On Complexity of Verification of Interacting Agents' Behavior

Michael Dekhtyar [a,1] Alexander Dikovsky [b,1] Mars Valiev [c,1]

[a] *Dept. of CS, Tver St. Univ., Tver, Russia, 170000*
Michael.Dekhtyar@tversu.ru

[b] *IRIN, Université de Nantes, 2, rue de la Houssinière, BP 92208 44322 Nantes Cedex 03 France*
Alexandre.Dikovsky@irin.univ-nantes.fr
and
Keldysh Inst. for Appl. Math., Moscow, Russia, 125047

[c] *Keldysh Inst. for Appl. Math. Moscow, Russia, 125047*
valiev@spp.keldysh.ru

Abstract

This paper studies the complexity of behavior of multi-agent systems. Behavior properties are formulated using classical temporal logic languages and are checked with respect to the transition system induced by the definition of the multi-agent system. We establish various tight complexity bounds of the behavior properties under natural structural and semantic restrictions on agent programs and actions.

Key words: Multi-Agent Systems, Temporal logics, μ-calculus, Model checking, Complexity

1 Introduction

The aim of this paper is to study the complexity of verification of behavior (dynamic) properties of deterministic, nondeterministic and asynchronous multi-agent systems [2] and continues our paper [10]. Although intelligent

[1] This work was sponsored by the Russian Fundamental Studies Foundation (Grants 04-01-00565, 01-01-00278 and 02-01-00652).

[2] The results of this paper were announced without proofs in preliminary publications [9,11].

agents have been the object of active study for at least two decades, research in this specific field (see [29,31,4,1,32]) is relatively scarce.

The terms *'Intelligent Agent'* (IA) and *'Multi-Agent System'* (MAS) refer to a promising and rather general metaphor of computing technology based on Artificial Intelligence. The range of IA applications extends from operating system interfaces, processing of satellite imaging data, web navigation to air traffic control, business process management and electronic commerce. Due to diversity of applications, and diversity of approaches, there is no unified definition of the notion of an Intelligent Agent. We refer the reader to [29] and several other publications [27,33,3,17,20,26] for a variety of interpretations of what Intelligent Agents are. For particular agent architectures, the intelligence capacity of an agent can vary from finite state control structures or IF-THEN rules to logic programs, non-monotone belief based systems or deontic logics (see [29] for a discussion and references).

Consider the following example.

Example 1 *"Resource-allocation"*
A resource allocation system $\mathcal{RA}$ consists of a manager-agent m that owns some resource r, which it distributes on orders from four user-agents u_1, u_2, u_3, u_4. Given a discrete timeline $t = \{t_1, t_2, \ldots\}$, each user has its own strategy for ordering resources:
1) u_1 is the first to order a resource; then it repeats its order on receipt of the resource;
2) u_2 orders the time instant after u_1 has ordered;
3) u_3 orders the time instant after u_1 has received the resource from m;
4) u_4 orders every time instant.
The manager m maintains a list of orders and fulfills the first order on the list, one order at a time. Only one order from each user-agent can be held on the list. So if m receives an order from some agent u_i before the previous order from it has been fulfilled, the new order is ignored.

We see that the five agents in Example 1 are autonomous in the sense that all of them can function continuously with or without stimuli from other agents. Meanwhile, these stimuli are necessary in order that the user-agents achieve their goals, i.e., obtain the resource r. To facilitate that, the agents communicate through messages, that allow them to find out when orders have been placed or fulfilled. The intelligence of the four user agents is rather primitive: just conditional actions. Meanwhile, agent m must be more intricate in order to control correctly the incoming orders and the states of the queue.

The behavior of the agents in $\mathcal{RA}$ is deterministic whereas generally this is not the case. In many applications agents have only partial knowledge of their medium, which causes a nondeterministic behavior of the MAS. Let us consider another example.

Example 2 *"Recruiting Committee".*
An academic recruiting committee consists of 5 member agents m_i, $0 \leq i \leq 4$ *(*m_0 *being the lab chief) and the secretary agent* s*. The seventh agent* c *simulates at each recruitment cycle the submission of a number of applications for a faculty position. The applications are of the form* $cand(C, Profile, Merits, Grants)$*, where* C *is a unique id of a candidate (taken from a given finite set of strings);* $Profile$ *identifies the field of research of the candidate: lp (logic programming), ai (artificial intelligence), cl (computational linguistics) or mm (multimedia);* $Merits$ *is a rank of the candidate's scientific accomplishments and, finally,* $Grants$ *is the sum of the grants the candidate has obtained (variables* $Merits$ *and* $Grants$ *take values from some finite sets of numbers). Agent* c *sends the secretary agent several such applications.*

The member agents can read, in addition to the applications, the information about the faculty member selected in the preceding recruitment cycle: $selected(C, Profile)$*. This fact combined with a special flag close and the applications form the shared database* GB *of the committee. On receipt of the applications, the secretary announces the beginning of a new recruiting cycle and deletes flag close from* GB*. Before the vote, the members may speak out on the candidates (through messages). They then vote by sending their secret vote messages to the secretary: the selected candidate or the abstention. On receiving all members' votes, the secretary updates the* GB *according to the tally: places the data of the candidate selected by the majority, if any (when no majority, no candidate is selected) and closes the session by putting the flag close back into* GB*. The secretary then deletes the applications of candidates who were not selected and enters the initial state of a new recruitment cycle.*

The five committee members have different vote tactics but all of them abstain when their criteria do not ensure the uniqueness or the existence of their choice.

$\mathbf{m_0}$ *(a.k.a. "the boss") has the following research area preferences:* $lp > ai > mm > cl$*. An area having been chosen, he selects a candidate with the largest sum of grants and announces his choice to all committee members;*
$\mathbf{m_1}$ *always joins to* m_0;
$\mathbf{m_2}$ *selects a candidate with the research area different from the area of the candidate chosen by* m_0 *who has the largest amount of grant money;*
$\mathbf{m_3}$ *votes for the candidate with the best scientific merits whose research area is different from that of the candidate hired in the previous cycle;*
$\mathbf{m_4}$ *votes with the majority, if any; if there is no majority, then* m_4 *joins to* m_2*.*
Agent c *in this example is nondeterministic. One cannot know beforehand either the exact number of the candidates or the specifics of their applications.*

The MAS we consider in this paper conform in general to the IMPACT architecture of Subrahmanian et al. introduced and described in detail in [29]. This architecture is very elaborate. It includes rather expressive agent specification means and control structures, e.g. adaptive action bases, logic programs articulating decision policies, constraints, belief-based meta-reasoning

about other agents, reasoning with uncertainty, reasoning about time, communication management, security related structures, interfacing, and some other facilities. Such abundance of expressive means makes this architecture well-adapted for practical applications. At the same time, it complicates the formal study of the properties of the agents. The agent semantics in IMPACT architecture is described in terms of transitions between the agent states and is shown in [29] to be intractable in general. In order to arrive at a polynomial time computable transition semantics, Subrahmanian et al. impose very complex limitations on the agents features. As a result, the definition of such "polynomial" agents becomes bulky.

In this paper, we impose other, easy-to-formulate limitations on IMPACT agents, which lead to a polynomial time semantics. We focus on the agent features that relate to actions, decision policies and communication. On the other hand, we do not consider features related to the legacy code, security, metaknowledge structures, temporal and uncertainty reasoning. Moreover, we simplify the internal agent's data structure to be a relational database (in IMPACT a more general structure is allowed), and consider conventional logic programs as a means of an agent action policy definition (IMPACT agent programs include deontic modalities "permitted action", "forbidden action", "action to be performed", etc.). Even after these simplifications, the MAS architecture remains very rich. We study behavior properties under various, more or less restrictive constraints on MAS parameters and semantics.

As the examples above show, agents can be deterministic or non-deterministic. But, when we consider the behavior of multi-agent systems, another aspect also becomes to be important: how do the agents of the system interact? If all the agents of the system are placed in a local network (in particular, in a standalone computer), we can assume that messages from one computer to another one go immediately, so, we say on a *synchronous mode* of interaction and synchronous multi-agent systems. On other side, if the transfer of messages from one computer to another one can take an indeterminate time (as in Internet), we say on *nondeterministic mode* of interaction and asynchronous multi-agent systems. It is clear that the behavior of an asynchronous MAS is non-deterministic, even if the agents of the system are deterministic. In fact, it will be shown that in a sense any synchronous non-deterministic MAS can be embedded in an asynchronous MAS with only deterministic agents.

In any case the behavior of the MAS is described as a set of trajectories (paths) in the state transition diagrams they induce: a single path in the (synchronous) deterministic case, and multiple paths in the nondeterministic (synchronous or asynchronous) case. This allows the use of classical temporal logics: $PLTL$, CTL, CTL^* [12], μ-calculus [21] and their first order variants to express the behavior properties of these systems.

The problem *"MA-BEHAVIOR"* of verifying that a temporal logic formula Φ holds on the trajectories of a given MAS, considered in this paper is, basically, a model checking problem. Model checking on abstract transition diagrams has been extensively studied since the early 1980s (see [30,24,12,13,7]). There is, however, a substantial difference between the classical problem statement and the one studied in this paper. Traditionally, the complexity results are established for transition diagrams that are *explicitly presented* or for some of their fixed representations (e.g., by finite state automata, by OBDD).

We establish the complexity bounds with respect to MAS whose operational semantics is presented in the form of transition systems. The novelty of this approach is in the fact that the problem complexity is determined by various structural and semantic constraints on MAS. MAS constitute a compact representation of the corresponding transition system. For example, even for a ground (i.e. variable-free) MAS A, the transition system $T(A)$ describing its trajectories may have the size exponential in $|A|$, because it may have $\mathbf{O}(2^{|A|})$ states. So, sometimes, our lower bounds are more pessimistic than in the classical case for the same classes of logics. As far as the upper bounds are concerned, they are either more informative and precise (in the case of polynomial time and space complexity), or they are simply translated from the corresponding classical results taking into consideration the size and the number of MAS states.

In our previous paper [10] we considered synchronous deterministic and nondeterministic MAS under some strong constraints, such as the monotonicity of intelligent components (logic programs) of the systems. For these classes of MAS the MA-BEHAVIOR problem turned out to be decidable in deterministic or nondeterministic polynomial time. In this paper we study the MA-BEHAVIOR problem for more general classes of synchronous and asynchronous MAS (under some weaker restrictions on their parameters). Naturally, in these cases the complexity of the MA-BEHAVIOR problem increases significantly and varies from polynomial space to double exponential time.

The remainder of this paper is structured as follows. In section 2, we describe the IA and the synchronous MAS architectures, their one step and transition system semantics, and specify several important classes of MAS corresponding to natural constraints imposed on their structural features. Then in section 3, we give a brief overview of some classical temporal logic notions and facts we use in the proofs. The two sections that follow study the problems of verifying dynamic properties for synchronous MAS, deterministic (section 4) and nondeterministic (section 5) cases. Then, in section 6 we introduce the asynchronous version of MAS and study the complexity of their verification.

2 Intelligent Agents and Multi-Agent Systems

In this section we present a simplified version of the IMPACT architecture of [29], for the synchronous case. Since the asynchronous MAS appear only in the last section, until that section we will use for brevity the simpler term "MAS" instead of "synchronous MAS".

A *(synchronous) multi-agent system* (*MAS*) $\mathcal{A}$ is a finite set $\{a_1, ..., a_n\}$ of *intelligent agents.* Each intelligent agent a has an *internal database (DB)* I_a, which is a finite set of ground atoms in its extensional signature $\mathbf{P}_a^e$ [3] and a finite message box $MsgBox_a$. Agents communicate through messages of the form *msg(Sender, Receiver, Msg)*, where *Sender* and *Receiver* are agent names (the source and the destination), and Msg is a ground atom (in the message signature $\mathbf{P}_a^m$) sent by *Sender* to *Receiver*. The internal DB and the current message box contents constitute the agent's current *local state* $IM_a = (I_a, MsgBox_a)$. To distinguish local properties of different agents of the system $\mathcal{A}$ we assume that extensional signatures of different agents are pairwise disjoint. Since the messages are not local the disjointness condition is not assumed for the message signatures of agents.

The set of local states $\{IM_a \mid a \in \mathcal{A}\}$ forms the current *global state* of the MAS.

Each agent a is capable of performing a number of parameterized actions constituting its action base AB_a. Any (parameterized) action has the form $\langle \alpha(X_1, \ldots, X_l), ADD_\alpha(X_1, \ldots, X_l), DEL_\alpha(X_1, \ldots, X_l), SEND_\alpha(X_1, \ldots, X_l)\rangle$, where $\alpha^{(l)}$ is a predicate from the action signature $\mathbf{P}_a^{act}$. We call $\alpha(X_1, ..., X_l)$ the parameterized name of the action. The sets $ADD_\alpha(X_1, \ldots, X_l)$ and $DEL_\alpha(X_1, \ldots, X_l)$ consist of atoms of the form $p(t_1, \ldots, t_k)$ where p is k-ary predicate (for some k) in the signature $\mathbf{P}_a^e$, $t_1, \ldots, t_k$ are terms which can include only variables $X_1, \ldots, X_l$. These sets determine updates of the internal DB (adding and deleting facts) when the corresponding action is executed. The set $SEND_\alpha(X_1, \ldots, X_l)$ consists similarly of atoms of the form $msg(a, b, p(t_1, \ldots, t_k))$ determining messages which will be sent to other agents. In further, when we define concrete agents, for brevity we will often use a short notation $(b, p(t_1, \ldots, t_k))$ instead of $msg(a, b, p(t_1, \ldots, t_k))$ in descriptions of sets $SEND_\alpha$, where α is an action of an agent a,

Let $c_1, \ldots, c_l$ be constants. Let us denote by $ADD_\alpha(c_1, \ldots, c_l)$ the set of facts obtained by substitution of $c_1, \ldots, c_l$ instead of $X_1, \ldots, X_l$ into atoms of

[3] We adopt a domain closure assumption: namely, some finite set $Const(\mathcal{A})$ of domain constants is connected with any MAS $\mathcal{A}$, and all built-in predicates and operations used by this MAS are defined on this set and computable in polynomial time with respect to the size of the MAS.

$ADD_\alpha(X_1, \ldots, X_l)$. The sets $DEL_\alpha(c_1, \ldots, c_l)$ and $SEND_\alpha(c_1, \ldots, c_l)$ are defined similarly. The ground atoms $\alpha(c_1, \ldots .c_l)$ are called *ground action names*.

The policy of the agent a for choosing actions from AB_a to execute depends on the current state IM_a^t (the local state of a at current time t) and is determined by a pair $\langle LP_a, Sel_a \rangle$. Here LP_a is a logical program which determines a set $Perm_a^t$ of ground action names permitted for execution at current time, and obligation operator Sel_a selects a subset Obl_a^t of $Perm_a^t$ consisting of ground action names which should be executed.

When Obl_a^t is determined its execution by the agent a is defined as follows. Let $AddObl_a^t$ be the union of all the sets $ADD_\alpha(c_1, \ldots, c_l)$ such that a ground name $\alpha(c_1, \ldots, c_l)$ from $Obl_{a,t}$ is unified with a parameterized name $\alpha(X_1, \ldots, X_l)$. The sets $DelObl_a^t$ and $SendObl_a^t$ are defined similarly. Then the next state of the internal base of a is obtained from the current state by deleting all the facts belonging to $DelObl_a^t$, and then adding all the facts belonging to $AddObl_a^t$. Moreover, the contents of $MsgBox_a$ at the next moment $t+1$ is defined as the set of all the messages $msg(b, a, Msg)$ such that $msg(b, a, Msg)$ belongs to $SendObl_b^t$, for all agents b of the system.

An action α is *expanding* if DEL_α is empty. Agent a is *expanding* if it has only expanding actions.

To complete the definition of a and one-step semantics for it we should define LP_a, Sel_a and how do they determine the sets $Perm_a^t$ and Obl_a^t.

LP_a is a logic program with the clauses of the form $H \leftarrow L_1, \ldots, L_n$, where $n \geq 0$, the head $H = \alpha(t_1, ..., t_l)$ is an action atom, i.e. α is an action predicate [4]; the literals L_i in its body are either action literals, or (extensional) internal DB literals, or atoms of the form $msg(b, a, Msg)$ or their negations $\neg msg(b, a, Msg)$, or built-in predicate calls $q(\bar{t})$

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 for every t the program
$LP_a^t = LP_a \cup \{p \leftarrow \mid p \in I_a^t\} \cup \{msg(b, a, Msg) \leftarrow \mid msg(b, a, Msg) \in MsgBox_a^t\}$ is *stratified* [2].

