ISSN 0361-7688, Programming and Computer Software, 2009, Vol. 35, No. 5, pp. 266–281. © Pleiades Publishing, Ltd., 2009.
Original Russian Text © M.K. Valiev, M.I. Dekhtyar, A.Ya. Dikovsky, 2009, published in Programmirovanie, 2009, Vol. 35, No. 5.

Systems of Agents Controlled by Logical Programs: Complexity of Verification

M. K. Valiev*, M. I. Dekhtyar, and A. Ya. Dikovsky*****

**Keldysh Institute of Applied Mathematics, Russian Academy of Sciences, Miusskaya pl. 1, Moscow, 125047 Russia*
e-mail: valiev@spp.keldysh.ru
***Tver State University, ul. Zhelyabova 3, Tver, 170013 Russia*
e-mail: Michael.Dekhtyar@tversu.ru
****University of Nantes, 2 rue de la Houssiniere, BP 92208, 44322 Nantes Cedex 3, France*
e-mail: Alexandre.Dikovsky@irin.univ-nantes.fr

Received October 17, 2008

Abstract—The complexity of the verification problem for the behavior (dynamical properties) of systems of interacting intelligent agents is considered. This paper is a continuation of our publications [1–3], in which this problem was mainly considered as applied to deterministic and nondeterministic systems, and largely focuses on asynchronous systems.

DOI: 10.1134/S036176880905003X

1. INTRODUCTION

The concepts of an intelligent agent and of a system of interacting agents (in short, a multiagent system, MA system, or MAS) were introduced more than twenty years ago. The study and application of these systems represent one of the most intensively developing areas in artificial intelligence and application programming (see, for example, [4–10]). One of important factors that have stimulated interest in agents is the development of network technologies (in particular, technologies related to the Internet).

The concept of an agent substantially generalizes the concept of an object in object-oriented programming due to the introduction of intelligent components that determine the operation policy of an agent depending on its state and messages received from other agents of the system and from the environment. The state of an agent may have a rather complicated structure; for example, it may include a database. Therefore, the behavior of these systems is rather complicated and requires the verification of whether their behavior satisfies certain conditions (verification of the dynamical properties of these systems). Naturally, this problem is unsolvable in the general case; therefore, we consider this problem (we call it the MA-BEHAVIOR problem) under various constraints imposed on the type of agents and on the class of properties to be verified. Note that, although the concepts of agents and MASs have been intensively studied over many years, the analysis of the verification of their behavior has started relatively recently [1, 11–15].

Multiagent systems have a wide scope of application, including diverse fields such as management of business processes, electronic trade, Internet navigation, social and military applications, etc. Such a variety of applications have given rise to a variety of approaches to the definition of the concepts of an agent and of an MAS (with various levels of abstraction from specific systems). In these cases, the architectures of agents mainly differ by the structure of their intelligent components, which may use solution trees, logical programs, inferences in various kinds of modal logic (epistemic, deontic, etc.), etc. Surveys of many of these approaches are given in the references cited above.

The definition of agent used in the present paper is largely based on the IMPACT architecture, which was introduced and described in detail in [9].[1] The full definition of this architecture is rather cumbersome. It includes different rather complicated mechanisms and means for specifying agents, which make them well adapted to practical applications. However, such an abundance of various means strongly complicates the formal analysis of the properties of these agents. In particular, as is shown in [9], the semantics of agents of the IMPACT architecture described in terms of transitions between states is unfeasible in the general case. To make the semantics of transitions computable in polynomial time, the authors of [17] (see also [9]) imposed some very complicated constraints on the type of agents. Unfortunately, the definition of such agents becomes hardly conceivable.

[1] Note that the agents considered here significantly differ from agents in the sense of [16].

Table 1. Complexity of verification for deterministic MA systems

Ground	Expand	Parameters	Logic	Complexity	N
Yes	Yes	positive	LTL	P	d1
		$m^2 * r = O(\log N)$	LTL	P	d2
		fixed $m \geq 2$	LTL	PSPACE	d3
		fixed $r \geq 1$	LTL	PSPACE	d4
	No		FO-LTL	PSPACE	d5
No	Yes	positive and fixed k	LTL	P	d6
		positive	LTL	EXPTIME	d7
	No	fixed k	FO-LTL	PSPACE	d8
			FO-LTL	EXPSPACE	d9

We propose some other rather easily formulable constraints on IMPACT agents, which also lead to the semantics of polynomial complexity. We focus on the means related to the definition of actions, decision-making policy, and communication of agents. Thus, we abstract from the details of the IMPACT architecture that are related to the agentization of an ordinary program code, security, meta-knowledge structures, and temporal and probabilistic reasoning. Moreover, in contrast to [9], where the authors admit more general structures of states, we consider sets of facts (an analog of relational databases) as the states of agents. Note that the architecture of an MAS remains rather representative even under these constraints, and a large part of examples in [9] fall under this kind of restricted architecture.

Agents can be either deterministic or nondeterministic, and the interaction between agents in an MAS can be either synchronous or asynchronous. For any initial state, synchronous systems of deterministic agents define a certain computational trajectory. In the other three cases, for a given initial state, a system defines a certain tree of possible computational trajectories.

