Concurrency with partial information

S. Borgo
Computer Science, Indiana University, Bloomington, IN 47405 (USA)
LACL, Université de Paris XII, 94010 Créteil Cedex (FR)
stborgo@indiana.edu

Abstract

The aim of this paper is to introduce a new formalism for multi-agent systems that captures (true) concurrency and independence. We begin describing a complete and decidable propositional multi-modal logic. The language, similar to Propositional Dynamic Logic in many aspects, differs from usual modal logic: modal operators are non-atomic (they are built out of basic actions) and capture concurrency at the syntactic level. Quantified logics are obtained by allowing quantifiers to occur in the modal operators. The semantics of quantified formulas uses both set-theoretic and game-theoretic notions (here introduced only informally.)

1 Introduction

The problems of multi-agent systems [1, 2] furnish interesting applications of information independence and concurrency issues. The relationship between agents, their actions, and the effective consequences of their knowledge is at the very center of many research areas across computer science, philosophy, logic, linguistics, social science, and economics.

In this paper we introduce a new formalism for multi-agent systems with strong emphasis on information issues. True concurrency, conflict, and causality have a central role in our work.

By *agent* we mean a rational and autonomous entity that has the power to execute actions. Our notion of rational agent is very broad, for instance, it includes computational as well as biological entities. By *rationality* we mean the ability to choose actions to achieve a given goal under the constraints imposed by the system [2]. Through actions, agents modify the state of the system, possibly causing some sentences to change their truth-value, and try to reach their goal. By *state* of the system we refer to the usual notion of global state. In our multi-agent logics, the system state is not partitioned with respect to agents, i.e., there are no local states for individual agents. In fact, the logics describe what agents, acting autonomously or in groups, can do to force or forbid the system to reach some given state.

The language contains a set of atomic actions which are combined to form modal operators. Generally speaking, the actions available to an agent depend on the state of the system at that point and on the concurrent actions performed by other agents. Note that time is not considered explicitly in the logics and there is no reasoning about history. The logics consider only the present and the future states.

Our formalism is part of the traditional modal approach to multi-agent systems as opposed to representational approaches. In particular, there is no reference to the internal structure of agents. Specific logics describing agent's (complex) attitudes can be found in [1, 3] and a broader analysis of (complex social) agents in [4].

In next section we informally present our multi-agent logics through an example. We give a detailed introduction of the propositional logic in section 3 and, in the following section, state completeness and decidability. After relating our system to Propositional Dynamic Logic in section 5, we give a (very) short description of one quantified extension of the logic. In section 7 we discuss our research and mention some related work.

2 A Guiding Example

Consider two non communicating agents α_1, α_2 and two boolean variables X_1, X_2.
Agent α_1 can change the value of X_1 and agent α_2 can change the value of X_2.
The set of available (constants for) actions is $\{1, 0, \epsilon\}$.
After agent α_i executes the action denoted by 1, X_i has value 1.
After agent α_i executes the action denoted by 0, X_i has value 0.
ϵ denotes the "null" action; whenever α_i executes the "null" action, the value of X_i does not change.

The system has four distinguished states s_{ij} $(0 \leq i, j \leq 1)$, where state $s_{i,j}$ corresponds to $(X_1 = i,\ X_2 = j)$. In the language, $s_{i,j}$ is described by proposition $\varphi_{i,j}$.

Consider the following 2-step run of the system:

- Initial state: $(X_1 = 0$ and $X_2 = 0)$

(I) agent α_1 performs action 1 and agent α_2 action ϵ

- $(X_1 = 1$ and $X_2 = 0)$

(II) agent α_1 perform action 1 and agent α_2 action 1

- $(X_1 = 1$ and $X_2 = 1)$

Step (I) corresponds to formula $\left[\begin{array}{c} 1 \\ \epsilon \end{array}\right] \varphi_{10}$, where the first row shows the action of agent α_1 and the second row the action of agent α_2, φ_{10} characterizes the state after this step. The modal operator in this formula has the shape of a 2×1 matrix built out of basic actions and it corresponds to the concurrent execution of the actions occurring in it.

