2013 IEEE/WIC/ACM International Conferences on Web Intelligence (WI) and Intelligent Agent Technology (IAT)

An Approach for the Verification of Multi-Agent Systems by Formally Guided Simulations

Paulo Salem da Silva
University of São Paulo
Department of Computer Science
São Paulo – Brazil
salem@ime.usp.br

Ana C. V. de Melo
University of São Paulo
Department of Computer Science
São Paulo – Brazil
acvm@ime.usp.br

***Abstract*—Multi-Agent Systems (MASs) can be used to model human and animal societies, for the purpose of analyzing their properties by computational means. We propose a verification technique that investigates such MASs by means of guided simulations. This is achieved by modeling the evolutions of an MAS as a transition system (implicitly), and the property to be verified as another transition system (explicitly). The former is derived (on-the-fly) from a formal specification of the MAS's environment. The latter, which we call a *simulation purpose*, is used both to verify the property and to guide the simulation. In this way, only the states that are relevant for the property in question are actually simulated. Algorithmically, this corresponds to building a synchronous product of these two transitions systems on-the-fly and using it to operate a simulator. This paper presents an overall account of this approach, whose several parts were developed in a number of previous works. Our main objective here is to provide an overall account of the technique in a succinct manner, so that its most important and general features are highlighted. To clarify the theoretical discussions and show their practical importance, we develop concrete working examples along the text.**

***Keywords*—multi-agent systems; social simulation; on-the-fly verification; model-based testing; behaviorism; environments; social networks.**

I. Introduction

This paper presents an approach to the verification and exploration of multi-agent systems (MASs) through simulations. We have been developing it in a number of previous works (especially [1], [2], [3]), culminating in a doctoral thesis [4]. Here we do not pursue in minute detail the many parts that form the approach, for this can be found in those works. Rather, our purpose is to provide an overall and integrated account of our method, something that has not yet been properly done outside the large body of the thesis. To do so, we employ concrete examples from our previous works to motivate, unify and illustrate the theoretical discussions. The ideas presented here may have broad applications beyond our particular realizations.

Our concern is with MASs in the context of social simulation. We propose a verification technique that investigates such MASs by means of guided simulations. This is achieved by modeling the evolutions of an MAS as a transition system (implicitly), and the property to be verified as another transition system (explicitly). The former is derived (on-the-fly) from a formal specification of the MAS's environment. The latter, which we call a *simulation purpose*, is used both to verify the property and to guide the simulation. In this way, only the states that are relevant for the property in question are actually simulated. Algorithmically, this corresponds to building a synchronous product of these two transitions systems on-the-fly and using it to operate a simulator.

In order to calculate the transition system that models the evolution of the MAS, we impose the following structure to the MAS. Agents are considered as executable black-boxes which receive inputs (i.e., stimuli) and produce outputs (i.e., actions). The environment, however, is examined in more detail. It is given as a formal specification which has a corresponding implementation capable of both providing inputs to and receiving outputs of agents. To generate the transition system, the formal specification of the environment is used to determine the next possible states. The agent implementations, in turn, are executed (i.e., the agents are simulated) to actually build these next states. That is to say, the environments determine an over-approximation of the state-space (this is the formal aspect), whereas agents' actions make this approximation precise (this is the simulation aspect). Figure 1 shows the elements of the proposed method.

There has been much research in both simulation [5] and, increasingly, formal verification of MASs (e.g., MCMAS [6]). However, very few attempts exist in combining these two approaches. A notable exception is the work of Bosse *et al.* [7], in which linear-time properties can be checked over simulation traces *a posteriori*. In contrast, our method actually guides the simulation.

Though our approach is quite general, our motivation and examples are based on a particular kind of social models, namely, those following behaviorist[1] principles, which contrasts with the more dominant cognitive school of thought found in the MAS literature. From the verification point of view, this has the advantage of allowing us to concentrate the formal manipulations on the environments.

[1]In this paper, the term "behavioral" (and derivatives) refers to the psychological approach, not the Artificial Intelligence movement that goes by this name. These are only superficially related, it is unfortunate that the same term is traditionally used in both cases.

978-1-4799-2902-3/13 $31.00 © 2013 IEEE
DOI 10.1109/WI-IAT.2013.119

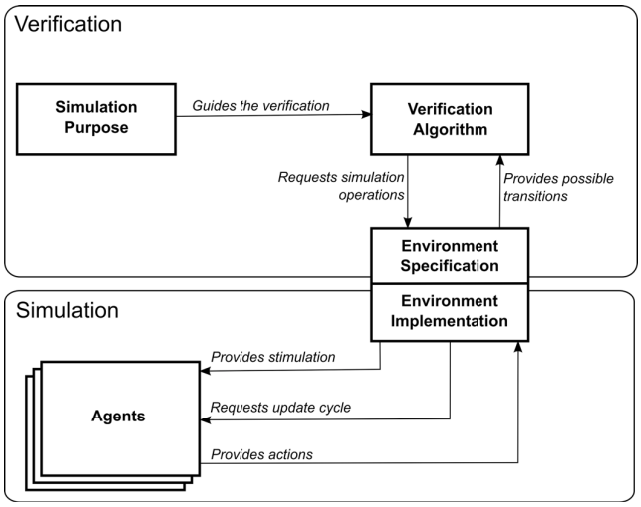

Fig. 1. Verification and simulation elements interaction. Notice, in particular, the important role that the environment has in relating verification and simulation. It acts as a coordinator which, on the one hand, formally defines what can be done, while on the other hand requests actual simulator operations.