In the first case, it is natural to apply linear time logic to describe the dynamical properties of the system; in the other cases, it is natural to apply some versions of branching time logic.

The MA-BEHAVIOR problem considered in the present paper consists in verifying if a certain formula of temporal logic is satisfied on the trajectories of a given MAS. Actually, this problem corresponds to the problem, known as model checking, which has been studied for many years as applied to abstract systems (diagrams) of transitions (see [18–24]). However, there is an essential difference between the classical statement of this problem and the version considered in the present study. Traditionally, complexity results are established for diagrams with unstructured states and explicitly defined transitions (or for other similar systems of finite-automata or OBDD types). In MASs, the states may have a complicated structure (sets of facts in our case), and transitions between the states are defined by some programs. This fact allows one to estimate the complexity of the verification problem on the basis of different structural and semantic properties of the systems.

MASs define a compact representation of a system of transitions. In particular, even in the case of a ground MAS (i.e., an MAS without variables), the system of transitions defined by it may have exponential size with respect to that of the initial MAS. Therefore, sometimes our lower bounds look more pessimistic than those in the classical case with the same logics. As regards the upper bounds, sometimes they are more informative and exact (when deterministic or nondeterministic polynomial upper bounds are obtained), or they are obtained by a simple translation of classical bounds with regard to the change in size when passing from an MAS to a system of transitions defined by this MAS.

The further part of the paper is organized as follows. Section 2 contains the basic definitions. In Section 3, we give an example that illustrates the possibilities of the MASs considered. Section 4 contains the description of a verification algorithm for deterministic MASs from [2]. In Section 6, this algorithm is used for constructing a nondeterministic algorithm that solves a verification problem for a class of asynchronous MASs in polynomial time. In Section 5, we give a survey of results from [2, 3] related to the complexity of verification of various classes of deterministic and nondeterministic MASs (in the form of Tables 1 and 2). Section 6 contains, in addition to the above-mentioned result, issues on the mutual reduction between MA-BEHAVIOR problems for nondeterministic and asynchronous MASs. The Conclusions contain a brief summary of the results and some discussion of the problem.

Table 2. Complexity of verification for nondeterministic MA systems

Ground	Expand	Parameters	Logic	Complexity	N
Yes	Yes	$m^2 * r = O(\log N)$	$\exists$ LTL	NP	n1
			$\forall$ LTL	co-NP	n2
		fixed $m \geq 2$	FO-Lμ_1	EXPTIME	n3
		fixed $r \geq 1$	FO-Lμ_1	EXPTIME	n4
	No		FO-Lμ_l, fixed l	EXPTIME	n5
			FO-Lμ	in NEXPTIME $\cap$ co-NEXPTIME	n6
No	Yes	fixed $r \geq 1$	$\exists$ LTL	NEXPTIME	n7
			$\forall$ LTL	co-NEXPTIME	n8
		positive	$\exists$ LTL	EXPSPACE-hard	n9
	No	fixed k	FO-Lμ	EXPTIME	n10
			FO-Lμ_l, fixed l	EXPEXPTIME	n11
			FO-Lμ	in NEXPEXPTIME $\cap$ co-NEXPEXPTIME	n12

2. AGENTS AND MULTIAGENT SYSTEMS

An *MA system* $\mathcal{A} = \{a_1, ..., a_n, P\}$ consists of a finite set $\{a_1, ..., a_n\}$ of intelligent agents with common predicate signature and a special *mail agent* P that simulates a communication channel between the agents a_i. An intelligent agent A has an *internal database (DB)* I_A that contains a finite set of ground atoms (i.e., expressions of the form $p(c_1, ..., c_k)$, where p is a predicate symbol and $c_1, ..., c_k$ are constants; the set of constants used by a given system is bounded) and a message box $MsgBox_A$.

Agents communicate via messages of the form *msg*(*Sender*, *Receiver*, *Msg*)), where *Sender* and *Receiver* are the names of agents (the source and the destination) and *Msg* is a (transferred) ground atom. An agent A sends messages to other agents of the system by transmitting these messages to the agent P and receives messages from agent P into its message box. Here one can consider two methods of operation: (i) a *synchronous* method, when it is assumed that messages are transferred from the source to the destination immediately (in this case the agent P can actually be excluded from the system) and (ii) an *asynchronous* method, when the transmission of a message from P to other agents can take arbitrary time. Synchronous MA systems are more suitable for describing practical systems that operate within a local network (or in a standalone computer), whereas asynchronous MA systems better reflect the properties of strongly distributed (for example, in the Internet) systems.

The state of the agent P contains all the messages that it has received but has not sent up to a current instant. The current contents of the internal DB and of the message box of agent A constitute its current local state $IM_A = \langle I_A, Msg, Box_A\rangle$. The *global state* of the MA system $\mathcal{A}$ consists of the local states of agents a_i and the state of the message box.