Formula $\left[\begin{array}{c} 1 \\ \epsilon \end{array}\right]\left[\begin{array}{c} 1 \\ 1 \end{array}\right] \varphi_{11}$ captures the full run. (Note that we use squared brackets for the operators. This notation resembles deliberately the use of square brackets in modal logic. Although in this example all operators are (total) functions, this is not a general assumption, that is, in general $\neg \left[\begin{array}{c} a \\ b \end{array}\right] \neg\varphi$ does not imply $\left[\begin{array}{c} a \\ b \end{array}\right] \varphi$.)

Suppose now that the second action for agent α_1 has not been chosen, that is, the 2-step run is determined as before but with no information about the action α_1 executes at (II). If α_1 prefers to move to a state where φ_{01} holds, i.e., to s_{01}, she faces this problem: what action should I execute at (II) to force the system to reach state s_{01} ?

To capture this situation we look at φ_{01} as a goal for agent α_1. Formula $\left[\begin{array}{c} 1 \\ \epsilon \end{array}\right]\left[\begin{array}{c} \exists z \\ 1 \end{array}\right] \varphi_{01}$, where the quantifier ranges over the set of actions, says that agent α_1 has to choose an

action for z. The first operator in $\left[\begin{array}{c} 1 \\ \epsilon \end{array}\right]$ is built out of constants, so the two agents α_1 and α_2 have no choice to make, they need to perform actions 1 and ϵ, respectively. As a result of these actions the system moves from state s_{00} to state s_{10}. At this point, it is public knowledge the effect of operator $\left[\begin{array}{c} 1 \\ \epsilon \end{array}\right]$, and the agents move to consider the second operator fully aware of being in state s_{10}. Now α_1 has to choose a value for z and α_2 has to execute 1. α_1 can choose to perform action 0, that is, choose 0 for z. This way, she forces the system to state s_{01}. Similarly, if α_1 chooses ϵ. Instead, if α_1 chooses 1 for z, then the formula becomes $\left[\begin{array}{c} 1 \\ \epsilon \end{array}\right]\left[\begin{array}{c} 1 \\ 1 \end{array}\right]\varphi_{01}$, which is false and does not satisfy her goal. Clearly, 0 or ϵ are her only options to reach state s_{01}.

The general case arises allowing quantifiers in any position of the modal operator. Suppose that both agents want to reach s_{01} in a 2-step run starting at s_{00} and that none of their actions has been fixed. In this case, all of the entries of the operator are quantified. We obtain the following formula:

$$\left[\begin{array}{c} \exists x \\ \exists y \end{array}\right]\left[\begin{array}{c} \exists z \\ \exists w \end{array}\right]\varphi_{01} \tag{1}$$

In this formula, both agents α_1 and α_2 have φ_{01} as final goal in the two steps of the run. Note that in this simple example there are several combinations of actions that bring them to the desired state.

If agent α_1, when choosing her first action, has no particular feeling for reaching state s_{01} or even desires to avoid it[1], the following expression gives the correct description:

$$\left[\begin{array}{c} \forall x \\ \exists y \end{array}\right]\left[\begin{array}{c} \exists z \\ \exists w \end{array}\right]\varphi_{01} \tag{2}$$

Here we can finally recognize the specific role of existential and universal quantifiers. Roughly speaking, existential quantifiers mark entries where the agent's choice stems from its wanting to make the formula true. The universal quantifiers instead mark entries where the agent chooses with no commitment to the truth-value of the formula or, in another reading, with the desire to make the formula false.

The first operator in (2) tells us that agent α_1 may cause the system to move to a state from which it is not possible to satisfy $\left[\begin{array}{c} \exists z \\ \exists w \end{array}\right]\varphi_{01}$ while α_2 tries to do the very opposite. The second operator tells us that at the second step both α_1 and α_2 want to end up in s_{01}. In the given system, both formulas (1) and (2) are true since the agents can always force the system to reach state s_{01} if they want. This is not always the case.

In full generality, we may have any combination of quantifiers in a modal operator. A sequence of modal operators like in

$$\left[\begin{array}{c} Q_1 x \\ Q_2 y \end{array}\right]\left[\begin{array}{c} Q_3 z \\ Q_4 w \end{array}\right]\varphi \quad (Q_i \in \{\forall, \exists\}) \tag{3}$$

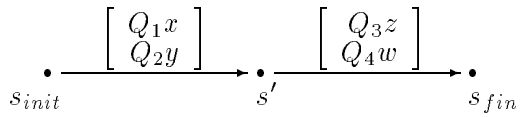
is evaluated from left to right one operator at a time. When choosing the action for the first operator in the sequence, agent α_1 does not know what α_2 is doing at this stage.

