Reasoning about Nondeterministic and Concurrent Actions: A Process Algebra Approach

Giuseppe De Giacomo
Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Salaria 113, 00198 Roma, Italy
degiacomo@dis.uniroma1.it

Xiao Jun Chen
Dipartimento di Scienze dell'Informazione
Università di Roma "La Sapienza"
Via Salaria 113, 00198 Roma, Italy
chen@dsi.uniroma1.it

Abstract

In this paper, we study reasoning about actions following a *model checking* approach in contrast to the usual *validity checking* one. Specifically, we model a dynamic system as a transition graph which represents all the possible system evolutions in terms of state changes caused by actions. Such a transition graph is defined by means of a suitable *process algebra* associated with an explicit *global store*. To reason about system properties we introduce an extension of modal μ-calculus. This setting, although directly applicable only when complete information on the system is available, has several interesting features for reasoning about actions. On one hand, it inherits from the vast literature on process algebras tools for dealing with complex systems, treating suitably important aspects like parallelism, communications, interruptions, coordinations among agents. On the other hand, reasoning by model checking is typically much easier than more general logical services such as validity checking.

Introduction

To describe the behaviors of a dynamic system, we may choose among several levels of abstraction.

1. At a very concrete level, we characterize the system by its unique *actual evolution*, which can be represented as a sequence of states/actions. At this level, we assume complete information on each state, and we assume knowledge of which action will be performed next.

2. At a more abstract level, we characterize the system by all its possible evolutions, which can be collectively represented as a transition graph (*transition system*). A single evolution at Level 1 is represented as a *path* on such graph. One of these paths is going to be the actual evolution of the system, yet we do not know which one it will be. Each node, representing a state, has several labeled out-arcs representing the actions that can be performed in that state. Each action causes the transition of the system from the current state to a possible successor. We remark that different out-arcs may be labeled by the same action: in this case, the action is *nondeterministic*. At this level, we assume complete information on all the possible evolutions of the system, i.e. on all its states, and on which actions *can* be performed in each state. However, we do not have knowledge on which action will be performed next.

3. At the third level, we do not assume anymore complete information on the possible evolutions of the system, and we model the system by selecting a set of transition systems instead of a unique one. Each of such transition systems represents an alternative possible behavior. In other words, we assume partial information both on the current state, and on the states resulting from performing an action.

While Level 1 is generally considered too concrete to be realistic, both Level 2 and 3 have been used in modeling dynamic systems. In particular, Level 3 is the one usually adopted by research on reasoning about actions (Rosenschein 1981; Reiter 1993; Lin & Shoham 1991; 1992; Lifschitz 1990; Lifshitz & Karta 1994). In this case, a certain *logic* (situation calculus, dynamic logic, etc.) is used *both* to represent and to reason about the dynamic systems. The typical reasoning problem of interest is *logical implication* (validity) in the form

$$\Gamma \models S_{init} \Rightarrow \Phi$$

where: Γ are axioms, used to select the set of transition systems that represent the dynamic system; S_{init} is a formula, which is a (partial) description of the initial state; Φ is a formula expressing the property we want to prove, e.g. the reachability of a state where a certain property (the goal) holds.

In this paper, we adopt the viewpoint of Level 2. Following the model checking approach proposed in (Halpern & Vardi 1991), we use a description formalism to define the *transition system* representing the possible evolutions of the system, and a reasoning formalism, i.e. a suitable *logic* for specifying properties we want to check. Notably, this setting is the one typically used in Process Algebras (e.g. CCS (Milner 1989), CSP (Hoare 1985), ACP (Bergstra & Klop 1984)) to model concurrent and reactive systems.[1]

[1] In Artificial Intelligence, research on search-based plan-

The state (*configuration*) of the system is composed of a passive component (*global store*) and an active component (*process*). The passive component is described as a set of primitive propositions, which characterizes univocally the properties held in the passive component of the configuration. The active component describes the activities of all the *agents* (e.g. robots, persons, pieces of software, subroutines, environment, etc.) in the system, in terms of *actions*, which cause the changes of the system. The state of the passive component can only be changed by the activities of the active component, which in fact make the system evolve.

