Checking Multi-Agent Systems Behavior Properties

Michael Dekhtyar*, Alexander Dikovsky† and Mars Valiev‡

* Dept. of CS, Tver State Univ. Tver, Russia
Email: Michael.Dekhtyar@tversu.ru
†IRIN, Université de Nantes, France
Email: Alexandre.Dikovsky@irin.univ-nantes.fr
and Keldysh Inst. for Appl. Math., Moscow, Russia
Email: dikovsky@spp.keldysh.ru
‡ Keldysh Inst. for Appl. Math., Moscow, Russia
Email: valiev@spp.keldysh.ru

***Abstract*—The complexity of checking multi-agent systems behavior properties is studied. The behavior properties are formulated using first order temporal logic languages and are checked relative to the state transition systems determined by the multi-agent system definition. Various tight complexity bounds of the behavior properties are established under natural structural and semantic restrictions on agent programs and actions. Several interesting deterministic or nondeterministic polynomial time problem classes are established.**

I. Introduction

In this paper, we study the complexity of checking behavior properties of intelligent agents systems.

We address the reader to the book [12] and several publications [14], [11], [10] discussing different readings and definitions of the terms. However, the intuitive appeal of the term 'Intelligent Agent' is quite clear :

- an IA is *autonomous,* i.e. it can function by itself in predetermined environments;
- it is *reactive,* i.e. it is capable of perceiving and responding to the stimuli of other agents or of its medium;
- it is *intelligent,* i.e. its actions are determined by a certain logic and estimation of its environment;
- and it is *goal oriented,* i.e. even if its functioning is continuous, it is oriented on reaching states with some predetermined properties.

Although Agents and Multi-Agent Systems (MA-systems) are a very important area of Artificial Intelligence since the early 80ies, the research into their behavior properties is yet rather scarce (see [3], [4], [12], [8]).

Basically, the MA-systems we consider conform to the so called IMPACT architecture introduced and described in detail in the book [12]. The IMPACT architecture is very elaborated. It includes rather expressive specification means and control structures, e.g. adaptive action bases, logic programs articulating decision policies, constraints, beliefs based meta-reasoning about other agents, uncertainty reasoning, reasoning about time, communication management, security related structures, interfacing, and some other facilities (see its comparison with other architectures in [5]). The IMPACT semantics is described in terms of transitions between the agents' states and is shown in [12] to be untractable in general. In order to arrive at a polynomial time computable transition semantics, the authors of [12] impose very complex limitations on the agents features. As a result, the definition of such "polynomial" agents becomes bulky.

This work was sponsored by the Russian Fundamental Studies Foundation (Grants 01-01-00278 and 02-01-00652).

In this paper, we propose rather simply formulated limitations leading to a polynomial time semantics. We focus only on the agents' actions, decision policies and communication. In particular, we do not consider security, metaknowledge, temporal and uncertainty reasoning and some other features. Moreover, we choose relational databases as agent's data structure (in IMPACT a more general structure is allowed), and consider conventional logic programs in the place of deontic programs of the IMPACT agents. Even after these simplifications, the MA-system architecture remains very rich. So we study the behavior properties under various more or less restrictive constraints on MA-system's parameters and semantics.

We distinguish deterministic and nondeterministic semantics of MA-systems. Their behavior properties are expressed as the properties of trajectories (paths) in the state transition diagrams they induce. This allows the use of classical temporal logics: $PLTL, CTL, CTL^*$ [6], μ-calculus [9] and their variants as behavior properties description languages. The *"MA-BEHAVIOR"* problem we consider is a model checking problem consisting in verifying that a temporal logic formula Φ holds on the tree of trajectories of a given MA-system. Model checking on explicitly presented transition diagrams has been extensively studied since the early 80ies (see [6], [7], [2], [13]).

