# Modular Interpreted Systems

Wojciech Jamroga
Department of Informatics
Clausthal University of Technology, Germany
wjamroga@in.tu-clausthal.de

Thomas Ågotnes
Department of Computer Engineering
Bergen University College, Norway
tag@hib.no

## ABSTRACT

We propose a new class of representations that can be used for modeling (and model checking) temporal, strategic and epistemic properties of agents and their teams. Our representations borrow the main ideas from *interpreted systems* of Halpern, Fagin et al.; however, they are also modular and compact in the way *concurrent programs* are. We also mention preliminary results on model checking alternating-time temporal logic for this natural class of models.

## Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent Systems*; I.2.4 [**Artificial Intelligence**]: Knowledge Representation Formalisms and Methods—*Modal logic*

## General Terms

Theory

## Keywords

open computational systems, temporal and strategic logics, modeling methodology, model checking

## 1. INTRODUCTION

The logical foundations of multi-agent systems have received much attention in recent years. Logic has been used to represent and reason about, e.g., knowledge [7], time [6], cooperation and strategic ability [3]. Lately, an increasing amount of research has focused on higher level representation languages for models of such logics, motivated mainly by the need for compact representations, and for representations that correspond more closely to the actual systems which are modeled. Multi-agent systems are *open* systems, in the sense that agents interact with an environment only partially known in advance. Thus, we need representations of models of multi-agent systems which are *modular*, in the sense that a component, such as an agent, can be replaced, removed, or added, without major changes to the representation of the whole model. However, as we argue in this paper, few existing representation languages are both modular, compact and computationally grounded on the one hand, and allow for representing properties of both knowledge and strategic ability, on the other.

In this paper we present a new class of representations for models of open multi-agent systems, which are modular, compact and come with an implicit methodology for modeling and designing actual systems.

The structure of the paper is as follows. First, in Section 2, we present the background of our work – that is, logics that combine time, knowledge, and strategies. More precisely: modal logics that combine branching time, knowledge, and strategies under incomplete information. We start with computation tree logic CTL, then we add knowledge (CTLK), and then we discuss two variants of alternating-time temporal logic (ATL): one for the perfect, and one for the imperfect information case. The semantics of logics like the ones presented in Section 2 are usually defined over *explicit models* (Kripke structures) that enumerate all possible (global) states of the system. However, enumerating these states is one of the things one mostly wants to avoid, because there are too many of them even for simple systems. Thus, we usually need representations that are more *compact*. Another reason for using a more specialized class of models is that general Kripke structures do not always give enough help in terms of methodology, both at the stage of design, nor at implementation. This calls for a semantics which is more *grounded*, in the sense that the correspondence between elements of the model, and the entities that are modeled, is more immediate. In Section 3, we present an overview of representations that have been used for modeling and model checking systems in which time, action (and possibly knowledge) are important; we mention especially representations used for theoretical analysis. We point out that the compact and/or grounded representations of temporal models do not play their role in a satisfactory way when agents' strategies are considered. Finally, in Section 4, we present our framework of *modular interpreted systems* (MIS), and show where it fits in the picture. We conclude with a somewhat surprising hypothesis, that model checking ability under imperfect information for MIS can be computationally cheaper than model checking perfect information. Until now, almost all complexity results were distinctly in favor of perfect information strategies (and the others were indifferent).

## 2. LOGICS OF TIME, KNOWLEDGE, AND STRATEGIC ABILITY

First, we present the logics CTL, CTLK, ATL and ATL$_{ir}$ that are the starting point of our study.

### 2.1 Branching Time: CTL

Computation tree logic CTL [6] includes operators for temporal properties of systems: i.e., path quantifier E ("there is a path"), to-


*AAMAS'07,* May 14–18, 2007, Honolulu, Hawai'i, USA.




978-81-904262-7-5 (RPS) ©2007 IFAAMAS

gether with temporal operators: $\bigcirc$ ("in the next state"), $\Box$ ("always from now on") and $\mathcal{U}$ ("until").[1] Every occurrence of a temporal operator is immediately preceded by exactly one path quantifier (this variant of the language is sometimes called "vanilla" CTL).

Let $\Pi$ be a set of atomic propositions with a typical element $p$. CTL formulae $\varphi$ are defined as follows:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \mathsf{E}\bigcirc\varphi \mid \mathsf{E}\Box\varphi \mid \mathsf{E}\varphi\,\mathcal{U}\,\varphi.$$

The semantics of CTL is based on Kripke models $M = \langle St, \mathcal{R}, \pi\rangle$, which include a nonempty set of states $St$, a state transition relation $\mathcal{R} \subseteq St \times St$, and a valuation of propositions $\pi : \Pi \to \mathcal{P}(St)$. A *path* $\lambda$ in $M$ refers to a possible behavior (or computation) of system $M$, and can be represented as an infinite sequence of states $q_0q_1q_2...$ such that $q_i\mathcal{R}q_{i+1}$ for every $i = 0, 1, 2, ....$ We denote the $i$th state in $\lambda$ by $\lambda[i]$. A *q-path* is a path that starts in $q$. Interpretation of a formula in a state $q$ in model $M$ is defined as follows:

- $M, q \models p$ iff $q \in \pi(p)$;
- $M, q \models \neg\varphi$ iff $M, q \not\models \varphi$;
- $M, q \models \varphi \wedge \psi$ iff $M, q \models \varphi$ and $M, q \models \psi$;
- $M, q \models \mathsf{E}\bigcirc\varphi$ iff there is a $q$-path $\lambda$ such that $M, \lambda[1] \models \varphi$;
- $M, q \models \mathsf{E}\Box\varphi$ iff there is a $q$-path $\lambda$ such that $M, \lambda[i] \models \varphi$ for every $i \geq 0$;
- $M, q \models \mathsf{E}\varphi\,\mathcal{U}\,\psi$ iff there is a $q$-path $\lambda$ and $i \geq 0$ such that $M, \lambda[i] \models \psi$ and $M, \lambda[j] \models \varphi$ for every $0 \leq j < i$.

## 2.2 Adding Knowledge: CTLK

CTLK [19] is a straightforward combination of CTL and standard epistemic logic [10, 7]. Let $\mathbb{A}\mathrm{gt} = \{1, ..., k\}$ be a set of agents with a typical element $a$. Epistemic logic uses operators for representing agents' knowledge: $K_a\varphi$ is read as "agent $a$ knows that $\varphi$". Models of CTLK extend models of CTL with epistemic indistinguishability relations $\sim_a \subseteq St \times St$ (one per agent). We assume that all $\sim_a$ are equivalences. The semantics of epistemic operators is defined as follows:

$$M, q \models K_a\varphi \quad \text{iff } M, q \models \varphi \text{ for every } q' \text{ such that } q \sim_a q'.$$

Note that, when talking about agents' knowledge, we implicitly assume that agents may have imperfect information about the actual current state of the world (otherwise the notion of knowledge would be trivial). This does not have influence on the way we model evolution of a system as a single unit, but it will become important when particular agents and their strategies come to the fore.

## 2.3 Agents and Their Strategies: ATL

*Alternating-time temporal logic* ATL [3] is a logic for reasoning about temporal and strategic properties of open computational systems (multi-agent systems in particular). The language of ATL consists of the following formulae:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle\langle A\rangle\rangle\bigcirc\varphi \mid \langle\langle A\rangle\rangle\Box\varphi \mid \langle\langle A\rangle\rangle\varphi\,\mathcal{U}\,\varphi.$$

where $A \subseteq \mathbb{A}\mathrm{gt}$. Informally, $\langle\langle A\rangle\rangle\varphi$ says that agents $A$ have a collective strategy to enforce $\varphi$. It should be noted that the CTL path quantifiers $\mathsf{A}, \mathsf{E}$ can be expressed with $\langle\langle\varnothing\rangle\rangle, \langle\langle\mathbb{A}\mathrm{gt}\rangle\rangle$ respectively.