Each agent a is assigned its base AB_a of *parameterized actions* of the form $\langle\alpha(X_1, ..., X_m), ADD_\alpha(X_1, ..., X_m), DEL_\alpha(X_1, ..., X_m), SEND_\alpha(X_1, ..., X_m)\rangle$, where $\alpha(X_1, ..., X_m)$ defines a *parameterized name of an action*. $ADD_\alpha(X_1, ..., X_m)$ and $DEL_\alpha(X_1, ..., X_m)$ are lists of atoms of the form $p(t_1, ..., t_k)$, where p is a k-ary predicate from the signature of the internal DB and $t_1, ... t_k$ are either constants or the parameters $X_1, ..., X_m$. These sets define the changes in the internal DB (additions and removals of facts) during appropriate actions. Similarly, $SEND_\alpha(X_1, ..., X_m)$ defines a list of messages of the form $msg(a, b, p(t_1, ... t_k))$ that are sent by agent a to other agents. To shorten expressions, we will sometimes write $(b, p(t_1, ... t_k))$ instead of $msg(a, b, p(t_1, ... t_k))$ in the definition of $SEND_\alpha$, where α is an action of agent a.

Let $c_1, ..., c_m$ be constants. Denote by $ADD_a(c_1, ..., c_m)$ a set of facts obtained by substituting $c_1, ..., c_m$ for $X_1, ..., X_m$ into the atoms from $ADD_a(X_1, ..., X_m)$. The sets $DEL_a(c_1, ..., c_m)$ and $SEND_a(c_1, ..., c_m)$ are defined similarly. Ground atoms of the form $a(c_1, ..., c_m)$ are called *ground names of actions*.

A specific choice of these actions to be executed depending on the local state of an agent are determined by the pair $\langle Pr_A, Sel_A\rangle$. Here Pr_A is an intelligent component that determines, together with the facts from the local state of the agent at a current instant t, a set of ground action names permitted for execution at current instant. As such components, one can take ordinary logical programs [25] whose signature contains action atoms. The rules of these programs have the form

$$H \longleftarrow L_1, \ldots, L_n,$$

where H is an action atom and L_i are either action literals, (extensional) literals with predicates from the signature of the internal DB, literals of messages of the form *msg*(*Sender*, *A*, *Msg*) or ¬*msg*(*Sender*, *A*, *Msg*), or some built-in predicates that can be calculated in polynomial time. An agent is said to be *positive* if its program does not contain negations.

We assume that the programs of agents are *safe* in the sense that (i) any variable on the left-hand side of a certain rule occurs positively in the right-hand side of the appropriate rule and (ii) a program Pr_A^{state} obtained by adding facts from the local *state* (the internal DB and the message box) of agent A to Pr_A is *stratified* [25].

Under these conditions, any logical program uniquely defines a set *Perm* consisting of ground names of actions that are derived by the rules of this program from the facts of a local state of an agent and are permitted to be executed at the current instant. An operator Sel_A chooses, from the set *Perm*, a (deterministic or nondeterministic) subset *Oblig* of actions that are actually executed at this step. The simplest operator of this type is the operator of unit choice: $Sel^{un}(S) = \{\{p\} | p \in S\}$.

Let $Oblig(=Oblig_t)$ be a set, chosen according to Sel_A, of actions that are actually executed at this step t. Then all actions from *Oblig* are executed in parallel in the following way. Let ADD_{Oblig} be a union of all sets $ADD_a(c_1, \ldots, c_m)$ such that the ground name $a(c_1, \ldots, c_m)$ from *Oblig* is unified with the parameterized name $a(X_1, \ldots, X_m)$. The sets DEL_{Oblig} and $SEND_{Oblig}$ are defined analogously. Then, the next state of the internal DB of the agent A is obtained from the current state by removing elements of DEL_{Oblig} and adding elements of ADD_{Oblig}. Moreover, all elements of $SEND_{Oblig}$ are added to the base of the mail agent, and the message box of the agent is emptied. Of course, one can also consider other addition and removal strategies from the internal DB and from the message box.

An action is said to be *expanding* if the set removed by it is empty. An agent is said to be *expanding* if all of its actions are expanding.

The one-step semantics of individual agents defined above is naturally extended to the semantics $Sem_{\mathcal{A}}$ of the entire MA system $\mathcal{A}$, that defines the transformations of its global states. Namely, at the beginning of each operation step of the system, messages of the form *msg*(*sender*, *receiver*, *msg*) are removed from the message box of the agent (all—for synchronous systems, and some—for asynchronous systems) and are immediately placed into the message boxes of appropriate agents–destinations. Then all agents change their internal states according to their one-step semantics and send their messages to the mail agent. After that the message boxes of these agents are emptied. This definition shows that even if each agent has deterministic semantics, the semantics of the entire system is nondeterministic in the asynchronous case.

Synchronous systems are classified under two classes: if all agents of a system are deterministic, the system is called deterministic, while otherwise a system is called nondeterministic.

2.1. Behavior of MA Systems

According to the above definitions, any MA system $\mathcal{A} = \{a_1, \ldots, a_n, P\}$ defines a one-step transition relation $\Rightarrow_{\mathcal{A}}$ on the set of its global states.