The novelty of our approach is that we consider diagrams with *structured* states and transitions specified by MA-systems, and we analyze the checking complexity under various constraints on the MA-system parameters and semantics. MA-systems constitute a compact representation of the corresponding transition system. For example, even for a ground (i.e. variable-free) MA-system A, the transition system $T(A)$ describing its trajectories may have the size exponential in $|A|$, because it can occur that it has $\boldsymbol{O}(2^{|A|})$ states. So sometimes, our lower bounds are more pessimistic as compared with the classical ones for the same classes of logics. The corresponding upper bounds either are more precise (in the case of polynomial time and space complexity), or they are translated from the corresponding classical results taking into consideration the size and the number of MA-system states. We establish some interesting problem classes in which the MA-BEHAVIOR problem turns out to be decidable in deterministic or nondeterministic polynomial time. And this is due to a new possibility of formulating natural constraints in terms of semantics and structural parameters of MA-systems.

The paper is organized as follows. In section 2, the architecture of MA-systems is defined and illustrated by a rather elaborated example of an MA-system. In section 3, the temporal logics we use are briefly discussed. In sections 4 – 7, the complexity results are presented concerning respectively ground deterministic, nonground deterministic, ground nondeterministic, and nonground nondeterministic MA-systems.

II. Agent and MA-system architecture

A. A simplified IMPACT architecture and semantics

A *multi-agent system* (*MA-system*) $\mathcal{A}$ is a finite set $\{a_1, ..., a_n\}$ of *intelligent agents* sharing the same signature. An intelligent agent a, as it shows up in this paper, has its *internal data base (DB)* I_a, which is a finite set of ground atoms. It communicates with the other agents in the system through its message box $MsgBox_a$. The messages are also ground atoms. The internal DB and the current message box contents make up the agent's current *local state* $IM_a = (I_a, MsgBox_a)$. The set of local states $\{IM_a \mid a \in \mathcal{A}\}$ makes up the *state* of the MA-system.

Each agent is capable of performing a number of parameterized actions constituting its action base AB_a. An action can be seen as an update of the agent's internal DB (defined by a finite list ADD_a of insertions and a finite list DEL_a of deletions of facts) accompanied by a finite list $SEND_a$ of messages to be loaded into the message boxes of recipient agents. Each action α is specified by an *action atom* p_α playing the role of a pattern for the action application. p_α can have variables in common with the action updates and the action messages in the three lists. For example, in the action

$\alpha = (check\_rel(X,Y),$
$\quad ADD([table1(X,1), table2(X,Y)]),$
$\quad DEL([table1(a_1,Y)]),$
$\quad SEND([(a_1, fax(X,Y,a))]))$
$p_\alpha = check\_rel(X,Y).$

Given a set GA of ground admissible action atoms (defined by the semantics below), each action $\alpha \in AB_a$ uniquely defines the set of admissible ground unifiers σ such that $p_\alpha\sigma \in GA$. As a result, GA uniquely determines the change in the local state $IM_a = (I_a, MsgBox_a)$ resulting from matching actions in AB_a with GA. Namely, the new internal DB $EFF_a(GA, I_a)$ is the result of:

(i) deleting from I_a all instances $\beta\sigma$ of atoms $\beta \in DEL_\alpha$, $\alpha \in AB_a$, such that $p_\alpha\sigma \in GA$,

(ii) and then adding to I_a all instances $\delta\sigma$ of atoms $\delta \in ADD_\alpha$, $\alpha \in AB_a$, such that $p_\alpha\sigma \in GA$.

For example, suppose that matching of the action α above against the current set GA of ground admissible actions defines two admissible unifiers: $\{X \mapsto 0.5, Y \mapsto 3\}$ and $\{X \mapsto 1, Y \mapsto 0\}$. Then the impact of α is that the atoms $table1(a_1, 3), table1(a_1, 0)$ will be deleted from I_a and then the atoms $table1(0.5, 1)$, $table2(0.5, 3), table1(1, 1), table2(1, 0)$ will be added to it.

The parallel effect on message boxes consists in creating for each agent $b \neq a$ the set of messages from a to b, i.e. the set $EFF_a(GA, MsgBox_b)$ of all message instances $\mu\sigma$, where $\mu \in SEND_\alpha$ is addressed to b and $p_\alpha\sigma \in GA$.

In our example, the messages $fax(0.5, 3, a), fax(1, 0, a)$ will be sent to the agent a_1.