The set $Perm_a^t$ of actions permitted for execution at time t is defined as the set of ground *action names* contained in the minimal model M_a^t of LP_a^t. As

[4] Auxiliary (intensional) predicates can be included in LP_a as actions with empty sets *ADD, DEL* and *SEND*.

is well known [2], this model is unique for the stratified logic programs and is computed by a polynomial time fixpoint computation procedure from the *groundization* $gr(LP_a^t)$ of the program LP_a^t [5]

We distinguish *deterministic* and *nondeterministic* agent obligation operators Sel. *Deterministic obligation operator* Sel is a total function which for a given set of ground action names A returns some its subset $Sel(A) \subseteq A$.

For instance, the *total deterministic* semantics defined by $Sel^{td}(A) = A$ belongs to this class. We can also imagine other types of deterministic obligation operators, e.g. priority driven deterministic operator that establishes some partial order $\prec$ on ground actions and is defined by $Sel^{\prec d}(A) = \{m \in A \mid \neg\exists m' \in A\ (m' \prec m)\}$.

Deterministic agents are those having a deterministic obligation operator. A MAS is called *deterministic* if all its agents are deterministic.

Nondeterministic one-step semantics is a total binary relation Sel on the subsets of the set of ground action names such that if $Sel(A, A')$ then $A' \subseteq A$.

The simplest nondeterministic operator in this class is the *unit choice* operator defined by $Sel^{un}(A) = \{\{p\} \mid p \in A\}$ which just guesses a single available action in A. Another example is the *spontaneous* operator defined by $Sel^{sn}(A) = \{A' \mid A' \subseteq A\}$. It guesses any subset of available actions in A. *Nondeterministic agents* are those with a nondeterministic obligation operator. A MAS is *nondeterministic* if it contains at least one nondeterministic agent.

It is natural to assume that a larger set of available actions leads to a larger set of chosen actions. Therefore, we assume that for every agent a its obligation operator Sel_a is a monotonic operator: $Sel_a(A) \subseteq Sel_a(A')$ for $A \subseteq A'$.

We will also assume that deterministic obligation operators are functions computable in polynomial time, and nondeterministic obligation operators are binary relations computable in polynomial time.

The one-step semantics of agent $a \in \mathcal{A}$ defines new local state of a and the set of messages which a sends to the other agents as described above.

The *one-step semantics* of the MAS $\mathcal{A}$ is a one step transition relation $\Rightarrow_{\mathcal{A}}$ on the set $\mathcal{S}_{\mathcal{A}}$ of global states of the form $S =< (I_{a_1}, MsgBox_{a_1}), \ldots, (I_{a_n}, MsgBox_{a_n}) >$ induced by one-step semantics of individual agents of $\mathcal{A}$

[5] I.e. from the set of all ground instances (with constants from $Const(\mathcal{A})$) of clauses in LP_a^t. It should be noted that the size of $gr(LP_a^t)$ can be exponential with respect to the size of LP_a^t.

in a natural way. We note that the relation $\Rightarrow_{\mathcal{A}}$ is total.

The pair $T(\mathcal{A}) = (\mathcal{S}_{\mathcal{A}}, \Rightarrow_{\mathcal{A}})$ constitutes a Kripke structure or state transition system (see, e.g. [7]). The behavior of MAS $\mathcal{A}$ in a global state S^0 from $\mathcal{S}_{\mathcal{A}}$ is determined by the set of paths in $T(\mathcal{A})$ starting in S^0. We will be interested in the behavior of MAS in initial states with empty message boxes.

For a MAS $\mathcal{A}$, and its initial global state $S^0 = < (I^0_{a_1}, MsgBox^0_{a_1}), ..., (I^0_{a_n}, MsgBox^0_{a_n}) >$, where $MsgBox^0_{a_i} = \emptyset$, $1 \leq i \leq n$, let $\mathcal{T}_{\mathcal{A}}(S^0)$ denote the set of infinite trajectories (execution paths in $T(\mathcal{A})$) of the form:

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

For a deterministic MAS $\mathcal{A}$, $\mathcal{T}_{\mathcal{A}}(S^0)$ consists of a single trajectory starting in S^0. If $\mathcal{A}$ is nondeterministic, then we will consider $\mathcal{T}_{\mathcal{A}}(S^0)$ as an infinite tree of trajectories with the root node labeled by S^0. The nodes of $\mathcal{T}_{\mathcal{A}}(S^0)$ are labeled by the global states $S \in \mathcal{S}_{\mathcal{A}}$ accessible from S^0 by the reflexive-transitive closure of $\Rightarrow_{\mathcal{A}}$. In what follows we do not distinct a node of $\mathcal{T}_{\mathcal{A}}(S^0)$ and its label.

This architecture covers systems of distributed autonomous parallel interacting agents. There are many applications well-suited for this framework. One example is distributed intelligent programs interacting in local networks. On the other hand, this architecture does not fit asynchronous interactions over the Internet. The nondeterministic semantics we propose here only partially cover such kind of interactions. In the last part of the paper we introduce the asynchronous version of MAS, and show that the main results obtained for nondeterministic MAS can be transferred to the asynchronous MAS.

2.1 Classes of Mullti-Agent Systems

We distinguish between two main classes of MAS: deterministic and nondeterministic. In both classes of MAS, we consider the following subclasses induced by natural constraints imposed on agent components. A MAS $\mathcal{A} = \{a_1, ..., a_n\}$ is

- *ground* if each program LP_{a_i} is ground [6] ;
- *k-dimensional* if the arities of all action atoms and all message atoms are bounded by k. 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*).

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

The following simple proposition characterizes the complexity of the one step semantics for MAS from these classes.

Proposition 1
(1) For each deterministic MAS $\mathcal{A}$ *, the transition function* $S \Rightarrow_{\mathcal{A}} S'$ *is computable in polynomial time w.r.t.* $|S| + |\mathcal{A}| + |S'|$ *if* $\mathcal{A}$ *is ground or dimension-bounded, and is computable in deterministic exponential time* [7] *in the general nonground case.*
(2) For each nondeterministic MAS $\mathcal{A}$ *, 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.*

2.2 Implementation of Examples

In this section we show how Examples 1 and 2 from Introduction can be implemented in terms of our MAS.

Example 3 *"Resource-allocation" revisited*
We specify the agents from Example 1 in the form of the following deterministic MAS $\mathcal{RA} = \{u_1, u_2, u_3, u_4, m\}$.

The states I_{u_1} *of* u_1 *can contain the facts* put_order *and* $receipt_1$. *The states* I_{u_i} $(i = 2, 3, 4)$ *can contain the fact* $receipt_i$ *stating that* u_i *received a resource at the previous step. In order to let* m *and other users know that* u_i *asks for a resource, this agent sends them the message order. When* m *fulfills an order of* u_i, *it sends to* u_i *the message ok. Agent* u_1 *sends to* u_3 *the message ok in order to confirm the receipt of a resource.*

Each agent u_i $(i = 1, 2, 3, 4)$ *has two actions:* $use\_resource_i$: $DEL_{use\_resource_i} = \{receipt_i\}$ *and* $receive_i$: $ADD_{receive_i} = \{receipt_i\}, DEL_{receive_i} = SEND_{receive_i} = \emptyset$ *for* $i = 2, 3, 4$, *and* $DEL_{receive_1} = \{put\_order\}, SEND_{receive_1} = \{(u_3, ok)\}$ *for* $i = 1$. *These actions are fired by the clauses:*
$use\_resource_i \leftarrow receipt_i$. *and*
$receive_i \leftarrow msg(m, u_i, ok)$.
Here are the other actions and program clauses of the agents.
Agent u_1 *has the action* put : $ADD_{put} = \{put\_order\}, SEND_{put} = \{(m, order), (u_2, order)\}$; *with the clause:*
$put \leftarrow \neg put\_order$

Agent u_2 *has the action* put : $SEND_{put} = \{(m, order)\}$ *with the clause:*
$put \leftarrow msg(u_1, u_2, order)$
Agent u_3 *has the action* put : $SEND_{put} = \{(m, order)\}$ *with the clause:*

[7] In fact, in polynomial time with respect to the size of the groundization of the program.

$put \leftarrow msg(u_1, u_3, ok)$
Agent u_4 *has the action* put : $SEND_{put} = \{(m, order)\}$ *with the clause:*
$put \leftarrow$.

Agent m *maintains the queue of orders, represented by facts* $first(A), next(A, B)$ *and* $last(A)$ *contained in its internal state* I_m. *It uses two auxiliary predicates:* $empty\_queue$ *which is true when the queue of orders is empty, and* $in\_queue(A)$ *which is true when the order of agent* A *is in the queue.*
$empty\_queue \leftarrow \neg first(u_1), \neg first(u_2), \neg first(u_3), \neg first(u_4)$.
$in\_queue(A) \leftarrow first(A)$.
$in\_queue(A) \leftarrow in\_queue(B), next(B, A)$.
When m *receives new orders it places them at the end of the list in the predefined order* $u_1 < u_2 < u_3 < u_4$.

For each $i = 1, 2, 3$ *and each sequence* $\beta = j_1, \ldots, j_i$ $(1 \leq j_1 < \ldots < j_i \leq 4)$ m *has an action* $insert_\beta(F, S, L, X_{j_1}, \ldots, X_{j_i})$:
$ADD_{insert_\beta}(F, S, L, X_{j_1}, \ldots, X_{j_i}) = \{first(S), next(L, X_{j_1}), next(X_{j_1}, X_{j_2}), \ldots,$
$next(X_{j_{i-1}}, X_{j_i}), last(X_{j_i})$;
$DEL_{insert_\beta}(F, S, L, X_{j_1}, \ldots, X_{j_i}) = \{first(F), next(F, S), last(L)\}$;
$SEND_{insert_\beta}(F, S, L, X_{j_1}, \ldots, X_{j_i}) = \{(F, ok)\}$.
This action is fired by the rule:
$insert_\beta(F, S, L, u_{j_1}, \ldots, u_{j_i}) \leftarrow new\_order(u_{j_1}), \ldots, new\_order(u_{j_i}),$
$\neg new\_order(u_{k_1}), \ldots, \neg new\_order(u_{4-i}), first(F), next(F, S), last(L)$.
(here $\{k_1, \ldots, k_{4-i}\} = \{1, 2, 3, 4\} \setminus \{j_1, \ldots, j_i\}$*).*
$new\_order(X) \leftarrow msg(X, m, order), \neg in\_queue(X)$.

When the queue is empty m *fires one of actions of the form* $insert1_\beta(X_{j_1}, \ldots, X_{j_i})$:
$ADD_{insert1_\beta(X_{j_1},\ldots,X_{j_i})} = \{first(u_{j_1}), next(u_{j_1}, u_{j_2}), \ldots, next(u_{j_{i-1}}, u_{j_i}), last(u_{j_i})\}$;
$DEL_{insert1_\beta(X_{j_1},\ldots,X_{j_i})} = SEND_{insert1_\beta(X_{j_1},\ldots,X_{j_i})} = \emptyset$. *This action is fired by the rule:*
$insert^0_\beta(u_{j_1}, \ldots, u_{j_i}) \leftarrow empty\_queue,\ new\_order(u_{j_1}), \ldots, new\_order(u_{j_i}),$
$\neg new\_order(u_{k_1}), \ldots, \neg new\_order(u_{4-i})$
(here again $\{k_1, \ldots, k_{4-i}\} = \{1, 2, 3, 4\} \setminus \{j_1, \ldots, j_i\}$*). (In the case* $i = 4$ *no negations of the form* $\neg new\_order(u_l)$ *are included.)*

All agents uses the total deterministic semantics defined by the obligation operator $Sel^{td}(A) = A$. *In fact, it can be shown that* $|Perm^t_a| \leq 1$ *for all* t *and* $a \in \mathcal{RA}$.

Example 4 *"Recruiting Committee" revisited .*
We implement the agents from Example 2 in the form of the following spontaneous nondeterministic MAS $\mathcal{RC}$. *This particular implementation does not use a shared database GB explicitely. It can be simulated by a database of a special agent who sends at each step the messages on all updates of GB to all agents in the system. In* $\mathcal{RC}$ *it is the secretary agent* s *who keeps its database* I_s *and provides the information on candidates for all the committee members. So, we suppose that all committee members have full access to this information and do not include the details of its transmission.*

Messages: *($(C = 0)$ means "abstain"; $C \neq 0$ identifies a candidate)*
preference messages the members exchange : $pref(C)$;
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.
Actions:
The action bases of the agents $\mathbf{m_i}$, $0 \leq i \leq 3$ *contain the single action* $act_i(C)$ *with* $ADD_{act_i}(C) = DEL_{act_i}(C) = \emptyset$, $SEND_{act_0}(C) = \{(s, vote(C)), (m_i, pref(C)) \mid i = 1, 2, 3, 4\}$, *and* $SEND_{act_i}(C) = \{(s, vote(C)), (m_4, pref(C)) \mid j = 1, 2, 3\}, i = 1, 2, 3$.

$\mathbf{m_4}$ *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* $ADD_{record}(Mb, Cd) = \{prefers(Mb, Cd)\}$,
$DEL_{record}(Mb, Cd) = SEND_{record}(Mb, Cd) = \emptyset$, *and*
$act_4(C)$ *with* $ADD_{act_4}(C) = \emptyset$, $DEL_{act_4}(C) = \{prefers(Mb, Cd) \mid for\ all\ Mb, Cd, \}$
$SEND_{act_4}(C) = \{(s, vote(C))$.

Programs:

LP_{m_0} :
$act_{m_0}(0) \leftarrow msg(s, m_0, begin\_vote), many\_best_0$.
$act_{m_0}(C) \leftarrow msg(s, m_0, begin\_vote), unique\_best_0(C)$.
$unique\_best_0(C) \leftarrow best_0(C), \neg many\_best_0$.
$many\_best_0 \leftarrow best_0(C), best_0(C_1), C \neq C_1$.
$best_0(C) \leftarrow \neg non\_best_0(C)$.
$non\_best_0(C) \leftarrow cand(C, P, M, G), cand(C_1, P_1, M_1, G_1), P < P_1$.
$non\_best_0(C) \leftarrow cand(C, P, M, G), cand(C_1, P, M_1, G_1), G < G_1$.

LP_{m_1} :
$act_{m_1}(C) \leftarrow msg(m_0, m_1, pref(C))$.

Programs LP_{m_2} *and* LP_{m_3} *have similar structure* $(i = 2, 3)$*:*
$act_{m_i}(0) \leftarrow rec_i(P_0), \neg unique\_best_i(C, P_0)$.
$act_{m_i}(C) \leftarrow rec_i(P_0), unique\_best_i(C, P_0)$.
$unique\_best_i(C, P_0) \leftarrow best_i(C, P_0), \neg many\_best_i(P_0)$.
$many\_best_i(P_0) \leftarrow best_i(C, P_0), best_i(C_1, P_0), C \neq C_1$.
$best_i(C, P_0) \leftarrow \neg non\_best_i(C, P_0)$.
$non\_best_i(C, P_0) \leftarrow cand(C, P, M, G), cand(C_1, P_1, M_1, G_1),$
$crit_i(P, P_0, P_1, M, M_1, G, G_1)$. *where*
$rec_2(P_0) \leftarrow msg(m_0, m_2, pref(C_0)), cand(C_0, P_0, M_0, G_0)$.
$rec_3(P_0) \leftarrow msg(s, m_3, begin\_vote), selected(C_0, P_0)$.
$crit_2(P, P_0, P_1, M, M_1, G, G_1) \leftarrow P \neq P_0, P_1 \neq P_0, G < G_1$.
$crit_3(P, P_0, P_1, M, M_1, G, G_1) \leftarrow P \neq P_0, P_1 \neq P_0, M < M_1$.

LP_{m_4} :
$record(Mb, Cd) \leftarrow msg(Mb, m_4, pref(Cd))$.
$act_{m_4}(C) \leftarrow prefers(m_0, C), prefers(m_1, C), prefers(m_i, C), C \neq 0.$ $(i = 2, 3)$
$act_{m_4}(C) \leftarrow prefers(m_2, C), prefers(m_3, C), C \neq 0.$
$act_{m_4}(C) \leftarrow prefers(m_0, 0), prefers(m_2, 0), prefers(m_3, C).$
$act_{m_4}(C) \leftarrow prefers(m_0, 0), prefers(m_3, 0), prefers(m_2, C).$
$act_{m_4}(C) \leftarrow prefers(m_0, 0), prefers(m_2, C_2), prefers(m_3, C_3),$
$C_2 \neq C_3, C_2 \neq 0, C_3 \neq 0.$
$act_{m_4}(C) \leftarrow prefers(m_0, C), prefers(m_2, C_2), prefers(m_3, C_3),$
$C_2 \neq C_3, C \neq 0.$

The program of c determines facts representing a set of all potential applications. Due to spontaneous choice, some subset of them is submitted through a single ADD-action for the new recruitment cycle.