The semantics of ATL is defined in so called *concurrent game structures (*CGS*s)*. A CGS is a tuple

$$M = \langle \mathbb{A}\mathrm{gt}, St, Act, d, o, \Pi, \pi\rangle,$$

consisting of: a set $\mathbb{A}\mathrm{gt} = \{1, \ldots, k\}$ of *agents*; set $St$ of *states*; *valuation of propositions* $\pi : \Pi \to \mathcal{P}(St)$; set $Act$ of atomic *actions*. Function $d : \mathbb{A}\mathrm{gt} \times St \to \mathcal{P}(Act)$ indicates the actions available to agent $a \in \mathbb{A}\mathrm{gt}$ in state $q \in St$. Finally, $o$ is a deterministic *transition function* which maps a state $q \in St$ and an action profile $\langle\alpha_1, \ldots, \alpha_k\rangle \in Act^k, \alpha_i \in d(i, q)$, to another state $q' = o(q, \alpha_1, \ldots, \alpha_k)$.

DEFINITION 1. *A (memoryless)* strategy *of agent $a$ is a function $s_a : St \to Act$ such that $s_a(q) \in d(a, q)$.*[2] *A* collective strategy *$S_A$ for a team $A \subseteq \mathbb{A}\mathrm{gt}$ specifies an individual strategy for each agent $a \in A$. Finally, the* outcome *of strategy $S_A$ in state $q$ is defined as the set of all computations that may result from executing $S_A$ from $q$ on:*

$$out(q, S_A) = \{\lambda = q_0q_1q_2... \mid q_0 = q \text{ and for every } i = 1, 2, ... \text{ there exists } \langle\alpha_1^{i-1}, ..., \alpha_k^{i-1}\rangle \text{ such that } \alpha_a^{i-1} = S_A(a)(q_{i-1}) \text{ for each } a \in A,\ \alpha_a^{i-1} \in d(a, q_{i-1}) \text{ for each } a \notin A, \text{ and } o(q_{i-1}, \alpha_1^{i-1}, ..., \alpha_k^{i-1}) = q_i\}.$$

The semantics of cooperation modalities is as follows:

- $M, q \models \langle\langle A\rangle\rangle\bigcirc\varphi$ iff there is a collective strategy $S_A$ such that, for every $\lambda \in out(q, S_A)$, we have $M, \lambda[1] \models \varphi$;
- $M, q \models \langle\langle A\rangle\rangle\Box\varphi$ iff there exists $S_A$ such that, for every $\lambda \in out(q, S_A)$, we have $M, \lambda[i]$ for every $i \geq 0$;
- $M, q \models \langle\langle A\rangle\rangle\varphi\,\mathcal{U}\,\psi$ iff there exists $S_A$ such that for every $\lambda \in out(q, S_A)$ there is a $i \geq 0$, for which $M, \lambda[i] \models \psi$, and $M, \lambda[j] \models \varphi$ for every $0 \leq j < i$.

## 2.4 Agents with Imperfect Information: ATL$_{ir}$

As ATL does not include incomplete information in its scope, it can be seen as a logic for reasoning about agents who always have complete knowledge about the current state of the whole system. ATL$_{ir}$ [21] includes the same formulae as ATL, except that the cooperation modalities are presented with a subscript: $\langle\langle A\rangle\rangle_{ir}$ indicates that they address agents with imperfect *information* and imperfect *recall*. Formally, the recursive definition of ATL$_{ir}$ formulae is:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle\langle A\rangle\rangle_{ir}\bigcirc\varphi \mid \langle\langle A\rangle\rangle_{ir}\Box\varphi \mid \langle\langle A\rangle\rangle_{ir}\varphi\,\mathcal{U}\,\varphi$$

Models of ATL$_{ir}$, *concurrent epistemic game structures* (CEGS), can be defined as tuples $M = \langle \mathbb{A}\mathrm{gt}, St, Act, d, o, \sim_1, ..., \sim_k, \Pi, \pi\rangle$, where $\langle \mathbb{A}\mathrm{gt}, St, Act, d, o, \Pi, \pi\rangle$ is a CGS, and $\sim_1, ..., \sim_k$ are epistemic (equivalence) relations. It is required that agents have the same choices in indistinguishable states: $q \sim_a q'$ implies $d(a, q) = d(a, q')$. ATL$_{ir}$ restricts the strategies that can be used by agents to *uniform strategies*, i.e. functions $s_a : St \to Act$, such that: (1) $s_a(q) \in d(a, q)$, and (2) if $q \sim_a q'$ then $s_a(q) = s_a(q')$. A collective strategy is uniform if it contains only uniform individual strategies. Again, the function $out(q, S_A)$ returns the set of all paths that may result from agents $A$ executing collective strategy $S_A$ from state $q$. The semantics of ATL$_{ir}$ formulae can be defined as follows:

- $M, q \models \langle\langle A\rangle\rangle_{ir}\bigcirc\varphi$ iff there is a uniform collective strategy $S_A$ such that, for every $a \in A$, $q'$ such that $q \sim_a q'$, and $\lambda \in out(S_A, q')$, we have $M, \lambda[1] \models \varphi$;

[1] Additional operators $\mathsf{A}$ ("for every path") and $\Diamond$ ("sometime in the future") are defined in the usual way.

[2] This is a deviation from the original semantics of ATL [3], where strategies assign agents' choices to *sequences* of states, which suggests that agents can by definition recall the whole history of each game. While the choice of one or another notion of strategy affects the semantics of the full ATL$^*$, and most ATL extensions (e.g. for games with imperfect information), it should be pointed out that both types of strategies yield equivalent semantics for "pure" ATL (cf. [21]).

$M, q \models \langle\langle A \rangle\rangle_{ir} \Box \varphi$ iff there exists $S_A$ such that, for every $a \in A$, $q'$ such that $q \sim_a q'$, and $\lambda \in out(S_A, q')$, we have $M, \lambda[i]$ for every $i \geq 0$;

$M, q \models \langle\langle A \rangle\rangle_{ir} \varphi \mathcal{U} \psi$ iff there exist $S_A$ such that, for every $a \in A$, $q'$ such that $q \sim_a q'$, and $\lambda \in out(S_A, q')$, there is $i \geq 0$ for which $M, \lambda[i] \models \psi$, and $M, \lambda[j] \models \varphi$ for every $0 \leq j < i$.

That is, $\langle\langle A \rangle\rangle_{ir} \varphi$ holds iff $A$ have a uniform collective strategy, such that for every path that can possibly result from execution of the strategy *according to at least one agent from* $A$, $\varphi$ is the case.

# 3. MODELS AND MODEL CHECKING

In this section, we present and discuss various (existing) representations of systems that can be used for modeling and model checking. We believe that the two most important points of reference are in this case: (1) the modeling formalism (i.e., the logic and the semantics we use), and (2) the phenomenon, or more generally, the domain we are going to model (to which we will often refer as the "real world"). Our aim is a representation which is reasonably close to the real world (i.e., it is sufficiently compact and grounded), and still not too far away from the formalism (so that it e.g. easily allows for theoretical analysis of computational problems). We begin with discussing the merits of "explicit" models – in our case, these are transition systems, concurrent game structures and CEGSs, presented in the previous section.

