A DYNAMIC LOGIC OF MULTIPROCESSING WITH INCOMPLETE INFORMATION

John H. Reif
Aiken Computation Lab., Harvard University
Cambridge, Mass., 02138

Gary L. Peterson
Computer Science Department, University of Rochester
Rochester, New York 14627

0. Summary

A crucial property of distributed multiprocessing systems is the lack of complete information by any given process about the states of other processes. The contribution of this paper is a fundamental modal logic, MPL, for multiprocessing with incomplete information. (Section 1.5 gives an informal introduction to MPL; the formal definitions are in Section 2.)

By way of this logic, we develop a solid (practical and theoretical) correspondence between distributed multiprocessing and multiplayer games of incomplete information.

Fischer and Ladner, [1979] have shown a log-space reduction from the outcome problem for two person games of perfect information to satisfiability of formulas in their propositional dynamic logic (PDL). We provide in Section 3 a log-space reduction from the outcome problem for multiplayer games of incomplete information to satisfiability of formulas in our logic MPL. Although in general satisfiability in out logic is undecidable, the satisfiability problem of formulae with hierarchical visibility structure is shown decidable (using the methods for solving hierarchical games developed in Reif [1979] and Peterson and Reif [1979], and also the model theoretic techniques of Fischer and Ladner [1979] and Pratt [1979b]).

At a more practical level, we argue that the game-like semantics of our logic provides a robust paradigm in which to view distributed multiprocessing problems. We apply our logic to describe total correctness properties of multi-process programs with shared variables as well as communicating processes with "handshake"-type message passing.

1. Introduction

1.1 Motivation: Incomplete Information in Computing Systems

Incomplete information arises naturally in a variety of computing system applications, for example:

(a) Security. Database Systems frequently contain information classified to be disseminated or modified only by a restricted class of users; other users unauthorized to gain direct access to this classified information may nevertheless infer some or all of its content from various incomplete information that these users are authorized to obtain (also see Dobkin, Jones and Lipton [1979] for other database security issues.)

Similarly, operating systems must be designed to be secure against unauthorized penetration and users which may conspire to degrade the system, using some incomplete information on the system specifications available to them.

(b) Reliability. Frequently, systems contain unreliable subcomponents but must be guaranteed to satisfy certain minimal operating specifications irregardless of situations in which subcomponents "conspire" to a series of breakdowns or otherwise modify their performance. In reliability problems such as these, controllers of the system generally have only incomplete information as to which subcomponents have brokendown or just how those still-operating subcomponents have modified their behavior. (The Bell Laboratories' electronic switching systems and BBN's network controllers are examples of computing systems which are designed to take into account such reliability issues.)

(c) Distributed Multiprocessing with Incomplete Information. Increasingly, multiprocessing systems are designed with processes physically distant from each other. Typically, no single process A has complete information about the current state of the entire system. Direct queries by A to other processes in the system might either be impossible (due to system configuration), useless (because of subsequent changes in state), or impractical (for efficiency or cost considerations). Generally, A can only infer the states of other processes by observing previous actions.

Preprint of paper appearing in 7th Annual ACM Symposium on Principles of Programming Languages, Las Vegas, NV, January 1980, pp. 193-202.

The above applications: Security, Reliability, Distributed Multiprocessing motivate our logic which can express system correctness in situations of incomplete information.

The next subsection defines games, which seem a robust model for computing systems in which incomplete information arises.

1.2 Game Definitions

A game G consists of:

1) players $\{0,1,\ldots,k\}$ partitioned into disjoint teams T_1, T_2

2) a set of positions P_i for each player i.

3) a next move relation $\xrightarrow[i]{} \subseteq P_i \times (\bigcup_j P_j)$

specifying possible next moves for each player i. A position is losing for a given player if he has no next move.

A play is a sequence of moves

$$\Pi_o \xrightarrow[i_1]{} \Pi_1 \xrightarrow[i_2]{} \Pi_2 \ldots \xrightarrow[i_k]{} \Pi_k$$

Games are assumed to be reasonable: the positions, encoded as strings in a fixed alphabet, do not grow in size during plays; and the next move relation is in polynomial time.

To introduce incomplete information to games, each position is partitioned into a fixed number of elements called resources; each player $i \in \{0,1,\ldots,k\}$ is allowed to view and modify only a restricted set $visible_i$ of these resources. A strategy τ of a player i is a mapping from plays of G ending in a position in P_i to a legal next move for i. We require that strategies of player i be independent of these resources in positions not in $visible_i$.

The outcome problem for game G given initial position Π_o is defined:

Is there a strategy for each player of T_1 such that in all resulting plays from Π_o result in a losing position for a player of T_2? (irregardless of the moves by players of T_2). A strategy τ is Markov if τ depends only on the last position of any play to derive the next move. Note that Markov strategies suffice in games of perfect information; where as in games of incomplete information non-Markov strategies are required for players to "infer" portions of positions and visible to them. (This will be an important consideration in our design of a multiprocess logic with incomplete information.)

It is interesting to note that the outcome of G is not effected if we consider team T_2 to be an omniscient (able to view all portions of positions) single player, since the strategy of T_1 must succeed against all counter moves of T_2. In the case of perfect information, T_1 can similarly be considered a single player; thus all games of perfect information are essentially two-person-games. However, in the general case of incomplete information, the team T_1 can not always be contracted to a a single player; the lack of complete information of players of T_1 about each other's position is essential to the game. Players can only be contracted to one if they view the same resources and are on the same team.

In general, the outcome of games of incomplete information with 3 or more players is undecidable [Peterson and Reif, 1979] but the outcome of two player games is decidable ([Reif, 1979] shows this problem complete in doubly exponential time). Also in the case of hierarchical games:

$$visible_o \supseteq visible_1 \supseteq \ldots \supseteq visible_k$$

the outcome is decidable for any number of players $\{0,1,\ldots,k\}$.

1.3 Games that Processors Play

It is our thesis that games of incomplete information are good models for distributed multiprocessing. This motivates the logic proposed in this paper.