Making use of a global store associated to a process, we specify the effects of an action in terms of the difference between the current and the resulting global stores. Properties not mentioned among such effects are kept unchanged (in this way we address the frame problem). Note that, this treatment is different from most approaches in the literature on logics of programs, where all properties of the state resulting from an action are specified explicitely.

We adopt a specifically designed process algebra to describe processes. Indeed, process algebras are generally recognized as a convenient tool for describing concurrent and multiagent systems. They offer useful techniques for constructing more complicated processes from simpler ones using various constructors. Especially, they provide us with the ability to express communications, interruptions, coordinations, etc. among processes, by way of synchronizations (or asynchronizations).

The reasoning problem of interest in this case is *model checking* in the form

$$\mathcal{T}, s_{init} \models \Phi$$

where: $\mathcal{T}$ is a transition system representing the possible evolutions of our dynamic system; s_{init} is a state of $\mathcal{T}$; Φ is a formula expressing the property we want to verify.

We develop a suitable extension of modal μ-calculus (Kozen 1983), a powerful logic of programs which includes *PDL* (Kozen & Tiuryn 1990), *CTL*, *CTL** (Emerson 1990). We show that model checking in our logic can be linearly reduced to model checking in standard modal μ-calculus. We remark that, in the context of process algebra, for finite state processes (processes that can be interpreted on finite transition systems), various model checking tools (e.g. (Boudol *et al.* 1990; Cleaveland, Parrow, & Steffen 1993)) have been developed and implemented to verify whether a given modal/temporal logic formulae is satisfied by the process. By means of our reduction, it is possible to use efficiently such existing implemented tools for reasoning about actions in our setting.

ning, including much work on STRIP (e.g. (Bylander 1991)) can be considered at this level. In contrast, research on deductive planning is typically carried out at Level 3.

Atomic and synchronized actions

The evolution of the system from one configuration to another is caused by one or more *atomic actions*. An *atomic action* is an elementary uninterruptible action executed by one agent. A set of atomic actions performed together by various agents constitutes a *synchronized action*. We use *Act* to denote the finite set of all possible atomic actions (ranged over by $a, b, \ldots$), and $\{a_1, \ldots, a_n\}$ the synchronized action composed by $a_1, \ldots, a_n \in Act$.

Each action causes certain *effects* on the global store. The effects of a synchronized action α are defined on the basis of the effects of the atomic actions in α.

Effects of an atomic action may depend on the current configuration. We introduce *premises* of an atomic action to distinguish different configurations under which the action is being executed. Furthermore, alternative effects of an action are still permitted under the same configuration. In this case, the actual effect is chosen *nondeterministically* among those that are possible.

Formally, let *Prop* denote the finite set of all possible primitive propositions (ranged over by $A, B, \ldots$). We define a function *effct* which associates to each action $a \in Act$, a finite set of pairs $effct(a) = \{(\psi_1, E_1), \ldots, (\psi_n, E_n)\}$, where for each pair (ψ_i, E_i):

- ψ_i is a propositional formula over *Prop* describing the properties the global store must satisfy to have effect E_i: ψ_i is the *premise* of E_i.
- E_i is a set of literals – atomic propositions or their negations – over *Prop* that describes a possible *effect* of the execution of action a under premise ψ_i. The literals in E_i are forced to hold in the successive configuration. We require that for no A, we have both $A \in E_i$ and $\neg A \in E_i$.

Given configuration σ and action a: (i) if no premise of a is satisfied by σ, then a will be performed without any effect on σ; (ii) it is possible that more than one pair in $effct(a)$ has its premise satisfied in σ: in this case, we say that action a is *nondeterministic*, i.e. it leads to more than one successive configuration.

We now define the change of a global store caused by an (atomic or synchronized) action. Note that, to each global store we can associate an interpretation over *Prop*. We first introduce a simple update operator $\circ$, which takes an interpretation σ and a set of non-contradictory literals $\mathcal{L}$, and returns a new interpretation σ'.