An action α is *expanding* if DEL_α is empty.

The policy of behavior of an agent a in its current local state IM_a is determined by its *program* P_a. The intension of P_a in a given local state serves to determine the set of admissible actions GA_a, which itself serves to change this local state through the actions in AB_a as we have explained above.

P_a is a logic program with the clauses of the form $H \leftarrow L_1, ..., L_n$, where $n \geq 0$, the head $H = \alpha(t_1, ..., t_l)$ is the pattern action atom of an action in AB_a, the literals L_i in its body are either action literals, or (extensional) internal DB literals, or atoms of the form $Received(Source\_agent, Message)$, or their negations $\neg Received(Source\_agent, Message)$, or else built-in predicate calls $q(\bar{t})$[1].

[1] We adopt a domain closure assumption fixing some finite set

An agent's program is *positive* if there are no negations in its clauses. An agent with positive program is also called *positive*.

We suppose that the program clauses are *safe* in the sense that all variables in the head H occur *positively* in the body $L_1, ..., L_n$, and that the program $P_a^{state} = P_a \cup \{p \leftarrow \mid p \in I_a\} \cup \{Received(Agent\_source, Message) \leftarrow \mid (Agent\_source, Message) \in MsgBox_a\}$ is *stratified* [1].

We consider different agent's one-step semantics and respectively different intentional admissible action sets GA_a. All these semantics are defined through the standard minimal model program semantics $Sem(P_a)(I_a, MsgBox_a)$ of P_a in the given local state. $Sem(P_a)(I_a, MsgBox_a)$ is the set M_a^{act} of ground *action atoms* in the minimal model M_a^{state} of P_a^{state}. As it is well known [1], this model is unique for the stratified logic programs and is computed by a polynomial time fixpoint computation procedure from the *groundization* $gr(P_a^{state})$ of the program P_a^{state} [2].

In other words, the semantics of P_a in the given local state is the set of ground actions implied by the program P_a^{state} and available for execution in the agent's current local state.

The agent's one-step semantics should choose (or guess) in polynomial time the set $GA_a \subseteq M_a^{act}$ of intended admissible actions in the set of all available actions M_a^{act}. So it is defined by an operator $Act_a(M)$ applied to a given set of available actions $M = M_a^{act}$. This semantics is *deterministic* if it is a function from M to M, and *nondeterministic* if it is a binary relation on M.

For instance, the identity operator $Act^{td}(M) = M$ is an example of a deterministic one-step semantics (called *total*). We can imagine other types of deterministic one-step semantics, e.g. priority driven deterministic semantics which presumes some partial order $\prec$ on ground actions and is defined by $Act^{\prec d}(M) = \{m \in M \mid \neg\exists m' \in M\ (m' \prec m)\}$.

The simplest nondeterministic one-step semantics is the *unit choice* one-step semantics defined by $Act^{un}(M) = \{\{p\} \mid p \in M\}$. It guesses some available action in M. Another example is the *spontaneous* one-step semantics defined by $Act^{sn}(M) = \{M' \mid M' \subseteq M\}$. It guesses any subset of available actions in M.

of constants **C** denoting the domain objects and considering a set **Π** of polynomial time computable built-in predicates and operations (e.g., the standard arithmetical operations over numbers).

[2] I.e. from the set of all ground instances of clauses in P_a^{state}. It should be noted that the size of $gr(P_a^{state})$ can be exponential with respect to the size of P_a^{state}. We remind that the domain closure assumption we have adopted includes the requirement of polynomial time calculability of the built-in predicates. So the polynomial time complexity of the fixed point computation is preserved.

Deterministic agents are those having a deterministic one-step semantics. An agent with a nondeterministic one-step semantics is *nondeterministic*

A particular one-step semantics operator Act_a being chosen, it uniquely defines the new local state of a in the way described above.