We do not present the details of the secretary implementation. Its program is straightforward and determines the following 8-step recruitment cycle: (1) s deletes close *from I_s and announces* begin_rec *to c; (2) c sends to s the information on candidates; (3) s puts it into I_s and sends* begin_vote *to the members; (4) m_0 and m_2 make their choice and send it to s and other agents; (5) m_1, m_2 and m_3 make their choice and send it to s and m_4, s puts the votes of m_0 and m_2 into I_s; (6) m_4 makes his choice and sends it to s, s puts the votes of m_1 and m_3 into I_s; (7) s puts the vote of m_4 into I_s; (8) s evaluates the result of recruitment, clears I_s and puts into I_s the facts selected(C, P) and* close.

The crucial point about the examples above is that the behavior of the systems $\mathcal{RA}$ and $\mathcal{RC}$ should satisfy some important properties, e.g. the behavior of $\mathcal{RA}$ in example 1 should be *fair* in the sense that each user-agent is repeatedly served by m (i.e. served sometimes in the future after its order has been fulfilled). At the same time, system $\mathcal{RC}$ has the following two properties:
no_consecutive_lp $=_{df}$ "if an *lp*-candidate had been selected in a recruitment cycle and in the next cycle there exists a unique non-*lp*-candidate who is best in both Merits and Grants among non-*lp*-candidates, then the 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 $\mathcal{RA}$ and $\mathcal{RC}$.

3 Complexity Classes and Logics

3.1 Complexity Classes

We assume that the reader is familiar with the basic notions of computational complexity such as deterministic, nondeterministic and alternating Tur-

ing machines (DTM, NTM, ATM), their time and space complexity measures and polynomial time reducibility (see [6,23]). We use standard notation for complexity classes *P, NP, PSPACE, EXPTIME, EXPSPACE* and *NEXPTIME*. By *EXPEXPTIME* and *NEXPEXPTIME* we denote the classes of problems decidable by deterministic and nondeterministic Turing machines in time $2^{2^{pol(n)}}$ for some polynomial $pol(n)$.

For a deterministic complexity class K, AK denotes the corresponding complexity class for alternating Turing machines. It is well known that $APSPACE = EXPTIME$ and $AEXPSPACE = EXPEXPTIME$ (see [6]).

3.2 Logics for MAS Behavior Properties

We follow the tradition of using temporal logic languages of discrete time [12,19] for expressing the properties of trajectories. In particular, in order to describe properties of deterministic MAS we will use a first order extension $FLTL$ of the propositional linear time logic $PLTL$ (see [12]), and in order to describe properties of nondeterministic MAS we use a first order extension μ^{FO} of the temporal μ-calculus [21] and some of its more efficiently decidable fragments.

The syntax and semantics of all these extensions are quite similar to those of their propositional variants.

$FLTL$ contains linear temporal operators **X** ("next") and **U** ("weak until") [8].

Its formulas are defined by the rules:
(s1) Any closed formula of the first order logic is a formula of $FLTL$ (we refer to these formulas as to basic state formulas).
(s2) If ϕ_1 and ϕ_2 are formulas, then $\neg\phi_1$, $\phi_1 \wedge \phi_2$ and $\phi_1 \vee \phi_2$, $\phi_1 \mathbf{U} \phi_2$ are formulas.

While useful as the means of specifying temporal relations between events, temporal logics are not strong enough to express the properties of trajectories branching through unlimited recursion. One such property is, for instance, the existence of a winning strategy in antagonistic games. *μ-calculus* introduced in [21] is an expressive branching time logic very well suited for expressing such properties. We use a simplified single transition version of this language. The formulas of the first order μ-calculus μ^{FO} are defined by the following rules.

[8] Other usual operators **V** ("unless"), **G** ("always") and **F** ("sometime") can be defined via **U**

(1) Atomic proposition variables $\mathbf{P}, \mathbf{Q}, ...$, and basic state sentences in the signature $\Sigma_{\mathcal{A}}$ are formulas.
(2) If ϕ and ψ are formulas, then $\mathbf{EX}\ \phi$, $\neg\phi$, $\phi \wedge \psi$ are formulas.
(3) If $\phi(\mathbf{P})$ is a formula in which the propositional variable $\mathbf{P}$ has only positive occurrences (is in the scope of an even number of negations), then $\mu\mathbf{P}.\phi(\mathbf{P})$ is a formula.
Intuitively, $\mathbf{EX}\ \phi$ ($\mathbf{AX}\ \phi$) means "ϕ is true at some (any) global state one-step reachable from the current global state", and $\mu\mathbf{P}.\phi(\mathbf{P})$ ($\nu\mathbf{P}.\phi(\mathbf{P})$) stands for the least (greatest) fixpoint of $\phi(\mathbf{P})$, considering $\phi(\mathbf{P})$ as a transformer of the set of states where $\mathbf{P}$ is true to the set of states where $\phi(\mathbf{P})$ is true.

The other connectives are introduced as abbreviations in the usual way: $\mathbf{AX}\ \phi$ abbreviates $\neg\mathbf{EX}\ \neg\phi$, $\nu\mathbf{P}.\phi(\mathbf{P})$ abbreviates $\neg\mu\mathbf{P}.\neg\phi(\neg\mathbf{P})$, etc.

In addition, the usual branching time operators $\mathbf{AG}$ ("in all states of all trajectories"), $\mathbf{EG}$ ("in all states of a trajectory") and their duals $\mathbf{EF}, \mathbf{AF}$ from the well-known logic CTL can be easily expressed in μ-calculus. For example, $\mathbf{EG}\phi$ is equivalent to $\nu\mathbf{P}.(\phi \wedge EX\mathbf{P})$.

We see that $FLTL$ and μ^{FO} differ from their propositional counterparts only by the use of first order (basic state) formulas in the place of propositional variables.

Usually, the semantics of temporal formulas is defined with respect to some Kripke-like structures. In this paper, we define the validity of a temporal formula on the MAS $\mathcal{A}$ trajectory tree $\mathcal{T} = \mathcal{T}_{\mathcal{A}}(S_0)$ with the root node S_0. Given a formula ϕ, $\mathcal{T}, S \models \phi$ denotes the fact that ϕ is *valid* in state S of $\mathcal{T}$.

The $\models$ relation for both $FLTL$ and μ^{FO} is defined inductively in the same way it is defined for their propositional counterparts: only the base cases differ. Namely, let $S =< (I_{a_1}, MsgBox_{a_1}), ..., (I_{a_n}, MsgBox_{a_n}) >$ be a global MAS state of $\mathcal{T}$, and ϕ be a basic state formula. Then $\mathcal{T}, S \models \phi$ iff $\bigcup_{i=1}^{n} I_{a_i} \models_{FO} \phi$ [9] ($\models_{FO}$ corresponds to the standard first order validity).

An essential syntactic complexity parameter of formulas of μ-calculus is their *alternation depth* [13] which, roughly speaking, measures the number of consecutive alternations of nested operators μ and ν. We let μ_k denote μ-calculus restricted to formulas of alternation depth at most k. It is well known that CTL is easily translated into μ_1 (see the translation of $\mathbf{EG}$ above).

Some of our results concern logics $\exists LTL$ and $\forall LTL$ with formulas of the form $\mathbf{E}(\phi), \mathbf{A}(\phi)$, where $\phi \in FLTL$. Here $\mathbf{E}$ and $\mathbf{A}$ indicate that the linear time formula ϕ is valid in some trajectory and respectively in all trajectories of

[9] We assume that there are no name conflicts between different agents.

MAS.

The problem *"MA-BEHAVIOR"* we consider in this paper applies to both deterministic and nondeterministic MAS. Given a system $\mathcal{A}$, its initial global state S_0 and a formula Φ of a temporal logic language 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 model checking problem, though applied to MAS in the role of a transition systems specification.

Example 5 *(Example 3 continued)*
For the MAS $\mathcal{RA}$ above, the formula $\mathbf{G}\ \mathbf{F}\ receipt_i$ *is valid on the trajectory generated by $\mathcal{RA}$ (it says "every agent receives the resource infinitely often"). Meanwhile, the following two formulas are not valid on this trajectory:* $\mathbf{F}\ (receipt_1 \wedge \mathbf{X}\ receipt_1)$ *(there are two consecutive moments when u_1 receives a resource) and* $\mathbf{G}\ (first(U) \wedge next(U, u_i) \rightarrow \mathbf{X}\ \mathbf{X}\ \neg receipt_i)$.

Example 6 *(Example 4 continued)*
The properties **no_consecutive_lp** *and* **worst_selected_lp** *of the MAS $\mathcal{RC}$ above can be easily expressed in the temporal logics we use. E.g.* **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* $\mathbf{AX}$ *eight times*[10]*, and best is a 1st order formula saying that "there is a unique non-lp candidate best in both Merits and Grants":*

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

One can verify that $\mathcal{T}_{(RC)}(S_0), S_0 \models$ **no_consecutive_lp** *holds in any initial state S_0.*
The property **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 that says: "there is an ai-candidate scientifically best, an mm-candidate financially best and an lp-candidate which is the worst in both parameters" (similar to best).
One can also verify that if at least one person submits an application and somebody was already selected in the initial state S_0, then $\mathcal{T}_{(RC)}(S_0), S_0 \models$ **worst_selected_lp**.

In the case where basic state formulas are always quantifier (and object variable)-free we do not distinguish these logics from their propositional counterparts and use the same names, because their model checking and satisfiability problems have the same complexity modulo polynomial time.

4 Behavior of deterministic MAS

In this section we consider the complexity of the MA-BEHAVIOR problem for deterministic MAS. At first we present a general algorithm DetCheck that

[10] We recall that 8 is the length of one recruitment cycle.

checks validity of formulas of FLTL against deterministic MAS. This algorithm was used in [10] to obtain deterministic and nondeterministic polynomial time complexity algorithms for the MA-BEHAVIOR problem for the deterministic and nondeterministic monotonic MAS with restrictions on some structural parameters. Here in subsections 4.2 and 4.3 we apply the algorithm DetCheck to get algorithms for the more general classes of ground and nonground deterministic MAS with complexity varying from polynomial space to exponential space. For the sake of completeness we also include below the polynomial time results from [10] without proofs.

4.1 Checking validity for determininstic MAS

The set of global states of any MAS $\mathcal{A}$ is finite. So when $\mathcal{A}$ is deterministic, the trajectory $\tau(\mathcal{A}, S^0)$ is periodic. Hence, even though $\tau(\mathcal{A}, S^0)$ is infinite, it can be folded into a finite structure. A straightforward algorithm for checking an $FLTL$-formula on this structure would require an explicit representation of this structure, and consequently, the space at least equal to the total size of its global states. However, in our situation, there exists a more intelligent way of model-checking which checks the structure by parts. It allows us to obtain significantly better upper bounds for the MA-BEHAVIOR problem. We note that the idea of constructing the trajectory structure by parts resembles that of the "on-the-fly" algorithms for model checking of transition systems (see [7]). In these algorithms the structure is constructed incrementally, i.e. states of the structure are added to it only when they are needed. This allows sometimes to use less space and/or time for refutation of the formula being verified. Our algorithm is not incremental: at each moment only a relatively small part of the structure is stored in the memory. It leads to significant economy of the space needed. Model checking literature discusses some other optimization approaches, such as symbolic model checking, abstraction, use of symmetry. Here we do not consider applying any of these optimizations to the MA-BEHAVIOR problem.

For a periodic trajectory $\tau = S^0, S^1, ..., S^t, ...$, let k and N be the smallest numbers such that $S^t = S^{t+N}$ for all $t \geq k$. In our model checking algorithm, we use three auxiliary functions. The first one, *move(t,i)*, given a time point t and a shift i, returns the time point $j < k + N$ that $S^j = S^{t+i}$:

move(t,i)= IF $t + i < k + N$ THEN $t + i$ ELSE $((t + i - k) \,\mathbf{mod}\, N) + k$.

The second function, F^τ, serves as the oracle. It returns the state $F^\tau(t) = S^t$ of trajectory τ at any time point t. The third function, FO_Check(S, Φ), is boolean-valued. Given a global state S and a closed first-order formula Φ, it returns $TRUE$ iff $S \models \Phi$.

Let $\tau = \tau(\mathcal{A}, S^0)$ be a periodic trajectory with parameters k and N, Φ be a $FLTL$ formula, and t be a time point. We set $s_{max}(\tau) = \max\{|S^t| \mid 0 \leq t \leq k+N\}$ and we denote by $s(F^\tau)$ and $t(F^\tau)$ the maximal space and time required for computing $F^\tau(t)$ for $0 \leq t \leq k+N$. We also denote by $s_{FO}(\tau, n)$ and $t_{FO}(\tau, n)$ the maximal space and time required to check whether $S^t \models \Psi$ for $0 \leq t \leq k+N$ and any first-order formula Ψ of length n.

The following recursive algorithm checks the property $\tau, S^t \models \Phi$ [11].
Algorithm DetCheck$(\tau, \mathbf{k}, \mathbf{N}, \mathbf{\Phi}, \mathbf{t})$

(1) $t :=$ move$(t, 0)$; $p := 0$;
(2) $r := 0$; $r' := 0$; $R := 0$;
(3) SELECT CASE of Φ
(4) CASE Φ is a basic state formula
(5) $S^t := F^\tau(t)$;
(6) return FO_Check(S^t, Φ);
(7) CASE $\Phi = \Phi_1 \oplus \Phi_2$ ($\oplus \in \{\wedge, \vee\}$)
(8) $b_1 :=$ DetCheck(τ, k, N, Φ_1, t);
(9) $b_2 :=$ DetCheck(τ, k, N, Φ_2, t),
(10) return $b_1 \oplus b_2$;
(11) CASE $\Phi = \neg\Phi_1$
(12) return $\neg$ DetCheck(τ, k, N, Φ_1, t);
(13) CASE $\Phi = \mathbf{X}(\Phi_1)$
(14) $t_1 :=$ move$(t, 1)$;
(15) return DetCheck$(\tau, k, N, \Phi_1, t_1)$;
(16) CASE $\Phi = \Phi_1 \mathbf{U} \Phi_2$
(17) IF $t < k$ THEN $R := k + N - t$
(18) ELSE $R := N$ END_IF
(19) FOR $i = 0$ TO $R - 1$ DO
(20) $r :=$ move(t, i); $p := i$;
(21) IF DetCheck$(\tau, k, N, (\neg\Phi_1 \vee \Phi_2), r)$
(22) THEN EXIT_FOR END_IF
(23) END_ DO
(24) IF $p = R - 1$ **OR** DetCheck$(\tau, k, N, (\Phi_1 \wedge \Phi_2), r)$
(25) THEN return *TRUE*
(26) ELSE return *FALSE*
(27) END_ IF ;
(28) END_SELECT

Lemma 1 *For given numbers k, N and t and an $FLTL$-formula Φ, the algorithm DetCheck checks whether $\tau^t \models \Phi$ for a periodic trajectory τ with parameters k and N, using F^τ and FO_Check as oracles. Its computation*

[11] For a deterministic MAS $\mathcal{A}$, the tree $\mathcal{T}_\mathcal{A}(S_0)$ consists of a single trajectory $\tau = \tau_\mathcal{A}(S_0)$. At any time point t, there is a one-to-one correspondence between the state S^t of τ and the trajectory suffix $\tau^t = S^t, S^{t+1}, \ldots$. Therefore, for the uniformity purposes, we use the notation $\tau, S^t \models \Phi$ for path formulas Φ as well as for state formulas.

takes space $\mathbf{O}(|t| + |\Phi| \log(k+N) + s_{max}(\tau) + s(F^\tau) + s_{FO}(\tau, |\Phi|))$, *and time* $pol(|t| + |\Phi|(k+N)(t(F^\tau) + t_{FO}(\tau, |\Phi|))$ *for some polynomial pol.*

The oracle F^τ in the lemma can be efficiently computed along the trajectories τ generated by MAS.

Lemma 2 *There is a polynomial pol and an algorithm, which for a MAS-system* $\mathcal{A}$ *, an initial state* S^0 *and a time point* $t \geq 0$*, computes the state* S^t *of the trajectory* $\tau(\mathcal{A}, S^0)$ *in space* $pol(|\mathcal{A}| + \max\{|S^r| \mid 0 \leq r \leq t\})$.

Proof. Immediately from Proposition 1.

The next assertion shows that the trajectories of the MAS are periodic. It provides some bounds on the parameters of these trajectories.

Lemma 3 *For any MAS* $\mathcal{A}$ *and initial state* S^0*, the trajectory* $\tau(\mathcal{A}, S^0)$ *is periodic with parameters* $k(\mathcal{A}, S^0)$ *and* $N(\mathcal{A}, S^0)$*. If* $\mathcal{A}$ *is ground, then* $k(\mathcal{A}, S^0) + N(\mathcal{A}, S^0) \leq 2^{pol(|\mathcal{A}| + |S^0|)}$*. In the general case,* $k(\mathcal{A}, S^0) + N(\mathcal{A}, S^0) \leq 2^{2^{pol(|\mathcal{A}| + |S^0|)}}$.

From Lemmas 1, 2 and 3, we obtain upper complexity bounds of verification of the properties of MAS behavior, expressible in $FLTL$.