## 3.1 Explicit Models

Obviously, an advantage of explicit models is that they are very close to the semantics of our logics (simply because they *are* the semantics). On the other hand, they are in many ways difficult to use to describe an actual system:

- Exponential size: temporal models usually have an exponential number of states with respect to any higher-level description (e.g. Boolean variables, $n$-ary attributes etc.). Also, their size is exponential in the number of processes (or agents) if the evolution of a system results from joint (synchronous or asynchronous) actions of several active entities [15]. For CGSs the situation is even worse: here, also the number of transitions is exponential, even if we fix the number of states.[3] In practice, this means that such representations are very seldom scalable.

- Explicit models include no modularity. States in a model refer to global states of the system; transitions in the model correspond to global transitions as well, i.e., they represent (in an atomic way) *everything* that may happen in one single step, regardless of who has done it, to whom, and in what way.

- Logics like ATL are often advertised as frameworks for modeling and reasoning about open computational systems. Ideally, one would like the elements of such a system to have as little interdependencies as possible, so that they can be "plugged" in and out without much hassle, for instance when we want to test various designs or implementations of the active component. In the case of a multi-agent system the need is perhaps even more obvious. We do not only need to "re-plug" various designs of a single agent in the overall architecture; we usually also need to change (e.g., increase) the number of agents acting in a given environment without necessarily changing the design of the whole system. Unfortunately, ATL models are anything but open in this sense.

[3]Another class of ATL models, alternating transition systems [2] represent transitions in a more succinct way. While we still have exponentially many states in an ATS, the number of transitions is simply quadratic wrt. to states (like for CTL models). Unfortunately, ATS are even less modular and harder to design than concurrent game structures, and they cannot be easily extended to handle incomplete information (cf. [9]).

Theoretical complexity results for explicit models are as follows. Model checking CTL and CTLK is P-complete, and can be done in time $O(ml)$, where $m$ is the number of transitions in the model, and $l$ is the length of the formula [5]. Alternatively, it can be done in time $O(n^2 l)$, where $n$ is the number of states. Model checking ATL is P-complete wrt. $m, l$ and $\mathbf{\Delta_3^P}$-complete wrt. $n, k, l$ ($k$ being the number of agents) [3, 12, 16]. Model checking ATL$_{ir}$ is $\mathbf{\Delta_2^P}$-complete wrt. $m, l$ and $\mathbf{\Delta_3^P}$-complete wrt. $n, k, l$ [21, 13].

## 3.2 Compressed Representations

Explicit representation of all states and transitions is inefficient in many ways. An alternative is to represent the state/transition space in a symbolic way [17, 18].

Such models offer some hope for feasible model checking properties of open/multi-agent systems, although it is well known that they are compact only in a fraction of all cases.[4] For us, however, they are insufficient for another reason: they are merely *optimized representations of explicit models*. Thus, they are neither more open nor better grounded: they were meant to optimize implementation rather than facilitate design or modeling methodology.

## 3.3 Interpreted Systems

Interpreted systems [11, 7] are held by many as a prime example of *computationally grounded* models of distributed systems. An interpreted system can be defined as a tuple $IS = \langle St_1, ..., St_k, St_{env}, \mathcal{R}, \pi \rangle$. $St_1, ..., St_k$ are local state spaces of agents $1, ..., k$, and $St_{env}$ is the set of states of the environment. The set of global states is defined as $St = St_1 \times ... \times St_k \times St_{env}$; $\mathcal{R} \subseteq St \times St$ is a transition relation, and $\pi : \Pi \rightarrow \mathcal{P}(St)$. While the transition relation encapsulates the (possible) evolution of the system over time, the epistemic dimension is defined by the local components of each global state: $\langle q_1, ..., q_k, q_{env} \rangle \sim_i \langle q'_1, ..., q'_k, q_{env} \rangle$ iff $q_i = q'_i$.

It is easy to see that such a representation is modular and compact as far as we are concerned with *states*. Moreover, it gives a natural ("grounded") approach to knowledge, and suggests an intuitive methodology for modeling epistemic states. Unfortunately, the way transitions are represented in interpreted systems is neither compact, nor modular, nor grounded: the temporal aspect of the system is given by a joint transition function, exactly like in explicit models. This is not without a reason: if we separate activities of the agents too much, we cannot model *interaction* in the framework any more, and interaction is the most interesting thing here. But the bottom line is that the temporal dimension of an interpreted system has exponential representation. And it is almost as difficult to "plug" components in and out of an interpreted system, as for an ordinary CTL or ATL model, since the "local" activity of an agent is completely merged with his interaction with the rest of the system.

## 3.4 Concurrent Programs

The idea of *concurrent programs* has been long known in the literature on distributed systems. Here, we use the formulation from [15]. A concurrent program $P$ is composed of $k$ concurrent processes, each described by a labeled transition system $P_i = \langle St_i, Act_i, \mathcal{R}_i, \Pi_i, \pi_i \rangle$, where $St_i$ is the set of local states of process

[4]Representation $R$ of an explicit model $M$ is *compact* if the size of $R$ is logarithmic with respect to the size of $M$.

$i$, $Act_i$ is the set of local actions, $\mathcal{R}_i \subseteq St_i \times Act_i \times St_i$ is a transition relation, and $\Pi_i, \pi_i$ are the set of local propositions and their valuation. The behavior of program $P$ is given by the product automaton of $P_1, ..., P_k$ under the assumption that processes work asynchronously, actions are interleaved, and synchronization is obtained through common action names.

Concurrent programs have several advantages. First of all, they are modular and compact. They allow for "local" modeling of components – much more so than interpreted systems (not only states, but also actions are local here). Moreover, they allow for representing explicit interaction between local transitions of reactive processes, like willful communication, and synchronization. On the other hand, they do not allow for representing implicit, "incidental", or not entirely benevolent interaction between processes. For example, if we want to represent the act of pushing somebody, the pushed object must explicitly execute an action of "being pushed", which seems somewhat ridiculous. Side effects of actions are also not easy to model. Still, this is a minor complaint in the context of CTL, because for temporal logics we are only interested in the flow of *transitions*, and not in the underlying *actions*. For temporal reasoning about $k$ asynchronous processes with no implicit interaction, concurrent programs seem just about perfect.

The situation is different when we talk about autonomous, proactive components (like agents), acting together (cooperatively or adversely) in a common environment – and we want to address their strategies and abilities. Now, particular actions are no less important than the resulting transitions. Actions may influence other agents' local states without their consent, they may have side effects on other agents' states etc. Passing messages and/or calling procedures is by no means the only way of interaction between agents. Moreover, the availability of actions (to an agent) should not depend on the actions that *will* be executed by other agents at the same time – these are the *outcome states* that may depend on these actions! Finally, we would often like to assume that agents act synchronously. In particular, all agents play simultaneously in concurrent game structures. But, assuming synchrony and autonomy of actions, *synchronization* can no longer be a means of coordination.

To sum up, we need a representation which is very much like concurrent programs, but allows for modeling agents that play synchronously, and which enables modeling more sophisticated interaction between agents' actions. The first postulate is easy to satisfy, as we show in the following section. The second will be addressed in Section 4.

We note that model checking CTL against concurrent programs is PSPACE-complete in the number of local states and the length of the formula [15].

## 3.5 Synchronous CP and Simple Reactive Modules

The semantics of ATL is based on synchronous models where availability of actions does not depend on the actions currently executed by the other players. A slightly different variant of concurrent programs can be defined via synchronous product of programs, so that all agents play simultaneously.[5] Unfortunately, under such interpretation, no direct interaction between agents' actions can be modeled at all.