Games of perfect information have been previously used as models for computations. Note that nondeterminism is essentially a solitaire (1-person) game of perfect information. The notion of alternation [Chandra, Kozen, Stockmeyer, 1979], introduced to model parallel systems, corresponds essentially to two-person games of perfect information.

Recently, multi-player alternation has been proposed by [Peterson and Reif, 1979] and corresponds essentially to multiple-person games of incomplete information. A restriction, called private alternation corresponds to multiple person hierarchical games.

Devillers [1977] and, more recently, Ladner [1979] have observed that total correctness properties of multiprocess systems can be formulated as games. For example, lockout can be stated:

> From some initial state of the system, can processes 1,2,...,k conspire so that process 0 cannot reach a given state?

Devillers and Ladner only considered these total correctness properties in situations of perfect information and from our previous discussion of games of perfect information, we can always reduce such games to 2-processor problems.

In this paper we consider the more general case when each process has only incomplete information about the state and memory of other processors.

To model correctness properties of a system of distributed processes by a game of incomplete information, we of course equate processes $\equiv$ players of the games. A key point is that we make no lower bound assumptions about the computing power of individual processors. The positions of an induced game consists of only certain fixed portions of the state and memory of the processors; other portions of the memory of processors (most likely, the majority of their memory if the processes are nontrivial) may not be contained in positions. As usual in games of incomplete information, each position is subdivided into portions visible to various processes and legal next moves for processes are determined only from visible portions of positions. However, strategies of processes may be arbitrarily complex, since we have assumed no lower bound on the size of the portion of processes' memory not represented by positions.

(Similarly, the relatively simple bidding rules of a game of poker are dependent only on the fixed sets of chips and cards in the hands of the players or on the table; however, the complexity of bidding strategies are bounded only by the intelligence or imagination of the players. A related issue occurs for the security applications in data base and operating systems discussed in Section 1.1; here the process controllers must determine a winning strategy (guaranteed security) irregardless of the computing power of the processors.

Another possible (but somewhat too restrictive, we believe), model for distributed multiprocessing is Markov games; Here we require that the positions of the induced game include all the state and memory of the processors. Consequently, process strategies are Markov (dependent only in the current position; see Section 1.2). The outcome of games restricted to Markov strategies is decidable ([Peterson and Reif, 1979]). (Within Section 1.5 we provide an alternate Markov semantics for our proposed multiprocess logic).

1.4 Previous Logics and Their Power of Encode Games

We have seen that correctness properties of distributed multiprocess systems can be posed as the outcome of games; it is natural therefore to consider the ability of previous process logics to succinctly encode games. More formally, for the logic L and a class of $\mathscr{G}$ games, we wish to determine if there is a log-space reduction from each game $G \in \mathscr{G}$ and initial position Π_0 to a formula p_{G,Π_0} of L such that p_{G,Π_0} is satisfiable iff the outcome of G from Π_0 is a win. For simplicity we consider only logics whose variables are propositional.

(a) Quantified Boolean Formulas (QBF) have only the power to succinctly encode the outcome of polynomial time bounded games of perfect information [Stockmeyer and Meyer, 1973]. Recently, [Peterson and Reif, 1979] have proposed a class of quantified boolean formulas with dependencies (DQBF), which can succinctly encode games of incomplete information with polynomial time bounds.

(b) Model Logics. Various model logics containing a single model operator were shown by Ladner [1977] to have precisely the power to succinctly encode games of perfect information with polynomial time bounds. Variants of these logics have appeared in:

(i) a temporal process logic [Pnueli, 1977] containing a model operator "eventually", with an extended logic containing also a model operator "nexttime" [Manna and Pnueli, 1979].

(ii) a logic for Data Bases with Incomplete Information [Lipski, 1979]

(iii) a logic for knowledge inference [McCarthy, et al., 1978].

Although these last two logics involve incomplete or hidden information, they seem much to weak for process logics since they can not succinctly encode even two-person games of incomplete information.

Pnueli's model operator "eventually" seems essential to any process logic and a similar model operator in our logic MPL and Pratt's. Pnueli has observed that his temporal logic is inherently endogenous; a single program must be provided to the logic via some fixed semantics; where as the following dynamic logics (and ours) are exogenous; various programs are provided within the formulas of the logic rather than by a fixed semantics.

(c) Dynamic Logics. Pratt's dynamic logic [Pratt 1976] features model operators which contain programs; a thorough investigation of this logic appears in [Harel, 1979]. The propositional dynamic logic (PDL) was shown by [Fischer and Ladner, 1979] to succinctly encode games of perfect information (thus PDL can in theory express correctness properties of multiprocessed systems with perfect information, as well as correctness properties of the nondeterministic (single process) systems for which PDL was intended to express).

The process logic recently proposed by Pratt [1979a] is a propositional dynamic logic with model operators $\_\rfloor$ ("afterwards"), $\_\lfloor\_$ ("during"), $\_\lceil$ ("preserves") which we generalized in our proposed logic MPL. (Pratt's $\lfloor\rfloor$ ("throughout") may be defined in terms of $\_\lceil$.)

Parikh [1978] has also proposed a powerful logic L^- which allows quantification over sequences.

In summary we note that no previous process logic can directly express incomplete information, which we find is a key element of distributed multiprocessing. Furthermore, the previous model logics incorporating incomplete information featured only a very restricted class of model operators and can not succinctly encode even decidable classes of games of incomplete information.

1.5 Innovative Features of our Logic MPL

Although the syntax of our logic MPL is essentially that of the process logic of Pratt [1979a], the semantics of our logic is unique in the inclusion of incomplete information. (We present only the innovative features of our logic here; a formal definition of our process logic is given in Section 2.)

The programs found within modalities in our logic are called actors. An actor A is basic if it consists of an indivisible program fragment.

To include incomplete information, each basic actor A is allowed to view and modify the values of only a restricted set of visible propositional variables. Furthermore, we subdivide each state into state elements and restrict each basic actor to view and modify only certain of these state elements. Thus, in general, each state contains elements public to all basic actors, certain state elements visible only to restricted sets of basic actors, and also state elements private to individual basic actors.