Using the Proposition 1 and Lemmas 1, 2 we can obtain by an analysis of the program DetCheck the following proposition which gives some upper complexity bounds of verification of the properties of deterministic MAS behavior, expressible in $FLTL$ (details of this analysis can be found in [10]).

Proposition 2 *Let a MAS* $\mathcal{A}$ *and an initial state* S^0 *be given. Then for some polynomial pol, the model checking of a* $FLTL$*-formula* Φ *over the trajectory* $\tau(\mathcal{A}, S^0)$ *can be accomplished within the space* $2^{pol(|\Phi| + |\mathcal{A}|)}$ *in the general case, and the space* $pol(|\Phi| + |\mathcal{A}|)$ *in the ground case.*

4.2 Ground deterministic MAS

If we suppose that MAS are ground, then by Proposition 2, it follows that the MA-BEHAVIOR problem MAS belongs to PSPACE. In this subsection we point out two interesting cases decidable in deterministic polynomial time.

Theorem 1 *(1) The MA-BEHAVIOR problem is decidable in polynomial time in the class of ground, expanding and positive MAS for the behavior properties* $\Phi \in PLTL$*.*
(2) The MA-BEHAVIOR problem is decidable in polynomial time in the class

of ground, expanding, and r-signal m-agent systems $\mathcal{A}$ *such that* $m^2 * r = \mathbf{O}(\log|\mathcal{A}|)$, *for the behavior properties* $\Phi \in PLTL$.

A proof of this theorem is given in [10]. It uses the following monotonicity lemma used also below.

Lemma 4 *Let* $\mathcal{A}$ *be an expanding and positive MAS (not necessarily ground),* S^0 *be its initial state, and* $\tau = \tau(\mathcal{A}, S^0) = S^0, S^1, \ldots S^t, S^{t+1}, \ldots$ *be its trajectory. Then for any time point* t *and two consecutive global states* $S^t =< (I^t_{a_1}, MsgBox^t_{a_1}), ..., (I^t_{a_n}, MsgBox^t_{a_n}) >$ *and* $S^{t+1} =< (I^{t+1}_{a_1}, MsgBox^{t+1}_{a_1}), ..., (I^{t+1}_{a_n}, MsgBox^{t+1}_{a_n}) >$ *of* τ, *the following inclusions hold for every* $1 \leq i \leq n$: $I^t_{a_i} \subseteq I^{t+1}_{a_i}$ *and* $MsgBox^t_{a_i} \subseteq MsgBox^{t+1}_{a_i}$.

Weakening the constraints imposed on the MAS by Theorem 1 will cause a substantial increase of complexity of the MA-BEHAVIOR problem. As we show in the next theorem, the problem becomes PSPACE-complete (which is the maximal complexity in the ground case) if only the number of agents or only the number of signals is bounded. An important consequence of this theorem is that distributivity of agents is really important: ground expanding MAS cannot be simulated in polynomial time by a single agent in this class.

Theorem 2 *(1) The MA-BEHAVIOR problem is PSPACE-complete for ground and expanding m-agent systems and the behavior properties* $\Phi \in PLTL$ *for any fixed* $m \geq 2$.
(2) The MA-BEHAVIOR problem is PSPACE-complete for ground, expanding and r-signal MAS and the behavior properties $\Phi \in PLTL$ *for any fixed* $r \geq 1$.
(3) The MA-BEHAVIOR problem is PSPACE-complete for the class of ground MAS and the properties of MAS behavior, expressible in $FLTL$.

Proof. (1) *Lower bound.* We show that any problem in PSPACE can be reduced in polynomial time to MA-BEHAVIOR problem for a ground and expanding 2-agent system. Let us fix a DTM $\mathcal{M}$ with workspace bounded by some polynomial $p(n)$, n being the length of an input word $x = a_{i_1}a_{i_2}...a_{i_n}$. We set $N = p(n)$. Let $A = \{a_0, a_1, \ldots, a_r\}$ be the tape alphabet ($a_0 = \wedge$ being the empty cell symbol) and $Q = \{q_0, \ldots, q_g\}$ be the state set of $\mathcal{M}$, in which q_0 is the initial state, q_{g-1} = *"no"* is the rejecting state, and q_g = *"yes"* is the accepting state. $P_{\mathcal{M}}$ will denote the program of $\mathcal{M}$ consisting of instructions of the form $q_j a_i \rightarrow q_u a_v S$, where $S \in \{-1, 0, 1\}$ are the head shifts. We assume that after $\mathcal{M}$ reaches one of its final states *"yes"* or *"no"*, it remains in this state forever. For any given $\mathcal{M}$ and x, we will construct an expanding 2-agent system $\mathcal{A} = \{A_1, A_2\}$, its initial state S^0, and a $PLTL$-formula Φ such that

(*) x is accepted by $\mathcal{M} \iff \tau(\mathcal{A}, S^0), S^0 \models \Phi$.

The simulation idea is that the two agents of $\mathcal{A}$ will exchange messages encoding instantaneous descriptions (*I-descriptions*) of $\mathcal{M}$ [12]. An agent receiving an I-description computes and returns the next I-description.

We fix the message signature $\mathbf{P}^m = Q \cup \{h_j \mid 1 \leq j \leq N,\ 0 \leq k \leq r\}$. These predicates describe the current state, the head position, and the symbols in the tape cells. The DB signature $\mathbf{P}^e_{A_1}$ consists of two facts $\{first, yes\}$ and $\mathbf{P}^e_{A_2} = \emptyset$. For each message $p \in \mathbf{P}^m$, the action bases AB_{A_1} and AB_{A_2} include an action with the same name p which does not change DB-state and sends p to the partner. Besides this, AB_{A_1} includes two special actions: *start* and *end*. Action *start* adds the fact *started* to I_{A_1} and sends to A_2 the starting I-description of $\mathcal{M}$:
$SEND_{start} = \{(A_2, h_1), (A_2, q_0), (A_2, a_{1,i_1}), (A_2, a_{2,i_2}), \ldots, (A_2, a_{n,i_n}),$
$(A_2, a_{n+1,0}), \ldots, (A_2, a_{N,0})\}$.
Action *end* adds to I_{A_1} the fact *yes*.

The program LP_A of agent $A \in \{A_1, A_2\}$ includes the following clauses (A' denotes the partner of A i.e. $A' = A_2$, if $A = A_1$ and $A' = A_1$ otherwise):

$a_{j,k} \leftarrow \neg msg(A', A, h_j), msg(A', A, a_{j,k})$

for all $1 \leq j \leq N,\ 0 \leq k \leq r$.
For each instruction $q_i a_k \rightarrow q_u a_v S \in P_{\mathcal{M}}$ and any $1 \leq j \leq N$, the program LP_A includes the set of clauses:

$q_u \leftarrow msg(A', A, h_j), msg(A', A, a_{j,k}), msg(A', A, q_i)$
$h_{j+S} \leftarrow msg(A', A, h_j), msg(A', A, a_{j,k}), msg(A', A, q_i)$
$a_{j,v} \leftarrow msg(A', A, h_j), msg(A', A, a_{j,k}), msg(A', A, q_i)$.

Besides these, the program LP_{A_1} includes two clauses:

$start \leftarrow \neg started$
$end \leftarrow msg(A', A, q_g)$.

Both agents $A_i, i = 1, 2$, here [13] use the total obligation operator $Sel^{td}_{A_i}$.

The property to check is expressed by the formula $\Phi = \mathbf{F}\ yes_{A_1}$ stating that the fact *yes* will appear eventually in I_{A_1}. The in initial state S^0 of $\mathcal{A}$ is empty: $I_{A_1} = I_{A_2} = \emptyset$. Let $\tau = \tau(\mathcal{A}, S^0) = S^0, S^1, \ldots, S^t, \ldots$ be the trajectory of $\mathcal{A}$ starting in this initial state. Let $\sigma = C_1, C_2, \ldots, C_t, \ldots$ be the sequence of I-descriptions representing the computation of $\mathcal{M}$ on the input x (C_1 represents the initial I-description in which the head in the state q_0 observes the first tape cell and the symbols of x are contained in the first n cells of the tape). The following assertion establishes the relationship between these two sequences.

Lemma 5 At any moment $t > 0$, the set of messages in $MsgBox^t_{A_i},\ i = (t$

[12] An I-description at a moment t is a word coding the global computation state at this moment: the current control state, the head position and the symbols written in tape cells.
[13] As well as all other agents in this section on deterministic MAS

mod 2) + 1, completely determines the I-description C_t, i.e. :
$(A', a_{j,k}) \in MsgBox^t_{A_i} \Leftrightarrow$ at the moment t, the cell j contains a_k,
$(A', h_j) \in MsgBox^t_{A_i} \Leftrightarrow$ at the moment t, $\mathcal{M}$ observes the cell j,
$(A', q_j) \in MsgBox^t_{A_i} \Leftrightarrow$ at the moment t, $\mathcal{M}$ is in the state q_j.

This lemma can be proven by a straightforward induction on t.

Now, the assertion (*) easily follows from lemma 5. Indeed,
x is accepted by $\mathcal{M} \Leftrightarrow$ there is an odd t such that at the step t $\mathcal{M}$ is in the state $q_g \Leftrightarrow (A_2, q_g) \in MsgBox^t_{A_1} \Leftrightarrow$ at the even step $t+1$, the action *end* is fired and inserts *yes* into $I^{t+1}_{A_1} \Leftrightarrow \tau(\mathcal{A}, S^0), S^0 \models \Phi$.

It is easy to see that $\mathcal{A}, S^0$ and Φ can be constructed from x in time polynomial in $|x|$ ($\mathcal{M}$ is fixed for all x). Therefore, MA-BEHAVIOR problem for ground and expanding 2-agent systems is PSPACE-hard.

(2) *Lower bound*. In this case, we first prove that any problem in PSPACE is reducible in polynomial time to MA-BEHAVIOR problem for ground, expanding and r-signal MAS for some fixed r. Then we show how to reduce the number r of signals to 1.

Let C be a PSPACE-complete problem and $\mathcal{M}$ be a DTM which recognizes C and works in space bounded by some polynomial $p(n)$, n being the input word's length $x = a_{i_1}a_{i_2}...a_{i_n}$. We set $N = p(n)$. We use the notation of the case (1) of the theorem for the alphabets of $\mathcal{M}$. As in the case (1), we construct from given $\mathcal{M}$ and x an expanding r-signal MAS $\mathcal{A}$, its initial state S^0 and a $PLTL$-formula Φ such that

(**) x is accepted by $\mathcal{M} \Leftrightarrow \tau(\mathcal{A}, S^0), S^0 \models \Phi$.

$\mathcal{A}$ consists of $N+1$ agents: $c_1, \ldots, c_N, s$. The first N agents c_k (cell-agents) simulate the corresponding tape cells k of $\mathcal{M}$. The information about the symbol contained in the cell k will be held in $MsgBox_{c_k}$. This time, the supervisor-agent s will receive messages from the cell-agents, compute the next I-description and return it to the corresponding agents.

The DB-states of cell-agents are always empty. The DB signature $\mathbf{P}^e_s$ includes two facts $\{started, yes\}$. We fix the message signature $\mathbf{P}^m = A \cup Q \cup \{h\}$. Intuitively, message a_i received by c_k from s means that the symbol a_i is written into cell k, message q_j received by c_k from s means that the current state of $\mathcal{M}$ is q_j, and message h received by c_k from s means that the head of $\mathcal{M}$ observes it. When a cell-agent receives some messages from s, it simply returns them to s at the next step. So the action base AB_{c_k} consists of actions $\{a_i |\ 0 \le i \le r\} \cup \{q_j |\ 1 \le j \le g\} \cup h$, and each action $ac \in AB_{c_k}$ puts the message (k, ac) into the message box of s.

Program LP_{c_k} of each $c_k,\ 1 \le k \le N,$ includes $(r+1)$ clauses of the form:

$$a_i \leftarrow msg(s, c_k, a_i), \neg msg(s, c_k, h), \quad (\ 0 \le i \le r),$$

$(g+1)$ clauses of the form:

$$q_j \leftarrow msg(s, c_k, q_j), \quad (0 \le j \le g),$$

and the clause

$$h \leftarrow msg(s, c_k, h).$$

For each $a_i \in A$, we supply the supervisor action base AB_s by N actions $s(k, a_i),\ 1 \le k \le N,$ each sending message a_i to the corresponding agent c_k. For each instruction $q_j a_i \rightarrow q_u a_v S \in P_{\mathcal{M}},$ the supervisor action base AB_s includes N actions $as(k, j, i),\ 1 \le k \le N.$ Action $as(k, j, i)$ sends message a_v to c_k and messages q_u and h to c_{k+S}. If $j = g$, then this action also adds to I_s the fact *yes*. Besides these, AB_s also includes action *start* adding to I_s the fact *started* and sending to cell-agents the facts determining the starting I-description of $\mathcal{M}$:

$$\text{SEND}(start) = \{(c_1, h), (c_1, q_0), (c_1, a_{i_1}),\ (c_2, a_{i_2}), \ldots,\ (c_n, a_{i_n}),\\ (c_{n+1}, a_0), \ldots, (c_N, a_0)\}.$$

Program LP_s has the clauses:

$$start \leftarrow \neg started,$$

$$s(k, a_i) \leftarrow msg(c_k, s, a_i), \neg msg(c_k, s, h)$$

for all $1 \le k \le N, a_i \in A$, and

$$as(k, j, i) \leftarrow msg(c_k, s, a_i), msg(c_k, s, q_j), msg(c_k, s, h)$$

for all $1 \le k \le N, 0 \le j \le g, 0 \le i \le r$.

As in case (1), the property to verify is expressed by the formula $\Phi = \mathbf{F}\ yes_s$ stating the fact that *yes* will appear eventually in I_s. Let all the agents' DB-states in the initial global state S^0 of $\mathcal{A}$ be empty. Let $\tau = \tau(\mathcal{A}, S^0) = S^0, S^1, \ldots, S^t, \ldots$ be a trajectory of $\mathcal{A}$ starting in this empty initial state. Let $\sigma = C_1, C_2, \ldots, C_t, \ldots$ be the sequence of I-descriptions constituting the computation of $\mathcal{M}$ on input x (in the starting I-description C_1, the current state is q_0, the head observes the first tape cell, the symbols of x are contained in the first n cells of the tape). The relationship between these two sequences is expressed by the following assertion.

Lemma 6 *At any moment $t > 0$, the set of messages in $MsgBox_s^{2t}$ completely determines the I-description C_t, i.e.*
$(c_k, a_i) \in MsgBox_s^{2t} \Leftrightarrow$ at the moment t, the cell k contains a_i,
$(c_k, h) \in MsgBox_s^{2t} \Leftrightarrow$ at the moment t, the head of $\mathcal{M}$ observes the cell k,
$(c_k, q_j) \in MsgBox_s^{2t} \Leftrightarrow$ at the moment t, $\mathcal{M}$ is in the state q_j.

This lemma can be proven by a straightforward induction on t. The induction condition should be extended as follows:
At any moment $t > 0$, the set of messages in $MsgBox_{c_k}^{2t-1}, 1 \le k \le N,$ completely determines the I-description C_t.

Lemma 6 implies the assertion (**) just as the assertion (*) follows from lemma 5 in case (1). It is easy to check that the number of rules in programs of all agents of $\mathcal{A}$ equals $2N(g+1)(r+1)+N(r+1)+1$. Since $N = p(|x|)$, the size of $\mathcal{A}$ is bounded by some polynomial in $|x|$. It is also evident that $\mathcal{A}, S^0$ and Φ can be constructed from x in time polynomial in $|x|$. Let $f = |\mathbf{P}^m| = g+r+3$. Then our construction shows that the MA-BEHAVIOR problem for ground, expanding and f-signal MAS is PSPACE-hard.

Now we outline the construction for a given $\mathcal{A}$ of a MAS $\mathcal{A}'$ which uses only one message 1 and is in a sense equivalent to $\mathcal{A}$. Each cell-agent c_k of $\mathcal{A}$ is replaced by f agents (one for each signal): $\{c_k(a_i) \mid 0 \leq i \leq r\}$ $\cup\{c_k(q_j) \mid 0 \leq j \leq g\} \cup \{c_k(h)\}$. Each of these agents $c_k(m)$ has only one action *act* sending the message 1 to s. Program $LP_{c_k(m)}$ consists of a single clause:

$$act \leftarrow msg(s, c_k(m), 1).$$

The supervisor s has the following actions:
Action $s(k, a_i)$, $1 \leq k \leq N$, sending message 1 to agent $c_k(a_i)$.
Action $as(k, j, i)$ simulating the instruction $q_j a_i \rightarrow q_u a_v S \in P_{\mathcal{M}}$ sends message 1 to $c_k(a_v)$, to $c_{k+S}(q_u)$ and to $c_{k+S}(h)$. If $j = g$, then it also adds *yes* to I_s.
Action *start* adds to I_s the fact *started* and sends the message 1 to the agents $c_1(h), c_1(q_0), c_1(a_{i_1})$, $c_2(a_{i_2}), \ldots,$ $c_n(a_{i_n}), c_{n+1}(a_0), \ldots, c_N(a_0)$.