We employ two working examples along this text. The first models a dog subject to experimentation, in the spirit of Pavlov's classical conditioning experiments (hereafter referred to as *Pavlovian Dog* example). It is a simple example designed to show the most basic features of the approach. The second defines a MAS formed by a teacher and several students (hereafter referred to as *School Children* example). The teacher wants the students to do their homework, but students may fail to do so owing to a number of environmental distractions (e.g., they may prefer to watch television instead). Both examples have actually been built and run in a proof-of-concept tool that we developed, called Formally Guided Simulator (FGS). As of this paper, FGS is also publicly available. [2]

This text is structured as follows. It begins by describing the important characteristics of agents (Section II) and environments (Section III), thereby establishing the MAS to be used. Next, the verification and exploration technique is presented (Section IV). Finally, we conclude (Section V). Related work is presented throughout the whole extent of the text, since in this manner it is easier to compare each of our developments with existing work.

II. Agents

The inner workings of agents, especially of so-called *intelligent* and *cognitive* agents, is often the focus of MAS research. These approaches center largely on what constitutes rational decisions, notably in the case of agents with limited computing capabilities. The *Beliefs-Desires-Intentions* (BDI) architecture [8] is a well-known example.

[2]Executables, source code (in Java) and full input examples (given as eXtensible Markup Language – XML – files) can currently be downloaded from `https://github.com/paulosalem/FGS`.The complete development of the examples, including the actual outputs, is given in [4]. Other similar examples are also available.

For the purpose of the technique presented here, however, such internal models are largely irrelevant. Detailed formal manipulations are reserved for the structure of the environment, and the behavior of agents is taken into account by simulating them as black-boxes which take inputs and produce outputs. So, as far as agent modeling is concerned, what counts is the *interaction* of the agent and its environment. Three factors are fundamental in this respect:

- The means by which the agent influences its environment (i.e., *actions*);
- The means by which the environment influences the agent (i.e., *stimuli*);
- The *autonomy* of the agent.

During verification and simulation, each agent is represented by an *agent profile* which specifies the actions it is capable of and the stimuli it can perceive, thereby establishing an interface that connects it to its environment.

Autonomy is a property usually ascribed to any agent. Informally, it denotes the fact that the agent is free to choose its actions. In our approach, this is formalized by assuming that, at any given state, any of the agent's actions may be emitted. This introduces a source of nondeterminism in the evolution of the MAS.

These general characteristics suffice to adopt the technique proposed here. Nevertheless, a concrete, executable, agent model is still necessary – something to run within the black-boxes. Since our examples were developed in the context of one such model, we briefly introduce it now.

A. Agent Architecture, Implementation and Simulation

We chose to develop and work with a behaviorist agent architecture [1], inspired by behavioral psychology, instead of the more popular cognitive ones. Our solution is called *Behaviorist Agent Architecture*. We formalized[3] (in the Z Notation [9]) and implemented (in Java) an agent architecture based on the Behavior Analysis theory of B. F. Skinner [10]. In this theory, the actions of agents (called "organisms") are seen as the result of past stimulation and certain innate parameters according to behavioral laws. The focus is not in mental qualities such as the nature of reason, but merely in the prediction and control of behavior by means of environmental stimulation.

Given the distinctive behaviorist point of view adopted, the problems we address differ considerably from those typically used in the MAS literature (e.g., auctions). Our examples are partly inspired by the literature on behaviorist psychology, but add innovations that arise from our particular computational approach (this will be clear in Section III).

[3]Even though the architecture is formalized, this formalization is not manipulated for the purpose of the verification technique presented in this paper. This is reserved for the formal description of environments. There are two main reason for this: (i) the formalization of agents is much more complex and thus more difficult to properly analyze by formal means; and (ii) in an abstract fashion, the global MAS evolution is more readily related to the environment, which permeates everything, than to each individual agent.

We start by showing some aspects of the agents considered in our two working examples. Owing to the limited space here, these and later specifications are only partial, tailored to illustrate particularly interesting points.

1) Example (Pavlovian Dog): The crucial point in classical conditioning is the creation of a relation between an initially neutral stimulus to a stimulus which already has value (i.e., a utility). Thus, the following is an excerpt of the necessary parametrization of the dog agent[4]:

$$
\begin{aligned}
&dog \in Organism\\
&\{food, injection, neutral, bark\_sound,\\
&\quad bell, whistle, veterinary\} \subset Stimulus_{dog}\\
&primaryStimuli_{dog} = \{food, injection, neutral,\\
&\quad bark\_sound\}\\
&primary\_utility_{dog}(food) = 0.9\\
&primary\_utility_{dog}(injection) = -0.6\\
&primary\_utility_{dog}(neutral) = neutral\_utility\\
&primary\_utility_{dog}(bark\_sound) = 0.1
\end{aligned}
$$

The above definitions specify an *organism* (i.e., the dog), a set of *stimuli* and set of *primary stimuli* which have predefined utilities (i.e., *primary utilities*). One possible experiment to perform with this dog would be to teach it that the stimulus *whistle* is always followed by the stimulus *food*, thereby indirectly assigning an utility to *whistle*.

