Decidability of a partial order based temporal logic

Kamal Lodaya[1], R. Ramanujam[1], P.S. Thiagarajan[2]

[1] The Institute of Mathematical Sciences, Madras 600 113, INDIA
[2] School of Mathematics, SPIC Science Foundation, 92 G.N. Chetty Road, T. Nagar, Madras 600 017, INDIA

1 Introduction

In the past decade, a variety of temporal logics have been proposed to reason about distributed programs and systems [Eme, Sti]. As opposed to the approach taken by Hoare logic [Apt] and dynamic logic [Har], where programs are explicitly used in the syntax of formulas, temporal logics have been interpreted over behaviours of programs rather than programs themselves. Typically, a transition system is used to model the state space of a program. A *run* (computation) of the program is a sequence of states permitted by the transition system. Formulas of the logic are interpreted over the states visited by a run. Depending on whether quantification is allowed over runs, one has branching or linear time temporal logics. In the case of concurrent systems, the states of the transition system represent *global* states of the distributed program, and a run denotes the *interleaved* execution of actions.

An alternative approach – which we have followed – is to have a single ("unfolded") structure represent the behaviour of a distributed program. In such a structure, causality, choice and concurrency can be explicitly defined and separated from each other. Furthermore, viewing a global state as only a potential state of affairs, one attaches logical assertions to *local* states of the computations generated by the program.

Axiomatizing models of distributed systems in this way is in the spirit of classical modal logic [HC], where one speaks of a logic *characterizing* a class of structures. The main motivation here is to identify operators which would directly express notions such as communication and concurrency. In the logic considered here, the focus of attention is communication between agents.

Consider a system of n (≥ 1) sequential agents which communicate with each other asynchronously via point-to-point unbounded buffers. Such systems often appear in the literature on distributed algorithms [CM]. One way of representing the "unfolded" behaviour of such a system is through n trees which are interconnected in some coherent fashion. This would be a generalization of the "timing diagrams" of Lamport [Lam]. (A timing diagram represents n interconnected linear orders.) The individual trees are meant to denote the local sequential behaviour of the agents. The interconnections capture the communication pattern established during the course of the runs permitted by the system.

Such behavioural objects can be formalized as *Communicating Sequential Agents* (CSAs) [LRT]. They can also be viewed as structured versions of prime event structures [NPW]. In [LRT] we proposed a temporal logic with future and indexed past modalities for CSAs. The semantics of the logic was chosen so that the modalities represent the notion of an agent gaining information about another through the *reception* of messages but not by *sending* them. We showed completeness results for this logic over CSAs and many subclasses of CSAs. In [LR], these results were further improved upon.

Decidability of the CSA-logics has turned out to be an interesting problem. Traditionally, decidability of tense logics has been proved via translation to $S2S$, the monadic second order theory of two successors (see [Bur] for an example). This gives a nonelementary decision procedure [Rab]. Encoding our logics into $S2S$ over trees seems difficult and generalizing $S2S$ to interpret over partial orders is a hard problem. On the other hand, the standard filtration techniques [HC] can be easily seen to violate the condition that each agent be tree-like.

In this paper, we consider the logic defined in [LR] and prove elementary decidability of that logic. This leads us to decidability results for the logics over subclasses of CSAs studied in [LRT]. Our results show that partial order based temporal logics, even with explicit reference to agents, need not be unmanageably expressive.

Our proof relies upon new techniques for performing filtrations over partial orders. Recording the right amount of information about the past at a local state turns out to be quite nontrivial.

It is perhaps worth noting that there are very few decidability results for logics based on the partial order semantics of concurrency. One such result is due to Pinter and Wolper [PW]. Their logic is interpreted over runs of the system. Our approach is different in the use of agents as well as in the interpretation over the entire structure rather than over runs. A similar remark applies to the logic of Penczek [Pen], which, although interpreted over event structures, has operators suited to describing (partially ordered) runs.

2 Definitions and Results

In this section, we present a model of distributed systems called n-ACSAs, a temporal logic for local reasoning in n-ACSAs and state the results of our paper. For the sake of brevity, we merely introduce the logic and models here; see [LR, LRT] for a detailed treatment.