The *one step semantics* of the MA-system $\mathcal{A}$ induces a one step transition relation $\Rightarrow_{\mathcal{A}}$ on the set $\mathcal{S}_{\mathcal{A}}$ of global states of the form $S =< (I_{a_1}, MsgBox_{a_1}), \ldots, (I_{a_n}, MsgBox_{a_n}) >$. The transition $S \Rightarrow_{\mathcal{A}} S'$ starts by calculating the sets of available actions $M_i^{act} = Sem(P_{a_i})(I_{a_i}, MsgBox_{a_i})$ for all agents $a_i \in \mathcal{A}$. Next, each agent's one-step semantics Act_{a_i} implemented by a deterministic or nondeterministic operator creates *admissible action sets* $GA_{a_i} = Act_{a_i}(M_i^{act})$. The message boxes of all agents in $\mathcal{A}$ are emptied thereafter (so the messages in S are forgotten). Then each agent's internal DB state I_{a_i} is replaced by $EFF_{a_i}(GA_{a_i}, I_{a_i})$ and the message box of each agent a_i is initiated by $\bigcup_{j \neq i} EFF_{a_j}(GA_{a_j}, MsgBox_{a_i})$.

B. Classes of MA-systems

We distinguish two main classes of MA-systems: deterministic and nondeterministic. A MA-system $\mathcal{A}$ is *deterministic* if all its agents are deterministic, otherwise it is *nondeterministic*.
In both classes of MA-systems, we consider the following subclasses induced by natural constraints imposed on agents' components. A MA-system $\mathcal{A} = \{a_1, ..., a_n\}$ is
- *ground* if each program P_{a_i} is ground[3];
- *k-dimensional* if the arities of the action atoms and of the message atoms are bounded by k (*dimension-bounded*, if *k-dimensional* for some k). In fact, this property fixes the maximal number of parameters involved in the actions and in the messages of $\mathcal{A}$;
- *expanding* if all its agents are *expanding*;
- *positive* if all its agents are *positive*;
- *m-agent* if $n \leq m$.
- *r-signal* if there are at most r different ground message atoms (*signals*).

The following proposition characterizes the complexity of the MA-system's one step semantics under some of these restrictions.

Proposition 1:
(1) For a deterministic MA-system $\mathcal{A}$, the transition function $S \Rightarrow_{\mathcal{A}} S'$ is computable in polynomial time with respect to $|S| + |\mathcal{A}| + |S'|$ if $\mathcal{A}$ is ground or dimension bounded, and is computable in deterministic

[3] I.e., all its clauses are ground.

exponential time [4] in the general nonground case.
(2) For each nondeterministic MA-system $\mathcal{A}$, the transition relation $S \Rightarrow_{\mathcal{A}} S'$ is recognizable in nondeterministic polynomial time with respect to $|S|+|\mathcal{A}|+|S'|$ if $\mathcal{A}$ is ground or dimension bounded, and is recognizable in nondeterministic exponential time in the general nonground case.

C. MA-System Behavior

We consider the behavior of MA-systems in an initial global state with empty message boxes. For a MA-system $\mathcal{A}$, its behavior in some initial global state $S^0 =< (I^0_{a_1}, MsgBox^0_{a_1}), ..., (I^0_{a_n}, MsgBox^0_{a_n}) >$, where $MsgBox^0_{a_i} = \emptyset,\ 1 \le i \le n,$ can be seen as the set $\mathcal{T} = \mathcal{T}_{\mathcal{A}}(S_0)$ of infinite trajectories (i.e. sequences of global states) of the form:

$\tau = (S^0 \Rightarrow_{\mathcal{A}} S^1 \Rightarrow_{\mathcal{A}} \ldots \Rightarrow_{\mathcal{A}} S^t \Rightarrow_{\mathcal{A}} S^{t+1} \Rightarrow_{\mathcal{A}} \ldots)$.

For a deterministic MA-system $\mathcal{A}$, $\mathcal{T}$ consists of a single trajectory starting in S^0. If $\mathcal{A}$ is nondeterministic, then $\mathcal{T}$ is an infinite tree of trajectories with the root node S^0. The nodes of $\mathcal{T}$ are the global states $S \in \mathcal{S}_{\mathcal{A}}$ accessible from S^0 by the reflexive-transitive closure of $\Rightarrow_{\mathcal{A}}$. If S is a node of $\mathcal{T}$, then the states in $Next_{\mathcal{A}}(S)$ are its immediate successors in $\mathcal{T}$. An infinite branch of $\mathcal{T}$ starting in some state is a *trajectory* in $\mathcal{T}$.