DEFINITION 2. *A* synchronous concurrent program *consists of $k$ concurrent processes $P_i = \langle St_i, Act_i, \mathcal{R}_i, \Pi_i, \pi_i \rangle$ with the following unfolding to a* CGS*:* $\mathbb{A}\mathrm{gt} = \{1, ..., k\}$, $St = \prod_{i=1}^{k} St_i$, $Act = \bigcup_{i=1}^{k} Act_i$, $d(i, \langle q_1, ..., q_k \rangle) = \{\alpha_i \mid \langle q_i, \alpha_i, q_i' \rangle \in \mathcal{R}_i$ *for some* $q_i' \in St_i\}$, $o(\langle q_1, ..., q_k \rangle, \alpha_1, ..., \alpha_k) = \langle q_1', ..., q_k' \rangle$ *such that* $\langle q_i, \alpha_i, q_i' \rangle \in \mathcal{R}_i$ *for every* $i$; $\Pi = \bigcup_{i=1}^{k} \Pi_i$, *and* $\pi(p) = \pi_i(p)$ *for* $p \in \Pi_i$.

We note that the *simple reactive modules* (SRML) from [22] can be seen as a particular implementation of synchronous concurrent programs.

DEFINITION 3. *A* SRML *system is a tuple* $\langle \Sigma, \Pi, m_1, \ldots, m_k \rangle$, *where* $\Sigma = \{1, \ldots, k\}$ *is a set of* modules *(or* agents*),* $\Pi$ *is a set of Boolean variables, and, for each* $i \in \Sigma$, *we have* $m_i = \langle ctr_i, init_i, update_i \rangle$, *where* $ctr_i \subseteq \Pi$. *Sets* $init_i$ *and* $update_i$ *consist of* guarded commands *of the form* $\phi \rightsquigarrow v_1' := \psi_1; \ldots; v_k' := \psi_k$, *where every* $v_j \in ctr_i$, *and* $\phi, \psi_1, \ldots, \psi_k$ *are propositional formulae over* $\Pi$. *It is required that* $ctr_1, \ldots ctr_k$ *partitions* $\Pi$.

The idea is that agent $i$ controls the variables $ctr_i$. The *init* guarded commands are used to initialize the controlled variables, while the *update* guarded commands can change their values in each round. A guarded command is *enabled* if the guard $\phi$ is true in the current state of the system. In each round an enabled update guarded command is executed: each $\psi_j$ is evaluated against the current state of the system, and its logical value is assigned to $v_j$. Several guarded commands being enabled at the same time model non-deterministic choice. Model checking ATL for SRML has been proved EXPTIME-complete in the size of the model and the length of the formula [22].

## 3.6 Concurrent Epistemic Programs

Concurrent programs (both asynchronous and synchronous) can be used to encode epistemic relations too – exactly in the same way as interpreted systems do [20]. That is, when unfolding a concurrent program to a model of CTLK or $\mathrm{ATL}_{ir}$, we define that $\langle q_1, ..., q_k \rangle \sim_i \langle q_1', ..., q_k' \rangle$ iff $q_i = q_i'$. Model checking CTLK against concurrent epistemic programs is PSPACE-complete [20]. SRML can be also interpreted in the same way; then, we would assume that every agent can see only the variables he controls.

Concurrent epistemic programs are modular and have a "grounded" semantics. They are usually compact (albeit not always: for example, an agent with perfect information will always blow up the size of such a program). Still, they inherit all the problems of concurrent programs with perfect information, discussed in Section 3.4: limited interaction between components, availability of local actions depending on the actual transition etc. The problems were already important for agents with perfect information, but they become even more crucial when agents have only limited knowledge of the current situation. One of the most important applications of logics that combine strategic and epistemic properties is verification of communication protocols (e.g., in the context of security). Now, we may want to, e.g., check agents' ability to pass an information between them, without letting anybody else intercept the message. The point is that the *action* of intercepting is by definition enabled; we just look for a protocol in which the *transition* of "successful interception" is never carried out. So, availability of actions *must* be independent of the actions chosen by the other agents under incomplete information. On the other hand, interaction is arguably the most interesting feature of multi-agent systems, and it is really hard to imagine models of strategic-epistemic logics, in which it is not possible to represent communication.

## 3.7 Reactive Modules

Reactive modules [1] can be seen as a refinement of concurrent epistemic programs (primarily used by the MOCHA model checker [4]), but they are much more powerful, expressive and

[5] The concept is not new, of course, and has already existed in folk knowledge, although we failed to find an explicit definition in the literature.

grounded. We have already mentioned a very limited variant of RML (i.e., SRML). The vocabulary of RML is very close to implementations (in terms of general computational systems): the *modules* are essentially collections of variables, states are just valuations of variables; events/actions are variable updates. However, the sets of variables controlled by different agents can overlap, they can change over time etc. Moreover, reactive modules support incomplete information (through *observability* of variables), although it is not the main focus of RML. Again, the relationship between sets of observable variables (and to sets of controlled variables) is mostly left up to the designer of a system. Agents can act synchronously as well as asynchronously.

To sum up, RML define a powerful framework for modeling distributed systems with various kinds of synchrony and asynchrony. However, we believe that there is still a need for a simpler and slightly more abstract class of representations. First, the framework of RML is technically complicated, involving a number auxiliary concepts and their definitions. Second, it is not always convenient to represent *all* that is going on in a multi-agent system as reading and/or writing from/to program variables. This view of a multi-agent system is arguably close to its computer implementation, but usually rather distant from the real world domain – hence the need for a more abstract, and more conceptually flexible framework. Third, the separation of the "local" complexity, and the complexity of interaction is not straightforward. Our new proposal, more in the spirit of interpreted systems, takes these observations as the starting point. The proposed framework is presented in Section 4.

# 4. MODULAR INTERPRETED SYSTEMS

The idea behind distributed systems (multi-agent systems even more so) is that we deal with several *loosely* coupled components, where most of the processing goes on *inside* components (i.e., locally), and only a small fraction of the processing occurs *between* the components. Interaction is crucial (which makes concurrent programs an insufficient modeling tool), but it usually consumes much less of the agent's resources than local computations (which makes the explicit transition tables of CGS, CEGS, and interpreted systems an overkill). Modular interpreted systems, proposed here, extrapolate the modeling idea behind interpreted systems in a way that allows for a tight control of the interaction complexity.

DEFINITION 4. *A* modular interpreted system *(*MIS*) is defined as a tuple*

$$S = \langle \mathbb{A}\mathrm{gt}, env, Act, \mathcal{I}n \rangle,$$

*where* $\mathbb{A}\mathrm{gt} = \{a_1, ..., a_k\}$ *is a set of agents, env is the environment, Act is a set of actions, and* $\mathcal{I}n$ *is a set of symbols called* interaction alphabet. *Each agent has the following internal structure:*

$$a_i = \langle St_i, d_i, out_i, in_i, o_i, \Pi_i, \pi_i \rangle, \text{ where:}$$

- $St_i$ *is a set of local states,*
- $d_i : St_i \to \mathcal{P}(Act)$ *defines local availability of actions; for convenience of the notation, we additionally define the set of* situated actions *as* $D_i = \{\langle q_i, \alpha \rangle \mid q_i \in St_i, \alpha \in d_i(q_i)\}$,
- $out_i, in_i$ *are* interaction functions*;* $out_i : D_i \to \mathcal{I}n$ *refers to the influence that a given situated action (of agent* $a_i$*) may possibly have on the external world, and* $in_i : St_i \times \mathcal{I}n^k \to \mathcal{I}n$ *translates external manifestations of the other agents (and the environment) into the "impression" that they make on* $a_i$*'s transition function depending on the local state of* $a_i$*,*
- $o_i : D_i \times \mathcal{I}n \to St_i$ *is a (deterministic) local transition function,*
- $\Pi_i$ *is a set of local propositions of agent* $a_i$ *where we require that* $\Pi_i$ *and* $\Pi_j$ *are disjunct when* $i \neq j$*, and*
- $\pi_i : \Pi_i \to \mathcal{P}(St_i)$ *is a valuation of these propositions.*

*The environment* $env = \langle St_{env}, out_{env}, in_{env}, o_{env}, \Pi_{env}, \pi_{env} \rangle$ *has the same structure as an agent except that it does not perform actions, and that thus* $out_{env} : St_{env} \to \mathcal{I}n$ *and* $o_{env} : St_{env} \times \mathcal{I}n \to St_{env}$.