[1]These two readings of the universal quantifier correspond to different logics.

Other information may be shared. For instance, in the quantified system we present in section 6 it is assumed that agent α_1 knows α_2's attitude toward the rest of the formula that is, what quantifiers Q_2 and Q_4 are. Analogously, switching α_1 and α_2.

Once the values of x and y are chosen, the first operator is determined and the system evolves to the next state as indicated by that operator. This change is known to all the agents. The new state becomes the present state in the evaluation of the rest of the formula $\left[\begin{array}{c} Q_3 z \\ Q_4 w \end{array} \right] \varphi$. In particular, both agents are fully aware of what changes their combined first actions caused before deciding their actions for the second operator.[2]

From the point of view of a transition system, formula (3) corresponds to the transition from the initial state, s_{init}, to the final state, s_{fin}, through an intermediate state, s' (after the instantiation of variables x and y, s' is the present state for the rest of the formula):



So far, we have informally presented a formalism for concurrent actions in a multi-agent system where agents are free to choose according to their desires. Although this allows us to describe an interesting set of problems, we have not taken advantage of all the elements in the formalism.

In our everyday experience, most of the time agents must choose and perform a sequence of actions without knowledge of what other agents are doing during the same period of time. To state that agents α_1 and α_2 choose two (n) successive actions with no knowledge of each other's choices, we write our formula using two-column (n-column) operators like

$$\left[\begin{array}{cc} Q_1 x & Q_3 z \\ Q_2 y & Q_4 w \end{array} \right] \tag{4}$$

The intended reading is that agent α_1, while choosing an instance for x and z, has no knowledge of α_2's choices for y and w. Similarly for α_2. The agents become aware of the effects of their combined actions only after both of them have executed both choices. In a transition system, this operator corresponds to the transition from s_{init} to s_{fin} without any intermediate state s'.

With this extension of the language, we enrich the meaning of the modal symbols '[' and ']' with respect to more traditional logics. In systems of modal logic brackets are used to represent grouping and are generally associated with the meaning of the necessity modality. In our formalism, the brackets maintain these roles and also receive an epistemic touch. In an operator of form $\left[\begin{array}{cc} Q_1 x & Q_3 z \\ Q_2 y & Q_4 w \end{array} \right] \varphi$, the brackets mark a situation where the agents perform a sequence of actions all the while receiving no feedback either about the evolution of the system or about other agents' actions. These brackets are like epistemic

[2] Operator $\left[\begin{array}{c} a \\ b \end{array} \right]$ may reach states different from the states reachable by, say, $\left[\begin{array}{c} a \\ \epsilon \end{array} \right]$ followed by $\left[\begin{array}{c} \epsilon \\ b \end{array} \right]$. Intuitively, the result of concurrent actions can be different from the result of any "linear" sequence of those same actions. This feature is known as true concurrency.

boundaries, and the agents, during the "time" spanned by the modal operator, act like isolated agents.

In formula (4), none of the agents knows what the precise consequences of her first actions are. The system changes depending on the effect of all the actions executed, if an agent does not know what the other agent executes and cannot observe the changes in the system, she has no way of knowing the concrete effects of her own actions. This happens in everyday situations. For example, when two people need to use the same phone line at the same time: one, say, wants to connect to the internet using a modem while the other decides to call a friend. These people start using the phone line (perhaps in different rooms) and only when they actually listen to the phone after dialing or check the modem activity to see why the connection does not go through, do they discover the disappointing output of their combined sequences of actions.

Thus, in general quantified modal operators are not equivalent to sequences of simpler operators:

$$\left[\begin{array}{cc} Q_1x & Q_3z \\ Q_2y & Q_4w \end{array}\right]\varphi \not\equiv \left[\begin{array}{c} Q_1x \\ Q_2y \end{array}\right]\left[\begin{array}{c} Q_3z \\ Q_4w \end{array}\right]\varphi \tag{5}$$

Nevertheless, the equivalence holds when only constants occur:

$$\left[\begin{array}{cc} a & c \\ b & d \end{array}\right]\varphi \equiv \left[\begin{array}{c} a \\ b \end{array}\right]\left[\begin{array}{c} c \\ d \end{array}\right]\varphi \qquad \text{(where } a, b, c, d \text{ denote fixed actions.)}$$

In these logics, we always assume that actions in the same column of an operator are performed concurrently by different agents, and all actions in a row are executed by the same agent in that order. From now on, we posit a fixed number k of agents.