Definition 2.1 *An* n-ACSA *(n Asynchronously Communicating Sequential Agents) is a triple $S = (E, \leq, \eta)$, where*

1. *E is a set of event occurrences.*
2. *$\leq$ is a partial order on E called the causality relation.*
3. *$\eta : E \to \{1, \ldots, n\}$ is a naming function.*
4. *$\forall e \in E : \ \forall i : 1 \leq i \leq n : \ \{e' \mid e' \leq e, \ \eta(e') = i\}$ is totally ordered by $\leq$.*

The following points can be noted about n-ACSAs:

1. *Causality* - We interpret $e_1 \leq e_2$ to mean that it is necessary for e_1 to occur before e_2 does. For instance, sending a message has to precede receiving it.
2. *Conflict* - Consider two incomparable event occurrences e_1 and e_2. We say they are in conflict, either if $\eta(e_1) = \eta(e_2)$, or if there exist e'_1 and e'_2 such that $\eta(e'_1) = \eta(e'_2), e'_1 \leq e_1, e'_2 \leq e_2$ and e'_1 and e'_2 are incomparable. Thus conflicting event occurrences never occur in the same computation. Now, we can see that condition (4) in Definition 2.1 ensures that no event occurrence causally depends on conflicting event occurrences. This forces agents to be tree-like - we refer to this property as backward linearity within agents.
3. *Concurrency* - Consider incomparable event occurrences e_1 and e_2 which are not in conflict. Necessarily, they are in distinct agents. We interpret this to mean concurrent occurrence of e_1 and e_2 and the same computation can include both.

A unique agent is associated with each event occurrence by the naming function η. Thus the sending of a message and its receipt are to be separated into distinct events performed by different agents. It is in this sense that our model supports asynchronous communication.

Let $e \in E$ and $\eta(e) = i$. ${\downarrow}e \stackrel{\text{def}}{=} \{e' \mid e' \leq e\}$ is the local state of agent i when e has "just" occurred. We say that an n-ACSA is *finitary* if for every e in E, ${\downarrow}e$ is finite. The motivation for defining finitary n-ACSAs is that any computation of a real system can be traced back to some starting point, so the past of any event occurring during the computation must be finite.

Figure 1 is an example of a 2-ACSA consisting of a *producer* agent (marked 1) and a *consumer* agent (marked 2), communicating via an unbounded buffer. The producer can produce zero or more items and then quit. The consumer can consume items produced by the producer as long as the items are available in the buffer. The "quit" events in agent 1 are labelled q.

n-ACSAs arise naturally as behavioural representations of distributed systems. Consider a structure $D = (TS_1, TS_2, \ldots, TS_n)$ where, for $i \in \{1, \ldots, n\}$, the TS_i are transition systems representing agent i. The alphabet of TS_i will

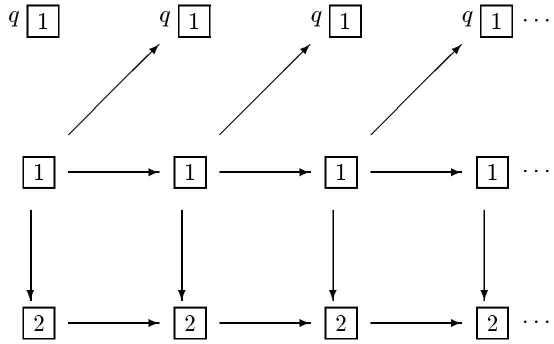

Fig. 1. A 2-ACSA

allow internal actions of agent i, send to j and receive from j actions, for $j \neq i$. Two action sequences are termed equivalent if their projections onto the alphabets of each individual agent are identical. The equivalence classes are called *traces*. The set of traces under a standard prefix ordering will give rise to an n-ACSA. The proof of this result follows that in [MT].

We now define a logic that is to be interpreted over n-ACSAs.

Let $Prop = \{p_0, p_1, \ldots\}$ be a countable set of atomic propositions. The formulas of the language $\mathcal{L}_n$ are defined inductively as follows:

– Every member of $Prop$ is a formula.
– If α and β are formulas, then so are $\neg\alpha, \alpha \vee \beta$, $\Diamond\alpha$ and $\diamondminus_i\alpha$, $i \in \{1, \ldots, n\}$.