Within our framework, we assume that every *action* is executed by an *actor*, that is, an agent. As a consequence, every actor is explicitly represented in a MIS as an agent, just like in the case of CGS and CEGS. The environment, on the other hand, represents the (passive) context of agents' actions. In practice, it serves to capture the aspects of the global state that are not observable by any of the agents.

The input functions $in_i$ seem to be the fragile spots here: when given explicitly as tables, they have size exponential wrt. the number of agents (and linear wrt. the size of $\mathcal{I}n$). However, we can use, e.g., a construction similar to the one from [16] to represent interaction functions more compactly.

DEFINITION 5. Implicit input function *for state* $q \in St_i$ *is given by a sequence* $\langle\langle \varphi_1, \eta_1 \rangle, ..., \langle \varphi_n, \eta_n \rangle\rangle$*, where each* $\eta_j \in \mathcal{I}n$ *is an interaction symbol, and each* $\varphi_j$ *is a boolean combination of propositions* $\hat{\eta}^i$*, with* $\eta \in \mathcal{I}n$*;* $\hat{\eta}^i$ *stands for "*$\eta$ *is the symbol currently generated by agent* $i$*". The input function is now defined as follows:* $in_i(q, \epsilon_1, ..., \epsilon_k, \epsilon_{env}) = \eta_j$ *iff* $j$ *is the lowest index such that* $\{\hat{\epsilon}^1_1, ..., \hat{\epsilon}^k_k, \hat{\epsilon}^{env}_{env}\} \models \varphi_j$*. It is required that* $\varphi_n \equiv \top$*, so that the mapping is effective.*

REMARK 1. *Every* $in_i$ *can be encoded as an implicit input function, with each* $\varphi_j$ *being of polynomial size with respect to the number of interaction symbols (cf. [16]).*

Note that, for some domains, the MIS representation of a system requires exponentially many symbols in the interaction alphabet $\mathcal{I}n$. In such a case, the problem is inherent to the domain, and $in_i$ will have size exponential wrt the number of agents.

## 4.1 Representing Agent Systems with MIS

Let $St_g = (\prod_{i=1}^{k} St_i) \times St_{env}$ be the set of all possible global states generated by a modular interpreted system $S$.

DEFINITION 6. *The unfolding of a* MIS $S$ *for initial states* $Q \subseteq St_g$ *to a* CEGS $cegs(S, Q) = \langle \mathbb{A}\mathrm{gt}', St', \Pi', \pi', Act', d', o', \sim'_1, ..., \sim'_k \rangle$ *is defined as follows:*

- $\mathbb{A}\mathrm{gt}' = \{1, ..., k\}$ *and* $Act' = Act$*,*
- $St'$ *is the set of global states from* $St_g$ *which are reachable from some state in* $Q$ *via the transition relation defined by* $o'$ *(below),*
- $\Pi' = \bigcup_{i=1}^{k} \Pi_i \cup \Pi_{env}$*,*
- *For each* $q = \langle q_1, \ldots, q_k, q_{env} \rangle \in St'$ *and* $i = 1, ..., k, env$*, we define* $q \in \pi'(p)$ *iff* $p \in \Pi_i$ *and* $q_i \in \pi_i(p)$*,*
- $d'(i, q) = d_i(q_i)$ *for global state* $q = \langle q_1, ..., q_k, q_{env} \rangle$*,*
- *The transition function is constructed as follows. Let* $q = \langle q_1, ..., q_k, q_{env} \rangle \in St'$*, and* $\alpha = \langle \alpha_1, ..., \alpha_k \rangle$ *be an action profile s.t.* $\alpha_i \in d'(i, q)$*. We define* $input_i(q, \alpha) = in_i\big(q_i, out_1(q_1, \alpha_1), \ldots, out_{i-1}(q_{i-1}, \alpha_{i-1}), out_{i+1}(q_{i+1}, \alpha_{i+1}), \ldots, out_k(q_k, \alpha_k), out_{env}(q_{env})\big)$ *for each agent* $i = 1, \ldots, k$*, and* $input_{env}(q, \alpha) = in_{env}\big(q_{env}, out_1(q_1, \alpha_1), \ldots, out_k(q_k, \alpha_k)\big)$*. Then,* $o'(q, \alpha) = \langle o_1(\langle q_1, \alpha_1 \rangle, input_1(q, \alpha)), \ldots, o_k(\langle q_k, \alpha_k \rangle, input_k(q, \alpha)), o_{env}(q_{env}, input_{env}(q, \alpha)) \rangle$*;*

- *For each* $i = 1, ..., k$: $\langle q_1, ..., q_k, q_{env}\rangle \sim'_i \langle q'_1, ..., q'_k, q'_{env}\rangle$ *iff* $q_i = q'_i$.[6]

REMARK 2. *Note that* MIS*s can be used as representations of* CGS*s too. In that case, epistemic relations* $\sim'_i$ *are simply omitted in the unfolding. We denote the unfolding of a* MIS $S$ *for initial states* $Q$ *into a* CGS *by* $cgs(S, Q)$.

Propositions 3 and 5 state that modular interpreted systems can be used as representations for explicit models of multi-agent systems. On the other hand, these representations are not always compact, as demonstrated by Propositions 7 and 8.

PROPOSITION 3. *For every* CEGS $M$, *there is a* MIS $S^M$ *and a set of global states* $Q$ *of* $S^M$ *such that* $cegs(S^M, Q)$ *is isomorphic to* $M$.[7]

PROOF. Let $M = \langle\{1, \ldots, k\}, St, Act, d, o, \Pi, \pi, \sim_1, \ldots, \sim_k\rangle$ be a CEGS. We construct a MIS $S^M = \langle\{a_1, \ldots, a_k\}, env, Act, \mathcal{I}n\rangle$ with agents $a_i = \langle St_i, d_i, out_i, in_i, o_i, \Pi_i, \pi_i\rangle$ and environment $env = \langle St_{env}, out_{env}, in_{env}, o_{env}, \Pi_{env}, \pi_{env}\rangle$, plus a set $Q \subseteq St_g$ of global states, as follows.

