The Logic of Distributed Protocols
(Preliminary Report)

Richard E. Ladner [1]
Department of Computer Science
University of Washington
Seattle, Washington 98195

John H. Reif [2]
Aiken Computation Laboratory
Harvard University
Cambridge, Massachusetts 02138

ABSTRACT

A propositional logic of distributed protocols is introduced which includes both the logic of knowledge and temporal logic. Phenomena in distributed computing systems such as asynchronous time, incomplete knowledge by the computing agents in the system, and game-like behavior among the computing agents are all modeled in the logic. Two versions of the logic, the *linear logic of protocols* (*LLP*) and the *tree logic of protocols* (*TLP*) are investigated. The main result is that the set of valid formulas in *LLP* is undecidable.

[1]Research supported by the National Science Foundation Grant No. DCR-8402565.

[2]Research supported by the Office of Naval Research Contract No. N00014-80-C-0647.

1 Introduction

Motivation and Background

Ever since the seminal work of Floyd [6], Naur [14], and Hoare [9] there has been an increasing interest in the development and analysis of formal tools to reason about program behavior. This interest has led to the creation of a large number of logical systems for handling various types of program behavior. Such logical systems are called *logics of programs*. Notable examples of logics of programs are dynamic logic originally defined by Pratt [19] and temporal logic of programs introduced by Pnueli [18]. The large body of literature in logics of programs has deepened our understanding about reasoning about program behavior, not only about how to reason, but also about what inherent limitations there are in our reasoning. Important results include the discovery of complete proof systems for some of the logics, computational upper and lower bounds for some of the logics, and the establishment of the relative expressive power of different logics.

More recently it has been recognized that the logics of programs developed so far are inadequate for reasoning about the behavior of distributed algorithms or, what we call *protocols*. Generally speaking, a protocol is an algorithm whose execution is shared by a number of independent participants or what we call *players*. Each player may be unaware of what the other players are exactly doing. Therefore, a key ingredient found in the behavior of many protocols (and not found in sequential or parallel algorithms) is the *lack of knowledge* about the complete state of the protocol by each of its players. A good example of this is found in the work on distributed agreement by Pease, Shostok, and Lamport [21]. They show that it is impossible for there to be a protocol such that the non-faulty players agree on a common value when there is a total of three players and at most one is faulty. The proof relies on the fact that, in the presence of inconsistent information coming from the other two players, it is possible for a non-faulty player to remain ignorant of which of the other two players is "lying". They also show that there is a protocol which allows the non-faulty players to agree on a common value when there are four players and at most one is faulty. The proof relies on the fact that in two rounds of exchanging information the non-faulty players can know enough to come to agreement. In one round they cannot know enough. Any logic of protocols must include as part of it a logic of knowledge. A number of researchers have begun the development of logics of distributed protocols which include the notion of knowledge [7] [8] [10] [12] [13] [16].

In this paper we present two new logics of distributed protocols and prove some basic theorems about them. The logics are basically logics of knowledge which incorporate a notion of global time. The logics are capable of implicitly modeling asynchronous time by allowing the players to be ignorant of global time. In addition, the logics are capable of implicitly modeling game-like behavior of distributed computation.

Any logic of protocols should also include a logic for time, or temporal logic. In particular, a key feature of time in a distributed computing setting is that it is asynchronous, that is, there is no global time that the independent computing agents know about. Asynchrony of time does not rule out the possibility of global time, only that if there is a global time then the individual computing participants have only limited knowledge of it. Temporal logics have been extensively studied in the context of both sequential and concurrent programs [1], [2], [3], [18], [25].

In addition, the behavior of distributed protocols exhibit game-like qualities which should be

modeled in the logic of protocols. For example, the distributed agreement problems described above are "competitions" of a sort. Some of the players must come to agreement despite the potential maliciousness of lying opponents. Another example of game-like property in distributed computing is the definition of the guaranteed lockout problem [11], [17]. Several formal treatments of game-theoretic aspects of computing have been introduced [17], [22], [23]. An early attempt to include game-like properties into a logic of programs is found in the work of Reif and Peterson [24]. Games and game-like behavior can be mathematically formalized in what is called *game theory*. It is not surprising that game theory should play an important role in distributed computing, because it already plays a significant role in Economics and Ecology, where multiple independent, yet interacting, players co-exist, compete, and cooperate [15]. We propose a (Church/Turing-like) Thesis for distributed computing.

- ***Distributed protocols are equivalent to (i.e., can be formally modeled as) games.***

The fact that a logic is capable of modeling game-like behavior is critical if it is claimed to accurately model properties of distributed protocols.

The work done so far on the logic of protocols which includes knowledge also includes, to varying degrees, time and to varying degrees is capable of implicitly modeling games [7] [8] [10] [12] [13] [16]. We feel, however, that the logics developed so far do not adequately model phenomena in protocols like asynchrony. In addition, they seem (with the possible exception of Parikh and Ramanujam's work [16]) to be capable of modeling only restricted classes of games. We feel that the logics of distributed protocols that we present go further than those of previous researchers in two ways:

- Our logics of protocols better model asynchronous time.
- Our logics better model the game-like behavior of protocols.

Indeed, the proof of our main theorem on the undecidability of one of the two logics relies heavily the fact that the logic is capable of modeling a certain three-player incomplete information game, where the players move asynchronously.

The Logic of Protocols

Informally, the logics we define are propositional logics with modal operators $\Box$, $\Box^*$, and K_i for $1 \leq i \leq t$ for some t. The number t refers to the number of players and $K_i p$ means that player i knows that p holds. The expression $\Box p$ means that in the next moment p must hold and $\Box^* p$ means that now and forever in the future p must hold. We can imagine that there is a global time which each of the players may or may not be knowledgeable of. This is akin to the common way of thinking about time in a distributed system where we imagine that the events taking place in the system are actually totally ordered but that each process in the system is unaware of the ordering except for those events that take place in its own process.