The semantics of a basic actor A are to extend a trajectory τ (a sequence of previous states) to a new trajectory $\tau' = \tau \cdot s$ consisting of τ followed by a new state s. A pair (τ,τ') is an action.† We require that actions of basic actors be independent of those portions of the old trajectory not visible to basic actor A; furthermore, the new state s may differ from the previous state only on those portions of s visible to A.

As a consequence of these semantic restrictions, relative to each basic actor A there is an equivalence relation over trajectories which A cannot distinguish. The induced computation tree contains nodes indistinguishable by various basic actors; as a consequence this semantics of incomplete information is not directly expressable in previous process logics, even in L- of Parikh [1978].

Actors (the programs of our logic) are constructed from basic actors by sequencing $(\alpha;\beta)$, nondeterministic choice $(\alpha|\beta)$, nondeterministic looping (α^*), and parallel execution $(\alpha||\beta)$. Formulas are constructed from propositional variables, logical connectives, and model expressions containing operators $\underline{\sqcup}$, $\underline{\underline{\sqcup}}$, $\underline{\overline{\sqcap}}$.

Given an actor α, let $\mathscr{A}_0(\alpha)$ be the basic actors of α. The $\mathscr{A}_0(\alpha)$ are considered to be players in a game, with the legal next-moves restricted by α considered actions of the $\mathscr{A}_0(\alpha)$, and plays corresponding to trajectories. A partitioning $\mathscr{A}_1$, $\mathscr{A}_2$ of $\mathscr{A}_0(\alpha)$ are considered teams of the game.

Given a formula p

$\alpha \underset{\mathscr{A}_1}{\underline{\sqcup}}$ p denotes "team $\mathscr{A}_1$ have a strategy yielding only trajectories where p holds afterwards" (i.e., p must hold on the last state of all trajectories, irregardless of the actions of team $\mathscr{A}_2$).

$\alpha \underset{\mathscr{A}_1}{\underline{\underline{\sqcup}}}$ p denotes "team $\mathscr{A}_1$ have a strategy yielding only trajectories during which p holds somewhere" (i.e., p holds on some state of all trajectories).

$\alpha \underset{\mathscr{A}_1}{\underline{\overline{\sqcap}}}$ p denotes "team $\mathscr{A}_1$ have a strategy yielding trajectories which preserve p" (i.e., once p holds on a given state of a trajectory, p holds on all later states).

In the case of perfect information and $\mathscr{A}_1 = \phi$, our model operators $\underset{\phi}{\underline{\sqcup}}$, $\underset{\phi}{\underline{\underline{\sqcup}}}$, $\underset{\phi}{\underline{\overline{\sqcap}}}$, are identical to the model operators of Pratt's process logic: $\underline{\sqcup}$, $\underline{\underline{\sqcup}}$, $\underline{\overline{\sqcap}}$ respectively.

1.6 Applications of MPL

Here we sketch some general domains of applications for MPL.

(a) Various Synchronization Primitives:

(i) Boolean Semaphores

MPL can be viewed as a natural logic for multiprocessing with shared variables as synchornization primitives (these shared variables are the propositional variables of MPL formulas viewed commonly by various basic actors).

(ii) Synchronization by Communication

MPL can also easily describe total correctness properties of a multiprocess language with communication primitives send and receive, such as described in [Hoar, 1978].

We assume a set C of communication channels; each channel $c \in C$ allows communication between two fixed processes transmitter(c), receiver(c) $\in \{0,1,\ldots,k\}$. The state of each channel $c \in C$ is either empty or full, and a full channel c contains value V(c) over a fixed, finite domain. Program statement send(c,v) is executed by process transmitter(c) only if c is initially empty; and immediately afterward c is full, with value v. Program statement X←receive(c) is executed by process receiver(c) only if c is initially full; X is set to the current value V(c), and immediately afterward c is empty.

(Note that these semantics require a "handshake" between transmitter and receiver; no queueing of messages is allowed.)

Programs including these communication primitives send and receive can be translated to programs with only shared variables as primitives (thus their total correctness properties can be described in MPL by introducing for each $c \in C$ a set of propositional variables encoding the value of c; these variables are viewed only by the transmitter and receiver of c. The program statements, X←receive(c) and send(c,v) can then be substituted by equivalent program segments containing the introduced variables.)

(b) Local vs. Distributed Control

An MPL actor α containing no instance of the interweaving operator $||$, corresponds to the case of a set of local processes sequentially executing a single program.

The interweaving operator $||$ (introduced also to PDL by [Abrahamson and Fischer, 1978]*) allows

†Note also that the semantics of a basic program of Pratt's process logic (which corresponds to a basic actor of our logic MPL) is a mapping from the current state (rather than from the entire old trajectory as in our logic) to a new trajectory. It is possible to similarly restrict the semantics of our logic. In Markov semantics, we require the actions of basic actors to be dependent only on the final state of the trajectory. The resulting logic is able only to describe Markov games (where players can only remember the current position and cannot infer about the private positions of their opponents from previous plays; see section 1.3). As we have noted, this Markov semantics is applicable to multiprocessing with incomplete information only if we assume the propositional variables of the logic encode all the memory of the processes; this assumption seems reasonable only in the case of very simple processes.

*We view our use of the interweaving operator $||$ as essentially only a syntactic device; any formula p of MP can be substituted by an equivalent formula p' of MP with no instance of $||$, by addition of Boolean variables as described in Abrahamson and Fischer [1978].

for distributed control. For example, in a distributed multiprocess system where each process $i \in \{0,\ldots,k\}$ executes a distinct program α_i (an actor), then

$$\alpha_0 \,||\, \alpha_1 \,||\, \ldots \,||\, \alpha_k$$

denotes their parallel execution.

(c) Multiprocess Games

We noted earlier that multiprocess correctness properties could be formulated as games of incomplete information. MPL formulas have the power to directly and succinctly express such games. This is proved in section 3; we give some examples below.

Consider a distributed multiprocess system with processor $0,1,\ldots,k$ and corresponding programs (actors) $\alpha_0,\alpha_1,\ldots,\alpha_k$ as above. We assume for each $i \in \{0,1,\ldots,k\}$ all the basic actors of α_i view the same set of propositional variables; (thus by the discussion in Section 1.2 the basic actors of α_i can be coalessed to a singular player i in our multiprocess game).