We will consider the behavior of $\mathcal{A}$ that starts to operate in a certain global state with empty message boxes, $S^0 = \langle (I^0_{a_1}, MsgBox^0_{a_1}), \ldots, (I^0_{a_n}, MsgBox^0_{a_n}), I^0_P \rangle$, in which $MsgBox_{a_i} = \emptyset$, $1 \le i \le n$, $I^0_P = \emptyset$. This behavior is determined by the tree $\mathcal{T}_{\mathcal{A}}(S^0)$ of possible infinite trajectories (i.e., sequences of global states) of the form

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

For a deterministic MA system $\mathcal{A}$, the tree $\mathcal{T}$ consists of a single trajectory that starts in S^0. If $\mathcal{A}$ is nondeterministic or asynchronous, then $\mathcal{T}$ is an infinite tree of trajectories with root S^0. The vertices of $\mathcal{T}$ are global states $S \in \mathcal{S}_{\mathcal{A}}$ that are accessible from S^0 by the reflexive–transitive closure of the relation $\Rightarrow_{\mathcal{A}}$. If S is a vertex of $\mathcal{T}$, then the states from $Next_{\mathcal{A}}(S)$ are direct successors of this vertex in $\mathcal{T}$. A finite or infinite branch of the tree $\mathcal{T}$ that starts at some of its vertices is called a *trajectory* from $\mathcal{T}$.

The problem of verifying the dynamical properties of multiagent systems (which we call the MA-BEHAVIOR problem) is formulated as follows. Suppose given an MA system $\mathcal{A}$, its initial global state S^0, and a formula F that expresses a certain property of the trees of trajectories. It is necessary to verify if formula F holds on the tree $\mathcal{T}_{\mathcal{A}}(S^0)$.

Due to the presence of the relational structure in the states of agents and the definition of transitions by logical programs, it is natural to consider the predicate extensions of temporal logics as languages for describing the dynamical properties of MA systems.

We use some versions of predicate extensions of the following propositional logics (see [22, 24, 26, 27]): linear time logic LTL, simple subsets ALTL and ELTL of the classical branching time logic, and the much richer μ-calculus Lμ (which uses, in addition to the usual temporal operators, the operators of constructing minimal and maximal fixpoints) and its subsets $L\mu_r$

($r = 1, 2, \ldots$) with constraints imposed on the alternation depth [22, 23].[2] For a predicate extension of logic L, we will use the notation FO-L (FO stands for first-order). A simple introduction to the theory of program and temporal logics can be found in [28].

2.2. Classes of MA Systems

In all the three classes of MA systems, deterministic, nondeterministic, and asynchronous ones, we distinguish the following subclasses obtained with different natural constraints imposed on the system and its agents. An MA system $\mathcal{A} = \{a_1, \ldots, a_n\}$ is

ground if the programs P_{a_i} of all its agents are ground;[3]

k-parameter if the number of arguments (arity) of all the predicates of actions of system's agents and all the predicates in their messages are bounded by the number k. It is obvious that this property specifies the maximal number of possible parameters in the actions and messages of the system $\mathcal{A}$;

expanding if all agents of the system are *expanding*;

positive if all agents of the system are *positive*;

m-agent if the number of agents $n \leq m$; and

r-signal if the number of different ground (propositional) predicates of the agents of the system (its signals) is no greater than r.

The following simple proposition characterizes the complexity[4] of executing one step by deterministic, nondeterministic, and asynchronous MA systems.

Proposition 1.

(1) *For any deterministic MA system $\mathcal{A}$, the transition function $S \Rightarrow_{\mathcal{A}} S'$ is computable in polynomial time with respect to the size $|S| + |\mathcal{A}| + |S'|$ of the input for subclasses of ground systems or systems with a fixed number of parameters*; *in the general case*, *the transition function is computable in (deterministic) exponential time.*[5]

(2) *For any nondeterministic or asynchronous MA system $\mathcal{A}$, the transition relation* (*multivalued function*) *$S \Rightarrow_{\mathcal{A}} S'$ is recognized in nondeterministic polynomial time with respect to the size $|S| + |\mathcal{A}| + |S'|$ of the input for subclasses of ground systems or systems with a fixed number of parameters*; *in the general case*, *this function is computable in nondeterministic exponential time.*

[2] In particular, the definition of the alternation depth in the m-calculus can be found on pp. 145, 146 of the Russian translation of [24].

[3] Recall that this means that there are no variables in the programs.

[4] The complexity concepts used in the paper can be found, for example, in [20, 30].

[5] In fact, in time polynomial with respect to the size of a ground system equivalent to the original one.

3. EXAMPLE OF AN MA SYSTEM: ALLOCATION OF RESOURCES

Consider a small resource allocation system $\mathcal{RA}$ that includes a manager-agent m, which possesses some (unbounded) resource to be distributed among four agents—users u_1, u_2, u_3, and u_4 and from which it gets orders for this resource. Each agent—user has its own resourse-request strategy:

(i) u_1 requests a resource at the initial instant and then repeats the request immediately after receiving a resource from m;

(ii) u_2 requests a resource immediately after u_1 has made a request;

(iii) u_3 requests a resource immediately after u_1 has received a resource from m;

(iv) u_3 requests a resource at every step;

Manager m keeps a record of orders, and if the record list is nonempty, executes the first order; i.e., the manager delivers a resource to the agent whose order is the first in the list of orders and deletes the order from the list. At every instant of time, the list of orders can contain at most one request from each agent—user. Therefore, if m gets an order from an agent u_i before its previous request has been executed, then the new request is ignored.