Program LP'_s has the following clauses:

$start \leftarrow \neg started$,

$s(k, a_i) \leftarrow msg(c_k(a_i), s, 1), \neg msg(c_k(h), s, 1)$,
$(1 \leq k \leq N, a_i \in A)$,

$as(k, j, i) \leftarrow msg(c_k(a_i), s, 1), msg(c_k(q_j), s, 1), msg(c_k(h), s, 1)$,
$(1 \leq k \leq N, 0 \leq j \leq g, 0 \leq i \leq r)$.

Let all the individual agents' DB-states be empty in the initial global state S'^0 of $\mathcal{A}'$. Then, as one can easily verify, $\tau(\mathcal{A}, S^0), S^0 \models \Phi \Leftrightarrow \tau(\mathcal{A}', S'^0), S'^0 \models \Phi$. Therefore, the MA-BEHAVIOR problem for ground, expanding and 1-signal MAS is PSPACE-hard.

(3) The upper bound directly follows from proposition 2, and the lower bound follows from assertions (1) or (2) of the theorem. □.

4.3 Nonground deterministic MAS

In this subsection, we lift the constraint of groundness and study the deterministic MAS whose programs may have rules with variables. We start by establishing the complexity of the MA-BEHAVIOR problem for the systems

with bounded arity of the predicates.

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

Proof. (1) Let $\mathcal{A}$ be an expanding, positive and k-dimensional MAS, S^0 be its initial state, $\tau = \tau(\mathcal{A}, S^0) = S^0, S^1, \ldots S^t, S^{t+1}, \ldots$ be its trajectory, n_{act} be the total number of possible ground actions and n_g be the total number of possible ground atoms in DB-states and in message boxes of agents in $\mathcal{A}$. By lemma 4, the inclusions $I_a^t \subseteq I_a^{t+1}$ and $MsgBox_a^t \subseteq MsgBox_a^{t+1}$ hold for each $a \in \mathcal{A}$ and all moments t. Therefore, if $S^t \neq S^{t+1}$, then there is a ground action α such that $\alpha \in Act_a^t \setminus Act_a^{t-1}$ for at least one agent $a \in \mathcal{A}$. Hence, the sum of parameters $k(\tau) + N(\tau)$ does not exceed n_{act} and the polynomial time bound of the theorem follows directly from lemma 1 and the following simple assertion.

Lemma 7 *Let n_{act} be the total number of possible ground actions and n_g be the total number of possible ground atoms in DB-states and in message boxes of agents in $\mathcal{A}$. Then for all k, there is a polynomial pol such that for any k-dimensional MAS $\mathcal{A}$ and a starting global state S_0, $n_{act} \leq pol(|S^0| + |\mathcal{A}|)$, $n_g \leq pol(|S^0| + |\mathcal{A}|)$ and $s^\tau_{max} \leq n\, pol(|S^0| + |\mathcal{A}|)$, where $\tau = \tau(\mathcal{A}, S^0)$ and n is the number of agents in $\mathcal{A}$.*

(2) Let $\mathcal{A}$ be a k-dimensional MAS, S^0 be some its initial state, and $\tau = \tau(\mathcal{A}, S^0) = S^0, S^1, \ldots S^t, S^{t+1}, \ldots$ be its trajectory. From lemma 7 it follows that $k(\tau) + N(\tau) \leq 2^{pol(|S^0|+|\mathcal{A}|)}$ for some polynomial pol. Then by lemma 1, we get a polynomial space upper bound for algorithm DetCheck. The lower bound follows from theorem 2, since ground MAS are in fact, 0-dimensional agents systems. □

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

Proof. *Upper bound.* Let $\mathcal{A}$ be an expanding and positive MAS, S^0 be its initial state, and $\tau = \tau(\mathcal{A}, S^0) = S^0, S^1, \ldots S^t, S^{t+1}, \ldots$ be its trajectory. As in the proof of theorem 3, $k(\tau) + N(\tau) \leq n_{act}$ and $n_{act} \leq 2^{pol(|S^0|+|\mathcal{A}|)}$ for some polynomial pol. The size of any state $S^t \in \tau$ is also bounded by $2^{pol(|S^0|+|\mathcal{A}|)}$. By our convention, all basic state subformulas of the $PLTL$-formula Φ are ground. Therefore, the time $t_{FO}(\tau, |\Phi|)$ needed to check the

first order subformulas of Φ on any state $S^t \in \tau$ is bounded by $pol_1(|S^t|)$. Now, by lemmas 1 and 2, we see that algorithm DetCheck has in this case the time bound $pol(|\Phi|(k+N)(t(F^\tau)+t_{FO}(\tau,|\Phi|)) \leq 2^{pol'(|\Phi|+|S^0|+|\mathcal{A}|)}$ for some polynomial pol'.

Lower bound. We will prove that even an expanding and positive 1-agent system can simulate any DTM running in exponential time. Let us fix a DTM $\mathcal{M}$ which works in time bounded by $2^{p(n)}$ for some polynomial $p(n)$, n being the length of an input word $x = a_{i_1}a_{i_2}...a_{i_n}$. We set $N = p(n)$.

We use in our construction the notation and the agreements in the proof of theorem 2 (1).

Given $\mathcal{M}$ and x, we construct an expanding and positive 1-agent system $\mathcal{A} = \{A\}$, its initial state S^0 and a $PLTL$-formula Φ such that

(*) x is accepted by $\mathcal{M} \iff \tau(\mathcal{A}, S^0), S^0 \models \Phi$.

Let $\overline{X}, \overline{Y}, \overline{T}, \overline{S}, \ldots$ denote lists of N boolean variables $(X_1, \ldots, X_N)$, $(Y_1, \ldots, Y_N), (T_1, \ldots, T_N), (S_1, \ldots, S_N) \ldots$. Let $\overline{m}$ denote the list of boolean constants $0, 1$ used as the binary representation of an integer $m, 0 \leq m \leq 2^N - 1$.

We fix the DB signature $\mathbf{P}^e_A = \{q_j(\overline{T}) | q_j \in Q\} \cup \{h(\overline{T}, \overline{S}), step(\overline{T}), yes\} \cup \{a_i(\overline{T}, \overline{S}) \mid 0 \leq i \leq r\}$. Intuitively, the atom $q_j(\bar{t})$ describes the state of $\mathcal{M}$ at the moment t, the atom $step(\bar{t})$ defines the current step t, the atom $h(\bar{t}, \bar{s})$ states that the head observes the cell s at the step t, the atom $a_i(\bar{t}, \bar{s})$ states that symbol a_i is written in the cell s at the step t, and the atom yes states that $\mathcal{M}$ accepts the input.

For each $p \in \mathbf{P}^e_A$, the action base AB_A contains an action with the same name p adding to DB-state I_A the fact p. It contains also the action end adding to I_A the fact yes.

The Program LP_A of agent A is positive. It defines two auxiliary predicates computing elementary arithmetic functions over the domain $[0, 2^N - 1]$:
$next(\overline{X}, \overline{Y}) \Leftrightarrow \overline{Y} = \overline{X + 1}$,
$shift(C, \overline{X}, \overline{Y}) \Leftrightarrow \overline{Y} = \overline{X + C}$, where $C \in \{-1, 0, 1\}$.

E.g., the function $next$ is defined by the clauses:
$next(X_1, ..., X_{N-1}, 0, X_1, ...X_{N-1}, 1)$.
$next(X_1, ..X_{N-2}, 0, 1, X_1, ...X_{N-2}, 1, 0)$.
. . .
$next(X_1, ..X_j, 0, 1, ..1, X_1, ...X_j, 1, 0, ..., 0)$.
. . .
$next(1, ...1, 1, ..., 1)$.

Besides this, it uses the facts $right(\overline{T}, \overline{S})$ and $left(\overline{T}, \overline{S})$ marking respectively the next and the preceding positions with respect to the head position at the moment T. The following two clauses of LP_A serve to recursively derive intensional facts marking the positions to the right (respectively to the left) of a position marked at a moment T :

$right(\overline{T}, \overline{S}) \leftarrow next(\overline{T}, \overline{S})$.
$right(\overline{T}, \overline{S}) \leftarrow right(\overline{T}, \overline{S_1}), next(\overline{S_1}, \overline{S})$.
$left(\overline{T}, \overline{S}) \leftarrow next(\overline{S}, \overline{T})$.
$left(\overline{T}, \overline{S}) \leftarrow left(\overline{T}, \overline{S_1}), next(\overline{S}, \overline{S_1})$.

The next group of clauses determines the initial DB state I_A of A :

$q_0(\overline{0})$. $\quad a_{i_1}(\overline{0}, \overline{1})$. $\quad a_0(\overline{0}, \overline{S}) \leftarrow right(\bar{n}, \bar{S})$
$h(\overline{0}, \overline{1})$. $\quad \ldots\ step(\overline{0})$. $\quad a_{i_n}(\overline{0}, \bar{n})$.

It is easy to see that the state I_A^1 represents exactly the starting I-description of $\mathcal{M}$ on the input x. The following clause of LP_A simulates the step counter:

$step(\overline{T}) \leftarrow step(\overline{T}_1), next(\overline{T}_1, \overline{T})$.

For each instruction $q_i a_k \rightarrow q_u a_v C$ of $\mathcal{M}$, the program LP_A contains five clauses simulating it:

$q_u(\overline{T}) \leftarrow next(\overline{T}, \overline{T}_1), q_i(\overline{T}_1), h(\overline{T}_1, \overline{S}), a_k(\overline{T}_1, \overline{S})$
$h(\overline{T}, \overline{S}_1) \leftarrow next(\overline{T}, \overline{T}_1), h(\overline{T}_1, \overline{S}), a_k(\overline{T}_1, \overline{S}), shift(C, \overline{S}, \overline{S}_1)$
$a_v(\overline{T}, \overline{S}) \leftarrow next(\overline{T}, \overline{T}_1), h(\overline{T}_1, \overline{S}), a_k(\overline{T}_1, \overline{S})$
$right(\overline{T}, \overline{S}_1) \leftarrow next(\overline{T}, \overline{T}_1), h(\overline{T}_1, \overline{S}), next(\overline{S}, \overline{S}_1)$
$left(\overline{T}, \overline{S}_1) \leftarrow next(\overline{T}, \overline{T}_1), h(\overline{T}_1, \overline{S}), next(\overline{S}_1, \overline{S})$

The clauses:

$a_k(\overline{T}, \overline{S}) \leftarrow next(\overline{T}, \overline{T}_1), right(\overline{T}, \overline{S}), a_k(\overline{T}_1, \overline{S})$
$a_k(\overline{T}, \overline{S}) \leftarrow next(\overline{T}, \overline{T}_1), left(\overline{T}, \overline{S}), a_k(\overline{T}_1, \overline{S})$
$(0 \leq k \leq r)$
serve to derive intensional facts stating that the symbol a_k is left unchanged in the cells to the right and to the left of a given position S.

Finally, the clause:
$end \leftarrow q_g(\overline{T})$
fires action *end* when the fact q_g appears in the current DB state.

We choose the formula $\Phi = \mathbf{F}\ yes_A$ as the property to verify. This formula states that the fact *yes* will appear eventually in I_A.

Let $\tau = \tau(\mathcal{A}, S^0) = S^0, S^1, \ldots, S^t, \ldots$ be the trajectory of $\mathcal{A}$ starting in the empty initial state $S^0 = \emptyset$. Let $\sigma = C_1, C_2, \ldots, C_t, \ldots$ be the sequence of

I-descriptions coding the computation of $\mathcal{M}$ on input x. In particular, this means that C_1 represents the initial I-description, where the current state is the starting state q_0, the head observes the first tape cell, the symbols of x are written in the first n consecutive cells, the resting cells containing a_0. The following assertion establishes the relationship between these two sequences.

Lemma 8 *(1)* $\max\{t' | step(\overline{t'}) \in I_A^t\} = t$ *for all* $t > 0$.
(2) For any $t > 0$, *the subset of facts of* I_A^t *with the time mark* $\bar{t}$ *completely determines the I-description* C_t, *i.e. :*
$a_k(\bar{t}, \bar{s}) \in I_A^t \Leftrightarrow$ *the tape cell* s *contains the symbol* a_k *at the moment* t,
$h(\bar{t}, \bar{s}) \in I_A^t \Leftrightarrow$ *the head observes the tape cell* s *at the moment* t, *and*
$q_i(\bar{t}) \in I_A^t \Leftrightarrow \mathcal{M}$ *is in the state* q_i *at the moment* t.

This lemma can be proven by a straightforward induction on t.

Now, the assertion (*) easily follows from lemma 8. Indeed, x is accepted by $\mathcal{M}$ $\Leftrightarrow$ there is a step $t \leq 2^N - 1$, where $\mathcal{M}$ reaches the accepting state q_g. By lemma 8, $q_g(\bar{t}) \in I_A^t$. Therefore, the action *end* is fired at the step $t + 1$, which adds *yes* to I_A^{t+1}. Hence, $\tau(\mathcal{A}, S^0), S^0 \models \Phi$. Conversely, if $\tau(\mathcal{A}, S^0), S^0 \models \Phi$ then lemma 8 implies that $\mathcal{M}$ starting with x reaches at some step the accepting state q_g.

It is easy to see that the size of $\mathcal{A}$ is bounded by some polynomial in $|x|$ and that $\mathcal{A}, S^0$ and Φ can be constructed from x and $\mathcal{M}$ in time polynomial in $|x|$. Therefore, by (*), it follows that the MA-BEHAVIOR problem for expanding and positive MAS is EXPTIME-hard. □

In fact, our proof shows that the MA-BEHAVIOR problem is intractable even in a very narrow class of MAS.

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

So it is no wonder that in the general case the problem is still more hard.

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

Proof. *Upper bound* follows immediately from proposition 2.
Lower bound. We will show that deterministic MAS can simulate DTM running in EXPSPACE. Let us fix a DTM $\mathcal{M}$ working in space bounded by $2^{p(n)}$ for some polynomial $p(n)$, n being the length of an input word $x = a_{i_1} a_{i_2} ... a_{i_n}$. We set $N = p(n)$.

In order to construct a MAS $\mathcal{A}$ simulating $\mathcal{M}$, we combine the ideas of lower bound proofs in theorems 2 (1) and 4. The constructed MAS $\mathcal{A} = \{A_1, A_2\}$ has two almost identical agents A_1 and A_2. They send each other in turn the I-descriptions of $\mathcal{M}$. The first agent a_1 has the DB signature $\mathbf{P}^e_{A_1} = \{started, yes\}$ and the DB signature $\mathbf{P}^e_{A_2}$ of the other is empty. We fix the message signature $\mathbf{P}^m = \{q_j \mid q_j \in Q\} \cup \{h(\overline{S}),\, yes\} \cup \{a_i(\overline{S}) \mid 0 \leq i \leq r\}$.

As in the proof of theorem 2 (1), for each $p \in \mathbf{P}^m$, the action bases AB_{A_1} and AB_{A_2} have an action named p, which does not change DB-states and sends p to the partner. AB_1 has also action *start* adding the fact *started* to I_{A_1}, and action *end* adding the fact *yes* to I_{A_1}.

Both programs LP_{A_i} $(i = 1, 2)$ define auxiliary predicates *next, shift, right* and *left* as in the proof of theorem 4. The following clauses of A_1 serve to send the starting I-description of $\mathcal{M}$ to A_2:

$q_0 \leftarrow \neg started$ $\quad$ $a_{i_1}(\overline{1}) \leftarrow \neg started$ $\quad$ $a_0(\overline{S}) \leftarrow right(\overline{n}, \overline{S}), \neg started$
$h(\overline{1}) \leftarrow \neg started$ $\quad$ $\ldots$
$start \leftarrow \neg started$ $\quad$ $a_{i_n}(\overline{n}) \leftarrow \neg started$

The last clause prevents from sending the starting I-description repeatedly. It is easy to see that on receiving these messages, the message box $MsgBox^1_{A_2}$ completely determines the starting I-description of $\mathcal{M}$ on input x.

In both programs LP_A $(A \in \mathcal{A})$, each instruction $q_i a_k \rightarrow q_u a_v C$ of $\mathcal{M}$ is simulated by the clauses:

$q_u \leftarrow msg(A', A, h(\overline{S})), msg(A', A, a_k(\overline{S})), msg(A', A, q_i)$
$h(\overline{S}_1) \leftarrow msg(A', A, h(\overline{S})), msg(A', A, a_k(\overline{S})), msg(A', A, q_i),\ shift(C, \overline{S}, \overline{S}_1)$
$a_v(\overline{S}) \leftarrow msg(A', A, h(\overline{S})), msg(A', A, a_k(\overline{S})), msg(A', A, q_i)$

The following clauses serve for informing the partner about cells left unchanged:

$a_k(\overline{S}_1) \leftarrow msg(A', A, h(\overline{S})), right(\overline{S}, \overline{S}_1), msg(A', A, a_k(\overline{S}_1))$ $\ (0 \leq k \leq r)$.
$a_k(\overline{S}_1) \leftarrow msg(A', A, h(\overline{S})), left(\overline{S}_1, \overline{S}), msg(A', A, a_k(\overline{S}_1))$ $\ (0 \leq k \leq r)$.