We consider process 0 to be distinguished, say a controller or scheduler. Our multiprocess games will have teams $T_1 = \{1,\ldots,k\}$ and $T_2 = \{0\}$. Let $\mathcal{A}_1$ be the basic actors of $\alpha_1,\ldots,\alpha_k$ and $\mathcal{A}_2$ be the basic actors of α_0.

Also let p_0 be a formula which holds precisely at the initial state of the multiprocess system, and let q be a formula.

(a) Total correctness: The formula

$$p_0 \wedge (\alpha_0||\alpha_1||\ldots||\alpha_k) \underset{\mathcal{A}_2}{\sqsupseteq} q$$

expresses that process 0 can guarantee that the correctness property q holds on termination.

(b) Accessibility: The formula

$$p_0 \wedge (\alpha_0||\alpha_1 \ldots ||\alpha_k) \underset{\mathcal{A}_1}{\perp} q$$

express that $1,\ldots k$ can conspire to guarantee that q is accessible (eventually holds). (For example, if q holds unless process 0 is in a distinguished state, the above formula expresses guaranteed lockout of process 0 from this state,)

(c) Avoidance: The formula

$$\neg q \wedge p_0 \wedge (\alpha_0||\alpha_1\ldots||\alpha_k) \underset{\mathcal{A}_2}{\sqsupset} (p_0 \rightarrow \text{true}) \vee \neg q$$

expresses that process 0 can guarantee that formula q never holds. (For example, the above formula can express lockout avoidance by process 0.)

2. Formal Definition of MPL

The distinguishing features of our logic were informally presented in the Introduction; we aim here for a rigorous definition.

2.1 The Syntax. We recursively define a set $\Phi = \{p,q,\ldots\}$ of *formulas* and a set $\mathcal{A} = \{\alpha,\beta,\ldots\}$ of *actors*. (These "actors" are similar to the "programs" of previous process logics of [Pratt, 1979a] and [Abrahamson and Fischer, 1979]; although as noted in Section 1.6 our "actors" have a somewhat more powerful semantics.)

For our basis we assume a set $\Phi_0 \subseteq \Phi$ of *basic formulas* which are simply prepositional variables $P,Q,\ldots$. Also, we assume a set of *basic actors* $\mathcal{A}_0 \subseteq \mathcal{A}$ which are symbols $A,B,\ldots$. There is a distinguished *null actor* $\theta \in \mathcal{A}_0$.

Let α,β be actors, let p,q be formulas and let $\mathcal{A}_1$ be a set of basic actors of α. Inductively, we may

(i) apply *logical connectives* $\neg$("negation") and $\wedge$("conjunction") to construct formulas $\neg p$ and $p \wedge q$.

(ii) apply *program constructs*; (";" ("sequencing") | ("nondeterministic choice"), * ("nondeterministic looping"), || ("parallel execution"), ? ("test") to construct the actors $(\alpha;\beta)$, $(\alpha|\beta)$, (α^*), $(\alpha||\beta)$, $p?$ (p must be propositional.)

(iii) apply modal operators $\underset{\mathcal{A}_1}{\sqsupseteq}$ ("after"), $\underset{\mathcal{A}_1}{\perp}$ ("during"), $\underset{\mathcal{A}_1}{\sqsupset}$ ("preserves") to construct the formulas

$$\alpha \underset{\mathcal{A}_1}{\sqsupseteq} p, \quad \alpha \underset{\mathcal{A}_1}{\perp} p, \quad \alpha \underset{\mathcal{A}_1}{\sqsupset} p .$$

(Informally, we augment our syntax with booleon connectives $\vee,\rightarrow,\leftrightarrow$ defined as usual from $\neg$ and $\wedge$).

We associate with each basic actor $A \in \mathcal{A}_0$ a fixed set $\text{visible}_A \subseteq \Phi_0$ defining those basic formulas whose value A may view. (We assume that for each $\Phi' \subseteq \Phi_0$, there is an infinity of $A \in \mathcal{A}_0$ such that $\text{visible}_A = \Phi'$.)

2.2 A Kripki-Like Semantics. A *model* $\mathcal{M}$ for MPL consists of

(i) a set $S^{\mathcal{M}}$ of *states*,

(ii) a relation $\models^{\mathcal{M}} \subseteq S^{\mathcal{M}} \times \Phi_0$ providing *basic facts*; $s \models^{\mathcal{M}} P$ denotes "propositional variable $P \in \Phi_0$ holds in state $s \in S^{\mathcal{M}}$".

(iii) For each basic actor $A \in \mathcal{A}_0$, a relation $\overset{\mathcal{M}}{\underset{A}{\rightarrow}} \subseteq S^{\mathcal{M}} \times S^{\mathcal{M}}$ providing *basic actions*; $s \overset{\mathcal{M}}{\underset{A}{\rightarrow}} s'$ denotes that "basic actor A may take state s into state s'."

(We will drop the superscript $\mathcal{M}$ for a fixed model $\mathcal{M}$.)

Let $\text{visible}_A(s)$ be those basic facts $s \models P$ such that $P \in \text{visible}_A$ (intuitively, $\text{visible}_A(s)$ are "those basic facts visible to basic actor A in state s"). Likewise, let $\text{invisible}_A(s)$ be those basic facts $s \models P$ such that $P \notin \text{visible}_A$.

We require for any model $\mathcal{M}$ that

if $s \xrightarrow{}_A s'$ is a basic action

then $\text{invisible}_A(s) = \text{invisible}_A(s')$ (thus basic actions of A only modify those portions of states visible to A).

Let a *trajectory* τ be a finite sequence of states in S (in Section 2.3 we extend our semantics to allow for infinite trajectories). Let $\Gamma(S)$ be the set of all such trajectories. Give a trajectory $\tau \in \Gamma(S)$, let $\text{visible}_A(\tau)$ be the sequence derived from τ by

(a) substituting $\text{visible}_A(s)$ in place of each s in τ.