One can see that all five agents in this example work by exchanging information on orders and on their execution. All of them work deterministically according to quite simple rules. However, the manager m must have a more complicated program that allows it to control the list of orders. The agents of the system $\mathcal{RA}$ are defined as follows.

The states I_{u_1} of agent u_1 may contain the facts *put_order* and *receipt*$_1$. The states I_{u_i} ($i = 2, 3, 4$) may contain the fact *receipt*$_i$, which means that u_i has received a resource at the previous step. When m executes an order of agent u_i, the latter sends a message *ok* to the former (i.e., sends the message *msg*(m, u_i, *ok*) to the mail agent); here and below, to simplify the notation when determining $SEND_\alpha$, where α is the action of an agent, we use the notation introduced in Section 2 in the definition of actions. Moreover, we omit some assumed indices α in the definition of the sets ADD_α, DEL_α, and $SEND_\alpha$.

Agent u_1 sends a message *order* to agent u_2 (when sending an order) and a message *ok* to agent u_3 (when receiving an order).

Each agent u_i ($i = 1, 2, 3, 4$) has two possible ways of action:

use_resource$_i$, consumption of a resource received, which is defined, for all i, by the sets $DEL = \{receipt_i\}$ and $ADD = SEND = \emptyset$, and *receive*$_i$, reception of a resource from m, which is defined, for $i = 2, 3, 4$, by the sets $ADD_i = \{receipt_i\}$ and $DEL_i = SEND_i = \emptyset$, while, for $i = 1$, the action $receive_1$ is defined by the sets $ADD_1 = \{receipt_1\}$, $DEL_1 = \{put\_order\}$, and $SEND_1 = \{(u_3, ok)\}$.

These actions are initiated by the following rules:

$use\_resource_i \longleftarrow receipt_i$ and

$receive_i \longleftarrow msg(m, u_i, ok)$.

In addition, the agents have an action *put* that sends an order (the message *order*), with specific effects and rules, to the manager.

For agent u_1, the action *put* is defined by the sets $ADD = \{put\_order\}$ and $SEND = \{(m, order), (u_2, order)\}$ and is initiated by the rule

$$put \longleftarrow \neg put\_order.$$

For agent u_2, the action *put* is defined by the set $SEND = \{(m, order)\}$ and is initiated by the rule

$$put \longleftarrow msg(u_1, u_2, order).$$

For agent u_3, the action *put* is defined by the set $SEND = \{(m, order)\}$ and is initiated by the rule

$$put \longleftarrow msg(u_1, u_3, ok).$$

For agent u_4, the action *put* is defined by the set $SEND = \{(m, order)\}$ and is initiated by the rule (fact)

$$put \longleftarrow .$$

The agent m controls the list of orders given by the following predicates of the internal DB:

$first(A)$, the order of agent A (stands first in the list;

$next(A, B)$, the order of agent B stands next after the order of agent A in the list;

$last(A)$, the order of agent A stands last in the list.

The manager m also uses auxiliary predicates (they can be considered as actions with empty sets *ADD*, *DEL*, and *SEND*): *empty_queue*, which checks if the list is empty, and $in\_queue(A)$, which checks if the order of A is in the list. These predicates are defined by the following propositions:

$$empty\_queue \longleftarrow \neg first(u_1),$$
$$\neg first(u_2), \neg first(u_3), \neg first(u_4), \text{ and}$$
$$in\_queue(A) \longleftarrow first(A),$$
$$in\_queue(A) \longleftarrow in\_queue(B), next(B, A).$$

When m gets new orders, it places them at the end of the list; if it should place several orders simultaneously, it places them in the following predetermined order: $u_1 < u_2 < u_3 < u_4$. For any subsequence $X_1 < \ldots < X_i$ of the sequence (u_1, u_2, u_3, u_4), m has the action $insert(F, S, L, X_1, \ldots, X_i)$:

$$ADD = \{first(S), next(L, X_1), next(X_1, X_2),$$
$$\ldots, next(X_{i-1}, X_i), last(X_i)\},$$
$$DEL = \{first(F), next(F, S), last(L)\}, \text{ and}$$
$$SEND = \{(F, ok)\}.$$

Each of these actions is initiated by the rule

$$insert(F, S, L, X_1, \ldots, X_i)$$
$$\longleftarrow new\_order(X_1), \ldots, new\_order(X_i),$$
$$\neg new\_order(Y_1), \ldots, \neg new\_order(Y_j),$$
$$first(F), next(F, S), last(L)$$