- $\mathcal{I}n = Act \cup St \cup (Act^{k-1} \times St)$,
- $St_i = \{[q]_{\sim_i} \mid q \in St\}$ for $1 \le i \le k$ (i.e., $St_i$ is the set of $i$'s indistinguishability classes in $M$),
- $St_{env} = St$,
- $d_i([q]_{\sim_i}) = d(i, q)$ for $1 \le i \le k$ (this is well-defined since $d(i, q) = d(i, q')$ whenever $q \sim_i q'$),
- $out_i([q]_{\sim_i}, \alpha_i) = \alpha_i$ for $1 \le i \le k$; $out_{env}(q) = q$,
- $in_i([q]_{\sim_i}, \alpha_1, \ldots, \alpha_{i-1}, \alpha_{i+1}, \ldots, \alpha_k, q_{env}) = \langle\alpha_1, \ldots, \alpha_{i-1}, \alpha_{i+1}, \ldots, \alpha_k, q_{env}\rangle$ for $i \in \{1, \ldots, k\}$; $in_{env}(q, \alpha_1 \ldots, \alpha_k) = \langle\alpha_1, \ldots, \alpha_k\rangle$; $in_i(\vec{x})$ and $in_{env}(\vec{x})$ are arbitrary for other arguments $\vec{x}$,
- $o_i(\langle[q]_{\sim_i}, \alpha_i\rangle, \langle\alpha_1, \ldots, \alpha_{i-1}, \alpha_{i+1}, \ldots, \alpha_k, q_{env}\rangle) = [o(q_{env}, \alpha_1, \ldots, \alpha_k)]_{\sim_i}$ for $1 \le i \le k$ and $\alpha_i \in d_i([q]_{\sim_i})$; $o_{env}(q, \langle\alpha_1, \ldots, \alpha_k\rangle) = o(q, \alpha_1, \ldots, \alpha_k)$; $o_i$ and $o_{env}$ are arbitrary for other arguments,
- $\Pi_i = \emptyset$ for $1 \le i \le k$, and $\Pi_{env} = \Pi$,
- $\pi_{env}(p) = \pi(p)$
- $Q = \{\langle[q]_{\sim_1}, \ldots, [q]_{\sim_k}, q\rangle : q \in St\}$

Let $M' = cegs(S^M, Q) = \langle\mathbb{A}\mathrm{gt}', St', Act', d', o', \Pi', \pi', \sim'_1, \ldots, \sim'_k\rangle$. We argue that $M$ and $M'$ are isomorphic by establishing a one-to-one correspondence between the respective sets of states, and showing that the other parts of the structures agree on corresponding states.

First we show that, for any $\hat{q}' = \langle[q']_{\sim_1}, \ldots, [q']_{\sim_k}, q'\rangle \in Q$ and any $\alpha = \langle\alpha_1, \ldots, \alpha_k\rangle$ such that $\alpha_i \in d'(i, \hat{q}')$, we have

$$o'(\hat{q}', \alpha) = \langle[q]_{\sim_1}, \ldots, [q]_{\sim_k}, q\rangle \text{ where } q = o(q', \alpha) \quad (1)$$

Let $\hat{q} = o'(\hat{q}', \alpha)$. Now, for any $i$: $input_i(\hat{q}', \alpha) = in_i([q']_{\sim_i}, out_1([q']_{\sim_1}, \alpha_1), ..., out_{i-1}([q']_{\sim_{i-1}}, \alpha_{i-1}), out_{i+1}([q']_{\sim_{i+1}}, \alpha_{i+1}), \ldots, out_k([q']_{\sim_k}, \alpha_k), out_{env}(q')) = in_i([q']_{\sim_i}, \alpha_1, \ldots, \alpha_{i-1}, \alpha_{i+1}, \ldots, \alpha_k, q') = \langle\alpha_1, \ldots, \alpha_{i-1}, \alpha_{i+1}, \ldots, \alpha_k, q'\rangle$. Similarly, we get that $input_{env}(\hat{q}', \alpha) = \langle\alpha_1, \ldots, \alpha_k\rangle$. Thus we get that $o'(\hat{q}', \alpha) = \langle o_1(\langle[q']_{\sim_1}, \alpha_1\rangle, input_1(\hat{q}', \alpha)), \ldots, o_k(\langle[q']_{\sim_k}, \alpha_k\rangle, input_k(\hat{q}', \alpha)), o_{env}(q', input_{env}(\hat{q}', \alpha))\rangle = \langle[o(q', \alpha_1, \ldots, \alpha_k)]_{\sim_1}, \ldots, [o(q', \alpha_1, \ldots, \alpha_k)]_{\sim_k}, o(q', \alpha_1, \ldots, \alpha_k)\rangle$. Thus, $\hat{q} = \langle[q]_{\sim_1}, \ldots, [q]_{\sim_k}, q\rangle$ for $q = o(q', \alpha_1, \ldots, \alpha_k)$, which completes the proof of (1).

We now argue that $St' = Q$. Clearly, $Q \subseteq St'$. Let $\hat{q} \in St'$; we must show that $\hat{q} \in Q$. The argument is on induction on the length of the least $o'$ path from $Q$ to $\hat{q}$. The base case, $\hat{q} \in Q$, is immediate. For the inductive step, $\hat{q} = o'(\hat{q}', \alpha)$ for some $\hat{q}' \in Q$, and then we have that $\hat{q} \in Q$ by (1). Thus, $St' = Q$.

Now we have a one-to-one correspondence between $St$ and $St'$: $r \in St$ corresponds to $\langle[r]_{\sim_1}, \ldots, [r]_{\sim_k}, r\rangle \in St'$. It remains to be shown that the other parts of the structures $M$ and $M'$ agree on corresponding states:

- $\mathbb{A}\mathrm{gt}' = \mathbb{A}\mathrm{gt}$,
- $Act' = Act$,
- $\Pi' = \bigcup_{i=1}^{k} \Pi_i \cup \Pi_{env} = \Pi$,
- For $p \in \Pi' = \Pi$: $\langle[q']_{\sim_1}, \ldots, [q']_{\sim_k}, q'\rangle \in \pi'(p)$ iff $q' \in \pi_{env}(p)$ iff $q' \in \pi(p)$ (same valuations at corresponding states),
- $d'(i, \langle[q']_{\sim_1}, \ldots, [q']_{\sim_k}, q'\rangle) = d_i([q']_{\sim_i}) = d(i, q)$,
- It follows immediately from (1), and the fact that $Q = St'$, that $o'(\langle[q']_{\sim_1}, \ldots, [q']_{\sim_k}, q'\rangle, \alpha) = \langle[r']_{\sim_1}, \ldots, [r']_{\sim_k}, r'\rangle$ iff $o(q', \alpha) = r'$ (transitions on the same joint action in corresponding states lead to corresponding states),
- $\langle[q']_{\sim_1}, \ldots, [q']_{\sim_k}, q'\rangle \sim'_i \langle[r']_{\sim_1}, \ldots, [r']_{\sim_k}, r'\rangle$ iff $[q']_{\sim_i} = [r']_{\sim_i}$ iff $q' \sim_i r'$ (the accessibility relations relate corresponding states), which completes the proof.

□

COROLLARY 4. *For every* CEGS $M$, *there is an* $\mathrm{ATL}_{ir}$*-equivalent* MIS $S$ *with initial states* $Q$, *that is, for every state* $q$ *in* $M$ *there is a state* $q'$ *in* $cegs(S, Q)$ *satisfying exactly the same* $\mathrm{ATL}_{ir}$ *formulae, and vice versa.*

PROPOSITION 5. *For every* CGS $M$, *there is a* MIS $S^M$ *and a set of global states* $Q$ *of* $S^M$ *such that* $cgs(S^M, Q)$ *is isomorphic to* $M$.