Let us see the following example.

Example. Recruiting committee (RC) of a Computer Science Institute. consists of 5 agents-members $m_i, 0 \le i \le 4$ (m_0 being the institute director) and the agent-secretary s. At any recruitment cycle, a number of agents-candidates c_j arrive. Agents-members access (in read mode) their common database GB. GB contains the facts $cand(C, Profile, Merits, Grants)$ describing the candidates. C stands for the candidates identification. $Profile$ stands for the candidates activity profile. In our example, the profiles are: lp (logic programming), ai (artificial intelligence), cl (computational linguistics) and mm (multimedia). An integer $Merits$ evaluates the candidates scientific merits. An integer $Grants$ represents the sum of the grants the candidate has obtained. Above the candidates data, GB also includes the information about the staff member selected in the preceding recruitment: $selected(C, Profile)$.

The candidates can only send their candidatures to the secretary. The candidates are invited by the secretary to take part in the competition. On receipt of candidatures, the secretary announces the recruitment session and deletes the flag *close* from GB. Before the vote, the members may speak out on the candidates (through messages). Then they vote by sending their secrete vote messages to the secretary: the selected candidate or the abstention. On receiving all members' votes, the secretary updates the GB according to the score: places the data of the one selected by the majority, if any (when no majority, no candidate is selected) and closes the session putting the flag *close* into GB. On the next step, secretary deletes the nonselected candidates data and enters the initial state of new candidatures wait.

The RC members in our example have different vote tactics but all of them abstain when their criteria do not ensure the uniqueness of choice:

$\mathbf{m_0}$ votes according to the following profile preferences: $lp > ai > mm > cl$, selects a candidate with the best grants sum and announces his choice to all RC members;

[4] In fact, polynomial time with respect to the groundization size.

$\mathbf{m_1}$ always votes for the candidate of m_0;
$\mathbf{m_2}$ selects a candidate with a profile different from that chosen by m_0 and having the best grants sum;
$\mathbf{m_3}$ votes for the candidate with the best scientific merits whose profile is different from that of the last recruited candidate;
$\mathbf{m_4}$ votes with the majority, if any; if there is no majority, then m_4 chooses the "antiboss" party candidate.

The activities of the RC can be represented by the following MA-system $\mathcal{RC}$:

Messages: (where $(C = 0)$ means "abstain"; $C \neq 0$ identifies a candidate)
between the members : $pref(C)$;
from a member to the secretary : $vote(C)$;
from the secretary to the candidates : $begin\_recr$;
from the secretary to the members : $begin\_vote$;

Agents
The action bases of the agents $\mathbf{m_i}$, $0 \le i \le 3$ contain the single action $act_i(C)$ with $ADD_i = DEL_i = \emptyset$, $SEND_0 = \{(s, vote(C)), (m_i, pref(C)) \mid i = 1, 2, 3, 4\}$ and $SEND_j = \{(s, vote(C)), (m_4, pref(C)) \mid j = 1, 2, 3\}$.

$\mathbf{m_4}$ keeps the preferences of other agents in his personal DB in the facts
$prefers(Member, Candidate)$. His action base contains two actions:
$record(Mb, Cd)$ with $ADD = \{prefers(Mb, Cd)\}$ and $DEL = SEND = \emptyset$ and
$act_4(C)$ with $ADD_4 = \emptyset$, $DEL_4 = \{prefers(Mb, Cd) \mid for\ all\ Mb, Cd\}$ and $SEND_4 = \{(s, vote(C))\}$.