Definition 1 Let σ be an interpretation over *Prop* and $\mathcal{L}$ a set of non-contradictory literals over *Prop*. We define update operator $\circ$ (infix) as follows: $\forall A \in Prop$,

$$(\sigma \circ \mathcal{L})(A) = \begin{cases} tt & A \in \mathcal{L} \\ ff & \neg A \in \mathcal{L} \\ \sigma(A) & A \notin \mathcal{L} \text{ and } \neg A \notin \mathcal{L} \end{cases}$$

Intuitively, the operator returns an interpretation which satisfies the non-contradictory literals in $\mathcal{L}$, and

retains the value of the original interpretation σ for those literals not occurring in $\mathcal{L}^2$.

Making use of this update operator, we can describe how an (atomic or synchronized) action affects the global store. Recall that, in general, actions are nondeterministic, hence several alternative changes to the global store can be produced. The set of possible global stores caused by a synchronized action $\{a_1, \ldots, a_n\}$ on σ, denoted by $\sigma/\{a_1, \ldots, a_n\}$, is defined below. As a special case, we have $\sigma/\{a\}$ when the action is in fact an atomic action a.

Definition 2 Let $\{a_1, \ldots, a_n\}$ be a set of atomic actions in Act, and σ an interpretation of the propositions in $Prop$. $\sigma/\{a_1, \ldots, a_n\}$ is defined as the set of all interpretations of the form $\sigma \circ (\mathcal{L}_1 \cup \ldots \cup \mathcal{L}_n)$ such that for $i = 1, \ldots n$:

- $\mathcal{L}_i = \begin{cases} E & \text{if } \exists(\psi, E) \in \mathit{effct}(a_i) \text{ s.t. } \sigma(\psi) = tt \\ \emptyset & \text{otherwise} \end{cases}$.
- $\mathcal{L}_1 \cup \ldots \cup \mathcal{L}_n$ does not contain contradictory literals.

Consider an atomic action a: (i) a is deterministic in σ iff $\sigma/\{a\}$ is singleton; (ii) if no effect E has its premises satisfied in σ, then $\sigma/\{a\} = \sigma$, i.e. the action has no effect (though it may still be performed).

In general, the effects of a synchronized action are the *sum* of the effects of the participating atomic actions. However, the effects of the atomic actions composing a synchronized action must be compatible. For example, pushing an pulling an handle cannot be synchronized. Intuitively, synchronizing two actions means not only to perform them at the same time, but also to perform each of them taking into account the feedback from the other. Consider action $\{a, b\}$ where the only applicable effect of a is $E_1 = \{A\}$ and the only applicable effect of b is $E_2 = \{\neg A\}$. Then the set of alternative global stores $\sigma/\{a, b\}$ resulting from executing $\{a, b\}$ is empty: a and b cannot be synchronized.

Observe the difference of the synchronized action $\{a, b\}$ and performing a and b simply in parallel. When a and b are performed in parallel, they are performed together, but independently. In such a case, it is reasonable to assume that they can be performed together even though they have contradictory effects. The contradiction can be resolved into nondeterminism. For example, let A and $\neg A$ be the only applicable effect of a and b respectively. Both a and b try to set the proposition A to the desired value independently. Nondeterministically, one of the two actions has "the last word" and succeeds, hence we have both the state in which A holds, and the state in which A does not hold. This intuition is often modeled by an interleaving model of parallel execution of actions. Interleaving is going to be the base of our treatment of parallel processes.

[2] We discuss possible extensions of form of update at the end of the paper.

Processes

We adopt CCS-style (Milner 1989) to combine actions into *processes*. Due to the appearances of recursions, we use process equations $P \doteq p$ to define processes. Here P is a process name and p is a process expression (or simply process). For each process name we associate a unique process definition. We will use $Proc$ to denote the set of process names. Processes follow the syntax below:

$$p ::= nil \mid P \mid (\phi \rightarrow a).p \mid p_1 + p_2 \mid p_1 \parallel p_2 \mid p\backslash\gamma$$

where nil denotes a predefined atomic process, P is a process name defined in $Proc$, ϕ denotes a propositional formula over $Prop$, a denotes an atomic action in Act, and γ denotes a set of expression of the form $\phi \rightarrow \varrho$ with ϕ a propositional formula over $Prop$ and ϱ a propositional formula over Act. Intuitively,