(here $\{Y_1, \ldots, Y_j\} = \{u_1, u_2, u_3, u_4\} \backslash \{X_1, \ldots, X_i\}$, $new\_order(X) \longleftarrow msg(X, m, order), \neg in\_queue(X)$. When the list is empty, m starts one of actions of the form $insert_0(X_1, \ldots, X_i)$:

$$ADD = \{first(X_1), next(X_1, X_2), \ldots,$$
$$next(X_{i-1}, X_i), last(X_i)\} \text{ and}$$
$$DEL = SEND = \emptyset$$

that are initiated by propositions of the form

$$insert_0(X_1, \ldots, X_i)$$
$$\longleftarrow empty\_queue, new\_order(X_1), \ldots,$$
$$new\_order(X_i), \neg new\_order(Y_1), \ldots,$$
$$\neg new\_order(Y_j)$$

(here again $\{Y_1, \ldots, Y_j\} = \{u_1, u_2, u_3, u_4\} \backslash \{X_1, \ldots, X_i\}$).

For the system $\mathcal{RA}$ defined here, the formula **G**, **F**, $\bigwedge_{i=1}^{4} receipt_i$ is true on a trajectory starting from the empty state $S_0 = \{\langle\emptyset, \emptyset\rangle, \langle\emptyset, \emptyset\rangle, \langle\emptyset, \emptyset\rangle, \langle\emptyset, \emptyset\rangle, \langle\emptyset, \emptyset\rangle\}$. Meaningfully, this formula implies that each agent receives the necessary resource infinitely frequently.[6]

At the same time, the following two formulas are not valid on this trajectory: $\mathbf{F}\,(receipt_1) \wedge \mathbf{X}\,(receipt_1)$ (there are two consecutive instants at which agent u_1 receives resources) and $\mathbf{G}\,(first(S) \wedge next(S, u_i) \longrightarrow \mathbf{XX} \neg receipt_i)$ (the agent that is second in the list will not receive a resource after two steps).

If we place these agents in an asynchronous system $\mathcal{RA}_{as}$, then simple formulas like $\forall\mathbf{G}\ \forall\mathbf{F}\ receipt_1$, or even $\forall\mathbf{F}\ receipt_1$, will not hold for $\mathcal{RA}_{as}$.[7] This is associated with the fact that, although u_1 requests a resource immediately at the first step, there exist trajectories on which this request will never be transferred to the agent m.

However, the formula $\forall\mathbf{G}\ \forall\mathbf{F}\ receipt_1$ holds for $\mathcal{RA}_{as}$. Indeed, let s be a vertex of the tree of trajectories. If the message $msg(m, u_1, ok)$ is in the message box of an agent in this state, then it can be transferred to u_1 at the next step and will cause the action $receive_1$. If this message is not in the message box of the agent, we consider the last message $Msg = msg(u_1, m, order)$ produced by the agent u_1 before the system reaches the state s. It is easy to understand that, at the instant when the system reaches the state s, the message Msg is either at the mail agent or in the list of the agent m

[6] The operators **G**, **F**, and **X** of linear time logic mean "always in future," "sometimes in future," and "at the next instant," respectively.

[7] The operators $\forall\mathbf{G}$, $\forall\mathbf{F}$, and $\exists\mathbf{F}$ of the branching time logic imply "in all possible future states," "on all trajectories, there is an instant," and "there is a possible future state," respectively.

(when this message disappears, the mail agent necessarily receives the message $msg(m, u_1, ok)$). When *Msg* is in the list, the agent *m*, while processing its list, reaches this message and sends the message $msg(m, u_1, ok)$ to the mail agent (we obtain the previous case). If *Msg* is at the mail agent, then the operation can be continued from the state *s*, and this message is sent to the agent *m* and appears in the list.

By similar arguments one can show that even the stronger formula $\forall \mathbf{G} \ \forall \mathbf{F} \ \bigwedge_{i=1}^{4} receipt_i$ holds for $\mathcal{RA}_{as}$. Some more complicated properties can be expressed by formulas of the modal μ-calculus Lμ.

4. BEHAVIOR OF DETERMINISTIC MA SYSTEMS

This section, where we discuss deterministic MA systems, is in fact a concise account of the contents of Section 4.1 from [2]. The results of this section will be used below in Subsection 6.1.

The set of global states of any MA system $\mathcal{A}$ considered here is finite. Therefore, the trajectory $\tau(\mathcal{A}, S^0)$ of this set in the deterministic case is periodic. Thus, in spite of the fact that $\tau(\mathcal{A}, S^0)$ is infinite, it can be folded into a certain finite structure. A direct algorithm for verifying an FO-LTL formula on this structure would require an explicit representation of this formula and, hence, memory no less than the total number of various global states. However, one can suggest a more careful method for verifying this formula on a model that will deal with small subsets of this formula point by point. This allows one to obtain much better complexity bounds for the MA-BEHAVIOR problem.

For a periodic trajectory $\tau = S^0, S^1, ..., S^t, ...$, denote by *k* and *N* minimal numbers such that $S^t = S^{t+N}$ for any $t \geq k$. In the algorithm given below, we will use three auxiliary functions. The first function, *move*(*t*, *i*), calculates, by a time instant *t* and a shift *i*, a time instant $j < k + N$ such that $S^j = S^{t+i}$:

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

The second function, F^τ, plays the role of an oracle, which yields a state $F^\tau(t) = S^t$ of the trajectory at any instant *t*. The third function verifies if a first-order formula Φ is true on state *S*: FO_Check(*S*, Φ) returns the logical value TRUE if $S \models \Phi$ and FALSE otherwise.