Finally, using the clause:

$end \leftarrow msg(A_2, A_1, q_g)$

the agent A_1 fires action *end* on receipt of the message stating that $\mathcal{M}$ has passed to the accepting state q_g.

As before, we choose the formula $\Phi = \mathbf{F}\ yes_{A_1}$ as the property to check. It states that the fact *yes* will appear eventually in I_{A_1}.

Let $\tau = \tau(\mathcal{A}, S^0) = S^0, S^1, \ldots, S^t, \ldots$ be the trajectory of $\mathcal{A}$ starting in the empty state $S^0 : \ I_{A_1} = I_{A_2} = \emptyset$. Let $\sigma = C_1, C_2, \ldots, C_t, \ldots$ be the sequence of I-descriptions coding the computation of $\mathcal{M}$ on input x. The following assertion establishes the relationship between the two sequences.

Lemma 9 *At each moment $t > 0$, the set of messages in $MsgBox^t_{A_i}$, $i = (t \bmod 2) + 1$), completely determines the I-description C_t, i.e.:*
$(A', a_k(\overline{s})) \in MsgBox^t_{A_i} \Leftrightarrow$ the tape cell s contains symbol a_k at the moment t,
$(A', h(\overline{s})) \in MsgBox^t_{A_i} \Leftrightarrow$ the head observes the tape cell s at the moment t, and
$(A', q_i) \in MsgBox^t_{A_i} \Leftrightarrow \mathcal{M}$ is in the state q_i at the moment t.

This lemma can be proven by a straightforward induction on t. Lemma 8 immediately implies that $\mathcal{M}$ accepts x iff $\tau(\mathcal{A}, S^0), S^0 \models \Phi$.

It is easy to check that the size of $\mathcal{A}$ is bounded by some polynomial in $|x|$ and that $\mathcal{A}, S^0$ and Φ can be constructed from x and $\mathcal{M}$ in time polynomial in $|x|$. So MA-BEHAVIOR problem is EXPSPACE-hard in the class of deterministic MAS.

5 Behavior of nondeterministic MAS

We start with a simple observation which will serve to adapt some well known model checking complexity results (e.g., see [13]) to upper complexity bounds for the MA-BEHAVIOR problem. Clearly, there is a simple algorithm T which, given a MAS A and its global state S_0, constructs a transition system $T(A, S_0)$ (in time polynomial in $|T(A, S_0)|$) such that:
1) the set of states of $T(A, S_0)$ coincides with the set of global states of A,
2) the set of trajectories of A starting in S_0 coincides with the set of trajectories of $T(A, S_0)$.

This remark relates the model checking complexity results with upper complexity bounds of MA-BEHAVIOR problem for some classes of MAS. In particular, we will use the following simple assertion.

Proposition 3
If $|T(A, S_0)| < f(|A|)$ for any system A in a class of MAS, and the model checking of a formula ϕ of a logic L on a transition system TS is executable with complexity $g(|TS|, |\phi|)$, then the complexity of MA-BEHAVIOR problem for this class of MAS and the logic L is bounded by $g(f(|A|), |\phi|)$.

5.1 Ground nondeterministic MAS

As for deterministic MAS above, we start with the ground case.

Theorem 6 *The MA-BEHAVIOR problem for the class of ground, expanding, and r-signal m-agent systems $\mathcal{A}$ such that $m^2 * r = \mathbf{O}(\log|\mathcal{A}|)$ and behavior properties $\Phi \in \exists LTL(\forall LTL)$ is NP-complete (respectively coNP-complete).*

A proof of this theorem is contained in [10]. It uses the following technical lemma used also below.

Lemma 10 *There is a polynomial $p(n)$ such that $\mathcal{T}_{\mathcal{A}}(S^0) \models \mathbf{E}\ \Psi$ iff there is a trajectory $\tau = S^0, \ldots, S^t, \ldots \in \mathcal{T}_{\mathcal{A}}(S^0)$ and a step $T \leq p(|\mathcal{A}| + |\Psi|)$ such that $\tau, S^0 \models \Psi$ and $I^t_a = I^T_a$ for all $t > T$ and every $a \in \mathcal{A}$.*

The next theorem shows that the complexity of MA-BEHAVIOR problem increases substantially if we weaken requirements to MAS. If in the class of expanding systems, we restrict only the number of agents or only the number of signals, then the problem becomes APSPACE-complete as it is in the general ground case. APSPACE denotes the set of problems decidable in polynomial space by ATM [6].

Theorem 7 *(1) The MA-BEHAVIOR problem is APSPACE (EXPTIME)-complete for nondeterministic ground and expanding m-agent systems and the behavior properties Φ in μ^{FO}_1 (for each fixed $m \geq 2$).*
(2) The MA-BEHAVIOR problem is APSPACE (EXPTIME)-complete for ground, expanding and r-signal MAS and the behavior properties Φ in μ^{FO}_1 (for each fixed $r \geq 1$).
(3) In both cases there exists a constant $c > 1$ such that the MA-BEHAVIOR problems are not decidable with deterministic time complexity $c^{n/logn}$.
(4) The MA-BEHAVIOR problem for ground MAS is:
(i) EXPTIME-complete for behavior properties Φ in μ^{FO}_r (for any fixed r);
(ii) in NEXPTIME $\cap$ coNEXPTIME for behavior properties Φ in μ^{FO}.

Proof. (1) *Lower bound.* The proof is a modified version of the proof of theorem 2 (1). Namely, we show that any problem in ASPACE can be reduced in polynomial time to the MA-BEHAVIOR problem for a nondeterministic ground and expanding 2-agent system.

Let us fix an ATM $\mathcal{M}$ working in space bounded by some polynomial $p(n)$, n being the length of an input word $x = a_{i_1}a_{i_2}...a_{i_n}$. We set $N = p(n)$. Without loss of generality, we can suppose that:
- the set $Q = \{q_0, \ldots, q_g\}$ of states of $\mathcal{M}$ is decomposed into two disjoint

subsets $Q = Q_u \cup Q_e$ of respectively universal and existential states, the starting state q_0 is universal, the accepting state q_{g-1} = *"yes"* and the rejecting state q_g = *"no"* are both existential,

- the set of instructions of $\mathcal{M}$ is also decomposed into two disjoint subsets $\mathcal{M} = \mathcal{M}_u \cup \mathcal{M}_e$, each instruction $q_j a_i \rightarrow q_u a_v S \in \mathcal{M}_u$ having $q_j \in Q_u$ and $q_u \in Q_e$ and each instruction $q_j a_i \rightarrow q_u a_v S \in \mathcal{M}_e$ having $q_j \in Q_e$ and $q_u \in Q_u$;

- each computation of $\mathcal{M}$ reaches either the success state *"yes"* or the failure state *"no"*, and neither of these states is in left hand sides of instructions.

Let $A = \{a_0, a_1, \ldots, a_r\}$ be the tape alphabet ($a_0 = \wedge$ being the empty cell symbol). From $\mathcal{M}$ and x, we will construct a ground expanding 2-agent system $\mathcal{A} = \mathcal{A}(M, x) = \{A_u, A_e\}$, its initial state S^0 and a μ_1-formula Φ such that

(*) x is accepted by $\mathcal{M} \iff \mathcal{T}_{\mathcal{A}}(S^0), S^0 \models \Phi$.

We follow the idea of the construction of theorem 2: the two agents in $\mathcal{A}$ send each other in turn I-descriptions of $\mathcal{M}$. When an agent receives the current I-description it computes the next I-description and sends it to its partner. The agent A_u will simulate the instructions of $\mathcal{M}_u$ and A_e those of $\mathcal{M}_e$.

We fix the message signature: $\mathbf{P}^m = Q \cup \{h_k \mid 1 \leq k \leq N\} \cup \{a_{k,j} \mid 1 \leq k \leq N, 0 \leq j \leq r\}$. These predicates describe the state, the head position, and the symbols written in the tape cells. The DB signatures of $\mathbf{P}_{A_u}$ and $\mathbf{P}_{A_e}$ consist of 0-ary predicates *started* and *success*.

Both action bases AB_{A_u} and AB_{A_e} contain $\mathbf{P}^m$, each action $p \in \mathbf{P}^m$ just sending p to the partner without changing the DB-state. Let $\mathcal{M}$ contain L instructions with left hand side $q_j a_i$, and the instruction $q_j a_i \rightarrow q_u a_v S$ has the number $l, 1 \leq l \leq L$. Let $v_{i,j,k,l}$, $1 \leq k \leq N$, denote the action which sends to the partner agent the messages $a_{k,v}, h_{k+S}, q_u$, and adds the fact *success* to the DB-state if $u = g-1$. Then the action $v_{i,j,k,l}$ belongs to AB_{A_u} if $q_j \in Q_u$ and to AB_{A_e}, otherwise.

Besides this, AB_{A_u} has a special action *start* which adds the fact *started* to I_{A_u} and sends to A_e the starting I-description of $\mathcal{M}$:
$\mathrm{SEND}(start) = \{(A_e, h_1), (A_e, q_0), (A_e, a_{1,i_1}), (A_e, a_{2,i_2}), \ldots, (A_e, a_{n,i_n}),$
$(A_e, a_{n+1,0}), \ldots, (A_e, a_{N,0})\}$.

Both programs LP_{A_u} and LP_{A_e} contain the clauses :

(*i*) $a_{k,j} \leftarrow \neg msg(A', A, h_k), msg(A', A, a_{k,j})$
for all $1 \leq k \leq N$ and $0 \leq j \leq r$ ($A' = A_e$, if $A = A_u$ and $A' = A_u$ otherwise).

Furthermore, for all $1 \leq k \leq N$, if there are L instructions in $\mathcal{M}$ with left hand side $q_j a_i$, then for each $1 \leq l \leq L$ the clause:

(ii_u) $v_{i,j,k,l} \leftarrow msg(A_e, A_u, h_k), msg(A_e, A_u a_{k,i}), msg(A_e, A_u, q_j)$

belongs to LP_{A_u} if $q_j \in Q_u$, and the clause

(ii_e) $v_{i,j,k,l} \leftarrow msg(A_u, A_e, h_k), msg(A_u, A_e, a_{k,i}), msg(A_u, A_e, q_j)$

belongs to LP_{A_e} otherwise.

Besides this, the program LP_{A_u} has the clause

(iii) $start \leftarrow \neg started.$

The agents A_u, A_e have the same nondeterministic obligation operator Sel, which keeps all permitted actions of the form $a_{k,j}$ and guesses a unique action $v_{i,j,k,l}$.

We choose the formula $\Phi = \mathbf{AX}\ \mu$ Z.(success $\vee$ **EX AX** Z) as the property to check. For a tree T, this formula states that it contains a finite $\forall\exists$-subtree ST [14] whose leaves are labeled by the fact *"success"*.

We choose the empty starting state S^0 of $\mathcal{A}$: $I_{A_u} = I_{A_e} = \emptyset$ and consider the trajectory tree $\mathcal{T} = \mathcal{T}_{\mathcal{A}}(S^0)$. It is easy to see that the following equivalence holds:

x is accepted by $\mathcal{M} \iff \mathcal{T}$ has a finite $\forall\exists$-subtree, whose leaves are odd-level nodes and contain the fact "success".

This equivalence is proven by a straightforward induction on computation steps (i.e. level numbers). It directly implies the property (*) for the constructed MAS $\mathcal{A}$, its starting state S^0 and formula Φ. $\mathcal{M}$ being fixed for all x, it is easy to see that $\mathcal{A}, S^0$ and Φ can be constructed from x in time polynomial in $|x|$. Therefore, MA-BEHAVIOR problem for nondeterministic ground, expanding and 2-agent systems is APSPACE-hard and consequently, EXPTIME-hard [6].

(2)*Lower bound.* The proof is by a modification of the proof of theorem 2 (2) similar to that we have used immediately above.

(3) The proof follows an argument in [16]. It is easily shown that using the usual binary coding of numbers, the size of descriptions of MAS $\mathcal{A}(M, x)$ in

[14] I.e., a subtree ST which 1) contains the root of T, 2) for all its even level nodes, contains all their daughters in T, and 3) for all its non-leaf odd level nodes,contains at least one of their daughters in T.

(1) and (2) can be bounded by $\mathbf{O}(n * \log n)$. Then the required assertion is obtained by using an ATM M recognizing in space $\mathbf{O}(n)$ a set recognizable by a DTM in time 3^n but not in time 2^n.

(4) *Upper bound.* It is well known (e.g., see [13,5]) that propositional μ-calculus model checking has time complexity upper bound $O((|TS| * |\phi|))^{ad(\phi)})$ for formulas ϕ with alternation depth $ad(\phi)$ on transition systems TS. On the other hand, this problem belongs to $NP \cap$ *co-NP*. These results are easily extended to first-order μ-calculus and finite transition systems with ground atom labels. A ground MAS A having no more than $2^{|A|}$ states, the upper bound follows from proposition 3. □

5.2 Nonground nondeterministic MAS

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

Theorem 8 *(1) The MA-BEHAVIOR problem is NEXPTIME-complete (coNEXPTIME-complete) in the class of expanding r-signal MAS (for any $r > 0$) and the behavior properties Φ expressed in $\exists LTL$ (respectively, in $\forall LTL$).*
(2) The MA-BEHAVIOR problem is APSPACE (EXPTIME)-complete for k-dimensional MAS (for any fixed $k > 0$) and the behavior properties Φ in μ^{FO}.

Proof. (1) *Upper bound.* The existence of a nondeterministic exponential time algorithm resolving the MA-BEHAVIOR problem for ground, expanding and r-signal MAS is based on the following upper bound on the length of trajectories in $\mathcal{T}_{\mathcal{A}}(S^0)$, similar to the one established in lemma 10.

Lemma 11 *There is a polynomial $p(n)$ such that for any ground, expanding and r-signal MAS $\mathcal{A}$, a state S^0 and a formula $\Psi \in PLTL$, $\mathcal{T}_{\mathcal{A}}(S^0) \models \mathbf{E}\ \Psi$ iff there is a trajectory $\rho = S^0, \ldots, S^t, \ldots \in \mathcal{T}_{\mathcal{A}}(S^0)$ and a step $T \leq 2^{p(|\mathcal{A}|+|\Psi|+|S^0|)}$ such that $\rho, S^0 \models \Psi$ and $I_a^t = I_a^T$ for all $t > T$ and for each agent $a \in \mathcal{A}$.*

Sketch of the proof. If not to say about the presence of variables in the clauses of agents' programs and of the condition $m^2 * r = \mathbf{O}(\log |\mathcal{A}|)$ (m being the number of agents in $\mathcal{A}$), we are in the conditions of lemma 10. Due to the use of variables, the size and the number of states grow: the number of atoms in DB-states of agents of $\mathcal{A}$ is now bounded by $2^{pol(|\mathcal{A}|+|S^0|)}$ for some polynomial *pol*. In particular, it means that the condition $m^2 * r =$

$\mathbf{O}(\log|\mathcal{A}|)$ is of no importance in this context. Since $\mathcal{A}$ is expanding, in any trajectory of $\mathcal{T}_{\mathcal{A}}(S^0)$, the number of steps where at least some agent's DB-state changes (i.e. increases) is bounded by $2^{pol(|\mathcal{A}|+|S^0|)}$. The number of different message box states of an agent is bounded by 2^{mr}. So the total number of global states of message boxes does not exceed $2^{m^2r} \leq 2^{r|\mathcal{A}|^2}$. Hence, for any trajectory $\rho \in \mathcal{T}_{\mathcal{A}}(S^0) = S^0, \ldots$ and any step i, if all agents' DB-states in $S^i, S^{i+1}, \ldots, S^{i+M}$ are the same for $M \geq 2^{r|\mathcal{A}|^2}$, then there exist two steps l and r, $i \leq l < r \leq i + M$ such that $S^l = S^r$. Applying assertion 5 in the proof of theorem 6, we bound the length M of such stable state subsequences by $2^{r|\mathcal{A}|^2+|\Psi|}$. Therefore, the stabilization step T in ρ can be bound by $2^{p(|\mathcal{A}|+|\Psi|+|S^0|)}$ for some polynomial p. □

Now, the only difference between the nondeterministic algorithm NdetCh1 checking that $\mathcal{T}_{\mathcal{A}}(S^0) \models \mathbf{E}\,\Psi$ and the algorithm NdetCh above is that NdetCh1 guesses in $\mathcal{T}_{\mathcal{A}}(S^0)$ a finite trajectory ρ of exponential length and then checks whether $\rho, S^0 \models \Psi$ on this trajectory. So it is a nondeterministic exponential time algorithm.