A *model* is a pair $M = (S, V)$ where S, the *frame*, is an n-ACSA and V, the *valuation*, is a function from E to 2^{Prop}.

The notion of a formula α being true in a local state $\downarrow\! e$ of a model M, denoted $M, e \models \alpha$, is defined inductively as follows:

(i) $M, e \models \alpha$ iff $\alpha \in V(e)$, for $\alpha \in Prop$.
(ii) $M, e \models \neg\alpha$ iff $M, e \not\models \alpha$.
(iii) $M, e \models \alpha \vee \beta$ iff $M, e \models \alpha$ or $M, e \models \beta$.
(iv) $M, e \models \diamondminus_i\alpha$ iff $\exists e' \leq e : \ \eta(e') = i$ and $M, e' \models \alpha$.
(v) $M, e \models \Diamond\alpha$ iff $\exists e' : \ e \leq e', \ \eta(e) = \eta(e')$ and $M, e' \models \alpha$.

Note that $\diamondminus_i$ behaves like a normal past modality – it covers all events that lie in the i-past of e. However $\Diamond\alpha$ is different: in agent j, it asserts that there is a possible future for agent j satisfying α. The asymmetry between past and future arises from the fact that in distributed systems, the past of other agents can be completely obtained by messages, while the possibilities for the future are only locally known.

Define $\boxminus_i\alpha \stackrel{\text{def}}{=} \neg\diamondminus_i\neg\alpha$ and $\Box\alpha \stackrel{\text{def}}{=} \neg\Diamond\neg\alpha$.

Examples of reasoning with this logic can be found in [LMRT, LR, LRT].

As usual, a formula α is *satisfiable* if there is a model $M = (S, V)$, $S = (E, \leq, \eta)$ and $\exists e \in E$ such that $M, e \models \alpha$. It is *finitary satisfiable* if it is satisfiable in a model $M = (S, V)$ where S is finitary. A formula α is *valid* if $\neg\alpha$ is not satisfiable. It is *finitary valid* if $\neg\alpha$ is not finitary satisfiable.

Here are some formulas. $(F1) - (F4)$ are valid, whereas $(F5)$ is not. However, $(F5)$ is finitary valid.

$(F1)$ $\boxminus_i\alpha \supset \boxminus_j\boxminus_i\alpha$
$(F2)$ $\boxminus_i\Box(\alpha \supset \diamondminus_i\alpha)$
$(F3)$ $\boxminus_i(\Diamond\alpha \equiv \boxminus_i\Diamond\alpha)$
$(F4)$ $\diamondminus_i\alpha \wedge \diamondminus_i\beta \supset \diamondminus_i(\alpha \wedge \diamondminus_i\beta) \vee \diamondminus_i(\beta \wedge \diamondminus_i\alpha)$
$(F5)$ $\diamondminus_i\alpha \supset \diamondminus_i(\alpha \wedge \boxminus_i(\neg\alpha \supset \boxminus_i\neg\alpha) \wedge \boxminus_j\boxminus_i\neg\alpha), \quad j \neq i$

The following theorem constitutes the main contribution of the paper:

Theorem 2.2 *1. Satisfiability of formulas in $\mathcal{L}_n$ is decidable in nondeterministic double exponential time.*

2. Finitary satisfiability of formulas in $\mathcal{L}_n$ is decidable in nondeterministic double exponential time.

We can also consider systems with unboundedly many agents. Let $\mathbf{N}$ denote the set $\{0, 1, \ldots\}$. We can define an *ACSA* (Asynchronously Communicating Sequential Agents) to be a tuple $S = (E, \leq, \eta)$, where E and $\leq$ are as before, but $\eta : E \to \mathbf{N}$ is a naming function such that the range of η is finite. Thus, the system allows a finite but unbounded number of agents. The logic, however, needs to allow for countably many past-indexed modalities. Define $\mathcal{L}$ to be the language whose formulas are built up as before, except that we include $\diamondminus_i\alpha$ as a formula, whenever α is a formula and $i \in \mathbf{N}$. The notions of satisfiability and finitary satisfiability for formulas in $\mathcal{L}$ over ACSAs and finitary ACSAs, respectively, can be suitably defined. We have the following result also:

Theorem 2.3 *1. Satisfiability of formulas in $\mathcal{L}$ over ACSAs is decidable in nondeterministic double exponential time.*

2. *Finitary satisfiability of formulas in $\mathcal{L}$ (over finitary ACSAs) is decidable in nondeterministic double exponential time.*

We can also extend our frames to allow synchronous communication.