PROOF. Let $M = \langle\mathbb{A}\mathrm{gt}, St, Act, d, o, \Pi, \pi\rangle$ be given. Now, let $\hat{M} = \langle\mathbb{A}\mathrm{gt}, St, Act, d, o, \Pi, \pi, \sim_1, \ldots, \sim_k\rangle$ for some arbitrary accessibility relations $\sim_i$ over $St$. By Proposition 3, there exists a MIS $S^{\hat{M}}$ with global states $Q$ such that $\hat{M}' = cegs(S^{\hat{M}}, Q)$ is isomorphic to $\hat{M}$. Let $M'$ be the CGS obtained by removing the accessibility relations from $\hat{M}'$. Clearly, $M'$ is isomorphic to $M$. □

COROLLARY 6. *For every* CGS $M$, *there is an* ATL*-equivalent* MIS $S$ *with initial states* $Q$. *That is, for every state* $q$ *in* $M$ *there is a state* $q'$ *in* $cgs(S, Q)$ *satisfying exactly the same* ATL *formulae, and vice versa.*

PROPOSITION 7. *The local state spaces in a* MIS *are not always compact with respect to the underlying concurrent epistemic game structure.*

PROOF. Take a CEGS $M$ in which agent $i$ has always perfect information about the current global state of the system. When constructing a modular interpreted system $S$ such that $M = cegs(S, Q)$, we have that $St_i$ must be isomorphic with $St$. □

[6] This shows another difference between the environment and the agents: the environment does not possess knowledge.

[7] We say that two CEGS are isomorphic if they only differ in the names of states and/or actions.

The above property is a part of the interpreted systems heritage. The next proposition stems from the fact that explicit models (and interpreted systems) allow for intensive interaction between agents.

PROPOSITION 8. *The size of $\mathcal{I}n$ in $S$ is, in general, exponential with respect to the number of local states and local actions. This is the case even when epistemic relations are not relevant (i.e., when $S$ is taken as a representation of an ordinary* CGS*).*

PROOF. Consider a CGS $M$ with agents $\mathbb{A}\mathrm{gt} = \{1, ..., k\}$, global states $St = \prod_{i=1}^{k} \{q_0^i, ..., q_i^i\}$, and actions $Act = \{0, 1\}$, all enabled everywhere. The transition function is defined as $o(\langle q_{j_1}^1, ..., q_{j_k}^k \rangle, \alpha_1, ..., \alpha_k) = \langle q_{l_1}^1, ..., q_{l_k}^k \rangle$, where $l_i = (j_i + \alpha_1 + ... + \alpha_k) \mod i$. Note that $M$ can be represented as a modular interpreted system with succinct local state spaces $St_i = \{q_0^i, ..., q_i^i\}$. Still, the current actions of all agents are relevant to determine the resulting local transition of agent $i$. □

We will call items $\mathcal{I}n, out_i, in_i$ the *interaction layer* of a modular interpreted system $S$; the other elements of $S$ constitute the *local layer* of the MIS. In this paper we are ultimately interested in model checking complexity with respect to the size of the local layer. To this end, we will assume that the size of interaction layer is polynomial in the number of local states and actions. Note that, by Propositions 7 and 8, not every explicit model submits to compact representation with a MIS. Still, as we declared at the beginning of Section 4, we are mainly interested in a modeling framework for systems of loosely coupled components, where interaction is essential, but most processing is done locally anyway. More importantly, the framework of MIS allows for separating the interaction of agents from their local structure to a larger extent. Moreover, we can control and measure the complexity of each layer in a finer way than before. First, we can try to abstract from the complexity of a layer (e.g. like in this paper, by assuming that the other layer is kept within certain complexity bounds). Second, we can also measure separately the *interaction complexity of different agents*.

## 4.2 Modular Interpreted Systems vs. Simple Reactive Modules

In this section we show that simple reactive modules are (as we already suggested) a specific (and somewhat limited) implementation of modular interpreted systems. First, we define our (quite strong) notion of equivalence of representations.

DEFINITION 7. *Two representations are equivalent if they unfold to isomorphic concurrent epistemic game structures. They are* CGS*-equivalent if they unfold to the same* CGS*.*

PROPOSITION 9. *For any* SRML *there is a* CGS*-equivalent* MIS*.*

PROOF. Consider an SRML $R$ with $k$ modules and $n$ variables. We construct $S = \langle \mathbb{A}\mathrm{gt}, Act, \mathcal{I}n \rangle$ with $\mathbb{A}\mathrm{gt} = \{a_1, ..., a_k\}$, $Act = \{\top_1, ..., \top_n, \bot_1, ..., \bot_n\}$, and $\mathcal{I}n = \bigcup_{i=1}^{k} St_i \times St_i$ (the local state spaces $St_i$ will be defined in a moment). Let us assume without loss of generality that $ctr_i = \{x_1, ..., x_r\}$. Also, we consider all guarded commands of $i$ to be of type $\gamma_{i,\psi}^{\top} : \psi \rightsquigarrow x_i := \top$, or $\gamma_{i,\psi}^{\bot} : \psi \rightsquigarrow x_i := \bot$. Now, agent $a_i$ in $S$ has the following components: $St_i = \mathcal{P}(ctr_i)$ (i.e., local states of $a_i$ are valuations of variables controlled by $i$); $d_i(q_i) = \{\top_1, ..., \top_r, \bot_1, ..., \bot_r\}$; $out_i(q_i, \alpha) = \langle q_i, q_i \rangle$; $in_i(q_i, \langle q_1, q_1 \rangle, ..., \langle q_{i-1}, q_{i-1} \rangle, \langle q_{i+1}, q_{i+1} \rangle, \langle q_k, q_k \rangle) = \langle \{x_i \in ctr_i \mid \langle q_1, ..., q_k \rangle \models \bigvee_{\gamma_{i,\psi}^{\top}} \psi\}, \{x_i \in ctr_i \mid \langle q_1, ..., q_k \rangle \models \bigvee_{\gamma_{i,\psi}^{\bot}} \psi\} \rangle$. To define local transitions, we consider three cases. If $t = f = \varnothing$ (no update is enabled), then $o_i(q_i, \alpha, \langle t, f \rangle) = q_i$ for every action $\alpha$. If $t \neq \varnothing$, we take any arbitrary $\hat{x} \in t$, and define $o_i(q_i, \top_j, \langle t, f \rangle) = q_i \cup \{x_j\}$ if $x_j \in t$, and $q_i \cup \{\hat{x}\}$ otherwise; $o_i(q_i, \bot_j, \langle t, f \rangle) = q_i \setminus \{x_j\}$ if $x_j \in f$, and $q_i \cup \{\hat{x}\}$ otherwise. Moreover, if $t = \varnothing \neq f$, we take any arbitrary $\hat{x} \in f$, and define $o_i(q_i, \top_j, \langle t, f \rangle) = q_i \cup \{x_j\}$ if $x_j \in t$, and $q_i \setminus \{\hat{x}\}$ otherwise; $o_i(q_i, \bot_j, \langle t, f \rangle) = q_i \setminus \{x_j\}$ if $x_j \in f$, and $q_i \setminus \{\hat{x}\}$ otherwise. Finally, $\Pi_i = ctr_i$, and $q_i \in \pi_i(x_j)$ iff $x_j \in q_i$. □

The above construction shows that SRML have more compact representation of states than MIS: $r_i$ local variables of agent $i$ give rise to $2^{r_i}$ local states. In a way, reactive modules (both simple and "full") are two-level representations: first, the system is represented as a product of modules; next, each module can be seen as a product of its variables (together with their update operations). Note, however, that specification of updates with respect to a single variable in an SRML may require guarded commands of total length $O(2^{\sum_{i=1}^{k} r_i})$. Thus, the representation of transitions in SRML is (in the worst case) no more compact than in MIS, despite the two-level structure of SRML. We observe finally that MIS are more general, because in SRML the current actions of other agents have no influence on the outcome of agent $i$'s current action (although the outcome can be influenced by other agents' current local states).

## 4.3 Model Checking Modular Interpreted Systems

One of our main aims was to study the complexity of symbolic model checking ATL$_{ir}$ in a meaningful way. Following the reviewers' remarks, we state our complexity results only as conjectures. Preliminary proofs can be found in [14].