3 Multi-Agent Propositional Logic

3.1 Syntax

The language contains denumerable many *constants* denoting actions ϵ, a_0, a_1, $a_2, \ldots$ (ϵ is a distinguished constant denoting the "null" action) and *atomic sentences* p_0, p_1, p_2, ... We write *Const* for the set of constants, $a, b, c, \ldots$ for arbitrary constants, and $p, q, \ldots$ for arbitrary atomic sentences.

Besides the propositional operators $\neg$ and $\rightarrow$ (from which $\vee, \wedge$, and $\leftrightarrow$ are defined in the usual way), new unary modal operators (multi-agent operators) are introduced. A *modal operator* $[M_n]$, where $n \geq 1$, is syntactically a matrix with k rows and n columns whose entries are constants (all the operators and matrices have k rows unless otherwise stated; we drop index n when not relevent or when it is clear from the context). We call *cOP* the set of all modal operators in the logic. Given any $k \times n$ matrix M ($n \geq 1$), we write $[M]$ if it is in *cOP*. The $k \times 1$ matrix whose entries are all ϵ is always in *cOP* and is called the *null operator*.

Although for theoretical purposes one would let *cOP* be the set of all possible operators built out of elements in *Const*, in this introduction to the logic we take a broader view and assume that *cOP* is simply a subset of all possible operators. We assume that the

set of operators contains only those combinations of actions that are actually compatible in the situation one wants to capture. For instance, if a_i corresponds to action "write in the i-th memory", then a system with two agents cannot meaningfully execute operator $\left[\begin{array}{c} a_i \\ a_i \end{array}\right]$. Taking cOP to be a subset of all possible operators, we leave open the possibility of discharging such an operator already at the syntactical level.

cOP is closed under juxtaposition: given modal operators $[A]$ and $[B]$, both in cOP, operator $[C]$, obtained by juxtaposition of $[A]$ and $[B]$, is also in cOP. We write $[A \mid B]$ for the juxtaposition of $[A]$ and $[B]$ in this order. For instance, assume $k = 2$ and let $[A] = \left[\begin{array}{cc} a_1 & a_3 \\ a_2 & a_4 \end{array}\right]$, $[B] = \left[\begin{array}{c} a_5 \\ a_6 \end{array}\right]$, then $[A \mid B] = \left[\begin{array}{ccc} a_1 & a_3 & a_5 \\ a_2 & a_4 & a_6 \end{array}\right]$ and it is in cOP if both $[A]$ and $[B]$ are.

In full generality, cOP is any set of modal operators (in the language) containing the null operator and closed under juxtaposition.

A modal operator with n columns is called an *n-operator*. There is no operator in the logic with zero columns. Nevertheless, to simplify some definitions, we refer to the matrix of dimension $k \times 0$ as the empty operator. If A is a fixed matrix, then operators obtained by juxtaposition of A are called *A-iterated*. An operator $[B]$ in cOP is a *suboperator* of $[M]$ if there exist $[A]$ and $[C]$ in cOP (or empty), such that $[M] = [A \mid B \mid C]$ and $[M] \neq [B]$.

The set of *formulas* (sentences in the propositional logic) is defined as follows:

1. all atomic sentences are formulas
2. $\neg\varphi$ is a formula if φ is a formula
3. $\varphi \rightarrow \psi$ is a formula if both φ and ψ are formulas
4. $[M]\varphi$ is a formula if $[M]$ is in cOP and φ is a formula

3.2 Semantics

A *multi-agent model* is a 4-tuple $\left\langle W, P; \{R^{k\times n} \mid n \in N^+\}; [\![\cdot]\!] \right\rangle$ such that:

- W is a non-empty set of states;
- P is a set of actions, one action for each constant of the language.[3] In particular, P contains ε, the "null" action;
- for all $n \in N^+$ and for all matrices Γ, if there exists $[A] \in cOP$ with $\Gamma = [\![A]\!]$ (see below), then $R^{k\times n}(\Gamma) \subseteq W \times W$. The relations satisfy the following properties:
  a) fix $R^{k,n}(\Gamma)$, $R^{k,m}(\Gamma')$, then $R^{k,n}(\Gamma) \circ R^{k,m}(\Gamma') = R^{k\times(n+m)}(\Gamma \mid \Gamma')$;[4]
  b) let Δ be the $k \times 1$ matrix $\left(\begin{array}{c} \varepsilon \\ \vdots \\ \varepsilon \end{array}\right)$, then $R^{k\times 1}(\Delta) = \{(w, w) \mid w \in W\}$;

[3] In other words, each element in the universe of actions P has a name in the language.