(b) substituting a single element x for maximal substrings of identical elements x,x,...,x.

Intuitively by (a), $\text{visible}_A(\tau)$ is that portion of the basic facts associated with trajectory which basic vector A may view, and by (b) we require that A be oblivious to modifications to basic facts not visible to A.

Let a *strategy* of basic actor $A \in \mathcal{A}_0$ be a partial mapping $\sigma: \Gamma(S) \to \Gamma(S)$ such that

(a) for each trajectory τ in the domain of σ, such that τ ends in some state $s \in S$,

$$s \xrightarrow{}_A \sigma(\tau).$$

(b) If τ' is also in the domain of σ and

$$\text{visible}_A(\tau') = \text{visible}_A(\tau), \text{ then}$$

$$\text{visible}_A(\sigma(\tau')) = \text{visible}_A(\sigma(\tau)).$$

We consider a set of basic actors $\mathcal{A}_1 \subseteq \mathcal{A}_0$ a *team*. A *team strategy* σ of $\mathcal{A}_1$ is a relation consisting of a union of strategies for each $A \in \mathcal{A}_1$.

A team strategy σ *restricts the basic actions* available to the members of $\mathcal{A}_1$, as follows.

For each basic actor $A \in \mathcal{A}_1$, let $s \xrightarrow[A,\sigma]{} s'$ iff

$$(s \xrightarrow{}_A s' \text{ and } A \notin \mathcal{A}_1) \text{ or } ((s,s') \in \sigma \text{ and } A \in \mathcal{A}_1)$$

(Thus the actions of basic actors not in $\mathcal{A}_1$ are not restricted.)

Given this fixed team strategy σ for $\mathcal{A}_1$, the actions of actors $\alpha \in \mathcal{A}$ are defined by the relation $\xrightarrow[\alpha,\sigma]{} \subseteq \Gamma(S) \times \Gamma(S)$.

(i) If α is a basic actor A, then $\tau \xrightarrow[A,\sigma]{} \tau'$ iff τ ends in state s, $\tau' = \tau \cdot s'$ and $s' = \sigma(\tau)$

(ii) If $\alpha = \beta_1;\beta_2$ then $\xrightarrow[\alpha,\sigma]{} = \xrightarrow[\beta_1,\sigma]{}$ composed with $\xrightarrow[\beta_2,\sigma]{}$.

(iii) If $\alpha = (\beta_1 | \beta_2)$ then

$$\xrightarrow[\alpha,\sigma]{} = \xrightarrow[\beta_1,\sigma]{} \cup \xrightarrow[\beta_2,\sigma]{}$$

(iv) If $\alpha = (\beta^*)$ then $\xrightarrow[\alpha,\sigma]{}$ is the reflexive, transition closure of $\xrightarrow[\beta,\sigma]{}$.

(v) If $\alpha = P?$ then $\tau \xrightarrow[P?,\sigma]{} \tau$ iff $\tau \models P$ is a basic action.

(The semantics of $\alpha = (\beta_1 || \beta_2)$ can be defined from the above by adding Boolean variables to the expressions, see [Abrahamson and Fischer, 1979].) Note that if $\tau \xrightarrow[\alpha,\sigma]{} \tau'$ then t is a prefix of τ'.

We wish now to extend $\models$ to a relation $\subseteq \Gamma(S) \times \Phi$ giving *facts* about all formulas.

For each propositional variable $P \in \Phi_0$, let

$\tau \models P$ iff $s \models P$, where s is the last state of τ.
$\tau \models \neg p$ iff not $(\tau \models p)$.
$\tau \models p \wedge q$ iff $(\tau \models p)$ and $(\tau \models q)$.

Finally, we define the semantics of the modal operators $\overset{\exists}{\underset{\mathcal{A}_1}{\exists}}$, $\overset{\exists}{\underset{\mathcal{A}_1}{\forall}}$, $\overset{\forall}{\underset{\mathcal{A}_1}{\exists}}$.

Let

$$\tau \models \left(\alpha \overset{\exists}{\underset{\mathcal{A}_1}{\exists}} p\right) \text{ iff}$$

$\exists$ team strategy σ for $\mathcal{A}_1$ with $\tau' \models p$

for all $\tau' \in \Gamma(S)$ such that $\tau \xrightarrow[\alpha,\sigma]{} \tau'$.

Let

$$\tau \models \left(\alpha \overset{\forall}{\underset{\mathcal{A}_1}{\exists}} p\right) \text{ iff}$$

$\exists$ team strategy σ for $\mathcal{A}_1$ such that $\forall \tau' \in \Gamma(S)$ with $\tau \xrightarrow[\alpha,\sigma]{} \tau'$, we have $\tau'' \models p$ for some $\tau'' \leq \tau'$ ($\leq$ denotes the (nonstrict) prefix relation).

Let

$$\tau \models \left(\alpha \overset{\forall}{\underset{\mathcal{A}_1}{\forall}} p\right) \text{ iff}$$

$\exists$ team strategy σ of $\mathcal{A}_1$ such that

$$\forall \tau' \in \Gamma(S) \text{ with } \tau \xrightarrow[\alpha,\sigma]{} \tau',$$

if $\tau''' \models p$ then $\tau'' \models p$ for $\tau''' \leq \tau'' \leq \tau'$.

Let a formula $p \in \Phi$ be *satisfiable* if there exists a model $\mathcal{M}$ with state $s \in S^{\mathcal{M}}$ such that $(s) \models^{\mathcal{M}} p$.

3. Encoding Games of Incomplete Information into MPL Formulae.

The complexity of satisfiability of classes of MPL formulas is related here to the complexity of the outcome of various classes of games of incomplete information considered in [Reif, 1979] and [Peterson and Reif, 1979].

3.1 The Visibility Structure of Games and Formulas of MPL

Let G be a game (see the game definitions of Section 1.2) with players $\{0,1,\ldots,k\}$ and each player $i \in \{0,1,\ldots,k\}$ allowed to view just the resources of the set $visible_i$. We assume each resource takes a value from a finite domain of constant size.