Definition 2.4 *$S = (E, \leq, \eta)$ is a* CSA *(Communicating Sequential Agents) when*

1. *$(E, \leq)$ is a partially ordered set of event occurrences.*
2. *$\eta : E \to \wp_{fin}(\mathbf{N})$ is a naming function assigning to each e in E a non-empty finite subset of $\mathbf{N}$.*
3. *$\forall e \in E : \ \forall j \in \mathbf{N} : \ \{e' \mid e' \leq e, \ j \in \eta(e')\}$ is totally ordered by $\leq$.*

We interpret $j \in \eta(e)$ as the agent j participating in the event e. Thus $\eta(e) = \{1, 2\}$ can stand for a synchronization "handshake" between agents 1 and 2. Note that only finitely many agents can synchronize at a time. *Finitary* CSAs can be defined by restricting each event to have a finite past.

Our logical language also needs to be changed slightly. We refer to [LRT] for details. Our result goes through for the richer frames as well:

Satisfiability as well as finitary satisfiability over CSAs of the logic defined in [LRT] is decidable in nondeterministic double exponential time.

3 Proof Ideas

In this section, we give a sketch of the proof of Theorem 2.2(1). The other theorems can be proved similarly. The full paper will contain detailed proofs of all the theorems mentioned in Section 2.

It can be easily checked that the logic $\mathcal{L}_n$ lacks the *finite model property*: consider the formula $\Box(\alpha \supset \Diamond\neg\alpha) \wedge \Box(\neg\alpha \supset \Diamond\alpha)$. We define a notion of *quasi-models* and show that the logic does possess the *finite quasi-model property.* Thus a formula is satisfiable if and only if it is finitely quasi-satisfiable. In fact, given a formula α of length m, we show that α is satisfiable iff it is satisfiable in a quasi-model of size $2^{2^{cm}}$ for some constant $c > 0$ determined by the logical language. This gives us the required decision procedure.

While models are based on partial orders, the quasi-models use pre-orders.

Definition 3.1 *A* quasi-frame *is a tuple $F = (W, \sqsubseteq, \eta)$, where $(W, \sqsubseteq)$ is a pre-order and $\eta : W \to \{1, \ldots n\}$ is a naming function such that*

$$\forall w \in W : \ \forall j \in \{1, \ldots, n\} : \ \{w' \sqsubseteq w \mid \eta(w') = j\} \text{ is totally ordered by } \sqsubseteq.$$

A *quasi-model* is a pair $Q = (F, V)$, where $F = (W, \sqsubseteq, \eta)$ is a quasi-frame and $V : W \to 2^{Prop}$ is the valuation. The notion $Q, w \models^q \alpha$ can be defined as for models. Quasi-satisfiability and quasi-validity are suitably defined. It can be checked that the formulas $(F1) - (F4)$ of the previous section are also quasi-valid.

The notion of *subformula closure* of a formula turns out to be crucial for proving decidability.

Definition 3.2 *Let α be a formula.*

1. *$CL'(\alpha)$ is the least set of formulas containing α and satisfying the conditions*
   (a) *$\neg\beta \in CL'(\alpha)$ implies $\beta \in CL'(\alpha)$.*
   (b) *$\beta \vee \gamma \in CL'(\alpha)$ implies $\{\beta, \gamma\} \subseteq CL'(\alpha)$.*
   (c) *$\diamondminus_i \beta \in CL'(\alpha)$ implies $\beta \in CL'(\alpha)$.*
   (d) *$\Diamond\beta \in CL'(\alpha)$ implies $\{\beta\} \cup \{\boxminus_j \Diamond\beta \mid j \in \{1, \ldots, n\}\} \subseteq CL'(\alpha)$.*
2. *$CL(\alpha) \stackrel{\text{def}}{=} CL'(\alpha) \cup \{\neg\beta \mid \beta \in CL'(\alpha)\}$.*

For any α, $CL(\alpha)$ is finite and linear in the size of α.