We will define two logics of protocols, the *linear logic of protocols* (*LLP*) and the *tree logic of protocols* (*TLP*). The semantics of *LLP* and *TLP* are related to but not the same as, respectively, the linear and branching time semantics for the temporal logic of programs [18] [1]. In the semantics of both *LLP* and *TLP* global time is allowed to branch, but in *LLP* from each player's point of view time is linear. This models the fact that in distributed systems, from one process' point of view there is only one visible history of events, but many system

histories may be consistent with that one visible history. In *TLP* global time can branch and players can be aware of it. The distinction between *LLP* and *TLP* can best be understood game theoretically. In a certain sense a model in a semantics for the logic of protocols will define a game, what players do and do not know about the game, and a strategy for each of the players in the game. In *LLP* each player receives no direct knowledge of the states of the other players. Hence, whatever knowledge about the other players states can be, at best, inferred. In *LLP* each player's strategy must be a blindfold strategy, that is, what each player does in a play of the game is preordained and cannot be changed as the game progresses. In *TLP* there is no preordained strategy for each player. A player can react differently in different plays of the game. In a practical sense, *LLP* models a distributed computing situation where the actions of each computing participant are determined as locally as possible. In *TLP* the actions of each participant are dependent on the interaction between the participants.

Asynchrony of time is modeled in *LLP* and *TLP* by the fact that, although there is a global time, the individual players may not have full knowledge about it. So the combination of global time and lack of knowledge about it by the players models asynchronous time. Time can be passing, but an individual player may have no idea if time has progressed at all. Game-like behavior is modeled in *LLP* and *TLP* in a combination of ways. First, time can branch which models a choice of possible moves in a game. Second, players have lack of knowledge which models incomplete information in a game. Third, there may be a number of players which models multi-person games. To understand the semantics of *LLP* and *TLP* it is sometimes helpful to imagine an omniscient player, player 0, different from the players who may have lack of knowledge, players 1 to t. Player 0 knows all. It knows about the global time and every thing the other players know. In a distributed computing system, player 0 corresponds to the system taken as a whole, while players 1 to t model the individual computing agents in the system.

The Results

The main results of this paper are:

- The introduction of a logic of protocols that adequately models knowledge and time, including asynchrony and game-like behavior.
- Theorems that reveal the mathematical structure of the models for the logic of protocols.
- A proof that the set of valid formulas in *LLP* is undecidable.

The third result is surprising for a number of reasons. First, *LLP* is a propositional logic. Second, taken separately, each of the logic of knowledge and the logic of time are decidable. In particular, the propositional temporal logic is decidable, in fact decidable in deterministic exponential time. (The temporal logic with just $\Box$ and $\Box^*$ is a special case of dynamic logic which has deterministic exponential time upper and lower bound complexity [5], [20].) Further, the propositional logic of knowledge without time is also decidable. (The logic of knowledge is PSPACE complete and the logic of knowledge limited to one player is Co-NP complete [7].) But combining the two logics together in the way we suggest in this paper leads to an undecidable logic. This result adds more strong evidence to the thesis that reasoning about distributed systems is much more difficult than reasoning about sequential programs. Finally, the proof of the undecidability of *LLP* is interesting in its own right. The proof genuinely uses

the asynchrony, lack of knowledge, and game-like behavior that are expressible in the logic of protocols.

There is still a lot that we do not know about the logics of protocols. For example, it is still an open question whether or not the set of valid formulas in *TLP* is decidable. It would seem that *LLP* is just a special case of *TLP*, but all we know at present is that every formula that is valid in *TLP* is also valid in *LLP*. There is no reason *prima facie* that the undecidability of *LLP* would imply the undecidability of *TLP*. Moreover, our proof of the undecidability of *LLP* relies heavily on the fact that we are using the linear semantics for the logic. Our proof of the undeciability of *LLP* also relies on there being more than one player. In the case there is just one player we have strong evidence that the logic of protocols with either the linear or tree semantics is decidable.

The paper is organized as follows: In Section 2 we give the formal definitions for the logics of protocols along with an example. In Section 3 we present some fundamental theorems which reveal the structure of models for the logics. In Section 4 we present the main result that the set of valid formulas in *LLP* is undecidable.

2 The Formal Syntax and Semantics

In this section we give the formal syntax and sematics for the logics of protocols with t players. We assume we have a set of propositional variables Φ, Boolean logical connectives, $\neg$, $\wedge$, Boolean constants 0 and 1, and modal operators $\Box$, $\Box^*$, and K_i for $1 \leq i \leq t$. The set of logic of protocols *formulas* are defined inductively:

1. If $P \in \Phi$ then P is a formula and 0 and 1 are formulas.

2. If p and q are formulas then so are $\neg p$, $p \wedge q$, $\Box p$, $\Box^* p$, and $K_i p$ for $1 \leq i \leq t$.

The set of formulas is the smallest set satisfying 1. and 2. above.

The additional boolean operators $\vee$, $\supset$, and $\equiv$ can be defined in the usual way from $\neg$, $\wedge$, 0, and 1. ($p \vee q = \neg(\neg p \wedge \neg q)$, $p \supset q = \neg p \vee q$, $p \equiv q = (p \supset q) \wedge (q \supset p)$). In addition, we can define the dual modal operators for $\Box$, $\Box^*$, and K_i. Define $\Diamond p = \neg\Box\neg p$, $\Diamond^* p = \neg\Box^*\neg p$.

In the semantics we are about to define formally we imagine some underlying distributed system with global time and t players. The system can move from one state of the system to another and as the system moves sequences of states of the system form histories, the last state in the history being the current state. One history extends a second if the sequence of states that forms the first history has as a prefix the second history. A successor of a history is an extension by just one move of the system. In general the system is nondeterministic in that there may be more than one successor. The propositional variables represent primitive statements that hold or not in given states. In general a formula will hold for a given history or not. Let w be a history of the system. Informally, $\Box p$ holds at w if p holds at every successor v of w, and $\Box^* p$ holds at w if p holds at w and at every history v that extends w. Thus, $\Diamond p$ holds at w if p holds in some successor history v, and $\Diamond^* p$ holds at w if p holds at w or at some history v that extends w. The formula $K_i p$ holds at w if p holds at every history v indistinguishable from w from player i's point of view. Shortly, we will make precise the notion of "indistinguishable from player i's point of view."