Let the visibility structure $\mathcal{V}$ of G be the directed bipartite graph with vertices consisting of

(i) minimal sets of players with the same visible resources

(ii) minimal sets of resources of G which are viewed by the same players.

For each vertex x of type (i), let the edges departing x enter precisely the resources viewed by players of X and let there be no other edges. Let p be a formula of MPL. The visibility structure of p is defined as in a game with the basic actors of p corresponding to players and the propositional variables of p corresponding to the resources of a game. We also introduce an additional omniscient player which can view all resources.

In Section 1.2 we noted various restrictions on the visibility structure which results in games with decidable outcomes. It is interesting to relate the outcome of games to the satisfiability of formuals of MPL with similar visibility structures.

Theorem 1: For each game of G of incomplete information, and initial position Π_o, we can construct in log-space a formula of p_{G,Π_o} of MPL with a visibility structure isomorphic to that of G and such that G has positive outcome iff p_{G,Π_o} is satisfiable.

To prove this theorem, we need only consider computation games, in which the positions are configurations of the multi-player alternation Turing machines of [Peterson and Reif, 1979]. We extend (to incomplete information) a proof technique of [Fischer and Ladner, 1978] which provided a log-space reduction from the acceptance problem of linear-space bounded alternating Turing machines (or equivalently, games of perfect information) to the satisfiability of PDL formulas.

3.2 Computation Games

A multiple-person alternating machine (see also [Peterson and Reif, 1979]) is essentially a Turing Machine consisting of:

(i) a state set Q with initial state q_I and accepting state $q_A \in Q$

(ii) input tape named o and work tapes $1,..,t_o$ with type alphabet

(iii) next move relation $\delta \subseteq (\Delta^{t_o+1} \times Q) \times (\Delta^{t_o+1} \times Q \times \{left,right\}^{t_o+1})$ defined as usual for multitape Turing Machines.

Fix on input string $\omega \in \Delta^n$. We assume M is linear space bounded.

Furthermore, M is considered a game G_M with

(iv) Q partitioned into sets $Q_o, Q_1, \ldots, Q_k$ correspondingly to players $0,1,\ldots,k$

(v) the teams are $T_1 = \{1,\ldots,k\}$ and $T_2 = \{0\}$

(vi) the positions are configurations of M, with initial position the configuration with the input tape containing ω, all work tapes empty, and all heads on the leftmost cell of each tape. A position is losing for T_2 if it contains the accepting state q.

(vii) the legal next moves of players are defined by δ,

(viii) each state $q \in Q$ is considered a boolean sequence of final length d and resources are negative integers $-1,\ldots,-d$ (indicating position of state) plus position integers $0,\ldots t_o$ indicating the types.

M accepts ω iff the outcome for G_M is a win for T_1.

A game G_u is universal for a class of games $\mathcal{C}$ if (i) $G_u \in \mathcal{C}$ and (ii) there is a log-space reduction from each $G \in \mathcal{C}$ and initial position Π_o of G, to initial position Π_{G,Π_o} of G_u and such that the outcome of G from Π_o is identical to the outcome of G_u from Π_{G,Π_o}.

If $\mathcal{V}$ is a visibility structure and $\mathcal{C}_{\mathcal{V}}$ is the class of games with visibility structure $\mathcal{V}$ then we can show (using the results of [Peterson and Reif, 1979]) there is a computation game universal to $\mathcal{C}_{\mathcal{V}}$. Thus, we need only consider computation games in the proof of Theorem 1.

3.3 Encoding Computation Games as Formulas of MPL.

Given a computation game G_M as in Section 3.2, we now provide a log-space construction of a formula p_M of MPL with visibility structure isomorphic to that of G_M and such that p_M is satisfiable iff M accepts ω.

We require propositional variables

P_u $1 \leq u \leq d$ (informally, "P_u is the boolean variable giving the uth portion of a state")

$C_{t,u,a}$ $0 \leq t \leq t_o$, $1 \leq u \leq d$, $a \in \Delta$ (informally, the head of the tape t is scanning the uth cell which has contents a").

We shall require basic actors $\vdash_0, \vdash_1, \ldots, \vdash_k$ and we let visible $\vdash_i$ to be those propositional variables P_u with $u \in visible_i$, plus those $C_{t,u,a}$ such that $t \in visible_i$.

It is also useful to define

$S_q \equiv \bigwedge_{1 \le u \le d} P_u = X_u$ where $q = (X_1,\ldots,X_d)$ is a state in Q (informally, "the current state is q")

$H_{t,u} \equiv \bigvee_{a \in \Delta} C_{t,u,a}$ (informally, "the head of the tape u is scanning cell u").

$TURN_i \equiv \bigvee_{q \in Q_i} S_q$ (informally, "it is player i s turn to move").

We let $p_M \equiv p_o \wedge \beta^* \underline{\sqcup}_1 p_1 \wedge p_2 \wedge p_3$ where

$\beta \equiv (TURN_o? \vdash_o) \mid \ldots \mid (TURN_k? \vdash_k)$

We can define formulae $p_o,\ldots,p_3$ for which informally:

p_o = "M is in its initial configuration"

p_1 = "there is exactly one state and one symbol per cell"

p_2 = "the cells not currently scanned by a type head are not modified by a move of any player"

p_3 = "the players move correctly"

The formulae p_o, p_1, and p_2 are easy to construct; we given here only p_3 (which is the only nontrivial formula).

$$p_3 \equiv \bigwedge_{0 \le i \le k} \bigwedge_{q \in Q_i} \bigwedge_{\vec{a} \in \Delta^{t_o+1}} \bigwedge_{\vec{u} \in \{0,\ldots,cn\}^{t_o+1}} \Big(\bigwedge_{0 \le t \le t_o} H_{t,u_t} \wedge C_{t,u_t,a_t} \Big) \rightarrow \Big(i \in T_1 \rightarrow \bigvee_{q',\vec{a}'} \vdash_i \{\underline{\sqcup}\}_i r \Big)$$

$$(q,\vec{a},q',\vec{a}',\vec{D}) \in \delta$$

$$\Big(i \notin T_1 \rightarrow \bigwedge_{q',a'} \vdash_i \underline{\sqcup}_{\not\in} r \Big)$$

$$(q,\vec{a},q',\vec{a}',\vec{D}) \in \delta$$

where $r \equiv S_q \wedge$

$$\bigwedge_{0 \le t \le t_o} (D_t = \underline{\text{left}} \rightarrow H_{t,u_t-1} \wedge C_{t,u_t-1,a'_t}$$

$$\vee (D_t = \underline{\text{right}} \rightarrow H_{t,u_t+1} \wedge C_{t,u_t+1,a'_t})$$