The agents $m_i, i \in \{0, 2, 3\}$ have the **programs:**

$act_{m_i}(C) \leftarrow$
  $receive(C_1)$,
  $find_i(L)$,
  $unique(C, L)$.
where:
$receive_i(\_) =_{df} received(s, begin\_vote)$, $(i = 0$ or $i = 3)$,
$receive_2(C) =_{df} received(m_0, pref(C))$,
$find_i(L) =_{df} findall(PC, (cand(C, P, M, G),$
  $criterion_i(C, P, M, G, PC)), L)$,
$unique(C, L) =_{df} L = [C]; C = 0$
and use the following candidate choice criteria:
$criterion_0(C, P, M, G, PC) =_{df}$
  $cand(PC, P, \_, G) \wedge$
  $\forall C' \neq PC\ (cand(C', P', \_, G') \rightarrow (P > P' \vee (P = P' \wedge G < G')))$
$criterion_2(C, P, M, G, PC) =_{df}$
  $cand(PC, P, \_, G) \wedge cand(C, P1, \_, \_) \wedge P \neq P_1 \wedge$
  $\forall C' \neq C\ (cand(C', P', \_, G') \wedge P_1 \neq P' \rightarrow G > G')$
$criterion_3(C, P, M, G, PC) =_{df}$
  $cand(PC, P, M, \_) \wedge selected(C', P') \wedge P \neq P' \wedge$
  $\forall C_1\ (cand(C_1, P_1, M_1, \_) \wedge P_1 \neq P' \rightarrow M > M_1)$

m_4 has the following **program:**
$record(Mb, Cd) \leftarrow$
  $received(Mb, pref(Cd))$.
$act_{m_4}(C) \leftarrow$
  $prefers(m_0, C), prefers(m_1, C), prefers(m_i, C),$
  $C \neq 0.\ (i = 2, 3)$
$act_{m_4}(C) \leftarrow$
  $prefers(m_2, C), prefers(m_3, C), C \neq 0.$
$act_{m_4}(C) \leftarrow$

$prefers(m_0, 0), prefers(m_2, 0), prefers(m_3, C)$.

$act_{m_4}(C) \leftarrow$

$prefers(m_0, 0), prefers(m_3, 0), prefers(m_2, C)$.

We do not present the details of the secretary implementation. Its program is straightforward and determines an 8-step recruitment cycle in which all persons putting up their candidatures are immediately registered as candidates.

III. Verification of behavior properties

The crucial point about the example above is that the behavior of $\mathcal{RC}$ satisfies certain important properties, e.g.:

no_consecutive_lp $=_{df}$ "if an *lp*-candidate has been selected in a recruitment cycle and in the next cycle there exists non-*lp*-candidate which is best in both Merits and Grants, then non-*lp*-candidate will be selected in the next cycle."

worst_selected_lp $=_{df}$ "it is possible that the candidate worst in $Merits$ and $Grants$ will be selected in each cycle."

These properties should be verified with respect to all runs of $\mathcal{RC}$.

We represent the MA-systems as transition systems on global states. In these terms, a run of a deterministic MA-system corresponds to a trajectory through states, each two adjacent states in the trajectory being related by a transition. As it concerns nondeterministic MA-systems, a run of such a system corresponds to a trajectory in the tree of trajectories with the same condition on the adjacent states. This makes the logics of linear or branching time a convenient tool of behavior specification and analysis. In particular, linear time logics are well suited for the deterministic MA-system behavior description, whereas the branching time logics and more expressive μ-calculus are more adapted to the nondeterministic MA-system behavior properties.

The *"MA-BEHAVIOR"* problem we consider in this paper, applies to deterministic MA-systems as well as to nondeterministic ones. Given such a system $\mathcal{A}$, an initial global state S_0 and a formula Φ expressing a property of trajectories, the MA-BEHAVIOR problem $(\mathcal{A}, S_0, \Phi)$ has a positive solution if Φ holds on the tree $\mathcal{T}_{\mathcal{A}}(S_0)$ of trajectories of $\mathcal{A}$ starting in S_0 (denoted $\mathcal{T}_{\mathcal{A}}(S_0), S_0 \models \Phi$). We see that it is a kind of the model checking problem, though applied to MA-systems in the role of transition diagram specification.

