easily checked that $(W, \Rightarrow)$ is a subgraph of G. To show that it is strongly good, first notice that it contains a good subgraph: simply collect all atoms which are μ-images along δ. Clearly $\alpha_0 \in \mu(\delta, 0)$, and $\mathbf{U}$ requirements are met within this subgraph, say H_0. Consider $H_0 \lceil Prop$. Because of the coherence of α_0, and the fact that α_0 is strongly satisfied in the given model, we can observe the following: for all $i \in Loc$, the sequence of sets of i-propositions which hold is the same for every $\delta' \in [\delta]$; with these n sequences, every shuffle of these corresponds to a run in $[\delta]$ which correspondingly generates a good subgraph. Thus, we see that $(W, \Rightarrow)$ is a strongly good subgraph of G.

In the other direction, when we are given a strongly good subgraph, we use the good subgraph in it to first build an infinite sequence σ with the property that every eventuality requirement is met. We then build n transition systems whose i-local states are sets of the form $\{p \mid p \in A_j \cap P \text{ and } i \in \eta(p)\}$, where A_j is the

j^{th} atom in the sequence, and the i-local transitions are defined by transitions in σ in which i participates. When we take the product of these, we get the required n-variable program, which has σ as a run. Now, when we consider any run σ' equivalent to σ, we need to show that eventuality requirements are met along σ' as well. For this it suffices to consider $\sigma' = v_1 v_2 \ldots$ such that $\sigma = u_1 u_2 \ldots$ and $u_1 \sim v_1$, $u_2 \sim v_2 \ldots$, where the u_j and v_j are finite sequences (this follows from [DGP95]). We now use the commuting property above for strongly good subgraphs to show that requirements are met along σ'.

Theorem 5.2 *Strong satisfiability, and hence inevitability, are decidable for coherent formulas α in time $2^{O(m^2)}$, where m is the length of α.*

A similar argument can be used to show the decidability of weak model checking for coherent formulas, where the complexity is again linear in the size of the system being checked.

References

[DGP95] Diekert, V., Gastin, P. and Petit, A., "Rational and recognizable complex trace languages", *Information and Computation*, vol 116, #1, 1995, 134-153.

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

[KP92a] Katz, S. and Peled, D., "Verification of distributed programs using representative interleaving sequences", *Distributed Computing*, vol. 6, 1992, 107-120.

[KP92b] Katz, S. and Peled, D., "Defining conditional independence using collapses", *TCS*, vol. 101, #3, 1992, 337-359.

[LPRT95] Lodaya, K., Parikh, R., Ramanujam, R., Thiagarajan, P.S., "A logical study of distributed transition systems", *Inf & Comp.*, vol. 119, #1, 1995, 91-118.

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

[MOP89] Mazurkiewicz, A., Ochmanski, E., and Penczek, W.,"Concurrent systems and inevitability", *TCS*, vol. 64, #3, 1989, 281-304.

[P92] Penczek, W., "On undecidability of propositional temporal logics on trace systems", *Inf. Proc. Letters*, vol. 43, 1992, 147-153.

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

[PP94] Peled, D., and Pnueli, A., "Proving partial order properties", *TCS*, vol. 126, 1994, 143-182.

[PWW96] Peled, D., Wilke, T. and Wolper, P., "An algorithmic approach to proving closure properties of ω-regular languages", Proc. CONCUR, *LNCS*, 1996.

[R95] Reisig, W., "Petri net models of distributed algorithms", *Tech. Rep. 58*, Humboldt Univ., Berlin, 1995 (to appear in *LNCS 1000*).

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

[T95] Thiagarajan, P.S., "A trace consistent subset of PTL", *LNCS 962*, 438-452.

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

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