It can be easily verified that p_M is satisfiable iff M accepts ω. Thus we have Theorem 1. □

4. Testing Satisfiability for Formulas with a Hierarchical Visibility Structure

Let G be a game of incomplete information with players $0,1,\ldots,k$. If

$$visible_0 \supseteq visible_1 \supseteq \ldots visible_k$$

then G has a hierarchical visibility structure, and the outcome is decidable. We can show:

Theorem 2. Satisfiability is decidable for the class of MPL formulae p_o with a hierarchical visibility structure:

$$visible_{A_0} \supseteq visible_{A_1} \supseteq \ldots visible_{A_k}$$

for basic actors $A_0,A_1,\ldots,A_k$ of p (This corresponds to the natural case where a single process, A_0, "spawns" a linear chain of processes $A_1,\ldots,A_k$ and A_i knows everything about A_j for $i \le j$.)

For simplicity, we assume

(a) p_o contains no instance of the $||$ operator and no model operator but $\underline{\sqcup}$

and (b) the basic actors in p_o are partitioned into disjoint sets $\mathcal{A}^0$, $\mathcal{A}^1$ where

(i) $\mathcal{A}^0$ is a set of basic actors which can view all resources.

(ii) the elements of $\mathcal{A}^1$ have the same visible resources.

Note that this visibility structure is hierarchical and corresponds to a two person game. (These assumptions (a) and (b) are dropped in our full algorithm, but in the above simple situation our algorithm is considerably easier to understand.) We provide a brief sketch of our algorithm.

We shall require the closure $cl(p_o)$ of formula p_o (see [Fischer and Ladner, 1978]) which is the minimal set of formulas such that

(i) $p \in cl(p_o)$

(ii) all formula subexpressions of elements of $cl(p)$ are in $cl(p_o)$

(iii) For $\alpha \underline{\sqcup}_{\mathcal{A}_1} q \in cl(p_o)$

if $\alpha = \beta_1;\beta_2$ then $\beta_1 \underline{\sqcup}_{\mathcal{A}_1} Q^{(\beta_2 \underline{\sqcup}_{\mathcal{A}_1} q)} \in cl(p_o)$

if $\alpha = (\beta_1 | \beta_2)$ then $(\beta_2 | \beta_1)$ and

$$\beta_1 \underline{\sqcup}_{\mathcal{A}_1} Q^{\beta_2 \underline{\sqcup}_{\mathcal{A}_1} q} \in cl(p_o)$$

if $\alpha = (\beta^*)$ then $\beta \underline{\sqcup}_{\mathcal{A}_1} Q^{\alpha \underline{\sqcup}_{\mathcal{A}_1} q} \in cl(p_o)$

The Q^p are new propositional variables which can be viewed by just those basic actors that can view all propositional variables in p.

Let $X = \{x \subseteq cl(p_o) \mid p \in x \Rightarrow \neg p \notin x$

$p \wedge q \in x \Rightarrow p,q \in x$

$p \vee q \in x \Rightarrow p \in x$ or $q \in x$

$(P?) \underset{\mathcal{A}_1}{\sqsupseteq} q \in x$ and $P \in x$

$\Rightarrow q \in x$

$(\alpha|\beta) \underset{\mathcal{A}_1}{\sqsupseteq} p \in x \Rightarrow \alpha \underset{\mathcal{A}_1}{\sqsupseteq} p \in x$

or $\beta \underset{\mathcal{A}_1}{\sqsupseteq} p \in x$

$(\alpha;\beta) \underset{\mathcal{A}_1}{\sqsupseteq} p \in x \Rightarrow (\alpha \underset{\mathcal{A}_1}{\sqsupseteq})(\beta \underset{\mathcal{A}_1}{\sqsupseteq} p) \in x$

$\alpha^* \underset{\mathcal{A}_1}{\sqsupseteq} p \in x \Rightarrow p,\ \alpha \underset{\mathcal{A}_1}{\sqsupseteq} p,$

$(\alpha \underset{\mathcal{A}_1}{\sqsupseteq})(\alpha^* \underset{\mathcal{A}_1}{\sqsupseteq} p) \in x \}.$

We define an initial model $\mathcal{M}_o$ with

(a) states $S \subseteq X^{|X|}$

(i.e., sequences of length $|X|$ containing elements of X)

(b) let $S \xrightarrow[A]{} S'$ be a basic action of $\mathcal{M}$ iff

($A \underset{\mathcal{A}_1}{\sqsupseteq} p \in S(i)$ for some $\mathcal{A}_1$ not containing A,
implies $p \in s'(i)$ for $1 \leq i \leq |X|$)

(c) let $S = P$ be a basic fact iff

$P \in s(i)$ for all $1 \leq i \leq |X|$.

(Intuitively, each $s \in S$ contains elements $s(i) \in X$ which could be formulas that hold at s. In general the basic actors of $\mathcal{A}'$ can not distinguish which $p \in s(i)$ holds of s, unless $p \in s(j)$ for all $1 \leq j \leq |X|$).

We can show that if p_o is satisfiable, then there is a substructure of $\mathcal{M}_o$ that satisfies p_o. We now proceed as in Pratt's [1979a] algorithm for testing satisfiability of PDL formulas, repeatedly deleting "inconsistent" states of S until convergence. (This phase of the algorithm is also similar to the game tree labeling algorithms of [Chandra, Kozen, Stockmeyer, 1978], [Reif, 1979] for determining a winning strategy of a two-person game. In the resulting model $\mathcal{M}$, there is some $s \in S$ such that $p_o \in s(i)$ for $i \in \{1,\ldots|X|\}$ iff p_o is satisfiable.