In order to specify the behavior properties of the deterministic MA-systems, we use the *propositional linear temporal logic* (PLTL) and its first order extension FLTL [6] with the standard linear time operators **X** ("nexttime"), **U** ("until"), **V** ("unless"), **G** ("always"), and **F** ("sometimes"). For nondeterministic MA-systems, we use simple subsets of classical branching time logics: ∀LTL and ∃LTL consisting respectively of the formulas of the form $\mathbf{E}(\phi), \mathbf{A}(\phi)$, where ϕ is in $FLTL$ [5], first order extension of the μ-calculus [9], [7] (FO-μ) and its alternation depth bounded subsets (FO-$\mu_l, l \geq 1$).

For example, the MA-system $\mathcal{RC}$ above being nondeterministic, the properties above can be expressed by the two formulas below.

no_consecutive_lp is expressed by the following CTL-formula:

$$\mathbf{AG}\ (best \wedge \neg close \wedge (\mathbf{AX}\ close) \wedge \exists\ C\ selected(C, lp)\ \rightarrow (\mathbf{AX}\ )^8\ \exists\ C_1, P\ (selected(C_1, P) \wedge P \neq lp)),$$

where $(\mathbf{AX}\ )^8$ denotes **AX** eight times, [6] and *best* is the following 1st order formula saying that "there is a non-*lp* candidate best in both $Merits$ and $Grants$":

$$\exists\ C, P, M, G\ (cand(C, P, M, G) \wedge P \neq lp\ \wedge \forall\ C', P', M', G'\ (cand(C', P', M', G') \wedge C \neq C'\ \rightarrow\ M\ > M' \wedge G > G').$$

Theorem 1: $\mathcal{T}_{(RC)}(S_0), S_0 \models$ **no_consecutive_lp** holds in any initial state S_0.

worst_selected_lp is expressed by the following μ-formula:

$$close\ \rightarrow\ \nu R.\ (Q \wedge \exists\ C\ selected(C, lp) \wedge \mathbf{EX}^8\ R),$$

where Q is a 1st order formula which says: "there is an *ai*-candidate scientifically best, an *mm*-candidate financially best and an *lp*-candidate which is the worst in both parameters". Clearly, such a formula can be written like *best*.

Theorem 2: If at least one person puts up his candidature and somebody was already selected in the initial state S_0 then $\mathcal{T}_{(RC)}(S_0), S_0 \models$ **worst_selected_lp**.

IV. Deterministic MA-systems

Theorem 3: (1) The MA-BEHAVIOR problem is decidable in polynomial time in the class of expanding, positive k-dimensional deterministic MA-systems, for behavior properties $\Phi \in PLTL$ and for any fixed k.
(2) The MA-BEHAVIOR problem is PSPACE-complete for k-dimensional deterministic MA-systems for any fixed k and for the properties of MA-systems behavior, expressible in $FLTL$.

Abandoning k-dimensionality leads to intractability.

Theorem 4: The MA-BEHAVIOR problem is EXPTIME-complete for expanding and positive deterministic MA-systems and the properties of MA-systems behavior, expressible in $PLTL$.

In fact, we show that the MA-BEHAVIOR problem is untractable even in a very narrow class of deterministic MA-systems.

[5] The path quantifiers **A** and **E** mean respectively "for all trajectories" and "for some trajectory".

[6] We remind that 8 is the length of one recruitment cycle.

Corollary 1: The MA-BEHAVIOR problem is EXPTIME-hard in the class of expanding, positive, and 0-signal deterministic 1-agent systems and behavior properties in $PLTL$.

In the general case the problem is still more hard.

Theorem 5: The MA-BEHAVIOR problem is EXPSPACE-complete in the class of deterministic MA-systems and the properties of MA-systems behavior, expressible in $FLTL$.

The space limits do not allow us to include more strong complexity results for ground MA-systems, both for deterministic and nondeterministic case.

V. Nondeterministic MA-systems

In this general case, we establish the complexity bounds of the MA-BEHAVIOR problem for MA-systems communicating through a bounded number of messages, or else using the predicates of bounded arity.