1. nil represents the termination of a process.
2. $(\phi \rightarrow a).p$ is the process which is capable of performing, under the *precondition* ϕ, action a, and then behaves as process p. This term can be viewed as an extension of CCS-term $a.p$ where no preconditions are specified.
3. $p_1 + p_2$ represents the alternative composition of p_1 and p_2.
4. $p_1 \parallel p_2$, the parallel composition of p_1 and p_2, is the process which can perform any interleaving or synchronizations of the actions of p_1 and p_2 – i.e. the performance of more actions together.
5. $p\backslash\gamma$ behaves like p but only actions in
$$A = \{\alpha \mid \forall \phi \rightarrow \varrho \in \gamma.\ \sigma \models \phi \Rightarrow \alpha \models \varrho\}$$
are allowed. This is an extention of CCS-term $p\backslash\gamma$ where γ is simply a set of atomic actions that are not allowed.

Given an initial configuration, the semantics of a system is given in terms of the transition relation $\_ \xrightarrow{\alpha} \_$ defined as the least relation satisfying the structural rules in Table 1. In this table, α (possibly with a subscript) denotes a set $\{a_1, \ldots, a_n\}$ of atomic actions executed together, and $\alpha(\varrho)$ denotes the truth value of ϱ in α.

Indeed, these rules permit us to associate to a configuration $(p_{init}, \sigma_{init})$, a transition system (Kripke structure) whose states are the configurations reachable from $(p_{init}, \sigma_{init})$, via the transitions inferred by the structural rules.

Definition 3 Given a set $\mathcal{P}$ of propositions, and set $\mathcal{A}$ of atomic actions, a *transition system* is a triple $(\mathcal{S}, \{\mathcal{R}_\alpha | \alpha \in 2^{\mathcal{A}}\}, \Pi)$, with a set of states $\mathcal{S}$, a family of transition relations $\mathcal{R}_\alpha \in \mathcal{S} \times \mathcal{S}$, and a mapping Π from $\mathcal{P}$ to subsets of $\mathcal{S}$.

We associate to each configuration $(p_{init}, \sigma_{init})$, a transition system $\mathcal{T} = (\mathcal{S}, \{\mathcal{R}_\alpha | \alpha \in 2^{\mathcal{A}}\}, \Pi)$, where $\mathcal{P} = Prop$, $\mathcal{A} = Act$, and

$$Act: \quad \frac{}{((\phi \rightarrow a).p, \sigma) \xrightarrow{\{a\}} (p, \sigma')}$$

where $\sigma(\phi) = tt, \quad \sigma' \in \sigma/\{a\}$

$$Def: \quad \frac{(p, \sigma) \xrightarrow{\alpha} (p', \sigma')}{(P, \sigma) \xrightarrow{\alpha} (p', \sigma')} \qquad P \doteq p$$

$$Sum_1: \quad \frac{(p_1, \sigma) \xrightarrow{\alpha} (p_1', \sigma')}{(p_1 + p_2, \sigma) \xrightarrow{\alpha} (p_1', \sigma')}$$

$$Sum_2: \quad \frac{(p_2, \sigma) \xrightarrow{\alpha} (p_2', \sigma')}{(p_1 + p_2, \sigma) \xrightarrow{\alpha} (p_2', \sigma')}$$

$$Int_1: \quad \frac{(p_1, \sigma) \xrightarrow{\alpha} (p_1', \sigma')}{(p_1 \parallel p_2, \sigma) \xrightarrow{\alpha} (p_1' \parallel p_2, \sigma')}$$

$$Int_2: \quad \frac{(p_2, \sigma) \xrightarrow{\alpha} (p_2', \sigma')}{(p_1 \parallel p_2, \sigma) \xrightarrow{\alpha} (p_1 \parallel p_2', \sigma')}$$

$$Syn: \quad \frac{(p_1, \sigma) \xrightarrow{\alpha_1} (p_1', \sigma_1') \qquad (p_2, \sigma) \xrightarrow{\alpha_2} (p_2', \sigma_2')}{(p_1 \parallel p_2, \sigma) \xrightarrow{\alpha_1 \cup \alpha_2} (p_1' \parallel p_2', \sigma')}$$

where $\sigma' \in \sigma/\alpha_1 \cup \alpha_2$

$$Res: \quad \frac{(p, \sigma) \xrightarrow{\alpha} (p', \sigma')}{(p \backslash \gamma, \sigma) \xrightarrow{\alpha} (p' \backslash \gamma, \sigma')}$$

where $\forall (\phi \rightarrow \varrho) \in \gamma \; s.t. \; \sigma(\phi) = tt : \alpha(\varrho) = tt$