[4] $\circ$ stands for the relational composition.

- $[\![\cdot]\!]$ is a valuation function mapping atomic sentences to sets of states; constant ϵ to action ε; other constants to elements in $P \setminus \{\varepsilon\}$; modal operators $[A] = \begin{bmatrix} a_{1,1} & \dots & a_{1,n} \\ \dots & \dots & \dots \\ a_{k,1} & \dots & a_{k,n} \end{bmatrix}$ to $[\![\,[A]\,]\!] = \begin{pmatrix} [\![a_{1,1}]\!] & \dots & [\![a_{1,n}]\!] \\ \dots & \dots & \dots \\ [\![a_{k,1}]\!] & \dots & [\![a_{k,n}]\!] \end{pmatrix}$. We write $[\![A]\!]$ for $[\![\,[A]\,]\!]$. (Note that the juxtaposition of $[\![A]\!]$ and $[\![B]\!]$ is equivalent to $[\![A \mid B]\!]$.)

For $s \in W$, we write $\mathcal{M}, s \models \varphi$ to mean that sentence φ is true at s in model $\mathcal{M}$. The notion is defined as follows:

1. $\mathcal{M}, s \models p_i$ if $s \in [\![p_i]\!]$
2. $\mathcal{M}, s \models \varphi \to \psi$ if $\mathcal{M}, s \models \varphi$ implies $\mathcal{M}, s \models \psi$
3. $\mathcal{M}, s \models \neg\varphi$ if not $\mathcal{M}, s \models \varphi$
4. $\mathcal{M}, s \models [M]\varphi$ if $[M] \in cOP$ and for all s' such that $(s, s') \in R^{k,n}([\![M]\!])$, $\mathcal{M}, s' \models \varphi$

A model is characterized by the atomic transitions in the usual way. However, an atomic transition may be associated with an operator with several columns. This happens whenever the multi-column operator at stake has no suboperators in cOP.

The definitions of satisfiability and validity of sentences are as in standard modal logic. Let $\mathcal{M}$ be a model and φ a sentence. If $\mathcal{M}, s \models \varphi$ for some state s, we say that φ is *satisfiable* in $\mathcal{M}$. If φ is satisfiable in some model $\mathcal{M}$, we say that φ is *satisfiable*. If $\mathcal{M}, s \models \varphi$ for all states s, we say that φ is *valid* in $\mathcal{M}$. If φ is valid in all models, we say that φ is *valid*.

3.3 Axiomatics

We need to provide rules only for the modal operators since $\neg$ and $\to$ are standard boolean connectives and are characterized by propositional logic in the usual way.

The null operator is determined by the *null schema*:

(NS) $\varphi \leftrightarrow [N_1]\varphi$, where $[N_1]$ is the 1-operator whose entries are all ϵ.

All operators in cOP satisfy (K) and necessitation (RN) [5]

(K) $[M](\varphi \to \psi) \to ([M]\varphi \to [M]\psi)$; (RN) $\dfrac{\varphi}{[M]\varphi}$

Given an operator $[M]$, define the dual operator $\langle M \rangle$ by $\neg[M]\neg$. Then, rules (K) and (RN) imply that the system is normal.

In the characterization of modal operators we add the (operator) *composition schema*:

(CS) For every $[A], [B] \in cOP$, $[A][B]\varphi \leftrightarrow [A \mid B]\varphi$.

Schema (CS) is a sort of composition principle for modal operators. Its semantic counterpart is given by the composition requirement on the relations in the model. (CS) ties the interpretation of constant operators to the interpretation of their suboperators.