Lower bound is established using the construction in the lower bound proof of theorem 4. The difference is that in the place of a DTM one should use a NTM running in exponential time and choose the nondeterministic unit-choice one-step semantics guessing at each step a single instruction to execute.

(2) *Upper bound.* Due to the condition of k-dimensionality, the size of the set of global states of $\mathcal{A}$ is bounded by $2^{pol(|S^0|+|\mathcal{A}|)}$ for some polynomial *pol*. Then, using proposition 3, we can follow the corresponding reasoning in point (4) in the proof of theorem 7 (4).
Lower bound trivially follows from theorem 7(1), since ground MAS are in fact, 0-dimensional MAS. □

It seems that if to delete the condition "expanding" in the assertion (1) of Theorem 8 then the complexity of the problem will essentially increase, but this is open now.

If in the proof of the lower bound in theorem 4 we simulate an ATM in place of a DTM, we obtain the following result.

Theorem 9 *The MA-BEHAVIOR problem is AEXPTIME-hard for nondeterministic expanding and positive MAS and the behavior properties 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 MAS, the problem is much more hard.

Theorem 10 *The MA-BEHAVIOR problem in the class of nondeterministic MAS*
1) is AEXPSPACE (EXPEXPTIME)-complete for the behavior properties expressed in μ_r^{FO} (for any fixed r);
2) is in NEXPEXPTIME $\cap$ co-NEXPEXPTIME for the behavior properties expressed in μ^{FO}.

Proof. *Upper bounds* follow by proposition 3 from the above mentioned results on model checking complexity for μ-calculus, because the size of a global state of $\mathcal{A}$ is estimated in the general case by the size of its groundization, i.e. $2^{pol(|\mathcal{A}|)}$ for some polynomial *pol*.

Lower bound. It suffices to prove that nondeterministic MAS can simulate an ATM running in exponential space. Let us fix an ATM $\mathcal{M}$ running in space bounded by $2^{p(n)}$ for some polynomial $p(n)$, n being the length of an input word $x = a_{i_1}a_{i_2}...a_{i_n}$. We set $N = p(n)$. We assume that M satisfies the restrictions used in the proof of theorem 7: no repetitions of I-descriptions and the universal states occur at the even steps (respectively, the existential states occur at the odd steps).

We construct a 2-agent system $\mathcal{A} = \{A_1, A_2\}$ simulating the machine M in a way very similar to that used in theorem 5. The only new thing to do is to define a nondeterministic one-step semantics of the agents. In order to simplify this semantics, we associate with each set $I_{i,j}$ of instructions of M having the same left hand side q_ia_j, the set of all actions of the form $\{v(i, j, \overline{s_1}, \overline{s_2}, \overline{s_3}, \overline{s_4}, l)|1 \leq l \leq L\}$, where L is the cardinality of $I_{i,j}$ and $\overline{s_i}$ are $0-1$-tuples of length N. For the instruction number l in $I_{i,j}$ of the form $q_ia_j \rightarrow q_ma_nC$, the ground action $v(i, j, \overline{s_1}, \overline{s_2}, \overline{s_3}, \overline{s_4}, l)$ sends to the partner agent the messages: $q_m, h(\overline{s_1}), a_n(\overline{s_2}), right(\overline{s_3}), left(\overline{s_4})$.

The instructions in $I_{i,j}$ are simulated in the programs of both agents by the rules:

$v(i, j, \overline{S_1}, \overline{S}, \overline{S_1}, \overline{S_2}, l) \leftarrow msg(A', A, h(\overline{S})), msg(A', A, a_k(\overline{S})),$
$msg(A', A, q_i), shift(C, \overline{S}, \overline{S_1}), next(\overline{S}, \overline{S_1}), next(\overline{S_2}, \overline{S}),$
for all $l, 1 \leq l \leq L$.

The agents have the same nondeterministic obligation operator *Sel* which guesses a unique possible ground action of the form $v(i, j, \overline{s_1}, \overline{s_2}, \overline{s_3}, \overline{s_4}, l)$ and keeps all the other actions at each step.

We choose the formula μZ.(*yes* $\vee$ **AX** (*yes* $\vee$ **EX** Z)) as the property to check. It states that the tree of trajectories of $\mathcal{A}$ has a finite $\forall\exists$-subtree whose leaf-states contain the fact *yes*.

Clearly, $\mathcal{M}$ accepts x iff $\mathcal{T}_{\mathcal{A}}(S^0), S^0 \models \Phi$.

It is easy to see that the size of $\mathcal{A}$ is bounded by a polynomial in $|x|$ and that the MAS $\mathcal{A}$, the state S^0, and the formula Φ can be constructed in time polynomial in $|x|$. So the MA-BEHAVIOR problem for the class of deterministic MAS is AEXPSPACE-hard. □

Remark. (1) Theorems 8, 9 and 10 can also be complemented by an assertion similar to that of theorem 7 (3), which gives absolute lower bounds of time complexity of the corresponding MA-BEHAVIOR problems.

(2) The upper complexity bounds established for the properties expressed in μ hold for CTL too, since it is polynomially translatable into μ-calculus.

6 Behavior of asynchronous MAS

In this section we consider a version of asynchronous MAS. An *asynchronous MAS* $\mathcal{A} = \{a_1, ..., a_n; PA\}$ consists of a finite set $\{a_1, ..., a_n\}$ of *intelligent agents* and a special post agent PA. The agent PA is used to simulate a communication network of the system which can deliver messages to their receivers asynchronously.

The agent a_i sends messages to the other agents in the system through the agent PA and receives messages from PA into its message box $MsgBox_{a_i}$. An internal state I_{PA} of PA includes all the messages which were received by PA and not yet sent in this time.

For the asynchronous MAS $\mathcal{A}$ its *one-step semantics* is a one step transition relation $\Rightarrow_{\mathcal{A}}$ on the set $\mathcal{S}_{\mathcal{A}}$ of global states of the form $S =< (I_{a_1}, MsgBox_{a_1}), \ldots, (I_{a_n}, MsgBox_{a_n}), I_{PA} >$ induced by one-step semantics of individual agents $a_i, i = 1, \ldots, n$, of $\mathcal{A}$ and the behavior of PA.

The transition $S^t \Rightarrow_{\mathcal{A}} S^{t+1}$ starts by calculating the sets $Perm_a^t$ of actions permitted for execution for all agents $a \in \mathcal{A}$. Next, each agent's selection operator Sel_a creates action set $Obl_a^t = Sel_a(Perm_a^t)$. The message boxes of all agents in $\mathcal{A}$ are emptied thereafter. Then each agent's internal DB state I_a^t is replaced by I_a^{t+1} by deleting of $DelObl_a^t$ and adding $AddObl_a^t$. and for each agent a all messages $SendObl_a^t$ sent by a are placed into I_{PA}^t. Then PA sends to every agent a *some* messages contained in I_{PA}^t which were sent to a by other agents (puts them into message box $MsgBox_a^{t+1}$) and deletes these messages from I_{PA}^t forming I_{PA}^{t+1}.

It follows from the definition that the transfer of a message from one agent to another can take an indeterminate amount of time (in particular, it can be lost). This reflects the asynchronous mode of agents' interaction.

For verifying asynchronous MAS it is important to take into account the internal state of PA which consists of message atoms $msg(a_i, a_j, p)$. But such a message can also occur in the message box of a_j. So, in order to correctly refer to truth values of such messages we should distinguish in formulas occurrences of these message atoms which have to be evaluated in PA and in message boxes of a_i. For this we take the following notation: $msg^j(a_i, a_j, p)$ will denote the atom to be evaluated in the message box of a_j, and $msg(a_i, a_j, p)$ denotes the atom evaluated in PA.

The most of the complexity results obtained in previous sections for MA-BEHAVIOR problem for synchronous non-deterministic systems can be transferred to asynchronous systems using similar arguments. But this is not very interesting, and we give here only two general theorems on mutual reducibility of MA-BEHAVIOR problem for synchronous and asynchronous MAS. Some of the above mentioned complexity results follow for asynchronous MAS from these theorems, although not for all classes of MAS considered for synchronous systems (it is caused by the generality of constructions used in the proof of the theorems).

The following theorem is proved by some simulation of nondeterministic MAS by asynchronous MAS.

Theorem 11 *Let dynamic properties to verify be formulated in the language μ_r^{FO}. Then MA-BEHAVIOR problem for nondeterministic MAS with obligation operator Sel^{un} is polynomial-time reducible to the MA-BEHAVIOR problem for asynchronous MAS with deterministic agents.*

Proof. For simplicity we give the proof for ground case only. The nonground case is somewhat more complicate, but similar. Let $\mathcal{A} = \{a_1, a_2, \ldots, a_n\}$ be a nondeterministic MAS such that each agent $a \in \mathcal{A}$ uses the obligation operator Sel^{un} to choose a set of executable actions. Let S^0 be an initial global state of $\mathcal{A}$. We construct from $\mathcal{A}$, S^0 and a μ-formula Φ an asynchronous MAS $\mathcal{B}$, its initial state R^0 and a μ-formula Ψ such that $\mathcal{A}, S^0 \models \Phi \Leftrightarrow \mathcal{B}, R^0 \models \Psi$.

We suppose that the agent a_i has a set $Act_i = \{\alpha_{i1}, \ldots, \alpha_{im_i}\}$ of ground action atoms.

The asynchronous MAS $\mathcal{B}$ consists of agents $a'_1, \ldots, a'_n$, two additional agents b and c and a postage agent PA. At each step the agent c sends to every a'_i three messages "1", "2" and "3" (we will be interested in trajectories along which a'_i will receive these messages in cyclic order 1-2-3).

For any action $\alpha \in Act_i$ the agent a'_i contains a new message atom α and a new duplicate action α' with the lists $ADD_{\alpha'} = DEL_{\alpha'} = \emptyset$ and $SEND_{\alpha'} = \{(a'_i, b, \alpha)\}$. Program $P_{a'_i}$ includes

(i) all clauses of P_{a_i} in which every occurence of each action atom α is changed to α', and to each body the atom $msg(c, a'_i, 1)$ is added;

(ii) a clause
$\alpha \leftarrow msg(b, a'_i, \alpha'), msg(c, a'_i, 3)$
for each $\alpha \in Act_i$.

Agent b for all a'_i and $\alpha \in Act_i$ includes action α^{a_i} with the lists $ADD_{\alpha^{a_i}} = DEL_{\alpha^{a_i}} = \emptyset$ and $SEND_{\alpha^{a_i}} = \{(b, a_i, \alpha)\}$. Program P_b for all a'_i and $\alpha \in Act_i$ includes a clause
$\alpha^{a_i} \leftarrow msg(a_i, b, \alpha)$.

Some trajectories of $\mathcal{B}$ simulate trajectories of $\mathcal{A}$, one step $S \Rightarrow_{\mathcal{A}} S'$ of $\mathcal{A}$ in three steps.

At the first of these steps each a'_i defines the set $Perm_{a_i} = Perm_{a_i}(S)$ of actions permitted in the state S and transfers the set of messages $\{msg(a_i, b, \alpha) | \alpha \in Perm_{a_i}\}$ to the agent b. At the second step b sends $\{msg(b, a_i, \alpha) | \alpha \in Perm_{a_i}\}$ to a'_i, and one of these messages is transfered to a'_i.

At the third step a'_i executes the action received from b (i.e. changes its internal database and transfers to all agents a'_j all the messages which should be sent to them by it), and PA resends to a'_i the rest of $Perm_{a'_i}$ and these messages will be "lost" at the next step.

The initial state $R^{(0)}$ of $\mathcal{B}$ is defined as follows: $I^{(0)}_{a'_i} = I^{(0)}_{a_i}$, $Msgbox^{(0)}_{a'_i} = Msgbox^{(0)}_{a_i} \cup \{msg(c, a_i, 1)\}$, $I^{(0)}_c = I^{(0)}_b = I^{(0)}_{PA} = Msgbox^{(0)}_b = Msgbox^{(0)}_c = \emptyset$.

The formula $\Psi = \Psi(\Phi, \mathcal{A})$ is obtained from Φ by inductively replacing all the subformulas of the form $\exists X \Theta$ by the formula $\exists X(f_1 \wedge \exists X(f_2 \wedge \exists X(f_3 \wedge \Theta)))$, where f_1, f_2, f_3 have the forms

$\bigwedge^n_{i=1}(\bigwedge^{m_i}_{j=1} \neg msg(a'_i, b, \alpha_{ij}) \wedge msg^i(c, a'_i, 2) \wedge \neg msg^i(c, a'_i, 1) \wedge \neg msg^i(c, a'_i, 3))$,

$\bigwedge^n_{i=1}(\bigvee^{m_i}_{j=1} msg^i(b, a'_i, \alpha_{ij}) \wedge \bigwedge^{m_i}_{k,l=1;k\neq l}(\neg msg^i(b, a'_i, \alpha_{ik}) \vee \neg msg^i(b, a'_i, \alpha_{il})) \wedge$
$msg^i(c, a'_i, 3) \wedge \neg msg(c, a'_i, 1) \wedge \neg msg^i(c, a'_i, 2))$,

$\bigwedge^n_{i=1}(\bigwedge^n_{j=1}(\bigwedge_{q^{(r)} \in \mathbf{P}^m_{a_i}} \forall Z_1, ..., Z_r(\neg msg(a'_i, a'_j, q^{(r)}(Z_1, ..., Z_r)) \wedge$
$msg^i(c, a'_i, 1) \wedge \neg msg^i(c, a'_i, 2) \wedge) \neg msg^i(c, a'_i, 3))$,

respectively.

The formula f_1 has the meaning "any agent a'_i has received the message 2 from c, and PA does not contain any action messages sent to the agent b" (i.e. all the action messages sent by agents a_i are transferred to b immediately: all these messages are from $Perm_{a_i}$).

The formula f_2 has the meaning "any agent a'_i has received the message 3 from c and exactly one action message α_{ij}" from b (in fact, the message α_{ij} belongs to the set $Perm_{a_i}$ which was sent to b by a_i in the previous step).

The formula f_3 has the meaning "any agent a'_i has received the message 1 from c, and for all i, j PA does not contain any information messages sent by a_i to a_j" (i.e. all the information messages sent by agents a_i are transferred to their receivers immediately).

It is clear that the system $\mathcal{B}$, the state $R^{(0)}$ and the formula Ψ are constructed in polynomial time with respect to sizes of $\mathcal{A}, S^{(0)}$ and Φ.

Let us define a similarity relation between global states of $\mathcal{A}$ and $\mathcal{B}$. Namely, a global state $R =< (I_{a'_1}, MsgBox_{a'_1}), \ldots, (I_{a'_n}, MsgBox_{a'_n}), (I_b, MsgBox_b), (I_c, MsgBox_c), I_{PA} >$ of $\mathcal{B}$ is similar to a global state $S =< (I_{a_1}, MsgBox_{a_1}), \ldots, (I_{a_n}, MsgBox_{a_n}) >$ of $\mathcal{A}$ iff $I_{a'_i} = I_{a_i}$, and $MsgBox_{a_i}$ is obtained from $MsgBox_{a'_i}$ by deleting all the messages received from the agents b anf c, for all i.

It is obvious that $R^{(0)}$ is similar to $S^{(0)}$.

Let us define a mapping G of the set of nodes of $\mathcal{T}_{\mathcal{A}}(S^{(0)})$ to the set of nodes of $\mathcal{T}_{\mathcal{B}}(R^{(0)})$ as follows. $G(S^{(0)}) = R^{(0)}$. Let $G(S) = R$ be defined for a node S from $\mathcal{T}_{\mathcal{A}}(S^{(0)})$ such that $msg(c, a'_i, 1)$ is in $MsgBox_{a'_i}$. Let S' be a successor of S. Then $G(S') = R'$, where R' is obtained by the three steps of $\mathcal{B}$ simulating the step $S \Rightarrow_{\mathcal{A}} S'$ as was described above.

It is clear that $G(S)$ is similar to S for each S from $\mathcal{T}_{\mathcal{A}}(S^{(0)})$.

Further we will use the notation $G(\mathcal{A})$ for the set $\{G(S)|S \in \mathcal{T}_{\mathcal{A}}(S^{(0)})\}$. Let $set_{\mathcal{A}}(\Phi) = \{S|\mathcal{T}_{\mathcal{A}}(S^{(0)}), S \models \Phi\}$, and $set_{\mathcal{B}}(\Psi) = \{R|\mathcal{T}_{\mathcal{B}}(R^{(0)}), R \models \Psi\}$.

The proof of the theorem is completed now by proving the following

Lemma 12 *For any formula Φ of μ^{FO} the equality $set_{\mathcal{B}}(\Psi(\Phi, \mathcal{A})) \cap G(\mathcal{A}) = \{G(S)|S \in set_{\mathcal{A}}(\Phi)\}$ holds.*

The lemma is proved by induction on the structure of Φ.