Table 1: Structural Rules

- $\mathcal{S} = \{(p, \sigma) \mid (p_{init}, \sigma_{init})(\xrightarrow{-})^*(p, \sigma)\}$, where $\_\xrightarrow{-}\_$ is the least relation satisfying all rules in Table 1, and $\_(\xrightarrow{-})^*\_$ is its reflexive transitive closure;
- $((p, \sigma), (p', \sigma')) \in \mathcal{R}_\alpha$ iff $(p, \sigma) \xrightarrow{\alpha} (p', \sigma')$;
- $\Pi(A) = \{(p, \sigma) | \sigma(A) = tt\}$.

Proving system properties

Once we have a description of a dynamic system, we can use it to infer properties of the system, like invariance of certain statements, the possibility to reach a configuration where a certain property holds (i.e. where a certain "goal" is satisfied).

Many temporal and modal logics have been proposed in the process algebra literature for verifying properties of concurrent systems (Emerson & Halpern 1986; Kozen 1983; Manna & Pnueli 1989). Among these, we focus on one of the most powerful logic of programs, *modal μ-calculus* (Kozen 1983; Streett & Emerson 1989)), which includes logics like PDL, CTL and CTL^*, and has been investigated (e.g. (Stirling 1992; Cleaveland 1990)) in the context of process algebra for expressing "temporal" properties of reactive and parallel processes.

Specifically, we introduce an extension of standard modal μ-calculus, called $\mathcal{M}_\mu$, which is suitable to verify properties of systems specified in our description formalism. $\mathcal{M}_\mu$ comprises (i) a set of *propositions* to denote properties of the global store; (ii) a family of *modal operators* to denote the capability of the active component to perform a certain action; and (iii) a family of *fixpoint operators* to denote "temporal" properties defined by *induction* and *coinduction*.

The formulae of $\mathcal{M}_\mu$ is defined on the base of *action formulae* generated by ($a \in \mathcal{A}$):

$$\varrho ::= a \mid \neg\varrho \mid \varrho_1 \wedge \varrho_2.$$

The meaning of such formulae is given by the following satisfaction relation, where α is a set of atomic actions denoting a synchronized action in general.

$\alpha \models a$ iff $a \in \alpha$

$\alpha \models \neg\varrho$ iff $\alpha \not\models \varrho$

$\alpha \models \varrho_1 \wedge \varrho_2$ iff $\alpha \models \varrho_1$ and $\alpha \models \varrho_2$

Formulae of $\mathcal{M}_\mu$ are formed from action formulae, propositions in $\mathcal{P}$, and variable symbols in *Var*, according to the following abstract syntax ($A \in \mathcal{P}$, $X \in Var$):

$$\Phi ::= A \mid \neg\Phi \mid \Phi_1 \wedge \Phi_2 \mid \langle\varrho\rangle\Phi \mid \mu X.\Phi \mid X$$

We make use of standard notions of scope, bound and free occurrences of variables, closed formulas, etc. As usual in μ-calculus, we require *syntactic monotonicity*: every variable X must occur in the scope of an even number of negation signs ($\neg$). The semantics of $\mathcal{M}_\mu$ is based on the notions of transition system and valuation, where, given a transition system $\mathcal{T}$, a *valuation* $\mathcal{V}$ on $\mathcal{T}$ is a mapping from variables in *Var* to subsets of the states in $\mathcal{T}$ (Kozen 1983).