2) Example (School Children): There are many agents in this example, but let us just examine some characteristics of a particular child, c_1. The child can perceive a number of possible stimuli, some of which have a primary utility. Importantly, he or she prefers to watch TV rather than to receive a prize from the teacher.

$$
\begin{aligned}
&c_1 \in Organism\\
&\{prize, disapproval, homework,\\
&\quad tv, cry\_sound\} \subset Stimulus_{c_1}\\
&primary\_utility_{c_1}(prize) = 0.5\\
&primary\_utility_{c_1}(disapproval) = -0.2\\
&primary\_utility_{c_1}(tv) = 0.6
\end{aligned}
$$

The child can also perform *actions*. However, some actions may *conflict*, which means that they cannot be performed both at the same time. For instance, *do_homework* and *watch_tv* conflict in this manner.

$$
\begin{aligned}
&\{do\_homework, study, watch\_tv, idle\} \subset Action_{c_1}\\
&conflict_{c_1}(do\_homework, watch\_tv) = conflicting
\end{aligned}
$$

III. Environments

In comparison with agents, environments of MASs have received little attention, as the survey of Weyns *et al.* [11] points out. However, environments can be of great use to the automated analysis of MASs. The reason is that they are often simple and amenable to formal descriptions, which in turn facilitates manipulating them and reasoning about their properties. In fact, the simplicity of environments may explain why they have historically received little attention.

To harness the potential of environments, we have developed the *Environment Model for Multi-Agent Systems (EMMAS)* in [2]. Its main features, which we present in this section, are the following:

- Non-determinism: it expresses several *possibilities* of how the MAS may evolve.
- It may define behaviors of the environment itself, in the form of *operations*, which can manipulate agents contained therein (this and the previous point are addressed in Section III-A).
- Agents exist within it as nodes in a social network, rather than as bodies in physical space. The environment mediates the relation between agents, and can also affect agents directly (Section III-B).
- It has a formal semantics in terms of transition systems. More specifically, EMMAS employs the π-calculus process algebra [12] to achieve this (Section III-C).

To facilitate reading, we use this color when writing EMMAS operations and tuples, and this color when writing sets in the context of an EMMAS specification.

A. Experimentation

An environment defines the context in which the agents exist, which is more than merely setting initial conditions. It includes behaviors pertaining to the environment itself, which may be executed either in response to an agent's actions or independently of any such action. This environmental context can be seen as an experimental setup, which defines all the possible experiments that can be performed in the agents. One may provide (in a process algebraic style) environment behaviors in EMMAS in order to define such an experimental setup. This works quite naturally with our behaviorist agent model, in which all behavioral phenomena can be defined in environmental terms. But, of course, such experimental environments can also be used with other kinds of agents.

From its very core, thus, EMMAS is concerned with the systematic exploration of possibilities (i.e., with formally-guided verifications). This focus contrasts with the other few existing approaches to environment formalization, such as [13], in which the main concerns are the definition and execution of MASs, and not their systematic analysis.

1) Example (Pavlovian Dog): There are a number of environment behaviors, as follows:

$$
\begin{aligned}
eb_1 = \;& ER(salivate, dog, NOP)\\
eb_2 = \;& ER(bark, dog, NOP)\\
eb_3 = \;& ER(push\_lever, dog, Stimulate(bell, dog))\\
eb_4 = \;& (Stimulate(bell, dog);\ Stimulate(food, dog)) +\\
& (Stimulate(whistle, dog);\ Stimulate(food, dog))
\end{aligned}
$$

Let us comment on the role of each of these:

[4] Although the actual formal specification originally used in [1], [4] is given in the Z Notation, here we use standard mathematical notation. This is possible because we do not develop the specification in detail in this text, and for the sake of clarity it is worth to translate it to a more readily understandable format.

- eb_1 and eb_2 define environment behaviors that react to the actions of *salivate* and *bark*, respectively (*ER* stands for environment response). These actions have no particular consequence in this environment, so the reaction is a do nothing operation (*NOP*). However, the actions will be relevant later, when we specify the simulation purpose to be checked. Though apparently pointless, eb_1 and eb_2 ensure that *salivate* and *bark* are possible in the environment, thus visible to the simulator.
- eb_3, on the other hand, is an environment behavior that actually does something. Whenever the dog pushes a lever, it gets stimulated with the sound of a bell. This can be understood as an apparatus available for the dog to manipulate.
- eb_4 has a more experimental role. It defines two alternative ways in which the environment can manipulate the dog, by means of the non-deterministic choice operator ($+$). In the first case, the *bell* stimulus is delivered, and later the *food* stimulus. In the second case, the *whistle* stimulus is delivered, and later the *food* stimulus. The objective of both is to try to condition either *bell* or *whistle* to *food*, a primary stimulus. The reason why these two alternatives are given here, and not merely one, is that the environment must express a range of possibilities. The actual experiments to be performed will be defined later by a simulation purpose, which will guide the choices among all the possibilities offered by the environment. The specification of multiple possibilities for the environment is an important contribution of our approach for the modeling and simulation of MASs.

2) Example (School Children): Here we have have three children, c_1, c_2 and c_3. A fragment of the environment behaviors to which they are subject is as follows:

$$\begin{aligned} eb_1 &= ER(study, c_1, Stimulate(information, c_1)) \\ eb_2 &= ER(study, c_2, Stimulate(information, c_2)) \\ eb_3 &= ER(study, c_3, Stimulate(information, c_3)) \\ eb_4 &= ER(watch\_tv, c_1, Stimulate(tv, c_1)) \\ eb_5 &= ER(watch\_tv, c_2, Stimulate(tv, c_2)) \\ eb_6 &= ER(watch\_tv, c_3, Stimulate(tv, c_3)) \end{aligned}$$

These behaviors allow students to either study or waste time watching television. This gives an opportunity for the children to procrastinate their homework. During simulation, this might interfere in the observed behaviors. Note that we assume that the children's environment is equal to all of them. This does not need to be the case, however. For instance, by suppressing eb_4 we would be defining that there is no TV available for c_1, even though c_1 would still be capable of emitting the *watch_tv* action.