The proof of decidability consists of two parts: unfolding and folding. The former shows that every quasi-satisfiable formula is also satisfiable. The latter shows that every model for a formula can be "folded down" into a finite quasi-model for it.

Lemma 3.3 (Unfolding) *If α is quasi-satisfiable in a finite quasi-model, then it is satisfiable.*

The proof of the unfolding lemma closely follows classical lines: for example, the one in [HC] to obtain irreflexive structures. We make countably many copies of every element in the quasi-model and connect up suitably so that antisymmetry is ensured. In fact, if $Q = ((W, \sqsubseteq, \eta), V)$ is the given quasi-model, we define a model $M = ((E, \leq, \eta'), V')$ where $E \stackrel{\text{def}}{=} W \times \mathbf{Z}$. (Here $\mathbf{Z}$ denotes the set of integers.)

Proving the second part is hard, and hence we give the proof strategy below.

Lemma 3.4 (Folding) *If α is a satisfiable formula, then α is quasi-satisfiable in a quasi-model of size $2^{2^{cm}}$, where $m = |\alpha|$ and $c > 0$ is a constant.*

Proof. Fix a satisfiable formula α_0. Let $M = (S, V)$ be the given model such that $S = (E, \leq, \eta)$ and for some $e_0 \in E$, we have $M, e_0 \models \alpha_0$. Throughout the rest of this section, we refer implicitly to $S = (E, \leq, \eta)$ and let CL denote $CL(\alpha_0)$.

Definition 3.5 *Define* $\mu : E \to 2^{CL}$ *by* $\mu(e) \stackrel{\text{def}}{=} \{\diamondminus_i\gamma \in CL \mid M, e \models \diamondminus_i\gamma\}$.

It is obvious from the definition that whenever we have $e_1 \leq e_2$ in M, we also have $\mu(e_1) \subseteq \mu(e_2)$. Further, when $\diamondminus_i\alpha \in \mu(e)$, there exists e' such that $e' \leq e$, $\eta(e') = i$ and $M, e' \models \alpha$. In fact, this assertion can be made stronger. Among all e' which satisfy these conditions, we can look for the "virtually earliest" one. This notion is defined below.

Definition 3.6 *Suppose* $\diamondminus_i\alpha \in \mu(e)$. *We say* e' *is a* witness *for* $(e, \diamondminus_i\alpha)$ *iff the following conditions hold:*

1. $e' \leq e$, $\eta(e') = i$ *and* $M, e' \models \alpha$.
2. *for every* $e'' \leq e$, *if* $\eta(e'') = i$ *and* $M, e'' \models \alpha$, *then* $\mu(e') \subseteq \mu(e'')$.

We say e' is a witness for e if it is a witness for $(e, \diamondminus_i\alpha)$ for some $\diamondminus_i\alpha \in \mu(e)$.

Proposition 3.7 *If* $\diamondminus_i\alpha \in \mu(e)$, *then there exists a witness* e' *for* $(e, \diamondminus_i\alpha)$. *Moreover,*

1. *If* e_1 *and* e_2 *are both witnesses for* $(e, \diamondminus_i\alpha)$, *then* $\mu(e_1) = \mu(e_2)$.
2. *If* e *is a witness for* $(e_1, \diamondminus_i\alpha)$ *and* $e_1 \leq e_2$, *then* e *is a witness for* $(e_2, \diamondminus_i\alpha)$ *as well.*

The idea of using witnesses allows us to "prune" the given model M. The pruning procedure retains the set E in the given model, but orders fewer event occurrences from distinct agents:

Proposition 3.8 $(E, \ll, \eta)$ *is a frame, where* $e_1 \ll e_2$ *iff*

- $e_1 \leq e_2$, *and*
- $\eta(e_1) = \eta(e_2)$ *or there is a witness* e_3 *for* e_2 *such that* $e_1 \leq e_3 \leq e_2$ *and* $\eta(e_1) = \eta(e_3)$.