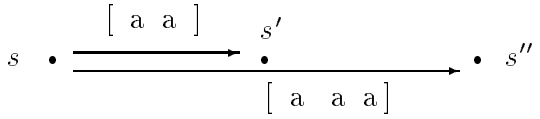


Figure 1: A transition system not satisfying (BP). (Simple loops are not shown)

Applicative concerns may suggest that more or less restricted sets cOP be adopted. Furthermore, the constraints on cOP affect the deductive properties of the whole system. Among the properties that are natural to consider, there is the *atomicity property*:

(AP) Let $[C_{n+1}]$, $n \geq 1$, be an operator in cOP such that $[C_{n+1}] = [A_1 \mid B_n]$. Then, $[A_1]$ is in cOP and so is $[B_n]$.

This property says that every n-operator can be obtained by juxtaposition of 1-operators.

Clearly, (AP) is very strong. Generally speaking, in the logic one can formalize a procedure at the syntactic level simply by introducing an n-operator that has no suboperators in cOP. Such an operator forces the system to evolve through a fixed path once the procedure is undertaken. This feature is particularly relevant when modeling, for instance, security protocols.

In contrast to this, property (AP) guarantees that any operator with 2 or more columns splits into suboperators thus preventing the purely syntactical characterization of procedures we hinted at above. On the other hand, assuming (AP) one obtains a simple system and has the possibility of reducing every complex evolution to a sequence of independent single steps.

Often it suffices to consider properties weaker than (AP). The *block (operator) property* turns out to be central to our approach:

(BP) Let $[D]$ be in cOP such that $[D] = [A \mid B \mid C]$ with $[B]$ in cOP. Then, $[A], [C]$ are in cOP or empty.

(BP) states that if an operator contains a suboperator in cOP, then the two pieces of the matrix on the left and on the right of the suboperator are also operators in cOP.

Without (BP) one might have operators[5] $[\,a\,a\,], [\,a\,a\,a\,]$ in cOP and, at the same time, $[\,a\,] \notin cOP$. Since these operators are interpreted by $[\,[\![a]\!]\,[\![a]\!]\,]$ and $[\,[\![a]\!]\,[\![a]\!]\,[\![a]\!]\,]$, respectively, one includes also frames as in Fig. 1, that is, frames where the interpretation of the two operators have no connection whatsoever. In our reading of the logic such a frame should be rejected. One cannot justify why after the execution of $[\,a\,a\,]$, the system cannot evolve through another execution of action a considering that it can evolve using $[\,a\,a\,a\,]$ from the very beginning.

In contrast to (AP), (BP) allows us to associate different "meanings" with the very same action according to the context in which it occurs. In a system satisfying (BP) but not (AP), the co-presence of, say, $[\,b\,a\,]$ and $[\,c\,a\,]$ in cOP implies neither the inclusion of

[5] For the sake of simplicity, we give the example in a one-agent system.

simpler operators nor any relationship between the states reachable by the two operators. In this way, action a may have different meanings depending on the actions that had been performed at the previous stage.

Similarly, sometimes we want to say that an agent begins to execute a protocol only if she is sure she will complete it (this happens, for instance, with some installation programs that do not allow the user to exit the process once it is started). In this case, it makes sense for modelling purposes to have operators that list all the (ordered) actions in the protocol and that cannot be obtained as combinations of smaller operators. A system satisfying (BP) is capable of such characterizations.

For these reasons, in the multi-agent propositional logic we always assume the closure of the operator set under (BP).

The axiomatization of the multi-agent logic is given by any complete set of axioms and rules for propositional logic, and axiom-schemata (NS), (K), (CS), and (RN).[6]
It is easy to establish that this system is sound.

4 Completeness, Decidability, and Complexity

One can build the canonical standard model (for multi-relational models) adapting the usual construction in modal logic to our axiomatic system. This is possible since the different modalities are independently axiomatized and the only schema involving more than one operator at a time, i.e. (CS), states that one needs to deal only with a base of operators for the set cOP.

Theorem 4.1 *The multi-agent propositional logic has the finite model property: if φ is satisfiable then it holds in a model with no more that $2^{\mathcal{O}(|\varphi|)}$ states.*

Corollary 4.1 *The satisfiability problem for the multi-agent propositional logic is decidable.*

Note that the block property (BP) is not dispensable.