Let $\mathcal{T} = (\mathcal{S}, \{\mathcal{R}_\alpha | \alpha \in 2^{\mathcal{A}}\}, \Pi)$ be a transition system and $\mathcal{V}$ a valuation on $\mathcal{T}$. We assign meaning to formulae of the logic by associating to $\mathcal{T}$ and $\mathcal{V}$ an *extension function* $(\cdot)^{\mathcal{T}}_{\mathcal{V}}$, which maps formulae to subsets of $\mathcal{S}$. The extension function $(\cdot)^{\mathcal{T}}_{\mathcal{V}}$ is defined inductively according to Table 2, where we use $\mathcal{V}[X/\mathcal{E}]$ to denote the valuation identical to $\mathcal{V}$ except for $\mathcal{V}[X/\mathcal{E}](X) = \mathcal{E}$.

Let us comment Table 2. The boolean connectives have the expected meaning. The modal operator $\langle\cdot\rangle$ is interpreted so that $\langle\varrho\rangle\Phi$ is satisfied by a configuration iff there is an execution of some action satisfying ϱ, that leads to a successive configuration where Φ holds. $(\mu X.\Phi)^{\mathcal{T}}_{\mathcal{V}}$ is interpreted as the least fixpoint of the operator $(\Phi)^{\mathcal{T}}_{\mathcal{V}[X/\mathcal{E}]}$: the unique existence of such fixpoint is guaranteed by Tarski's Theorem, since $(\Phi)^{\mathcal{T}}_{\mathcal{V}[X/\mathcal{E}]}$

$$
\begin{array}{lcl}
(X)^{\mathcal{T}}_{\mathcal{V}} & = & \mathcal{V}(X) \subseteq \mathcal{S} \\
(A)^{\mathcal{T}}_{\mathcal{V}} & = & \Pi(A) \subseteq \mathcal{S} \\
(\neg\Phi)^{\mathcal{T}}_{\mathcal{V}} & = & \mathcal{S} - (\Phi)^{\mathcal{T}}_{\mathcal{V}} \\
(\Phi_1 \wedge \Phi_2)^{\mathcal{T}}_{\mathcal{V}} & = & (\Phi_1)^{\mathcal{T}}_{\mathcal{V}} \cap (\Phi_2)^{\mathcal{T}}_{\mathcal{V}} \\
(\langle\varrho\rangle\Phi)^{\mathcal{T}}_{\mathcal{V}} & = & \{s \in \mathcal{S} \mid \exists \alpha, s'.\ s.t. \\
 & & \quad \alpha \models \varrho,\ (s,s') \in \mathcal{R}_\alpha,\ s' \in (\Phi)^{\mathcal{T}}_{\mathcal{V}}\} \\
(\mu X.\Phi)^{\mathcal{T}}_{\mathcal{V}} & = & \bigcap\{\mathcal{E} \subseteq \mathcal{S} \mid\ (\Phi)^{\mathcal{T}}_{\mathcal{V}[X/\mathcal{E}]} \subseteq \mathcal{E}\ \}
\end{array}
$$

Table 2: Definition of the extension function $(\cdot)^{\mathcal{T}}_{\mathcal{V}}$

is monotonic according to the restriction of syntactic monotonicity.

Let us consider some examples. If only deterministic atomic actions are considered, then

$$\mu X.\phi_g \vee \langle \mathbf{any} \rangle X$$

where **any** denotes $a \vee \neg a$, expresses the existence of a plan (sequence of actions) to reach the goal ϕ_g from the initial configuration. If nondeterministic actions are present, the formula above does not suffice anymore, and needs to be replaced by

$$\mu X.\phi_g \vee \bigvee_{a \in Act} \langle \mathbf{any} \rangle tt \wedge [a]X.$$

A more complicated expression

$$\nu X.\mu Y.[a]((\langle b \rangle tt \wedge X) \vee Y)$$

states that b is possible infinitely often throughtout any infinite length run consisting wholly of a actions. Here $\nu X.\Phi = \neg\mu X.\neg\Phi[X/\neg X]$ where $\Phi[X/\neg X]$ is the formula obtained substituting all free occurrences of X by the formula $\neg X$. Note that $\nu X.\Phi$ is a greatest fixpoint.