Observe that for e_1, e_2 such that $\eta(e_1) = \eta(e_2)$, we have $e_1 \ll e_2$ iff $e_1 \leq e_2$.

The following proposition assures us that we might as well forget about the given model and work with the "pruned" structure $(E, \ll, \eta)$.

Proposition 3.9 *Let* $e \in E$.

1. *If* $M, e \models \diamondminus_i\alpha$ *then* $\exists e' \ll e$: $\eta(e') = i$ *and* $M, e' \models \alpha$.

2. *If $M, e \models \Diamond\alpha$, then $\exists e'$: $\eta(e') = \eta(e)$, $e \ll e'$ and $M, e' \models \alpha$.*

We now proceed to define the equivalence relation on E which will give us the required filtration.

Definition 3.10 *1. $\Pi : E \to 2^{(\{1,\ldots,n\}\times 2^{CL})}$ is given by:*

$$\Pi(e) \stackrel{\text{def}}{=} \{(\eta(e'), \mu(e')) \mid e' \ll e\}.$$

2. *$\sim\, \subseteq E \times E$ is given by: $e_1 \sim e_2$ iff $(\eta(e_1) = \eta(e_2)$ and $\Pi(e_1) = \Pi(e_2))$.*

Proposition 3.11 *Suppose $e_1 \sim e_2$. Then the following hold:*

1. *$\mu(e_1) = \mu(e_2)$.*
2. *If e_3 is a witness for $(e_1, \Diamond\!\!\!\cdot_i\alpha)$, then there exists a witness e_4 for $(e_2, \Diamond\!\!\!\cdot_i\alpha)$ such that $\mu(e_3) = \mu(e_4)$.*

The following lemma asserts that $\sim$ behaves like a "past bisimulation" with respect to $\ll$.

Lemma 3.12 *Suppose $e_1 \sim e_2$ and $e_3 \ll e_1$. Then there exists $e_4 \ll e_2$ such that $e_3 \sim e_4$.*

To get an idea of how this lemma is proved, assume the hypothesis. By definition of $\sim$, we can find $e_4 \ll e_3$ such that $\mu(e_3) = \mu(e_4)$ and $\eta(e_3) = \eta(e_4)$. This is far from proving Π equality for e_3 and e_4. Now suppose $e \ll e_3$. We can find $e' \ll e_2$ of the same μ and η as e. If we can manage to transport e' to the past of e_4, we are done, as that would prove $\Pi(e_3) \subseteq \Pi(e_4)$. If $\eta(e') = \eta(e_4)$, we can use backward linearity to get the result. Otherwise, $\eta(e)$ and $\eta(e_3)$ are also different: so find e'' between e and e_3 which is a witness for some formula $\Diamond\!\!\!\cdot_i\alpha$, use Proposition 3.7(2) and Proposition 3.11 to find a witness in the past of e_2, observe that $\Diamond\!\!\!\cdot_i\alpha \in \mu(e_4)$ and thus use Proposition 3.7 to transport $\mu(e'')$ to $\Pi(e_4)$. The rest of the proof proceeds as in the previous case.

With these preliminaries, we are now ready to define the desired filtration of the given model. So far we have considered only modal equivalence; that is, two equated events (in the given model) satisfy the same modal formulas in CL. To take care of the propositions, we now introduce a refinement of the equivalence relation we have been using so far:

Definition 3.13 *$\approx\, \subseteq E \times E$ is defined by:*

$$e_1 \approx e_2 \text{ iff } e_1 \sim e_2 \text{ and } V(e_1) \cap CL = V(e_2) \cap CL.$$

Below, we use $[e]$ to denote the equivalence class of e under $\approx$, where $e \in E$.

Proposition 3.14 *$Q \stackrel{\text{def}}{=} (F, V')$ is a quasi-model, where $F = (W, \sqsubseteq, \eta')$ and V' are defined as follows:*

1. $W \stackrel{\text{def}}{=} \{[e] \mid e \in E\}$.
2. $[e_1] \sqsubseteq [e_2]$ *iff* $\exists e, e' : \ e \sim e_1, \ e' \sim e_2$ *and* $e \ll e'$.
3. $\eta'([e]) \stackrel{\text{def}}{=} \eta(e)$.
4. $V'([e]) \stackrel{\text{def}}{=} V(e) \cap CL$.