Definition of Models

Let Φ be a set of propositional variables. A *model* or *tree model* for the logic of protocols with t players is a $(t+4)$-tuple

$$M = (W, W_0, \pi, \rho, \preceq_1, ..., \preceq_t)$$

where W is a set of states, $W_0 \subseteq W$ is a set of initial states, $\pi : W \rightarrow 2^{\Phi}$ is a mapping indicating which propositional variables hold at which states, $\rho \subseteq W \times W$ is a relation indicating the next move relation, and for each i, $\preceq_i \subseteq W^* \times W^*$ indicating what player i does and does not know, where W^* is the set of all finite sequences (strings) of members of W. In addition M must satisfy properties listed $A_1 - A_3$ below.

Given a nonempty sequence $w \in W^*$ define $last(w)$ to be the last member of the sequence. We define the set of *histories* of M, W^ρ, inductively as follows. If $w \in W_0$ then $w \in W^\rho$ and if $w \in W^\rho$, $(last(w), a) \in \rho$ then $wa \in W^\rho$, and there are no other members of W^ρ. If u and v are two histories then define $u \subseteq^\rho v$ if u is a prefix of v and $u \subset^\rho v$ if u is a proper prefix of v, that is $u \subset^\rho v$ if $u \subseteq^\rho v$ and $u \neq v$. In what follows we let i be arbitrary where $1 \leq i \leq t$. In general, u, v, w, x, y will range over W^ρ unless otherwise specified. The first additional property of M is:

A_1 : $\preceq_i$ is a reflexive and transitive relation on W^ρ.

This first property begins to express the meaning of the relation $\preceq_i$. Intuitively, $u \preceq_i v$ means that from player i's point of view the history v is an extension of the history u. Property A_1 expresses that player i's view of history is reflexive and transitive. We can now express the idea that two histories are indistinguishable from player i's point of view by defining the equivalence relation, $\approx_i$, on W^ρ:

$$u \approx_i v \text{ if and only if } u \preceq_i v \text{ and } v \preceq_i u.$$

The relation $\approx_i$ is an equivalence relation because $\preceq_i$ is reflexive and transitive. Precisely, $u \approx_i v$ means that histories the u and v are indistinguishable from player i's point of view. Further define: $u \prec_i v$ if and only if $u \preceq_i v$ and $u \not\approx_i v$.

The remaining properties are:

A_2 : If $u \subseteq^\rho v$ then $u \preceq_i v$,

A_3 : If $u \preceq_i v$ then $v' \approx_i u$, for some $v' \subseteq^\rho v$.

Properties A_2 and A_3 express more clearly how player i can come to view two histories to be indistinguishable. First, if one history extends another then player i's view is consistent with that extension. Second, if one history extends another from player i's point of view then there must be an earlier time in the first history when player i's view was the same as his view of the second history.

A *linear model* is a model which satisfies the following additional property:

A_4 : $u \preceq_i v$ or $v \preceq_i u$.

Some fundamental properties of models and linear models will be given in the next section.

Definition of Truth

Let $M = (W, W_0, \pi, \rho, \preceq_1, ..., \preceq_t)$ be a model and $w \in W^\rho$. If p is a formula then we define the relation $M, w \models p$ inductively on the structure of p.

1. $M, w \models 1$ and $M, w \not\models 0$,
2. $M, w \models P$ if and only if $P \in \pi(last(w))$, if $P \in \Phi$,
3. $M, w \models \neg p$ if and only if $M, w \not\models p$,
4. $M, w \models p \wedge q$ if and only if $M, w \models p$ and $M, w \models q$,
5. $M, w \models \Box p$ if and only if for all $a \in W$ such that $wa \in W^\rho$, $M, wa \models p$,
6. $M, w \models \Box^* p$ if and only if for all $v \in W^*$ such that $wv \in W^\rho$, $M, wv \models p$,
7. $M, w \models K_i p$ if and only if for all $v \in W^\rho$ such that $v \approx_i w$, $M, v \models p$.

Define TLP_t and LLP_t to be the logic of protocols with t players where the interpretation of formulas is made in, repectively, tree and linear models for the logic of protocols with t players. Define TLP and LLP to be the logic of protocols corresponding to TLP_t and LLP_t where the number of players t is unbounded.

We say that a formula p is *satisfiable in* TLP_t if there is a tree model $M = (W, W_0, \pi, \rho, \preceq_1, ..., \preceq_t)$ and a $w \in W_0$ such that $M, w \models p$. Further, p is *valid in* TLP_t if for every tree model $M = (W, W_0, \pi, \rho, \preceq_1, ..., \preceq_t)$ and every $w \in W_0$, $M, w \models p$. A formula is *satisfiable in* LLP_t if it is satisfiable in some linear model and *valid in* LLP_t if it is valid in all linear models. We say that a formula p is *valid (satisfiable) in* TLP (LLP) if it is *valid (satisfiable) in* TLP_t (LLP_t) for some t where the knowledge operators, K_i in p are such that $i \leq t$.

Example

For example, consider a model $M = (\{a, b, c\}, \{a\}, \pi, \rho, \preceq_1)$ where

1. $\pi(a) = \pi(b) = \{P\}$, $\pi(c) = \emptyset$,
2. $\rho = \{(a, b), (b, c), (c, c)\}$,
3. $\preceq_1 = \{(a, a), (a, abc^i), (abc^i, abc^j) : i, j \geq 0\}$.

Figure 1. graphically illustrates what this model looks like.

In the model M, player 1 can distinguish the history a from the history ab, but cannot distinguish the histories ab, abc, $abcc$, ... Now consider the formula

$$p = K_1 \Box P \wedge \neg \Box K_1 P.$$

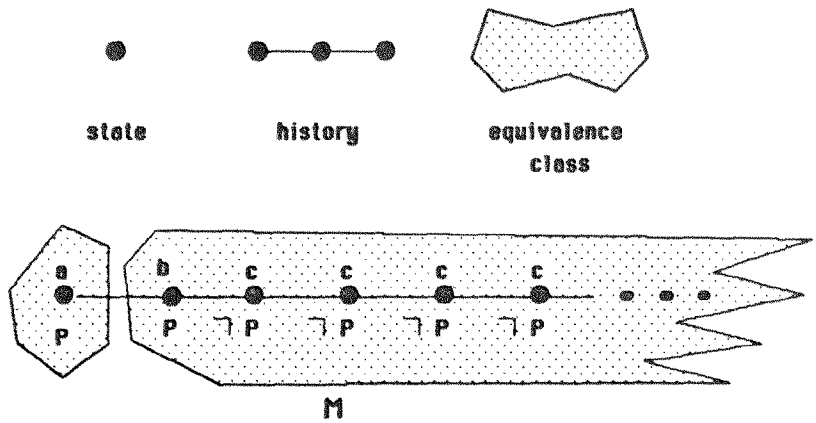

Figure 1: Illustration of the model M. Indistinguishable histories are enclosed.