Let $\tau = \tau(\mathcal{A}, S^0)$ be a periodic trajectory with parameters *k* and *N*, Φ be an arbitrary FO-LTL formula, and *t* be a time instant. Set $s_{\max}(\tau) = \max\{|S^t| \mid 0 \leq t \leq k + N\}$ and denote by $s(F^\tau)$ and $t(F^\tau)$, respectively, the maximal memory and time needed for calculating $F^\tau(t)$ for $0 \leq t \leq k + N$. Denote by $s_{\text{FO}}(t, n)$ and $t_{\text{FO}}(\tau, n)$, respectively, the maximal memory and time needed for verifying the condition $S^t \models \Psi$ for $0 \leq t \leq k + N$ and any first-order formula Ψ of length at most *n*.

The following recursive algorithm verifies the property $\tau, S^t \models \Phi$.

```
Algorithm DetCheck(τ, k, N, Φ, 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^τ(t);
(6) return FO_Check(S^t, Φ);
(7) CASE Φ = Φ_1 ⊕ Φ_2 (⊕ ∈ {∧, ∨});
(8) b_1 := DetCheck(τ, k, N, Φ_1, t);
(9) b_2 := DetCheck(τ, k, N, Φ_2, t);
(10) return b_1 ⊕ b_2;
(11) CASE Φ = ¬Φ_1;
(12) return ¬ DetCheck(τ, k, N, Φ_1, t);
(13) CASE Φ = X(Φ_1);
(14) t_1 := move(t, 1);
(15) return DetCheck(τ, k, N, Φ_1, t_1);
(16) CASE Φ = Φ_1 U Φ_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 DetChek(τ, k, N, Φ_2, r);
(22) THEN EXIT_FOR END_IF;
(23) END_DO;
(24) IF p = R − 1;
(25) THEN return TRUE;
(26) ELSE;
(27) b := TRUE;
(28) FOR j = 0 TO p − 1 DO;
(29) r' := move(t, j);
(30) IF ¬DetChek(τ, k, N, Φ_1, r');
(31) THEN b := FALSE; EXIT_FOR;
(32) END_IF END_DO;
(33) return b END_IF
(34) END_SELECT
```

Lemma 1. *For given numbers k, N, and t and an FO-LTL formula* Φ, *the algorithm DetCheck verifies the condition* $\tau^t \models \Phi$ *on a periodic trajectory* τ *with parameters k and N using the functions* F^τ *and FO_Check as oracles. The memory used by the algorithm is* $\mathbf{O}(|t| + |\Phi| + d^*(\Phi)\log(k + N) + s_{\max}(\tau) + s(F^\tau) + s_{\text{FO}}(t, |\Phi|))$, *and the time is limited by* $pol(|t| + |\Phi|(k + N)(t(F^\tau) + t_{\text{FO}}(t, |\Phi|))$ *for a polynomial pol.*

The oracle F^τ, which provides information on the trajectory τ in the algorithm, can be efficiently calculated for trajectories τ generated by MA systems.

Lemma 2. *There exists an algorithm that calculates the state* S^t *of the trajectory* $\tau(\mathcal{A}, S^0)$ *at instant t by the MA system* $\mathcal{A}$, *its initial state* S^0, *and the time instant* $t \geq 0$.

For a polynomial pol, the upper bound for the memory used for this calculation is $pol(|\mathcal{A}| + \max\{|S^r| \mid 0 \le r \le t\})$.

The following lemma establishes the boundaries of the periodicity parameters for the trajectories of MA systems.

Lemma 3. *For any MA system* $\mathcal{A}$ *and its 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 a ground system, then* $k(\mathcal{A}, S^0) + N(\mathcal{A}, S^0) \le 2^{pol(|\mathcal{A}| + |S^0|)}$. *In the general case, the inequality* $k(\mathcal{A}, S^0) + N(\mathcal{A}, S^0) \le 2^{2^{pol(|\mathcal{A}| + |S^0|)}}$ *holds.*

From Lemmas 1, 2, and 3 we can obtain upper bounds for the complexity of verification of the properties of MA systems that can be expressed in terms of FO-LTL formulas.

Proposition 2. *Suppose given an MA system* $\mathcal{A}$ *and its initial state* S^0, *and let* $k = k(\mathcal{A}, S^0)$ *and* $N = N(\mathcal{A}, S^0)$. *Then, for a polynomial pol, the validity of the FO-LTL formula* Φ *on the trajectory* $\tau(\mathcal{A}, S^0)$ *can be verified with the use of memory of size* $2^{pol(|\Phi| + |\mathcal{A}|)}$ *in the general case and with memory of size* $pol(|\Phi| + |\mathcal{A}|)$ *in the case of ground systems.*

5. COMPLEXITY OF VERIFICATION OF DETERMINISTIC AND NONDETERMINISCTIC MA SYSTEMS

In our previous studies [1–3], we solved the problem of the complexity of the MA-BEHAVIOR problem for classes of deterministic and nondeterministic MA systems and a number of their subclasses. The main results obtained in those papers are summarized in Tables 1 and 2.