Note that in the definition of $\sqsubseteq$ we have used $\sim$ and not $\approx$. To show that F is indeed a quasi-frame, we make heavy use of Lemma 3.12. The lemma also enables us to prove:

Lemma 3.15 $\forall \gamma \in CL : \forall e \in E : \ M, e \models \gamma$ *iff* $Q, [e] \models^q \gamma$.

Since $M, e_0 \models \alpha_0$, we get $Q, [e_0] \models^q \alpha_0$ and thus α_0 is quasi-satisfiable. Further, the quasi-model we constructed has size bounded by 2^{2^m}, where $m = |CL|$. This completes the proof of Lemma 3.4. $\square$

When proving Theorem 2.2(2) for the finitary case, our unfolding cannot use the construction of Lemma 3.3 since the model produced is not finitary. We have to discover a condition on quasi-models which allows unfolding in the future direction, yet preserves the satisfaction of past formulas. Formulating such a condition, folding models so that they produce quasi-models satisfying this condition and discovering an unfolding construction make the proof quite nontrivial. The details will be provided in the full paper.

Admittedly, our decision procedures have a very high complexity. It is far from clear that this is the best that can be done. For instance, using automata-theoretic techniques as in [VW] might give a better decision procedure.

Acknowledgement: We thank Madhavan Mukund for lengthy discussions on the decidability of this and other logics.

References

[Apt] K.R. Apt: Ten years of Hoare logic: a survey – Part I, *ACM Trans. Prog. Lang. Syst.* **3**,4 (1981) 431-483.

[Bur] J.P. Burgess: Decidability for branching time, *Studia Logica* **XXXIX** (1980) 203-218.

[CM] K.M. Chandy, J. Misra: *Parallel program design: a foundation* (Addison-Wesley, 1988).

[Eme] E.A. Emerson: Temporal and modal logic, in J. van Leeuwen (ed.): *Handbook of theoretical computer science* **B** (Elsevier, 1990) 995-1072.

[HC] G.E. Hughes, M.J. Creswell: *A companion to modal logic* (Methuen, 1982).

[Har] D. Harel: Dynamic logic, in D. Gabbay, F. Guenthner (eds.): *Handbook of philosophical logic* **II** (Reidel, 1984) 497-604.

[Lam] L. Lamport: Time, clocks and the ordering of events in a distributed system, *Commun. ACM* **21**,7 (Jul 1978) 558-565.

[LMRT] K. Lodaya, M. Mukund, R. Ramanujam, P.S. Thiagarajan: Models and logics for true concurrency, *Sādhanā* **17**, Part 1 (Indian Academy of Sciences, Mar 1992) 131-165.

[LR] K. Lodaya, R. Ramanujam: Tense logics for local reasoning in distributed systems, *Proc. 11th FST & TCS, LNCS* **560** (1991) 71-88.

[LRT] K. Lodaya, R. Ramanujam, P.S. Thiagarajan: Temporal logics for communicating sequential agents: I, *Int. J. Found. Comput. Sci.* **3**,2 (1992) 117-159.

[MT] M. Mukund, P.S. Thiagarajan: A Petri net model of asynchronously communicating sequential processes, in R. Narasimhan (ed.): *A perspective in theoretical computer science – commemorative volume for Gift Siromoney* (World Scientific, 1989) 165-198.

[NPW] M. Nielsen, G. Plotkin, G. Winskel: Petri nets, event structures and domains I, *Theoret. Comput. Sci.* **13** (1980) 86-108.

[Pen] W. Penczek: A temporal logic for event structures, *Fund. Inform.* **XI** (1988) 297-326.

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

[Rab] M.O. Rabin: Decidability of second order theories and automata on infinite trees, *Trans. Amer. Math. Soc.* **141** (1969) 1-35.

[Sti] C. Stirling: Temporal logics for CCS, *Proc. REX 88, LNCS* **354** (1989) 660-672.

[VW] M. Vardi, P. Wolper: Automata theoretic techniques for modal logics of programs, *J. Comput. Syst. Sci.* **32**,2 (1986) 183-221.