In the environment, there is nothing that forbids a child from both studying and watching TV at the same time. However, recall that in Section II we defined that *study* and *watch_tv* conflict within the agent. Therefore, during simulation, it will never be the case that *study* and *watch_tv* are emitted at the same time by a child. This is an example of how the formal specification of the environment depends on the actual observation of agent actions (and thereby on the unobservable internal mechanisms of the agent) to generate the appropriate simulation.

B. Structure

In EMMAS, instead of representing the physical position of agents, we represent their relationships. That is to say, the MAS is viewed as a social network. Given the behaviorist point of view that we adopt, these relationships are modeled by defining how the actions of an agent are transformed in stimuli for other agents by means of *action transformers*. An agent is related to another if it can stimulate the other in this manner.

Even though a social network structure has technical advantages in certain circumstances (e.g., to model relationships), the fundamental point here is not that they are indispensable. Rather, it is simply that *some* well-defined structure is necessary so that it can be *formalized* and thus *systematically manipulated*. Our choice for social networks is justified by the fact that there exist convenient formalisms to denote both these networks and the operations of Section III-A. In particular, we have used the π-calculus process algebra to formalize EMMAS.

1) Example (Pavlovian Dog): In this simple example, the relation to other agents is not relevant at all, since there is only one agent. Nevertheless, this agent is represented as a lonely node in an otherwise empty social network. This node is given as an *agent profile*, which is a tuple composed by an identifier (in this case, 0), a set of stimuli and a set of actions. As far as the environment specification is concerned, this information is sufficient to interact with the agent.

$$\begin{aligned} S &= \{food, bell, whistle, injection, veterinary, \\ &\quad neutral, bark\_sound\} \\ A &= \{salivate, bark, sit, push\_lever\} \\ dog &= \langle 0, S, A \rangle \end{aligned}$$

2) Example (School Children): In this example we have two types of agents, namely, a teacher (denoted by t) and the children (denoted by c_1, c_2, c_3). The teacher has an agent profile that allows him to provide homework, prizes, and rewards to students, as well as to receive money as payment from the school. The children, in turn, can perceive the relevant stimuli and perform the necessary actions to interact with such a teacher, but are also capable of interacting both among themselves and with other elements (e.g., television).

$$\begin{aligned} S_t &= \{money, homework_1, homework_2, homework_3, \\ &\quad see\_annoying_1, see\_annoying_2, see\_annoying_3\} \\ A_t &= \{assign\_homework, reward_1, reward_2, reward_3, \\ &\quad punish_1, punish_2, punish_3\} \\ S_c &= \{prize, disapproval, homework, provocation, \\ &\quad information, tv, cry\_sound, neutral\} \\ A_c &= \{do\_homework, study, annoy, watch\_tv, cry, idle\} \end{aligned}$$

$$\begin{aligned} t &= \langle 0, S_t, A_t \rangle \\ c_1 &= \langle 1, S_c, A_c \rangle \\ c_2 &= \langle 2, S_c, A_c \rangle \\ c_3 &= \langle 3, S_c, A_c \rangle \end{aligned}$$

Owing to the authority of the teacher and to the configuration of the class, there is a fixed social network that allows the agents to interact in several ways. The links in this network are action transformers, which are tuples of the form $\langle ag_1, a, s, ag_2 \rangle$ defining that when ag_1 emits action a, agent ag_2 will receive stimulus s. Note that the same action may participate in multiple transformers. Here is a fragment of the specification for the set of action transformers.

$$
\begin{aligned}
&AT = \\
&\{\langle t, reward_1, prize, c_1\rangle, \langle t, reward_2, prize, c_2\rangle, \\
&\langle t, reward_3, prize, c_3\rangle, \\
&\\
&\langle t, punish_1, disapproval, c_1\rangle, \\
&\langle t, punish_2, disapproval, c_2\rangle, \\
&\langle t, punish_3, disapproval, c_3\rangle, \\
&\\
&\langle t, assign\_homework, homework, c_1\rangle, \\
&\langle t, assign\_homework, homework, c_2\rangle, \\
&\langle t, assign\_homework, homework, c_3\rangle, \\
&\\
&\langle c_1, do\_homework, homework, t\rangle, \\
&\langle c_2, do\_homework, homework, t\rangle, \\
&\langle c_3, do\_homework, homework, t\rangle, \\
&\\
&\langle c_1, annoy, provocation, c_2\rangle, \langle c_2, annoy, provocation, c_1\rangle, \\
&\langle c_1, annoy, provocation, c_3\rangle, \langle c_3, annoy, provocation, c_1\rangle, \\
&\langle c_2, annoy, provocation, c_3\rangle, \langle c_3, annoy, provocation, c_2\rangle, \\
&\\
&\langle c_1, annoy, see\_annoying_1, t\rangle, \\
&\langle c_2, annoy, see\_annoying_2, t\rangle, \\
&\langle c_3, annoy, see\_annoying_3, t\rangle, \\
&\\
&\ldots\}
\end{aligned}
$$

The intuitive meaning of these several action transformers can be grasped from their definition. Nonetheless, let us comment on some of them:

- $\langle t, reward_1, prize, c_1\rangle$ specifies that the teacher has the power to reward student c_1 with a prize. Other similar action transformers specify other powers with respect to the children, such as the power to punish and assign homework. These are formalizations of the social norm that dictates that the teacher has a certain authority over the students.
- $\langle c_1, annoy, provocation, c_2\rangle$ specifies that the child c_1 can annoy c_2, an act that is perceived as a provocation. This does not mean that c_1 will annoy c_2, but only that it has the possibility of doing so. This can be the case, for example, if c_1 and c_2 sit close to one another in the class. However, EMMAS abstracts any such physical location properties, and preserves only the logical relation between the agents.
- $\langle c_1, annoy, see\_annoying_1, t\rangle$ specifies that whenever c_1 annoys another child, the teacher will observe it. This is an example of how the same action can have more than one kind of consequence (i.e., annoy another child and allow the teacher to see this).