As mentioned above, the reasoning problem we are interested in is model checking. Let $\mathcal{T} = (\mathcal{S}, \{\mathcal{R}_\alpha | \alpha \in 2^{\mathcal{A}}\}, \Pi)$ be a transition system, $s \in \mathcal{S}$, Φ a closed $\mathcal{M}_\mu$ formula. The related model checking problem (denoted by $\mathcal{T}, s \models \Phi$) is to verify whether $s \in (\Phi)^{\mathcal{T}}_{\mathcal{V}}$ where $\mathcal{V}$ is any valuation, since Φ is closed.

To relate $\mathcal{M}_\mu$ (where modalities have action formulae as indices) with the standard μ-calculus (where modalities are indexed by single actions), we can construct two transformations F and H on transition systems $\mathcal{T}$ and formulae Φ respectively, so that the verification of $\mathcal{T}, s \models \Phi$ (s is a state of $\mathcal{T}$) can be linearly reduced into the verification of standard modal μ-calculus formulae. This is formally stated below:

Proposition 1 *Let $\mathcal{S}$ denote the set of transition systems and $\mathcal{L}$ the set of standard modal μ-calculus formulae. There exist transformations $F : \mathcal{S} \to \mathcal{S}$ and $H : \mathcal{M}_\mu \to \mathcal{L}$ such that for any $\mathcal{T} \in \mathcal{S}, \Phi \in \mathcal{M}_\mu$:*

(i) *$F(\mathcal{T})$ and $H(\Phi)$ are linearly bounded by $\mathcal{T}$ and Φ respectively;*

(ii) *For any state s of $\mathcal{T}$, let $F(s)$ denote the image of state s in $F(\mathcal{T})$, then*

$$\mathcal{T}, s \models \Phi \ \textit{iff}\ F(\mathcal{T}), F(s) \models H(\Phi)$$

The construction of such F is based on reifying transitions, i.e. on introducing a new state for each transition, so that the action formula is transformed into a formula on the new state. These transformations of $\mathcal{T}$ and Φ by F and H respectively, allow us to exploit efficiently the existing model checking tools, e.g. (Cleaveland, Parrow, & Steffen 1993).

Discussion and conclusion

It is not difficult to build a formula that denotes exactly a given transition system $\mathcal{T}$ wrt a given state s. Such a formula $\chi_{\mathcal{T},s}$, which has essentially the same size as $\mathcal{T}$, is called *characteristic formula* (Steffen & Ingolfsdottir 1994; Halpern & Vardi 1991), and can be used to reduce model checking into validity checking. Indeed, we have $\mathcal{T}, s \models \Phi$ iff $\models \chi_{\mathcal{T},s} \Rightarrow \Phi$. This shows that model checking is in fact a specilization of validity checking, the one you get when you have complete information on the system.

Further extensions of the present work are possible along several directions. We outline some of them.

The first extension concerns the form of the update for the global store. The only essential point, in order to retain precisely the proposed setting, is to have some function returning $\sigma/\{a_1, \ldots, a_n\}$ from the inputs $\{a_1, \ldots, a_n\}$ and σ. It follows that we may adopt a more complex form of update, based, for example, on some notion of distance among global stores. In this way, we can specify effects of actions as general formulae over *Prop* instead of literals, and furthermore, we can address indirect effects by specifying domain constrains that must hold in each global store. Observe that the update we are interested in applies to interpretations, and thus is much simpler than update of theories discussed, e.g., in (Katsuno & Mendelzon 1991).

Another possible extension is to consider the global store as a set of multi-valued variables instead of boolean variables, or even as a first order interpretation over some fixed domain. Such extension can be easily accommodated in our setting. Indeed the way transition systems are built remains essentially the same, while the logic used for verification needs to be extended in order to take into account the new kind of properties expressed in the global store. Research in Databases on query languages based on first-order logic plus fixpoints e.g. (Abiteboul, Hull, & Vianu 1995),

and that on complex transactions e.g. (Bonner & Kifer 1995) are relevant.

Finally, we believe that it is of great interest moving from Level 2 to Level 3 by mixing the process algebra approach presented here, with the usual logical approach. This would allow us to introduce incomplete information in a better controlled way. For example, we could specify agents whose behavior is completely known by means of process description presented here, and agents whose behavior is only partially known (as happens typically for the environment) by logical axioms. To this end the research on "loose specification" in process algebras (Boudol & Larsen 1992), as well as research in knowledge representation on description logics that include assertions on "individuals" (which can be interpreted as a partial description of a transition system) (De Giacomo & Lenzerini 1994), is relevant.