First, we prove the following proposition.

(#) If the assertion of the lemma holds for formulas Θ and Θ' then it holds for formulas $\neg\Theta, \Theta \wedge \Theta', \Theta \vee \Theta', \exists X\Theta, \forall X\Theta$.

For boolean connectives it is obvious.

For formulas of the form $\exists X\Theta$ the proposition follows from the assertion

(*) For any node S of $\mathcal{T}_{\mathcal{A}}(S^{(0)})$ the following is true:

$\mathcal{T}_{\mathcal{A}}(S^{(0)}), S \models \exists X\Theta$ iff $\mathcal{T}_{\mathcal{B}}(R^{(0)}), G(S) \models \Psi(\exists X\Theta, \mathcal{A})$

Suppose Φ is $\exists X\Theta$, and $\mathcal{T}_{\mathcal{A}}(S^{(0)}), S \models \Phi$. Then, by definition of $\exists X$, $\mathcal{T}_{\mathcal{A}}(S^{(0)}), S' \models \Theta$, for some S' such that $S \Rightarrow_{\mathcal{A}} S'$. Then, by the induction hypothesis, $\mathcal{T}_{\mathcal{B}}(R^{(0)}), G(S') \models \Psi(\Theta, \mathcal{A})$. By the definition above, $\Psi(\Phi, \mathcal{A})$ $=\exists X(f_1 \wedge \exists X(f_2 \wedge \exists X(f_3 \wedge \Psi(\Theta, \mathcal{A}))))$. We note that, by the definition of G, there exists a path $G(S) = R \Rightarrow_{\mathcal{B}} R_1 \Rightarrow_{\mathcal{B}} R_2 \Rightarrow_{\mathcal{B}} R_3 = G(S')$ in $\mathcal{T}_{\mathcal{B}}(R^{(0)})$. It is clear that $\mathcal{T}_{\mathcal{B}}(R^{(0)}), R_i \models f_i, i = 1, 2, 3$. It follows that $\mathcal{T}_{\mathcal{B}}(R^{(0)}), G(S) \models \Psi(\Phi, \mathcal{A})$.

Conversely, suppose that $\mathcal{T}_{\mathcal{B}}(R^{(0)}), G(S) \models \Psi(\Phi, \mathcal{A})$. Then there exists a path $G(S) = R \Rightarrow_{\mathcal{B}} R_1 \Rightarrow_{\mathcal{B}} R_2 \Rightarrow_{\mathcal{B}} R_3$ such that 1) $\mathcal{T}_{\mathcal{B}}(R^{(0)}), R_i \models f_i, i = 1, 2, 3$, and 2) $\mathcal{T}_{\mathcal{B}}(R^{(0)}), R_3 \models \Psi(\Theta, \mathcal{A})$. It follows from 1) that each agent a_i' in R_2 executes some action $\alpha_{ij} \in Perm_{a_i}(S)$. Then each $a_i \in \mathcal{A}$ can execute the same action in state S, because its obligation operator $Sel^{un}_{a_i}$ can choose any action from $Perm_{a_i}(S)$. After this the system $\mathcal{A}$ goes to state S' such that $G(S') = R_3$. Then by the induction hypothesis $\mathcal{T}_{\mathcal{A}}(S^{(0)}), S' \models \Theta$ and therefore $\mathcal{T}_{\mathcal{A}}(S^{(0)}), S \models \exists X\Theta$.

The assertion (*) is proved.

If Φ is $\forall X\Theta$ the proof follows from the equivalence $\forall X\Theta$ with $\neg\exists X\neg\Theta$.

The proposition (#) is proved.

If Φ is basic then the assertion of the lemma follows from the similarity of S and $G(S)$. Hence, from the proposition (#) we deduce that the lemma holds for any formula Φ in μ_0^{FO} (i.e. not containing the operators μ and ν).

If Φ has the form $\mu Z.\Theta(Z)$ or $\nu Z.\Theta(Z)$ then the lemma is proved by a straighforward but cumbersome induction on the computation of fixpoints for these formulas. As an example we consider here only the following simple case.

Let Φ have the form $\mu Z.\Theta(Z)$ where $\Theta(Z)$ does not contain μ and ν.

Then $\Psi(\Phi, \mathcal{A}) = \mu Z.\Psi(\Theta(Z), \mathcal{A})$. By definition of μ we have $set_{\mathcal{A}}(\Phi) = \bigcup_{i=0}^{\infty} set_{\mathcal{A}}(\Theta^i(false))$, and $set_{\mathcal{B}}(\Psi(\Phi, \mathcal{A})) = \bigcup_{i=0}^{\infty} set_{\mathcal{B}}((\Psi(\Theta, \mathcal{A}))^i(false))$, Then we have $\Psi(\Theta^i(false), \mathcal{A}) = (\Psi(\Theta, \mathcal{A}))^i(false)$, and $set_{\mathcal{B}}((\Psi(\Theta, \mathcal{A}))^i(false)) \cap$

$G(\mathcal{A}) = \{G(S)|S \in set_{\mathcal{A}}(\Theta^i(false))\}$. It follows that $set_{\mathcal{B}}(\Psi(\Phi, \mathcal{A})) \cap G(\mathcal{A}) = \{G(S)|S \in set_{\mathcal{A}}(\Phi)\}$.

If Φ has the form $\nu Z.\Theta(Z)$ where $\Theta(Z)$ does not contain μ and ν then the proof is similar.

In the following theorem asynchronous MAS are simulated by nondeterministic MAS.

Theorem 12 *Let dynamic properties to verify be formulated in the language μ_r^{FO}. Then MA-BEHAVIOR problem for asynchronous nondeterministic MAS is polynomial-time reducible to the MA-BEHAVIOR problem for (synchronous) nondeterministic MAS.*

Proof.

Let $\mathcal{A} = \{a_1, ..., a_n; PA\}$ be a nondeterministic asynchronous MAS, and Φ be a formula to verify. We construct from $\mathcal{A}$ a nondeterministic MAS $\mathcal{B} = \{a'_1, \ldots, a'_n, pa\}$ which simulates $\mathcal{A}$. Nondeterministic agent pa will simulate the work of PA by saving some part of messages in its data base and then sending some of them to receivers.

For each message predicate $q \in \mathbf{P}^m_{a_i}$ we put into $\mathbf{P}^m_{a'_i}$ a predicate q_{ij} for all $j \neq i$. The heads α of clauses of the logical components of the agents a'_i are the same as of the agents a_i, but each message of the form $msg(a_i, a_j, q)$ in the list $SEND_\alpha$ of action α is changed to $msg(a'_i, pa, q_{ij})$. In the bodies of all clauses of P_{a_i} each atom of the form $msg(a_j, a_i, q)$ is changed to $msg(pa, a'_i, q_{ji})$, and a new atom $msg(pa, a'_i, 1)$ is added to the body of each clause.

The action base AB_{pa} of the nondeterministic post agent pa includes for each $q^{(k)}_{ij}$ three actions $save(q_{ij}), resend(q_{ij})$ and $send(q_{ij})$ with the following lists:
$ADD_{save(q_{ij}} = \{q_{ij}(X_1, \ldots, X_k)\}, DEL_{save(q_{ij}} = SEND_{save(q_{ij}} = \emptyset$,
$ADD_{resend(q_{ij}} = DEL_{resend(q_{ij}} = \emptyset, SEND_{resend(q_{ij}} =$
$\{msg(pa, a_j, q_{ij}(X_1, \ldots, X_k)\}$, $ADD_{send(q_{ij}} = \emptyset$,
$DEL(send(q_{ij}) = \{q_{ij}(X_1, \ldots, X_k)\}, SEND_{send(q_{ij}} =$
$\{msg(pa, a_j, q_{ij}(X_1, \ldots, X_k)\}$.

AB_{pa} also includes two actions add_{one} and del_{one} to count odd and even steps:
$ADD_{add\_one} = \{1\}, DEL_{add\_one} = \emptyset, SEND_{add\_one} =$
$\{msg(pa, a'_i, 1)|i = 1, \ldots, n\}$. $ADD_{del\_one} = \emptyset, DEL_{del\_one} = \{1\}$,
$SEND_{del\_one} = \emptyset$.

To fire these actions program P_{pa} of pa includes two clauses:
$add\_one \leftarrow \neg 1$.
$del\_one \leftarrow 1$.

For each $q_{ij}^{(k)}$ program P_{pa} includes three clauses:
$save(q_{ij})(X_1, \ldots, X_k) \leftarrow msg(a_i, pa, q_{ij}(X_1, \ldots, X_k)$.
$resend(q_{ij})(X_1, \ldots, X_k) \leftarrow msg(a_i, pa, q_{ij}(X_1, \ldots, X_k)$.
$send(q_{ij})(X_1, \ldots, X_k) \leftarrow q_{ij}(X_1, \ldots, X_k)$.

The obligation operator Sel_{pa} chooses from $Perm_{pa}$ any subset of actions of the form $send(q_{ij})(t_1, \ldots, t_k)$, and Sel_{pa} selects one and only one atom from each pair of action atoms of the form $\{save(q_{ij})(t_1, \ldots, t_k), resend(q_{ij})(t_1, \ldots, t_k)\} \subseteq Perm_{pa}$.

Satisfiability of Φ in $\tau(\mathcal{A}, S^0)$ can be reduced to satisfiability of a formula Ψ of μ_r^{FO} in $\tau(\mathcal{B}, S^0)$. Ψ is constructed as in the previous theorem, but in a simpler way.

It is clear that the reduction is polynomial-time computable.

In particular, these theorems have the following corollaries.

Corollary 1 . *MA-BEHAVIOR problem for ground asynchronous MAS with deterministic agents*
1) is $EXPTIME$-hard for verifying formulas from μ_r^{FO}, for any fixed $r \geq 1$, and
2) is in $EXPTIME$ for verifying formulas from all μ^{FO}.

Corollary 2 . *MA-BEHAVIOR problem for non-ground asynchronous MAS with deterministic agents*
1) is $EXPEXPTIME$-hard for verifying formulas from μ_r^{FO}, for any fixed $r \geq 1$, and
2) is in $EXPEXPTIME$ for verifying formulas from all μ^{FO}.

7 Conclusion

Multi-Agent Systems represent a class of general parallel and/or distributed software systems. Many well known techniques of behavior analysis and verification for concurrent and parallel programs apply to MAS as well. At the same time, specific architectural features of MAS require significant rework of these approaches.

For MAS, with their rich architecture, the adequacy and the results of the behavior analysis are closely related with the exact choice of the level of detail of important architecture features and parameters and with the adopted restrictions on them. In this paper we have defined a specific fragment of the

IMPACT architecture [29]. Within this architecture, MAS can be either deterministic or nondeterministic, depending on the one-step semantics of the agents. To account for this, we use two different classes of temporal logics to express the properties of the MAS behavior. For each class of MAS we have considered some natural structural constraints: on the number of agents, on the number of messages available, on the dimensionality (arity) of actions and messages. We have also considered some important semantic constraints limiting expressivity and the effect of actions: the use of variables and/ or negation in agent programs, the possibility/ impossibility of deleting facts from agent states. Our goal was to determine computational complexity of the corresponding MA-BEHAVIOR problem for every combination of these restrictions. In many but not in all cases, we have established tight complexity bounds. In particular, our study has shown that under some of these reasonable restrictions, it is possible to capture the complexity of behavior properties described by means of classical linear and branching time logics within relatively low complexity classes: even in deterministic or nondeterministic polynomial time under some natural restrictions. Despite the fact that our agent's architecture is substantially simpler than the original IMPACT architecture of [29], many of our results can be extended to the general case, as the main features of the original one-step semantics of agents are computable in polynomial space, as is shown in Chapter 11 of [29].

Intelligent Agents and Multi-Agent 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 just one such specific architecture. However, it illustrates the way in which penetrating deeply into a complex MAS architecture permits, in some cases, a deeper understanding of the behavior properties of agents. Considered in this light, this paper creates a framework for applying similar analysis to other MAS architectures in order to find interesting subclasses of MAS with efficiently verifiable behavior properties. We also note that we considered here only 'naive' variants of checking algorithms, leaving to further research the application of different optimization techniques such as symbolic model checking, abstraction, using symmetry properties of MAS, introduced in the model checking literature.

Acknowledgment. We would like to thank an anonymous referee for many helpful comments.

References

[1] Araragi, T., Attie, P., Keidar, I., Kogure, K., Luchangco, V., Lynch, N., and Mano, K., On Formal Modeling of Agent Computations. In: *NASA Workshop on Formal Approaches to Agent-Based Systems.* April, 2000.

[2] Apt, K. R., Logic Programming. In: J. van Leeuwen (Ed.) *Handbook of Theoretical Computer Science. Volume B. Formal Models and Semantics, Chapter 10*, Elsevier Science Publishers B.V. 1990, 493-574.

[3] Barringer, H., Fisher, M., Gabbay, D., Gough, G., and Owens R. METATEM: An Introduction. *Formal Aspects of Computing,* 1995, 7:533-549.

[4] Benerecetti, M., Guinchiglia, F., and Serafini, L. Model Checking Multiagent Systems. *Technical Report # 9708-07.* Instituto Trentino di Cultura, 1998.

[5] Bernholz, O., Vardi, M. Y., Wolper, P. An automata-theoretic approach to branching time model checking. *Proc. Int. Workshop "Computer aided verification",* Stanford, 1994 (LNCS).

[6] Chandra, A. K., Kozen, D. C., Stockmeyer, L. J., Alternation. *J. Ass. Comput. Mach.,* 1981, 28(1):114-133.

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

[8] Dekhtyar, M. I., Dikovsky, A. Ja., Dynamic Deductive Databases with Steady Behavior. In: *Proc. of the 12th International Conf. on Logic Programming,* (L. Sterling Ed.), The MIT Press, 1995, 183-197.

[9] Dekhtyar, M., Dikovsky, A., and Valiev, M., Complexity of Multi-Agent Systems Behavior. In: S. Flesca and G. Ianni (Eds.) *JELIA 2002, Lect. Notes in AI,* N. 2424, 2002, 125-136.

[10] Dekhtyar, M., Dikovsky, A., Valiev, M., On feasible cases of checking multi-agent systems behavior. *Theoretical Computer Science*, Elsevier Science, 2003, v.303, no.1, 63-81.

[11] Dekhtyar, M., Dikovsky, A., and Valiev, M., Complexity of Asynchronous Behavior of Multi-Agent Systems. Abstracts of "Second St.Petersburg Days of Logic and Computability", August 24-26, 2003, St. Petersburg, 26-27.

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

[13] 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.

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

[15] Fisher, M., Dixon, C., Peim, M., Clausal temporal resolution, *ACM Transactions on computational logic,* 2001, 2(1).

[16] Fischer, M. J., Ladner, R. J. Propositional dynamic logic of regular programs. *J. Comp. Sys. Sci.,* 8(1979), 194-211.

[17] Jennings, N., Sycara, K. and Wooldridge, M. A roadmap of agent research and development *Autonomous Agents and Multi-Agent Systems,* 1998, 1(1):7-38.

[18] Lichtenstein, O., Pnueli, A., Checking that finite state concurrent programs satisfy their linear specification. In: *Proc. 12th ACM Symposium on Principles of Programming Languages.* 1985, 97-107.

[19] Manna, Z., Pnueli, A. *The temporal logic of reactive and concurrent systems: Specification.* Springer Verlag, 1991.

[20] Müller, J. P., Architectures and applications of intelligent agents: A survey. *The Knowledge Engineering Review,* 1998, 13(4):353-380.

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

[22] Luck, M., d'Inverno, M. A Conceptual Framework for Agent Definition and Development *The Computer Journal,* 2001, 44(1),1-20.

[23] Papadimitriou, C. H., *Computational complexity.* Addison Wesley, 1994.

[24] Queille, J. P., Sifakis, J., Specification and verification of concurrent programs in CESAR. In: *Proc. of the 5th International Symposium on Programming, Lecture Notes in Computer Science,* N. 137, 1982, 195-220.

[25] Rao, A. S. and Georgeff, M. P., A model-theoretic approach to the verification of situated reasoning systems. In: *Proceedings of the Thirteen's International Joint Conference on Artificial Intelligence (IJCAI-93).* 1993, 318-324.

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

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

[28] Stirling, C., Walker, D., Local model checking in the Mu-calculus. *Lecture Notes in Computer Science,* N. 351, 1989.

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

[30] 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.

[31] Wooldridge, M., Dunne, P.E., The Computational complexity of Agent Verification. In: *J.-J. Meyer and M. Tambe (eds.), Intelligent Agents VIII. Springer-Verlag Lecture Notes in AI Volume,* March 2002.

[32] M. Wooldridge, M. Fisher, M.-P. Huget, and S. Parsons. Model Checking Multiagent systems with MABLE. In: *Proc. of the First Intern. Conf. on Autonomous Agents and Multiagent Systems (AAMAS-02)*, Bologna, Italy, July 2002.

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