C. Formal Semantics

In order to allow the execution and systematic exploration of the environment constructs seen above, it is necessary to assign precise meanings to them. That is to say, they must have a formal semantics.

Ultimately, what we need is a way to transform the environment specification in a transition system that represents all the possible evolutions of the MAS. Each path in such a transition system represents one possible simulation of the MAS. As we shall see in Section IV-B, this is used to check whether certain properties hold for the MAS by systematically simulating the relevant paths.

More precisely, we employ an *annotated transition system (ATS)* to this end. Essentially, an ATS is tuple $\langle S, E, P, \rightarrow, L, s_0\rangle$ of states (S), events (E), propositions (P), a transition relation from states to states through events ($\rightarrow: S \times E \times S$), a labeling function that assign propositions to states and, finally, an initial state $s_0 \in S$.

In EMMAS, an ATS is built in two steps. First, the EMMAS specification is translated into a (large) π-calculus expression. We do not explore the details of this translation (nor of π-calculus) here (see [2], [4] for this). For our purposes, it suffices to point out that π-calculus:

- provides a way to express processes algebraically, with operators to specify non-deterministic choice and parallel composition, among others. This is very similar to EMMAS itself, which adds abstractions on top of such fundamental elements. For example, the EMMAS expression $(BeginStimulation(s_1, ag) + BeginStimulation(s_2, ag))$ is translated as $\overline{beginning^n_{s_1}}(x).\overline{stable^n_{s_1}}(x).\overline{done}(x) + \overline{beginning^n_{s_2}}(x).\overline{stable^n_{s_2}}(x).\overline{done}(x)$ in π-calculus (where n is the identifier assigned to agent ag, and x is an arbitrary free variable).
- defines channel-based communication between processes as the fundamental concept that allows agents to interact.
- has an operational semantics. This means that any π-calculus expression can be converted to a transition system.

The second step follows from this last point. Once the EMMAS specification is translated into π-calculus, an ATS is generated by employing the π-calculus' operational semantics, annotating it with some more information and finally pruning it using certain constraints. It is worth noting that a special event in this ATS, !*commit*, signals when it is appropriate to change the stimulation being transmitted to and check what actions are being emitted by the agents, as well as giving these agents a chance to modify their internal state. That is to say, whenever !*commit* is found during a simulation, the verification and the simulation aspects of the MAS interact (i.e., the rift in Figure 1 is surmounted).

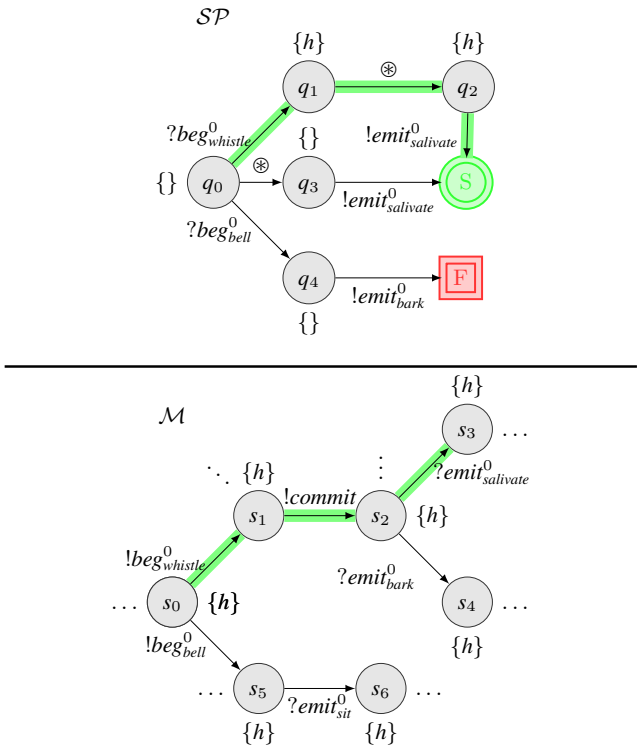

Fig. 2. In this example, the simulation purpose $\mathcal{SP}$ guides the simulation so that a whistle stimulus is delivered to an organism (a dog), then anything can happen, and finally the organism salivates. Only the shaded runs can synchronize to achieve success. States are annotated with propositions and transitions with events. The state labeled with S is `success`, and the one labeled with F is `failure`. The dots (...) denote that $\mathcal{M}$ continues beyond the states shown (it is possibly infinite).

IV. Verification and Exploration

One models an MAS in order to study its properties. We propose a way to do so by formulating hypotheses about the MAS and checking whether they hold or not (e.g., "every time the agent does X, will it do Y later?"). If a hypothesis does not hold, it means that either the hypothesis is false or the MAS has not been correctly specified. The judgement to be made depends of our objectives in each particular circumstance. Are we trying to discover some law about the MAS? In this case, if a hypothesis that represents this law turns out to be false, it is the hypothesis that is incorrect, not the MAS. Are we trying to engineer an MAS that obey some law? In this case we have the opposite, a falsified hypothesis indicates a problem in the MAS. Hence, the verification to be carried out is actually an experimentation procedure to learn about MAS models, not only to find implementation errors, as it is often the case with verification methods.

In what follows we examine how such hypotheses are formulated (Section IV-A) and then the algorithms actually used for verification and exploration (Section IV-B). Further details about both of these aspects, in the more general context of discrete-event simulation, can be found in [3] and [4].