We have $M, a \models p$. This formula is interesting because it is an instance of the negation of one of Lehman's axioms for his logic of knowledge and time [12]. This points out a fundamental difference between previous models and ours. Our models allow for asynchrony. Player 1 is aware of going from the history a to the history ab, but is totally unaware of going from history ab to abc. In history a he knows that in one step P will be true, but in one step he does not know that P will be true because in one step his knowledge of time suddenly becomes limited. On the other hand the slightly modified formula $K_1\Box^*P \wedge \neg\Box^*K_1P$ is not satisfiable which naturally implies that the formula $K_1\Box^*P \supset \Box^*K_1P$ is valid.

3 Structure of Models

The properties $A_1 - A_4$ by themselves do not give a clear picture of what tree and linear models look like. The goal of this section is to present by way of two theorems a concrete picture of tree and linear models. The following describes in word what models look like. In any model the histories, W^ρ, form a family of trees, where each non-initial history has a unique immediate predecessor history and every history has zero or more immediate successor histories. Consider each player i individually. The histories are partitioned into equivalence classes by the relation $\approx_i$. In the tree model case the equivalence classes themselves form a family of trees, where the predecessor and successors of an equivalence class are implicitly induced by the relation $\preceq_i$. In the linear model case the equivalence classes form a sequence where the linear order of the sequence is implicitly induced by the relation $\preceq_i$. Figure 3 describes the models graphically.

To lead to our two theorems we are required to pass through a sequence of lemmas. In what follows we let $M = (W, W_0, \pi, \rho, \preceq_1, ..., \preceq_t)$ be a model and let $1 \leq i \leq t$. Define $w \in W^\rho$ to be *i-minimal* if $v \preceq_i w$ implies $v \approx_i w$. The proofs of Lemmas 3.1 - 3.6 are omitted to save space.

Lemma 3.1 A member w of W^ρ is i-minimal if and only if $w \approx_i w_0$ for some $w_0 \in W_0$.

If $w \in W^\rho$ is not i-minimal define $pred_i(w)$ to be the unique w' such that for some $a \in W$, $w' \subseteq^\rho w'a \subseteq^\rho w$, and $w' \prec_i w'a \approx_i w$. Such a string exits by Lemma 3.1 and by property A_2.

Lemma 3.2 If $u, v \in W^\rho$ are such that $pred_i(u) \preceq_i v \preceq_i u$ then either $v \approx_i pred_i(u)$ or $v \approx_i u$.

The final lemma in this series shows that the $pred_i$ function is invariant under $\approx_i$.

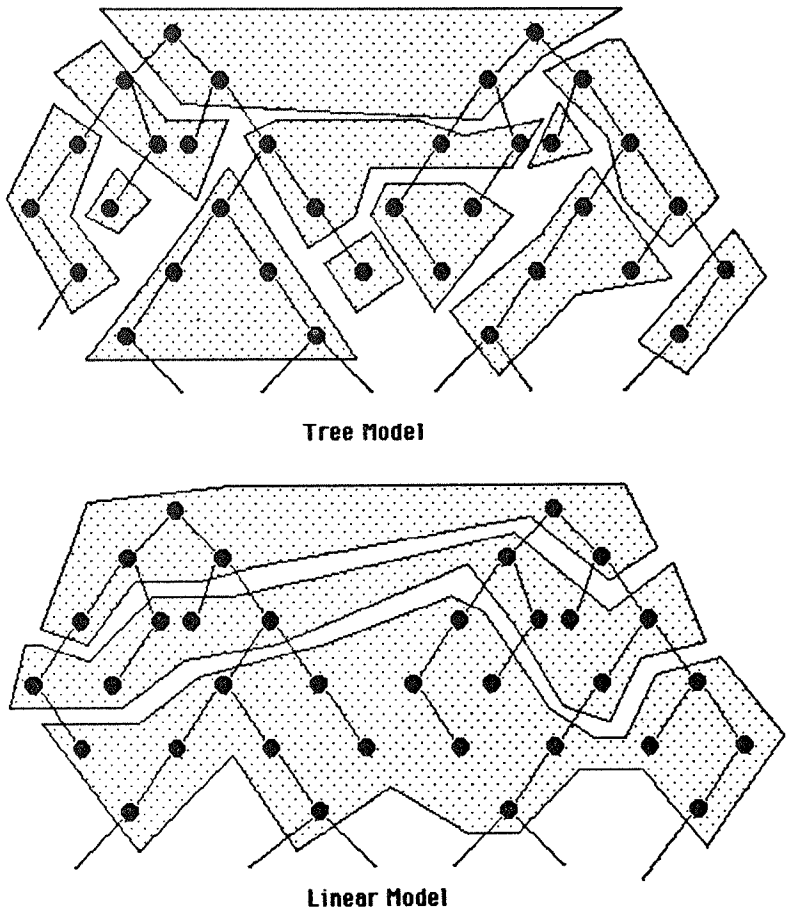

Figure 2: Graphical depiction of a tree model and a linear model.

Lemma 3.3 If $u, v \in W^\rho$ are such that $u \approx_i v$ and u is not i-minimal then $pred_i(u) \approx_i pred_i(v)$.

If $w \in W^\rho$ then define $succ_i(w)$ to be the set of all $wva \in W^\rho$ such that $v \in W^*$, $a \in W$, and $w \approx_i wv \prec_i wva$.

Lemma 3.4 If $u, v \in W^\rho$ are such that $v \in succ_i(u)$ then $pred_i(v) \approx_i u$.

Lemma 3.5 If $u, v, x \in W^\rho$ are such that $u \in succ_i(v)$ and $v \preceq_i x \preceq_i u$ then $x \approx_i v$ or $x \approx_i u$.

We also get a corresponding result to Lemma 3.3 if the model M is linear.