In a one-agent logic, consider the set of axioms given in section 3.3 and drop (BP). Let operators $[\,a\,a\,]$ and $[\,a\,a\,a\,]$ be in cOP and assume there is no operator $[\,a\,]$. Then for all formulas φ the confluence axiom holds:

$$[\,a\,a\,a\,][\,a\,a\,]\varphi \leftrightarrow [\,a\,a\,][\,a\,a\,a\,]\varphi$$

One can produce an infinite number of operators commuting this way, for instance, considering operators with a prime number of occurrences of a. If these operators are in cOP and cannot be generated by simpler operators, then they are semantically independent, thus the logic resembles the product of infinite normal modal logics $\mathbf{K}$.[7] It is known [6] that any logic which is the product of three or more normal systems $\mathbf{K}$ is undecidable and non-finitely axiomatizable.

The following results are established using standard technics [7]

[6] Schema (R), that is, $[M](\varphi \wedge \psi) \leftrightarrow ([M]\varphi \wedge [M]\psi)$ is derivable in the logic [5].

[7] The commutativity and the confluence axioms hold for every product logic.

Theorem 4.2 *(Completeness) If a sentence is valid, then it is provable in the logic.*

Theorem 4.3 *The satisfiability of a formula φ can be established in exponential time on the length of φ.*

5 Relationship to Propositional Dynamic Logic

The propositional logic we introduced is a multi-modal language with features similar to those of *Propositional Dynamic Logic* (*PDL*) [7].

The idea of using actions as indices for modalities, the possibility of combining actions, the first condition on relations in multi-agent models, and the composition schema (CS) are all elements already introduced in *PDL*. Nonetheless, the two systems are quite different in other aspects. Condition (AP) is always satisfied in *PDL* but not in our logic and *PDL* cannot express the connection between actions and agents. The two approaches diverge considerably in their quantified versions as one can see in next section.

In a system with only one agent, our propositional system with (AP) can be considered a fragment of *PDL* where construct ";" is substituted by juxtaposition "|". In extending our multi-agent logic with *PDL* constructs like "$\cup$" and "$*$", one can decide to add these at the operator level as in $\left[\begin{array}{c} a \\ c \end{array}\right] \cup \left[\begin{array}{c} b \\ c \end{array}\right]$ and/or at the entry level as in $\left[\begin{array}{c} a \cup b \\ c \end{array}\right]$. These two options may generate languages with different characteristics. For instance, applying "$*$" to single entries of modal operators one obtains an asynchronous version of the logic.

6 A Multi-Agent Quantified Logic

We briefly introduce a simplified version of the *Multi-Agent Basic Logic* (*MBL*) which is just one system in the family determined by our formalism. The reader can find a formal introduction to multi-agent quantified systems in [8]. For the sake of simplicity, here cOP is taken to be the set of all operators built out of constants in the language, and neither relations nor functions are considered.

First, we add to the language of the propositional logic a denumerable list of variables for actions x_0, $x_1, x_2, \ldots$ A new set qOP of unary modal operators is obtained substituting one or more constants in an operator of cOP with quantified variables, that is, $\forall x_i$ or $\exists x_i$ for some i. We require that no variable occurs more than once in a quantified operator.

Let us write $A(i,j)$ for the element of $[A]$ at entry (i,j). An operator $[A_n] \in cOP$ is said to be an *instance* of an operator $[M_n]$ (possibly with quantifiers) if all the constants in $[M_n]$ equals the constants in the corresponding entries of $[A_n]$, i.e., $M_n(i,j) = A_n(i,j)$ for all constants $M_n(i,j)$.

Formulas in *MBL* are inductively generated as in section 3.1 adding the following clause: $[M_n]\varphi$ is a formula if $[M_n] \in qOP$ and φ is a formula.

A *model* for *MBL* is defined as in the propositional case. The semantics on non-quantified formulas is as before. For lack of space, we describe the semantics of formulas with quantified operators only informally.

Consider the set **IN** of possible instances of $[M]$. Consider the subset **T** of all operators $[A] \in \mathbf{IN}$ such that $\mathcal{M}, s \models [A]\varphi$ and let $\mathbf{T} \neq \emptyset$. A formula $[M]\varphi$ is set to be true if the agents can force $[M]$ to be instantiated by operators in **T** only.

To instantiate $[M]$, we proceed as follows. Whenever there is an existential quantifier in row i, agent α_i picks any action that occurs in the corresponding entry of some operator in **T**. This is justified by the fact that the agent wants to end up with an operator in this set. Whenever there is an universal quantifier in row i, an action is selected randomly. These rules determine an agent strategy. In particular, there might be several strategies for an agent.

Consider the set of instances for $[M]$ obtained combining one strategy for α_1, one strategy for α_2, ... , one strategy for α_k. The formula turns out to be true if and only if this set of operators is a subset of **T**.