A. Simulation Purposes and Satisfiability Relations

A hypothesis is defined by specifying a simulation purpose and a satisfiability relation. If the MAS satisfies the specified simulation purpose with respect to the desired satisfiability relation, than the hypothesis is corroborated. Otherwise, it is falsified. The idea of using such a simulation purpose is inspired by the TGV [14] approach to model-based testing, in which formal *test purposes* are used to select relevant test cases. Here, a formal simulation purpose is used to select relevant simulations executions, though of course the criteria of relevance, among other technicalities, are quite different.

Formally, a simulation purpose is an ATS subject to further restrictions. In particular, it is a finite transition system and defines two special states, `success` and `failure`. All runs that lead to `success` denote desirable simulations, whereas all that lead to `failure` denote undesirable ones.

The satisfiability relations, in turn, require the introduction of another technical definition, namely, the notion of *synchronous product*. Given an ATS $\mathcal{M}$ that models an MAS, and a simulation purpose $\mathcal{SP}$, their synchronous product (denoted by $\mathcal{SP} \otimes \mathcal{M}$) is another ATS in which every run represents a possible evolution of $\mathcal{M}$ which has been allowed by the $\mathcal{SP}$. Each state in $\mathcal{SP} \otimes \mathcal{M}$ takes the form of (q, s), where s is a state of $\mathcal{M}$, and q is a state of $\mathcal{SP}$. There are precise rules that determine when states and events synchronize. For example, an output event (e.g., $!n$) at the simulation level (i.e., $\mathcal{M}$) can synchronize with an input (e.g., $?n$) event with the same name (n) at the verification level (i.e., $\mathcal{SP}$). Figure 2 shows an example of $\mathcal{SP}$, $\mathcal{M}$ and runs that synchronize to form $\mathcal{SP} \otimes \mathcal{M}$.

These runs may terminate in a state (q, s) where $q =$ `success` or $q =$ `failure`, meaning that the run in question is desirable or undesirable, respectively. Different satisfiability relations are defined, namely:

- *Feasibility*: $\mathcal{SP}$ is feasible with respect to $\mathcal{M}$ if there is at least one run in $\mathcal{SP} \otimes \mathcal{M}$ which terminates in a state (q, s) such that $q =$ `success`. There are weak and strong variants of this.
- *Refutability*: $\mathcal{SP}$ is refutable with respect to $\mathcal{M}$ if there is at least one run in $\mathcal{SP} \otimes \mathcal{M}$ which terminates in a state (q, s) such that $q =$ `failure`. There are weak and strong variants of this.
- *Certainty*: $\mathcal{SP}$ is certain with respect to $\mathcal{M}$ if all runs in $\mathcal{SP} \otimes \mathcal{M}$ terminate in a state (q, s) such that $q =$ `success`.
- *Impossibility*: $\mathcal{SP}$ is impossible with respect to $\mathcal{M}$ if all runs in $\mathcal{SP} \otimes \mathcal{M}$ terminate in a state (q, s) such that $q =$ `failure`.

1) Example (Pavlovian Dog): Let us now define a simulation purpose that can perform some experiments in the dog in order to show its classical conditioning capabilities. We begin by defining the states (Q) and events (E) of the simulation purpose.

$$Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8, q_9, q_{10}, q_{11}, q_{12}, \texttt{success}, \texttt{failure}\}$$

$$E = \{!emit^0_{salivate}, !stop^0_{salivate}, !emit^0_{push\_lever}, ?beginning^0_{food}, ?stable^0_{food}, ?beginning^0_{whistle}, ?stable^0_{whistle}, ?beginning^0_{bell}, ?stable^0_{bell}, ?ending^0_{bell}, ?absent^0_{bell}\}$$

Events follow a peculiar naming convention. The prefix ! denote an event that is generated by an agent, while the prefix ? denotes an event that is generated by the environment. For each event, a superscript indicates the agent to which it applies and a subscript denotes either an action or a stimulus. The names *emit* and *stop* refer to the start and termination of an agent action, respectively. The names *absent*, *beginning*, *stable* and *ending* refer to the several stages through which a stimulation must pass.

The interesting thing now is to define, through the possible transitions, the experiments to perform. We will define two such experiments. Below, the $\circledast$ denotes a special event that can synchronize with any other (i.e., it specifies that any event found at this point is acceptable).

$$\begin{aligned}\rightsquigarrow = \{&(q_0, ?beginning^0_{whistle}, q_1), (q_1, \circledast, q_1),\\ &(q_1, ?stable^0_{whistle}, q_3), (q_3, \circledast, q_3),\\ &(q_3, ?beginning^0_{food}, q_4), (q_4, \circledast, q_4),\\ &(q_4, ?stable^0_{food}, q_4), (q_4, !emit^0_{salivate}, q_4),\\ &(q_4, ?beginning^0_{whistle}, \texttt{failure}),\\ &(q_4, !stop^0_{salivate}, q_5), (q_5, \circledast, q_5),\\ &(q_5, ?beginning^0_{whistle}, q_6), (q_6, \circledast, q_6),\\ &(q_6, !emit^0_{salivate}, \texttt{success}),\\ \\ &(q_0, ?beginning^0_{bell}, q_{10}), (q_{10}, ?stable^0_{bell}, q_{10}),\\ &(q_{10}, ?ending^0_{bell}, q_{10}), (q_{10}, ?absent^0_{bell}, q_{10}),\\ &(q_{10}, \circledast, q_{10}), (q_{10}, ?beginning^0_{whistle}, \texttt{failure}),\\ &(q_{10}, ?beginning^0_{food}, q_{12}), (q_{12}, \circledast, q_{12}),\\ &(q_{12}, !emit^0_{push\_lever}, \texttt{success})\}\end{aligned}$$

The first experiment consists in stimulating the dog with *whistle*, then giving it *food* later, and finally stimulating it again with *whistle* to check whether it emits a *salivate* action. If it does, it means that *whistle* was successfully conditioned to *food*.