Lemma 3.6 Let M be a linear model. If $u, v, x, y \in W^\rho$ are such that $x \in succ_i(u)$, $y \in succ_i(v)$, and $u \approx_i v$ then $x \approx_i y$.

The properties listed in Lemmas 3.1 - 3.6 essentially give us the structure of the equivalence classes of W^ρ under the equivalence relation $\approx_i$. Let $[w]_i$ be the equivalence class of w under $\approx_i$. Define $[W^\rho]_i = \{[w]_i : w \in W^\rho\}$. The reflexive, transitive relation $\preceq_i$ induces a partial order $\leq_i$ on $[W^\rho]_i$ in a natural way. We say that $[u]_i \leq_i [v]_i$ if $u \preceq_i v$. The relation $\leq_i$ is reflexive, transitive, and antisymmetric on its domain $[W^\rho]_i$. We can further, define $[u]_i <_i [v]_i$ if $[u]_i \leq_i [v]_i$ and $[u]_i \neq [v]_i$.

Theorem 3.7 Let $M = (W, W_0, \pi, \rho, \preceq_1, ..., \preceq_t)$ be a tree model. The set $[W^\rho]_i$ ordered by $\leq_i$ is a family of trees. Specifically:

1. The roots of $[W^\rho]_i$ are $\{[w_0]_i : w_0 \in W_0\}$,
2. If $[w]_i$ is not a root then its parent is $[pred_i(w)]_i$,
3. The children or $[w]_i$ are $\{[v]_i : v \in succ_i(w)\}$.

The proof follows immediately from Lemmas 3.1 - 3.6.

Theorem 3.8 Let $M = (W, W_0, \pi, \rho, \preceq_1, ..., \preceq_t)$ be a linear model. The set $[W^\rho]_i$ is linearly ordered by $\leq_i$. Specifically:

1. The least member of $[W^\rho]_i$ is $[w_0]_i$ for any $w_0 \in W_0$,
2. If $[w]_i$ is not the least member of $[W^\rho]_i$ then its immediate predecessor is $[pred_i(w)]_i$,
3. If $[w]_i$ is not the greatest member of $[W^\rho]_i$ then its immediate successor is $[v]_i$ for any $v \in succ_i(w)$.

The proof follows from Lemmas 3.1 - 3.6.

The result of Theorem 3.8 will be used in the proof of the undecidability of the set of valid formulas in *LLP*.

4 The Undecidability Result

In this section we prove that the set of valid formulas in *LLP* is undecidable. More specifically we show that the set of satisfiable formulas in LLP_2 is not recursively enumerable. Since a formula p in the logic of protocols with two players is valid in *LLP* if and only if the formula $\neg p$ is not satisfiable in LLP_2, it follows that validity in *LLP* is undecidable if satisfiabilty in LLP_2 is undecidable.

Theorem 4.1 The set of satisfiable formulas in LLP_2 is not recursively enumerable.

Proof Some of the ideas behind this proof come from a proof of the undecidability of a certain three-player game which is an instance of the guaranteed lockout problem defined by Peterson and Reif [17]. The motivation behind this proof is implicitly game theoretic. The general plan is to show that given a Turing machine T a formula f_T can be constructed with the property that T runs without halting with a blank input if and only if the formula f_T is satisfiable in some linear model. Since the set of Turing machines that do not halt on a blank tape is not a recursively enumerable set then the set of satisfiable formulas in *LLP* is not recursively enumerable.

Let M be a one tape Turing machine with states Q, tape symbols Γ, $Q \cap \Gamma = \emptyset$, blank symbol $B \in \Gamma$, start state $q_0 \in Q$, and transition partial funtion $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{R, L\}$. Let $\Delta = Q \cup \Gamma \cup \{\#, ¢\}$, where $\#$ and $¢$ are new symbols. Define an *ID* to be a string of the form $¢a_1¢a_2¢...¢a_m¢$ where $a_1a_2...a_m \in \Gamma^*Q\Gamma^*$. Define the *initial ID* to be $¢q_0¢B¢$. Finally, define an *infinite computation* to be an infinite string of the form

$$\#^{m_0}C_0\#^{m_1}C_1\#^{m_2}C_2\#^{m_3}...$$

where

1. C_i is an *ID* for all i,

2. C_0 is the initial *ID*,

3. C_{i+1} follows from C_i in one move of T for all i,

4. $|C_i| = 2i + 5$ for all i,

5. $m_i > 0$ for all i,

The reason why infinite computations have this unusual form will be clarified later. Let $q = \lceil log_2|\Delta|\rceil$. With each symbol $\sigma \in \Delta$ we assign a specific Boolean vector $(\sigma_1, ..., \sigma_q)$ of length q. When we talk about σ in the context of a formula we are referring to its representation as a Boolean vector. The formula we will construct will have propositional variables X, and S_{ij} for $i = 1, 2$ and $1 \leq j \leq q$. Let S_1 and S_2 be the vectors $(S_{11}, S_{12}, ..., S_{1q})$ and $(S_{21}, S_{22}, ..., S_{2q})$. In general, we will think of S_1 and S_2 as variables that take on values in Δ. If U and V are two Boolean vectors of the same length m. We use the abbreviation "$U = V$" to stand for $(U_1 \equiv V_1) \wedge ... \wedge (U_m \equiv V_m)$.

We define a function *collapse* : $\Delta^\omega \rightarrow \Delta^\omega \cup \Delta^*$ by *collapse*(s) is the string obtained from s by collapsing multiple contiguous occurrences of the same symbol to one occurrence. That is, if a_i is a member of Δ for all i, $a_i \neq a_{i+1}$ for all i, and $m_i > 0$ for all i then

$$collapse(a_0^{m_0} a_1^{m_1} ...) = a_0 a_1 a_2 ....$$

Further, if a_i is a member of Δ for $0 \leq i \leq r$, $a_i \neq a_{i+1}$ for $0 \leq i < r$ and $m_i > 0$ for $0 \leq i < r$ then $collapse(a_0^{m_0} a_1^{m_1} ... a_r^\omega) = a_0 a_1 ... a_r$ Two members s_1 and s_2 of Δ^ω are ***identical collapsed*** if $collapse(s_1) = collapse(s_2)$.