In the case p_o is a general formula of MPL with a hierarhical visibility structure, we take the states

$$S = X^{|X|^{\cdot^{\cdot^{\cdot^{|X|}}}}} \Big\} \text{ k' times}$$

(when k' is number of basic actors A with distinct $visible_A$) and the rest of the construction may be suitably generalized.

6. Conclusion

6.1 Summary

We have (a) introduced a logic MPL for multiprocessing with incomplete information, (b) shown that games of incomplete information may be succinctly encoded into formulas of MPL with the same visibility structure, (c) developed an algorithm for testing satisfiability of formulas with a hierarchical visibility structure.

6.2 Extensions to MPL

We have attempted to define MPL with as few as possible program constructs, logical operations and model operators, without sacrificing the expressibility and power of the logic. Various extensions may be made, for example:

(a) Syntatic Modifications. We could introduce the while-loop and if-then-else program constructs as basic; as has been done previously in PDL. Another possible extension is to introduce message communication as the fundamental process synchronization primitive; this is again actually only a syntactic modification as noted in Section 1.6.

(b) Infinite Trajectories. We can easily allow for trajectories which are initial sequences of states. Here we could (as in [Pratt, 1979a]) introduce a special abort state Ω from which no action may be taken. Failing trajectories are those which contain Ω ; we require that a trajectory can only contain Ω as the last state. (This restriction requires some tedious redefinition of the program constructs such as

(c) A First Order Logic. MPL may be extended to allow quanitification fo variables. We are considering the dependency quantification of variables defined in [Peterson and Reif, 1979] since these restricted quantifications allow expression of strategies which are independent of given private variables.

6.3 Open Problems.

(a) Harel [1979b] has shown that the model operations $\sqsupseteq$, $\perp$, $\sqsupseteq^{\sqsupseteq}$ of Pratt's process logic are independent and there is a further model operator independent of these; are the model operators $\underline{\sqsupseteq}$, $\underline{\perp}$, $\underline{\sqsupseteq^{\sqsupseteq}}$ similarly independent?

(b) Are there natural restrictions to visibility structures, besides hierarchical, for which games and formulas of MPL are decidable?

(c) Most importantly, provide a consistent and complete axiomization for a decidable class of MPL formulas.

6.4 Acknowledgments

We wish to thank Albert Meyer and David Harel for their careful reading and enlightening comments on a preliminary draft of this paper.

Bibliography

Abrahamson, K. and M.J. Fischer, "Applications of Boolean Variables to Automata Theory and Dynamic Logic," Computer Science Department, University of Washington, Seattle, May 1978.

Chandra, A.K., D.C. Kozen, and L.J. Stockmeyer, "Alternation," IBM Research Report RC 7489, Yorktown Heights, N.Y., January 1978.

Devillers, R., "Game Interpretation of the Deadlock Avoidance Problem," Communications of the ACM 20:10 (1977), pp. 741-745.

Dobkin, D., A.K. Jones, and R.J. Lipton, "Secure Databases: Protection against User Interference," ACM Trans. on Database Systems 4:1 (1979), pp. 97-106.

Fischer, M.J. and R.E. Ladner, "Propositional Dynamic Logic of Regular Programs," Journal of Computer and System Sciences 18 (1979), pp. 194-211.

Harel, D., "First-Order Dynamic Logic," Lecture Notes in Computer Science 68, Springer-Verlag, New York, 1979.

Harel, D., "Two Results in Process Logic," Information Processing Letters 8:4 (1979), pp. 195-198.

Harel, D. and V.R. Pratt, "Nondeterminism in Logics of Programs," Proceedings of the 5th ACM Symposium on Principles of Programming Languages, Tucson, Arizona, January 1978.

Hoare, C.A.R., "Communicating Sequential Processes," Communications of the ACM 21:8 (1978), pp. 666-677.

Ladner, R.E., "The Computational Complexity of Provability in Systems of Modal Propositional Logic," SIAM Journal of Computing 6:3 (1977), pp. 467-480.

Ladner, R.E., "The Complexity of Problems in Systems of Communicating Sequential Processes," 11th ACM Symposium on Theory of Computing, April 1979

Lipski, W., Jr., "On Semantic Issues Connected with Incomplete Information Data Bases," ACM Trans. on Database Systems, September 1979.

Manna, Z., and A. Pnueli, "The Model Logic of Programs," Technical Report, Computer Science Dept., Stanford University, 1979.

McCarthy, J., M. Sato, T. Hayashi, and S. Igarashi, "On the Model Theory of Knowledge," AI Memo 312, Computer Science Dept., Stanford University, April 1978.

Parikh, R., "A Decidability Result for a Second Order Process Logic," Proceedings of the 19th IEEE Symposium on Foundations of Computer Science, Ann Arbor, Michigan, October 1978.

Peterson, G.L., and J.H. Reif, "Multiple-Person Alternation," 20th Symposium on Foundations of Computer Science, Puerto Rico, 1979.

Pnueli, A., "The Temporal Logic of Programs, 18th IEEE Symposium on Foundations of Computer Science, October 1977.

Pratt, V.R., "Semantical Considerations on Floyd-Hoare Logic," Proceedings 17th IEEE Symposium on Foundations of Computer Science, 1976.

Pratt, V.R., "Process Logic," Proceedings 6th ACM Symposium on Principles of Programming Languages, San Antonio, Texas, January 1979.

Pratt, V.R., "Models of Program Logics," 12th FOCS, San Juan, Puerto Rico, 1979.

Reif, J.H., "Universal Games of Incomplete Information," 11th ACM Symposium on Theory of Computing, Atlanta, Georgia, 1979.

Stockmeyer, L.J., and A.R. Meyer, "Word Problems Requiring Exponential Time," Proceedings 5th ACM Symposium on Theory of Computing (1973), pp. 1-9.

Stockmeyer, L.J., and A.K. Chandra, "Provably Difficult Combinatorial Games," SIAM Journal of Computing 8:2 (1979), pp. 151-173.