In the second experiment we condition *bell*, instead of *whistle*, to *food*. It is assumed that the dog was already taught that action *push_lever* leads to stimulus *bell*. The experiment, then, consists in checking whether *push_lever* is emitted by the dog, which shows how stimulus conditioning influences learned behavior.

With all this we have the simulation purpose $\langle Q, E, \{\}, \rightsquigarrow, \{\}, q_0\rangle$.

2) Example (School Children): Let us build a simulation purpose to check whether homework is being done. Again, we define some states (Q) and events (E).

$$\begin{aligned}Q &= \{q_0, q_1, q_2, q_3, \texttt{success}, \texttt{failure}\}\\ E &= \{!emit^0_{assign\_homework}, !emit^1_{do\_homework}\\ &\quad !emit^2_{do\_homework}, !emit^3_{do\_homework}\}\end{aligned}$$

The transitions, in turn, are as follows. First we require the teacher to assign some homework. Then anything can happen, but in the end we require that at least one of the students actually do the homework.

$$\begin{aligned}\rightsquigarrow = \{&(q_0, \circledast, q_0),\\ &(q_0, !emit^0_{assign\_homework}, q_1), (q_1, \circledast, q_1),\\ \\ &(q_1, !emit^1_{do\_homework}, \texttt{success}),\\ &(q_1, !emit^2_{do\_homework}, \texttt{success}),\\ &(q_1, !emit^3_{do\_homework}, \texttt{success})\}\end{aligned}$$

Thus we have the simulation purpose $\langle Q, E, \{\}, \rightsquigarrow, \{\}, q_0\rangle$.

B. Verification Algorithms

The satisfiability relations define conditions that one may require of the MAS, but we still need algorithms to check whether they hold. Despite the fact that there are several satisfiability relations, their verification is carried out in a similar manner. All the algorithms have the following main characteristics:

- They perform a depth-first search on the synchronous product of $\mathcal{SP}$ and $\mathcal{M}$. This search has a maximum depth, $depth_{max}$;
- $\mathcal{SP} \otimes \mathcal{M}$ is computed on-the-fly (i.e., it is not computed *a priori*; rather, at each state, the algorithms calculate the next states necessary to continue), because $\mathcal{M}$ itself is obtained on-the-fly from: (i) the π-calculus expressions present in each of its states; and (ii) the relevant actions emitted by the simulated agents in such states.
- A simulator interface is assumed to exist. This is used to control the simulation execution, including the possibility of storing simulator states and backtracking to them later.
- The complexity in space is polynomial w.r.t. to the size of the environment and other parameters, and the complexity in time is exponential w.r.t. $depth_{max}$. The actual complexity formulas are rather intricate, but can be found in [4].
- Nevertheless, contrary to pure formal verification methods, our algorithms do not seek to explore the state-space exhaustively. Rather, they aim only at employing the available computational resources to explore as much as possible of the state-space in a systematic and automated manner.

What the search is looking for is what changes from one algorithm to another. In the case of feasibility, the objective is to find one *run* that leads to `success` (i.e., a *feasible run*), thus showing an example of how to perform a successful simulation. In certainty, it is to find one run that leads to `failure`, and therefore provides a counter-example (if no such run is found, it means that certainty holds w.r.t. to the available observations). The other satisfiability relations are checked analogously to these. Furthermore, if the verification of a satisfiability relation requires a search depth greater than $depth_{max}$, then the result of the algorithm is `inconclusive`, instead of `success` or `failure`.

1) Example (Pavlovian Dog): FGS verifies in two seconds that the simulation purpose is indeed weakly feasible. It outputted the verdict `success` and the following feasible run:

$$((q_0, s_0), ?\mathit{beginning}^0_{\mathit{bell}}, (q_{10}, s_1), \circledast, (q_{10}, s_2), \circledast, (q_{10}, s_3), \circledast, (q_{10}, s_4), \circledast, (q_{10}, s_5), \circledast, (q_{10}, s_6), \circledast, (q_{10}, s_7), \circledast, (q_{10}, s_8), \circledast, (q_{10}, s_9), \circledast, (q_{10}, s_{10}), \circledast, (q_{10}, s_{11}), \circledast, (q_{10}, s_{12}), \circledast, (q_{10}, s_{13}), ?\mathit{ending}^0_{\mathit{bell}}, (q_{10}, s_{14}), \circledast, (q_{10}, s_{15}), \circledast, (q_{10}, s_{16}), \circledast, (q_{10}, s_{17}), \circledast, (q_{10}, s_{18}), \circledast, (q_{10}, s_{19}), ?\mathit{absent}^0_{\mathit{bell}}, (q_{10}, s_{20}), \circledast, (q_{10}, s_{21}), \circledast, (q_{10}, s_{22}), \circledast, (q_{10}, s_{23}), \circledast, (q_{10}, s_{24}), ?\mathit{beginning}^0_{\mathit{food}}, (q_{12}, s_{25}), \circledast, (q_{12}, s_{26}), \circledast, (q_{12}, s_{27}), !\mathit{emit}^0_{\mathit{push\_lever}}, (\texttt{success}, s_{28}))$$

The above run shows how the *bell* stimuli was delivered to agent 0, and how eventually the agent emits the action *push_lever*.