Informally, the formula f_T describes a situation involving two players, player 1 and player 2. For all time player 1 knows the value of S_1 and player 2 knows the value of S_2. Neither player ever knows the value of X. As time passes the successive values of S_1 form an infinite string $s_1 \in \Delta^\omega$. Even though player 1 knows S_1 he is unaware of time passing if the value of S_1 does not change form one moment to the next. Intuitively, as time passes player 1 sees *collapse*(s_1), not s_1 itself. Similarly, s_2 is the successive values of S_2 and player 2 only sees *collapse*(s_2). Because neither player knows X it will turn out that if the formula f_T is satisfied in a linear model it will be satisfied in a linear model with two infinite histories, one corresponding to each of the possibilities $X = 0$ and $X = 1$.

Consider the case $X = 0$. As time progresses the successive values of S_1 form an infinite string $e_1 \in \Delta^\omega$. Similarly, successive values of S_2 form the infinite string e_2. The formula will force $e_1 = e_2$, $collapse(e_1) = e_1$, and $collapse(e_2) = e_2$.

If $X = 1$ then infinite strings m_1 and m_2 are formed from successive values of S_1 and S_2 respectively. The formula f_T forces the two strings to match in a way such that, when m_1 and m_2 are interpreted as infinite computations, the first *ID* of m_1 "matches" the second *ID* of m_2, the second *ID* of m_1 "matches" the third *ID* of m_2, and so on. This "matching" guarantees that if m_1 and m_2 are identical collapsed then each of m_1 and m_2 are infinite computations. In the linear model satisfying the formula it will be guaranteed that e_1 and m_1 are identical collapsed and that e_2 and m_2 are identical collapsed. Thus, in the linear model satisfying f_T all four strings e_1, m_1, e_2, and m_2 are identical collapsed and each is an infinite computation. The formula f_T guarantees that embedded in any linear model satisfying it is a "proof" that an

infinite computation exists. The "proof" can be found by examining the two infinite histories arising from the lack of the players knowledge about the variable X. Thus T does not halt on the blank input if and only if f_T is satisfied in a linear model.

We now describe the "matching" that is forced by the formula f_T Consider the two infinite computations:

$$\begin{aligned} s &= \#^5\#^3C_0\#^3C_1\#^3C_2\#^3C_3\#^3\ldots \\ t &= \#C_0\#C_1\#C_2\#C_3\#C_4\#\ldots \end{aligned}$$

Because $|C_0| = 5$ and $|C_{i+1}| = |C_i| + 2$ then the i-th ID of s "lines up" with the $(i+1)$-st ID of t. The initial padding in s matches the initial ID of t the rest of the padding in s aligns the middle # of each $\#^3$ with a corresponding # in t. There is a function $F : \Delta^7 \to \Delta \cup \{\varepsilon\}$ which can be used to verify the matching after the 5-th character of s and the 6-th character of t. Let s_i and t_i be the $(i+1)$-st characters of s and t respectively, so that $s = s_0s_1s_2\ldots$ and $t = t_0t_1t_2\ldots$. The symbol ε, which is needed to make F a totally defined function, is a new symbol with its own representation as a Boolean vector. The function F can be defined in such a way that for all $i \geq 5$, $F(s_i, s_{i+1}, \ldots, s_{i+6}) = t_{i+1}$. We can define F as follows:

$$F(\sigma_0, \sigma_1, \ldots, \sigma_6) = \begin{cases} \# & \text{if } \sigma_0 = \sigma_1 = \sigma_2 = \#, \\ B & \text{if } \sigma_2 = \sigma_3 = \sigma_4 = \#, \\ \not\varepsilon & \text{if } \sigma_0 = \not\varepsilon \text{ or } \sigma_2 = \not\varepsilon, \\ \sigma_2 & \text{if } \sigma_1 = \not\varepsilon \text{ and } (\{\sigma_0, \sigma_2, \sigma_4\} \cap Q = \emptyset \text{ and } \sigma_2 \neq \# \text{ or} \\ & \qquad \sigma_4 \in Q \text{ and } \delta(\sigma_4, \sigma_6) = (p, \tau, R)), \\ \sigma_1 & \text{if } \sigma_1 = \not\varepsilon \text{ and } \sigma_2 \in Q \text{ and } \delta(\sigma_2, \sigma_4) = (p, \tau, L), \\ \tau & \text{if } \sigma_1 = \not\varepsilon \text{ and } (\sigma_0 \in Q \text{ and } \delta(\sigma_0, \sigma_2) = (p, \tau, L) \text{ or} \\ & \qquad \sigma_2 \in Q \text{ and } \delta(\sigma_2, \sigma_4) = (p, \tau, R)), \\ p & \text{if } \sigma_1 = \not\varepsilon \text{ and } (\sigma_0 \in Q \text{ and } \delta(\sigma_0, \sigma_2) = (p, \tau, R) \text{ or} \\ & \qquad \sigma_2 \in Q \text{ and } \delta(\sigma_4, \sigma_6) = (p, \tau, L)), \\ \varepsilon & \text{otherwise.} \end{cases}$$

The following Propostion describes completely the properties that F must possess.

Proposition 1. Let $s = s_0s_1s_2\ldots$ and $t = t_0t_1t_2\ldots$ be infinite strings in Δ^ω with the properties:

1. $s \in \#^8\not\varepsilon q_0\not\varepsilon B\not\varepsilon((\neg\not\varepsilon\not\varepsilon)^*\#^3\not\varepsilon)^\omega$,
2. $t \in \#(\not\varepsilon\neg\not\varepsilon)^\omega$,
3. For all $i \geq 5$, $F(s_i, s_{i+1}, \ldots, s_{i+6}) = t_{i+1}$,
4. $collapse(s) = collapse(t)$.

Then, s and t are infinite computations.

The proof of the Proposition is straightforward but tedious.