References

Abiteboul, S.; Hull, R.; and Vianu, V. 1995. *Foundations of Databases.* Addison Wesley Publ. Co., Reading, Massachussetts.

Bergstra, J., and Klop, J. 1984. Process algebra for synchronous communication. *Information and Control* 60:109–137.

Bonner, A. J., and Kifer, M. 1995. Concurrency and communication in transaction logic. In *Proceedings of ICDT'95.*

Boudol, G.; de Simone, R.; Roy, V.; and Vergamini, D. 1990. Process calculi, from theory to practice: Verification tools. *Lecture Notes in Computer Science* 407.

Boudol, G. and Larsen, K. 1992. Graphical versus logical specifications. *Theoretical Computer Science* 106:3–20.

Bylander, T. 1991. Complexity results for planning. In *Proceedings of IJCAI-91*, 274–279.

Cleaveland, R.; Parrow, J.; and Steffen, B. 1993. The concurrency workbench: A semantics-based tool for the verification of concurrent systems. *ACM Transaction on Programming Languages and Systems* 15:36–72.

Cleaveland, R. 1990. Tableaux-based model checking in the propositional mu-calculus. *Acta Informatica* 27:725–747.

De Giacomo, G. and Lenzerini, M. 1994. Boosting the correspondence between description logics and propositional dynamic logics. In *Proceedings of AAAI-94*, 205–212.

Emerson, E., and Halpern, J. 1986. "sometimes" and "not never" revisited: on branching time versus linear time temporal logic. *Journal of ACM* 33(1):151–178.

Emerson, E. A. 1990. *Handbook of Theoretical Computer Science*, volume B. Elsevier Science Publishers B.V. chapter 16.

Halpern, J., and Vardi, M. 1991. Model checking vs. theorem proving: a manifesto. In *Proceedings of KR-91*, 325–334.

Hoare, C. 1985. *Communicating Sequential Processes.* London: Prentice Hall Int.

Katsuno, H., and Mendelzon, A. 1991. On the difference between updating a knowledge base and revising it. In *Proceedings of KR-91*, 387–394.

Kozen, D., and Tiuryn, J. 1990. Logics of programs. In van Leeuwen, J., ed., *Handbook of Theoretical Computer Science.* Elsevier Science Publishers. 790–840.

Kozen, D. 1983. Results on the propositional mu-calculus. *Theoretical Computer Science* 27:333–355.

Lifschitz, V. 1990. Frame in the space of situations. *Artificial Intelligence* 46:365–376.

Lifshitz, V., and Karta, G. 1994. Actions with indirect effects (preliminary report). In *Proceedings of KR-94*, 341–350.

Lin, F., and Shoham, Y. 1991. Provably correct theories of action (preliminary report). In *Proceedings of AAAI-91*, 349–354.

Lin, F., and Shoham, Y. 1992. Concurrent actions in the situation calculus. In *Proceedings of AAAI-92*, 590–595.

Manna, Z., and Pnueli, A. 1989. The anchored version of the temporal framework. In *Lecture Notes in Computer Science 354*, 201–284.

Milner, R. 1989. *Communication and Concurrency.* London: Prentice Hall.

Reiter, R. 1993. Proving properties of states in the situation calculus. *Artificial Intelligence* 64:337–351.

Rosenschein, S. 1981. Plan synthesis: a logical approach. In *Proceedings of IJCAI-81.*

Steffen, B., and Ingolfsdottir, A. 1994. Characteristic formulae for processes with divergence. *Information and Computation* 110(1):149–163.

Stirling, C. 1992. Modal and temporal logic. In Abramsky, S.; Gabbay, D. M.; and Maibaum, T. S. E., eds., *Handbook of Logic in Computer Science.* Oxford: Clarendon Press. 477–563.

Streett, R. S., and Emerson, E. A. 1989. An automata theoretic decision procedure for the propositional mu-calculus. *Information and Control* 81:249–264.