2) Example (School Children): FGS took six seconds to check weak feasibility. In the feasible run found, it determined that at least agent identified by 1 did its assigned homework. In the process of doing so, other events took place, such as children annoying each other and crying. These, however, did not prevent that agent from doing the homework. The feasible run found is the following:

$$((q_0, s_0), \circledast, (q_0, s_1), \circledast, (q_0, s_2), \circledast, (q_0, s_3), \circledast, (q_0, s_4), !\mathit{emit}^0_{\mathit{assign\_homework}}, (q_1, s_5), \circledast, (q_1, s_6), \circledast, (q_1, s_7), \circledast, (q_1, s_8), \circledast, (q_1, s_9), !\mathit{emit}^1_{\mathit{do\_homework}}, (\texttt{success}, s_{10}))$$

V. Conclusion

In this paper we presented the foundations of a framework for the verification and exploration of MASs. This framework is based on formal descriptions of agent interfaces and their environments, as well as formal definitions of simulation purposes. These elements are used to perform simulations in a systematic and guided manner in order to determine whether certain precisely defined satisfiability relations hold or not. It is thus a significant step forward in bridging the gap between simulation and verification techniques.

Nonetheless, the approach has certain disadvantages too. Notably, we observed problems with respect to scalability. When translating the specification of an environment in π-calculus (or any similar formalism), one gets very large expressions. Owing to the nature of the calculus, calculations which are quadratic to the size of the expression have to be performed at each simulation step. Our implementation of the π-calculus includes a number of optimizations to address this issue (e.g., caching previous calculations), but it is not clear to what extent one may carry out such a strategy.

Our realization of the proposed method is based on a particular (behaviorist) agent architecture, as well as a custom simulator. However, as we emphasized in the text, the fundamental aspects of the technique can be applied to other cases – Figure 1 offers the general blueprint. We do hope that our FGS will be but one among many formally guided simulators yet to come.

Acknowledgements

The authors would like to thank Prof. Dr. Marie-Claude Gaudel, from University Paris-Sud, for crucial suggestions that led to the present work.

This project benefited from the financial support of *Coordenação de Aperfeiçoamento de Pessoal de Nível Superior* (CAPES) and *Conselho Nacional de Desenvolvimento Científico e Tecnológico* (CNPq).

References

[1] P. S. da Silva and A. C. V. de Melo, "A simulation-oriented formalization for a psychological theory," in *FASE 2007 Proceedings*, ser. Lecture Notes in Computer Science, M. B. Dwyer and A. Lopes, Eds., vol. 4422. Springer-Verlag, 2007, pp. 42–56.

[2] ——, "A formal environment model for multi-agent systems," in *Formal Methods: Foundations and Applications*, ser. Lecture Notes in Computer Science, J. Davies, L. Silva, and A. Simao, Eds. Springer Berlin / Heidelberg, 2011, vol. 6527, pp. 64–79.

[3] ——, "On-the-fly verification of discrete event simulations by means of simulation purposes," in *Proceedings of the 2011 Spring Simulation Multiconference (SpringSim'11)*. The Society for Modeling and Simulation International, 2011.

[4] P. S. da Silva, "Verification of behaviourist multi-agent systems by means of formally guided simulations," Ph.D. dissertation, Universidade de São Paulo and Université Paris-Sud 11, November 2011, joint thesis between the two universities. Can be downloaded at: `http://tel.archives-ouvertes.fr/tel-00656809`.

[5] N. Gilbert and S. Bankers, "Platforms and methods for agent-based modeling," *Proceedings of the National Academy of Sciences of the United States*, vol. 99, no. Supplement 3, 2002.

[6] A. Lomuscio, H. Qu, and F. Raimondi, "MCMAS: A model checker for the verification of multi-agent systems," in *Computer Aided Verification*, ser. Lecture Notes in Computer Science, A. Bouajjani and O. Maler, Eds. Springer Berlin / Heidelberg, 2009, vol. 5643, pp. 682–688.

[7] T. Bosse, C. M. Jonker, L. van der Meij, A. Sharpanskykh, and J. Treur, "Specification and verification of dynamics in agent models," *Int. J. Cooperative Inf. Syst.*, vol. 18, no. 1, pp. 167–193, 2009.

[8] A. S. Rao and M. P. Georgeff, "BDI agents: From theory to practice," in *ICMAS*, V. R. Lesser and L. Gasser, Eds. The MIT Press, 1995, pp. 312–319.

[9] J. Woodcock and J. Davies, *Using Z: Specification, Refinement, and Proof*. Prentice Hall, 1996.

[10] B. F. Skinner, *Science and Human Behavior*. The Free Press, 1953.

[11] D. Weyns, H. V. D. Parunak, F. Michel, T. Holvoet, and J. Ferber, "Environments for multiagent systems: State-of-the-art and research challenges," in *Proceedings of the 1st International Workshop on Environments for Multi-agent Systems (Lecture Notes in Computer Science, 3374)*, D. W. *et al.*, Ed. Springer, 2005, pp. 1–47.

[12] R. Milner, *Communicating and Mobile Systems: the π-Calculus*. Cambridge University Press, 1999.

[13] J. Ferber and J.-P. Müller, "Influences and reactions: a model of situated multiagent systems," in *Proceedings of ICMAS'96 (International Conference on Multi-Agent Systems)*. AAAI Press, 1996.

[14] C. Jard and T. Jéron, "TGV: theory, principles and algorithms: A tool for the automatic synthesis of conformance test cases for non-deterministic reactive systems," *Int. J. Softw. Tools Technol. Transf.*, vol. 7, no. 4, pp. 297–315, 2005.