We are now ready to define f_T. We will specify eight formulas $f_1 - f_8$ in such a way that

$$f_T = \bigwedge_{m=1,2} K_m[\bigwedge_{n=1}^{8} f_n].$$

To simplify the formulas we make some abbreviations. We already mentioned the abbreviation "$S = T$" when S and T are Boolean vectors. We use the notation "$\Box^k p$" to stand for $\Box\Box\ldots\Box p$ where the number of $\Box$'s is k. So $\Box^0 p = p$. If $\sigma \in \Delta$ then we use "$\neg\sigma$" to abbreviate $\Delta - \{\sigma\}$. After each conjunct f_i we will provide some intuition about what the conjunct means. Intuitively we can imagine time as a sequence of *moments*, so that each conjunct must hold in the initial moment (moment 0). Conjuncts of the form $\Box^* p$ force p to hold for all moments. A subformula of a conjunct of the form $\Box^k p$ forces p to hold in the k-th moment from the time the whole subformula holds.

$$f_1 : \Box^* \bigwedge_{i=1,2} \bigvee_{\sigma\in\Delta} K_i(S_i = \sigma)$$

The formula f_1 states that at all time each player i knows the value of its variable S_i which is one of the symbols in Δ.

$$f_2 : \Box^* \bigwedge_{i=1,2} \bigwedge_{j=0,1} \neg K_i(X = j)$$

The formula f_2 states that for all time neither player knows the value of X.

$$f_3 : \Box^* \bigwedge_{j=0,1} (X = j \supset (\Diamond(X = j) \wedge \Box(X = j)))$$

The formula f_3 states that for all time, if X has a certain value then there is a next moment when it has the same value and in every next moment it has the same value. This formula guarantees that there is always a next moment. That is, time does not stop. This formula also guarantees that if initially X has a certain value then it has that value forever. This formula also guarantees that, as time passes, the successive values of S_i form an infinite string in the language Δ^ω.

$$f_4 : S_2 = ¢ \wedge \Box^*((S_2 = ¢ \supset \Box(S_2 \neq ¢)) \wedge (S_2 \neq ¢ \supset \Box(S_2 = ¢)))$$

The formula f_4 states that initially S_2 has the value ¢ and for all time if the value of S_2 is ¢ then in the next moment the value is not ¢ and *vice versa*. Thus, as time passes the successive values of S_2 form an infinite string in the language $(¢\neg¢)^\omega$.

$$f_5 : X = 0 \supset \Box^*(S_1 = S_2)$$

The formula f_5 states that if at the beginning the value of X is 0 then for all time the value of S_1 equals the value of S_2. Thus, if the value of X is 1 then the infinite string formed by successive values of S_1 is identical to the infinite string formed by successive values of S_2.

$$f_6 : X = 1 \supset S_1 = ¢ \wedge \bigwedge_{1\le k\le 8} \Box^k(S_1 = \#) \wedge \Box^9(S_1 = ¢) \wedge$$
$$\Box^{10}(S_1 = q_0) \wedge \Box^{11}(S_1 = ¢) \wedge \Box^{12}(S_1 = B) \wedge \Box^{13}(S_1 = ¢)$$

$$f_7 : X = 1 \supset \Box^{13}\Box^*((S_1 = ¢ \supset \Box(S_1 \neq ¢)) \wedge ((S_1 \neq ¢ \wedge S_1 \neq \#) \supset \Box(S_1 = ¢)) \wedge$$
$$((S_1 = ¢ \wedge \Box(S_1 = \#)) \supset (\Box^2(S_1 = \#) \wedge \Box^3(S_1 = \#) \wedge \Box^4(S_1 = ¢))))$$

The formulas f_6 and f_7 combined together state, if $X = 1$ then the successive values of S_1 form an infinite string in the language $¢\#^8¢q_0¢B¢((\neg¢¢)^*\#^3¢)^\omega$.

The final formula in f_T is:

$$f_8 : X = 1 \supset \Box^6\Box^* \bigvee_{\sigma_0,\ldots,\sigma_6,\tau\in\Delta} \Big(\bigwedge_{0\le k\le 6} \Box^k(S_1 = \sigma_k) \wedge \Box(S_2 = \tau) \wedge F(\sigma_0,\ldots,\sigma_6) = \tau\Big)$$

The formula f_8 states that if the value of X is 1 then the infinite strings formed by successive values of S_1 and S_2 must match in a nice way. For $i \ge 6$, i-th, $(i+1)$-st, ... , $(i+6)$-th characters of the infinite string formed by successive values of S_1 and the $(i+1)$-st character of the infinite string formed by successive values of S_2 must match according to the function F.

Claim I. If T does not halt on a blank input tape then f_T is satisfiable.

There is not enough space to include the details of the proof of this claim. Given an infinite computation of T on a blank tape a linear model satisfying f_T can be constructed.

Claim II. If f_T is satisfiable then T does not halt on a blank input tape.

Let $M = (W, W_0, \pi, \rho, \preceq_1, \preceq_2)$ be a linear model and let w_0 be a member of W_0 such that $M, w_0 \models f_T$. By Theorem 3.8 for each $i = 1, 2$ and for each $x_0 \in W_0$, $x_0 \approx_i w_0$. Hence, for all $x_0 \in W_0$ and for $1 \le n \le 8$, $M, x_0 \models f_n$. Choose $u_0, v_0 \in W_0$ such that $M, u_0 \models X = 0$ and $M, v_0 \models X = 1$. Such u_0 and v_0 exist because $M, w_0 \models f_2$. Because $M, u_0 \models f_4$ and $M, v_0 \models f_3$ there must exist two infinite sequences $u_0u_1u_2\ldots$ and $v_0v_1v_2\ldots$ in W^ω, each of which is an infinite history. Let $u[k] = u_0u_1\ldots u_k$ and $v[k] = v_0v_1\ldots v_k$. Again, by f_4 we have $M, u[k] \models X = 0$ and $M, v[k] \models X = 1$ for all $k \ge 0$.

Because $M, u[k] \models f_1$ and $M, v[k] \models f_1$, there exist infinite sequences $e_1 = e_{10}e_{11}e_{12}\ldots$, $e_2 = e_{20}e_{21}e_{22}\ldots$, $m_1 = m_{10}m_{11}m_{12}\ldots$, and $m_2 = m_{20}m_{21}m_{22}\ldots \in \Delta^\omega$ such that

$$\begin{array}{ll} M, u[k] \models S_1 = e_{1k} & M, u[k] \models S_2 = e_{2k} \\ M, v[k] \models S_1 = m_{1k} & M, v[k] \models S_2 = m_{2k} \end{array}$$

Leaving out the details the formulas f_1, f_4, f_5 and the fact that M is a linear model guarantee the following:

$$collapse(m_2) = m_2 = e_2 = e_1 = collapse(m_1).$$

The formulas $f_4, f_6 - f_8$ guarantee the following:

- $m_2 \in (¢\neg¢)^\omega$ (f_4),
- $m_1 \in ¢\#^8¢q_0¢B¢((\neg¢¢)^*\#^3¢)^\omega$ (f_6, f_7),
- $F(m_{1i}, m_{1(i+1)}, \ldots, m_{1(i+6)}) = m_{2(1+1)}$ for all $i \ge 6$ (f_8).