Theorem 6: (1) The MA-BEHAVIOR problem is NEXPTIME-complete (coNEXPTIME-complete) in the class of expanding r-signal MA-systems, for any fixed r, and the behavior properties Φ expressed in $\exists LTL$ (respectively, in $\forall LTL$).
(2) The MA-BEHAVIOR problem is EXPTIME-complete for k-dimensional MA-systems, for any fixed k, and the behavior properties Φ expressed in first order μ.

Theorem 7: The MA-BEHAVIOR problem is EXPSPACE-hard for expanding positive MA-systems and behavior properties expressed in $\exists LTL$.

We think that the corresponding tight upper bound holds too, but this problem is left open.

In the general case of nondeterministic nonground MA-systems, the problem is still harder.

Theorem 8: The MA-BEHAVIOR problem for general MA-systems
1) is EXPEXPTIME-complete for the behavior properties expressed in first order μ_r (for any fixed r);
2) is complete in NEXPEXPTIME $\cap$ co-NEXPEXPTIME for the behavior properties expressed in first order μ.

VI. Conclusion

The agent and MA-system architectures published within the past few years are dissimilar and diversified because they represent various application domains of this new software technology. Our study concerns one such specific architecture which covers the systems of distributed autonomous parallel interacting agents. This architecture is not universal. For example, it does not fit asynchronous interactions in the Internet. The nondeterministic semantics we propose are only partially covering such kind of interactions.

This paper illustrates the way in which penetrating in much detail in a complex MA-system architecture permits in some cases to better understand the behavior properties and in this way, to obtain more deep results. Considered from this point of view, this paper creates a perspective of applying similar analysis to other MA-system architectures in order to find interesting subclasses of MA-systems with efficiently checked behavior properties.

References

[1] Apt, K. R., Logic Programming. In: J. van Leeuwen (Ed.) *Handbook of Theoretical Computer Science. Volume B.* , Elsevier Science Publishers B.V. 1990, 493-574.

[2] Clarke, E. M., Grumberg, O. and Peled, D., *Model Checking,* MIT Press, 2000.

[3] Dekhtyar, M. I., Dikovsky, A. Ja., On Homeostatic Behavior of Dynamic Deductive Data Bases. In: *Lect. Notes in CS,* N. 1181, 1996, 420-432.

[4] Dekhtyar, M. I., Dikovsky, A. Ja., Valiev, M. K. Applying temporal logic to analysis of behavior of cooperating logic programs. *Lect. Notes in CS,* N. 1755, 2000, 228-234.

[5] Eiter, T. and Mascardi, V. Comparing Environments for Developing Software Agents. *Techn. Rept. INFSYS RR-1843-01-02*, March 2001, Inst. für Informationssysteme, Technische Universität Wien, Vienna, Austria.

[6] Emerson, E. A. Temporal and modal logic. In: J. van Leeuwen (Ed.), *"Handbook of Theor. Comput. Sci.",* Elsevier Sci. Publishers, 1990.

[7] Emerson, E. A. Model checking and the mu-calculus. In: N. Immerman, P. H. Kolaitis (Eds.), "Descriptive Complexity and Finite Models". *Proc. of a DIMACS Workshop,* 1996, 185-214.

[8] Fisher, M., Wooldridge, M., Specifying and Verifying Distributed Intelligent Systems. In: M. Filgueiras and L. Damas (Eds.) *Lect.Notes in AI*, N. 727, 1993, pp.13-28, Springer-Verlag: Berlin, Germany.

[9] Kozen, D., Results on the Propositional μ-calculus. *Theoretical Computer Science*, 1983, v. 27, pp. 333–354.

[10] Reiter, R. *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems.* MIT Press, 2001.

[11] Shoham, Y., Agent oriented programming. *Artificial Intelligence*, 1993, 60:51-92.

[12] Subrahmanian, V. S., Bonatti, P., Dix, J., et al., *Heterogeneous Agent Systems,* MIT Press, 2000.

[13] Vardi, M., Wolper, P., An automata-theoretic approach to automatic program verification. In: *Proc. of the IEEE Symposium on Logic in Computer Science,* 1986, 332-344.

[14] Wooldridge, M., Jennings N. Intelligent agents: Theory and practice. *The Knowledge Engineering Review,* 1995, 10(2).