7 Conclusions and Related Work

We have presented a formal language that can express true concurrency in a system with independent agents. The formalism is apt to capture several real-life situations where the agents' knowledge and their (perhaps changing) attitude toward a given goal are important.

A formula $C\varphi$ in this logic, where C is a constant modal operator, describes the evolution of the multi-agent system similarly to the description of the evolution of a 1-agent system provided by a formula in Propositional Dynamic Logic (PDL).

In contrast, a formula $Q\varphi$, where Q is a quantified modal operator, "posits" the problem of moving to a state where φ holds. The quantified operator describes also the number of actions each agent has to execute and the attitude of the agents toward the posited goal. According to the constraints in Q, the agents independently decide which actions to execute and the choices determine the evolution of the system. Thus, the combined choices of all the agents affect the set of reachable states.

The connection between quantified logics in our formalism and epistemic issues (knowledge, strategy, information exchange) is important. We have developed several quantified systems describing different types of agent. The systems differ mainly in the reasoning skills of the agents and in the amount of information available to the agents. Applications to concrete problems (communication, security, biological systems) are under development. On the proof-theoretical side, these quantified systems are generally undecidable.

Research on multi-agent systems is rapidly growing and has received new stimuli from the mixture of logic and game-theory.

Parikh introduced Game Logic (GL) in [9] to reason about program correctness (a survey including new results of Pauly can be found in [10]).

An extensive study of the relationship between logic, game theory and language has been carried out by Ahti Pietarinen. In his work on epistemic logic, Pietarinen considers also operators with several columns [11]. Such operators should not be confused with those introduced in this paper.

Finally, [12] and [13] are closely related to the logics presented above. The latter paper presents an interesting extension of modal logic along the lines of Hintikka's work in first-order logic.

Acknowledgements

Research partially supported by NSF grant CCR-0105651. The author is grateful to D. Leivant and A. Carbone for the interesting discussions on the topic and for their support. The paper was written in part while the author was at the Laboratory for Applied Ontology (ISTC-CNR) at Trento (IT) and at the Ladseb-CNR at Padova (IT).

References

[1] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi, *Reasoning about Knowledge*, MIT Press, 1995.

[2] M. J. Wooldridge and N. R. Jennings, Intelligent Agents: Theory and Practice, *Know. Eng. Review*, 10(2):115–152, 1995.

[3] W. van der Hoek, B. van Linder, and J.-J. Ch. Meyer, An Integrated Modal Approach to Rational Agents, In *Foundations of Rational Agency*, 14 ALS. Kluwer, 1999.

[4] C. Castelfranchi, Modelling Social Action for AI Agents, *Artificial Intelligence*, 103(1-2):157–182, 1998.

[5] B. F. Chellas, *Modal Logic: An Introduction*, Cambridge Univ. Press, 1980.

[6] R. Hirsch, I. Hodkinson, and A. Kurucz, On Modal Logics Between $\mathbf{K} \times \mathbf{K} \times \mathbf{K}$ and $\mathbf{S5} \times \mathbf{S5} \times \mathbf{S5}$, *Journal of Symbolic Logic*, 67:221–234, 2002.

[7] D. Harel, D. Kozen, and J. Tiuryn, *Dynamic Logic*, MIT Press, 2000.

[8] S. Borgo, Multi-Agent Logics Based on Information Independence and Concurrency, (submitted).

[9] R. Parikh, The Logic of Games and its Applications, *Annals of Discrete Mathematics*, 24, 1985.

[10] M. Pauly, An Introduction to Game Logic, In M. Faller, S. Kaufmann, and M. Pauly, editors, *Formalizing the Dynamics of Information*, pages 69–84. CSLI, 2000.

[11] A. Pietarinen, Games and Logics of Knowledge for Multi-agent Systems, In *Mexican International Conference on Artificial Intelligence 2002*, number 2313 in LNAI, 2002.

[12] R. Alur, T. Henzinger, and O. Kupferman, Alternating-time Temporal Logic, In de Roever W.-P., L. H., and P. A., editors, *Compositionality - The Significant Difference*, LNCS 1536, pages 23–60, 1999.

[13] J. C. Bradfield, Independence: Logics and Concurrency, In *CSL'00*, LNCS 1862, pages 247–261, 2000.