CONJECTURE 10. *Model checking* ATL *for modular interpreted systems is EXPTIME-complete.*

CONJECTURE 11. *Model checking* ATL$_{ir}$ *for the class of modular interpreted systems is PSPACE-complete.*

A summary of complexity results for model checking temporal and strategic logics (with and without epistemic component) is given in the table below. The table presents completeness results for various models and settings of input parameters. Symbols $n, k, m$ stand for the number of states, agents and transitions in an explicit model; $l$ is the length of the formula, and $n_{local}$ is the number of local states in a concurrent program or modular interpreted system. The new results, conjectured in this paper, are printed in italics. Note that the result for model checking ATL against modular interpreted systems is an extension of the result from [22].

| | $m, l$ | $n, k, l$ | $n_{local}, k, l$ |
|---|---|---|---|
| CTL | P [5] | P [5] | PSPACE [15] |
| CTLK | P [5, 8] | P [5, 8] | PSPACE [20] |
| ATL | P [3] | $\mathbf{\Delta_3^P}$ [12, 16] | *EXPTIME* |
| ATL$_{ir}$ | $\mathbf{\Delta_2^P}$ [21, 13] | $\mathbf{\Delta_3^P}$ [13] | *PSPACE* |

If we are right, then the results for ATL and ATL$_{ir}$ form an intriguing pattern. When we compare model checking agents with perfect vs. imperfect information, the first problem appears to be much easier against explicit models measured with the number of transitions; next, we get the same complexity class against explicit models measured with the number of states and agents; finally, model checking imperfect information turns out to be *easier* than model checking perfect information for modular interpreted systems. Why can it be so?

First, a MIS unfolds into CEGS and CGS in a different way. In the first case, the MIS is assumed to encode the epistemic relations explicitly (which makes it explode when we model agents with perfect, or almost perfect information). In the latter case, the epistemic

aspect is ignored, which gives some extra room for encoding the transition relation more efficiently. Another crucial factor is the number of available strategies (relative to the size of input parameters). The number of all strategies is exponential in the number of global states; for uniform strategies, there are usually much less of them but still exponentially many in general. Thus, the fact that perfect information strategies can be synthesized incrementally has a substantial impact on the complexity of the problem. However, measured in terms of *local states and agents*, the number of all strategies is *doubly exponential*, while there are "only" exponentially many uniform strategies – which settles the results in favor of imperfect information.

## 5. CONCLUSIONS

We have presented a new class of representations for open multi-agent systems. Our representations, called modular interpreted systems, are: *modular*, in the sense that components can be changed, replaced, removed or added, with as little changes to the whole representation as possible; more *compact* than traditional explicit representations; and *grounded*, in the sense that the correspondences between the primitives of the model and the entities being modeled are more immediate, giving a methodology for designing and implementing systems. We also conjecture that the complexity of model checking strategic ability for our representations is higher if we assume perfect information than if we assume imperfect information.

The solutions, proposed in this paper, are not necessarily perfect (for example, the "impression" functions $in_i$ seem to be the main source of non-modularity in MIS, and can be perhaps improved), but we believe them to be a step in the right direction. We also do not mean to claim that our representations should replace more elaborate modeling languages like Promela or reactive modules. We only suggest that there is a need for compact, modular and reasonably grounded models that are more expressive than concurrent (epistemic) programs, and still allow for easier theoretical analysis than reactive modules. We also suggest that MIS might be better suited for modeling simple multi-agent domains, especially for human-oriented (as opposed to computer-oriented) design.

## 6. ACKNOWLEDGMENTS

We thank the anonymous reviewers and Andrzej Tarlecki for their helpful remarks. Thomas Ågotnes' work on this paper was supported by grants 166525/V30 and 176853/S10 from the Research Council of Norway.

## 7. REFERENCES

[1] R. Alur and T. A. Henzinger. Reactive modules. *Formal Methods in System Design*, 15(1):7–48, 1999.

[2] R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time Temporal Logic. *Lecture Notes in Computer Science*, 1536:23–60, 1998.

[3] R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time Temporal Logic. *Journal of the ACM*, 49:672–713, 2002.

[4] R. Alur, T.A. Henzinger, F.Y.C. Mang, S. Qadeer, S.K. Rajamani, and S. Tasiran. MOCHA user manual. In *Proceedings of CAV'98*, volume 1427 of *Lecture Notes in Computer Science*, pages 521–525, 1998.

[5] E.M. Clarke, E.A. Emerson, and A.P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, 1986.

[6] E.A. Emerson and J.Y. Halpern. "sometimes" and "not never" revisited: On branching versus linear time temporal logic. In *Proceedings of the Annual ACM Symposium on Principles of Programming Languages*, pages 151–178, 1982.

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

[8] M. Franceschet, A. Montanari, and M. de Rijke. Model checking for combined logics. In *Proceedings of the 3rd International Conference on Temporal Logic (ICTL)*, 2000.

[9] V. Goranko and W. Jamroga. Comparing semantics of logics for multi-agent systems. *Synthese*, 139(2):241–280, 2004.

[10] J. Y. Halpern. Reasoning about knowledge: a survey. In D. M. Gabbay, C. J. Hogger, and J. A. Robinson, editors, *The Handbook of Logic in Artificial Intelligence and Logic Programming, Volume IV*, pages 1–34. Oxford University Press, 1995.

[11] J.Y. Halpern and R. Fagin. Modelling knowledge and action in distributed systems. *Distributed Computing*, 3(4):159–177, 1989.

[12] W. Jamroga and J. Dix. Do agents make model checking explode (computationally)? In M. Pěchouček, P. Petta, and L.Z. Varga, editors, *Proceedings of CEEMAS 2005*, volume 3690 of *Lecture Notes in Computer Science*, pages 398–407. Springer Verlag, 2005.

[13] W. Jamroga and J. Dix. Model checking abilities of agents: A closer look. Submitted, 2006.

[14] W. Jamroga and T. Ågotnes. Modular interpreted systems: A preliminary report. Technical Report IfI-06-15, Clausthal University of Technology, 2006.

[15] O. Kupferman, M.Y. Vardi, and P. Wolper. An automata-theoretic approach to branching-time model checking. *Journal of the ACM*, 47(2):312–360, 2000.

[16] F. Laroussinie, N. Markey, and G. Oreiby. Expressiveness and complexity of ATL. Technical Report LSV-06-03, CNRS & ENS Cachan, France, 2006.

[17] K.L. McMillan. *Symbolic Model Checking: An Approach to the State Explosion Problem*. Kluwer Academic Publishers, 1993.

[18] K.L. McMillan. Applying SAT methods in unbounded symbolic model checking. In *Proceedings of CAV'02*, volume 2404 of *Lecture Notes in Computer Science*, pages 250–264, 2002.

[19] W. Penczek and A. Lomuscio. Verifying epistemic properties of multi-agent systems via bounded model checking. In *Proceedings of AAMAS'03*, pages 209–216, New York, NY, USA, 2003. ACM Press.

[20] F. Raimondi and A. Lomuscio. The complexity of symbolic model checking temporal-epistemic logics. In L. Czaja, editor, *Proceedings of CS&P'05*, 2005.

[21] P. Y. Schobbens. Alternating-time logic with imperfect recall. *Electronic Notes in Theoretical Computer Science*, 85(2), 2004.

[22] W. van der Hoek, A. Lomuscio, and M. Wooldridge. On the complexity of practical ATL model checking. In P. Stone and G. Weiss, editors, *Proceedings of AAMAS'06*, pages 201–208, 2006.