If we let $s = m_{11}m_{12}m_{13}\ldots$ and $t = m_{21}m_{22}m_{23}\ldots$ then s and t satisfy Proposition 1. Hence, T does not halt on a blank tape. $\Box$

5 Conclusion

Although we have introduced two logics of protocols, *TLP* and *LLP*, we have only been able to show undecidability for *LLP* which is arguably the less interesting of the two logics. *TLP* can model general games of incomplete information while *LLP* only can model blindfold games. It is our belief that *TLP* is also undecidable, but direct application of the techniques in this paper does not seem sufficient to prove it. Fischer and Immerman have recently developed a logic which seem to model Markov games, games where strategies do not depend on complete histories but only on the current position of the game [4]. The Fischer-Immerman logic is decidable.

It is not yet clear which logics which include knowledge and time will turn out to be the most useful. What is clear already is that the study of logics for distributed computation is enriching our understanding of what the foundations of distributed computation are.

References

[1] Ben-Ari, M., Manna, Z., and Pnueli, A. The Temporal Logic of Branching Time. Eighth ACM Symposium on Principles of Programming Languages, 1981, pp. 164-176.

[2] Emerson, E.A. and Halpern J.Y. Decision Procedures and Expressiveness in the Temporal Logic of Branching Time. Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing, 1982, pp. 169-180.

[3] Emerson, E.A. and Sistla, A.P. Deciding Branching Time Logic. Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing, 1984, 14-24.

[4] Fischer, M.J. and Immerman, N. Foundations of Knowledge in Distributed Systems. Conference on Theoretical Aspects of Reasoning About Knowledge, Asilomar, March 19-22, 1986.

[5] Fischer, M.J. and Ladner, R.E. Propositional Dynamic Logic of Regular Progams. *Journal of Computer and System Sciences, Vol. 18, No. 2,* 1979, pp. 194-211.

[6] Floyd, R.W. Assigning Meaning to Programs. In J.T. Schwartz (ed.) *Mathematical Aspects of Computer Science,* Proc. Symp. in Applied Math. 19. Providence, R.I. American Math. Soc., 1967, pp 19-32.

[7] Fagin, R., Halpern,J.Y., and Vardi, M.Y. A Model-Theoretic Analysis of Knowledge: Preliminary Report. 25th Annual Symposium on Foundations of Computer Science, 1984, pp. 268-287.

[8] Halpern, J.Y. and Fagin, R. A Formal Model of Knowledge, Action, and Communication in Distributed Systems. Proceedings of the Fourth Annual ACM Symposium on Principles of Distributed Computing, 1985, pp 224-236.

[9] Hoare, C.A.R. An Axiomatic Basis for Computer Programming. *Communications of the ACM, Vol. 12,* 1969, pp. 576-580.

[10] Halpern, J.Y. and Moses, Y. Knowledge and Common Knowledge in a Distributed Environment. Proceedings of the Third Annual ACM Conference on Principles of Distributed Computing, 1984.

[11] Ladner, Richard E. The Complexity of Problems in Systems of Communicating Processes. *Journal of Computer and System Sciences, Vol. 21, No. 2,* 1980, pp. 179-194.

[12] Lehman, Daniel. Knowledge, Common Knowledge and Related Puzzles. Proceedings of the Third Annual ACM Conference on Principles of Distributed Computing, 1984, pp. 62-67.

[13] Moses, Y., Dolev D., and Halpern, J.Y. Cheating Husbands and Other Stories: A Case Study of Knowledge, Action, and Communication. Proceedings of the Fourth Annual ACM Symposium on Principles of Distributed Computing, 1985, pp 215-223.

[14] Naur, P. Proof of Algorithms by General Snapshots. *BIT, Vol. 6*, 1966, pp 310-316.

[15] Owen, G., *Game Theory*, Academic Press, 1982.

[16] Parikh, Rohit and Ramanujam, R. Distributed Processes and the Logic of Knowledge. *Logic of Programs, Lecture Notes in Computer Science, No. 193*, Ed. by Rohit Parikh, Springer-Verlag, N.Y., 1985, pp 256-268.

[17] Peterson, Gary L. and Reif, John H. Multiple-Person Alternation. 20th Annual Symposium on Foundations of Computer Science. IEEE Computer Society, 1979, pp. 348-363.

[18] Pnueli, A, The Temporal Logic of Programs. 18th Annual Symposium on Foundations of Computer Science, 1977, pp 46-57.

[19] Pratt, V.R. Semantical Considerations on Floyd-Hoare Logic. 17th Annual Symposium on Foundations of Computer Science, 1976, pp.109-121.

[20] Pratt, V.R. A Near Optimal Method for Reasoning about Action. MIT Technical Report, LCS/TM-138, 1979.

[21] Pease, M., Shostak, R., and Lamport, L. Reaching Agreement in the Presence of Faults. *Journal of the ACM, Vol. 27, No. 2*, 1980. pp 228-234.

[22] Reif, John H. Universal Games of Incomplete Information. Proceedings of the Eleventh Annual ACM Symposium on Theory of Computing. ACM, 1979, pp. 288-308.

[23] Reif, John H. The Complexity of Two-Player Games of Incomplete Information. *Journal of Computer and System Sciences, Vol. 29, No. 2*, 1984, pp. 274-301.

[24] Reif. John H. and Peterson, Gary L. A Dynamic Logic of Multiprocessing with Incomplete Information. Seventh Annual ACM Symposium on Principles of Programming Languages, 1980, pp. 193-202.

[25] Sistla, A.P. and Clarke, E.M. The Complexity of Propositional Linear Temporal Logics. Ninth Annual ACM Symposium on Principles of Programming Languages, 1982, 159-168.