The first three columns in these tables determine the classes of MA systems considered. The columns "Ground" and "Expand" allow one to distinguish between ground and nonground systems and between expanding (without removals) systems and systems with removals. The column "Parameters" indicates some constraints imposed on the parameters of the architecture of MA systems that lead to lower complexity of verification. In particular, N denotes the size of the total description of a system, m is the number of its agents, k is the maximal size of the action atoms and messages (in the propositional case, $k = 0$), and r is the number of different signals (i.e., messages). The column "Logic" indicates the logical language in which the properties of the behavior of appropriate MA systems are formulated. The column "Complexity" shows the complexity bound of the appropriate MA-BEHAVIOR problem.

In all the rows except for n6, n9 and n12, the appropriate MA-BEHAVIOR problem is complete in a given complexity class. The rows n6 and n12 show only upper complexity bounds, and the row n9, only a lower bound.

It is no wonder that the MA-BEHAVIOR problem is very complex in the general case; it is EXPSPACE-complete for deterministic systems (d9) and EXPEXTIME-complete for nondeterministic systems (n11). However, we could distinguish some subclasses of systems in which this problem is solved efficiently. These systems either have strong monotonicity properties (d1 and d6) or do not remove any facts from their databases and have constraints on their structure: $m^2 * r = O(\log N)$ (d2). The rows d3, d4, and d7 show that the removal of any of the constraints leads to an immediate increase in the complexity.

As regards nondeterministic systems, one can hardly expect efficient solution of the verification problem even for very limited subclasses of these systems. Propositions n1 and n2 describe the cases when this problem is solvable in (co)nondeterministic polynomial time. The bounds n3–n5 show that relaxing the conditions leads to the complication of the problem. Note that, in all these cases one can obtain a sharper lower bound of the form $\mathbf{o}(c^{n/\log n})$. Note also that upper bounds for the languages FO-Lμ, FO-Lμ_1, and FO-Lμ_l are obtained by generalizing the known general results of [23] on the complexity of verification on models for the μ-calculus.

6. BEHAVIOR OF ASYNCHRONOUS SYSTEMS

To verify asynchronous MASs, it is important to take into consideration the internal state of a mail agent, which consists of atoms of the form $msg(a_i, a_j, p)$. However, these atoms are also encountered in the message boxes of agents. Therefore, to correctly refer to these atoms in formulas, we adopt the following notation: $msg^j(a_i, a_j, p)$ in the formulas denotes an atom the truth value of which is evaluated with respect to the message box of the agent a_j, and $msg(a_i, a_j, p)$ denotes an atom evaluated over the state of the mail.

Nondeterministic and asynchronous systems are rather similar to each other. This fact is formally confirmed by the results of Subsection 6.2 on the mutual reducibility of MA-BEHAVIOR problems for nondeterministic and asynchronous MASs. This allows one to directly translate the results on the complexity of verification for nondeterministic MASs to asynchronous MASs, but only for general systems. This is associated with the fact that the modeling of expanding nondeterministic systems involves nonmonotonic asynchronous systems and vice versa. Nevertheless, in Subsection 6.1 we show that the result on the NP-completeness of the MA-BEHAVIOR problem for a subclass of nondeterministic expanding MASs is carried over to a similar class of asynchronous MASs.

6.1. A Subclass of Asynchronous MASs with an NP-Complete MA-BEHAVIOR Problem

Theorem 1. *The MA-BEHAVIOR problem with the behavior properties expressed by formulas $\Phi \in \exists LEL$ ($\forall LTL$) that do not contain atoms–messages for a class of ground, expanding, r-signal, m-agent asynchronous systems $\mathcal{A}$ such that $m^2 * r = \mathbf{O}(\log|\mathcal{A}|)$ is NP-complete (respectively, coNP-complete).*

Proof. We give a proof for $\exists LTL$ formulas. The case of $\forall LTL$ formulas is obtained by the equivalence $\mathcal{T}_{\mathcal{A}}(S^0) \models \mathbf{E}(\Psi) \Leftrightarrow \mathcal{T}_{\mathcal{A}}(S^0) \not\models \mathbf{A}(\neg\Psi)$.

Upper bound. Let $\mathcal{A}$ be a ground, expanding, r-signal, m-agent asynchronous system such that $m^2 * r = \mathbf{O}(\log|\mathcal{A}|)$. A nondeterministic algorithm that solves the MA-BEHAVIOR problem in polynomial time is based on the following stabilization property of some trajectories from $\mathcal{T}_{\mathcal{A}}(S^0)$.

Lemma 4. *There exists a polynomial $p(n)$ such that $\mathcal{T}_{\mathcal{A}}(S^0) \models \mathbf{E}\,\Psi$ if and only if there exist a trajectory $\mu = S^0, \ldots, S^t, \ldots \in \mathcal{T}_{\mathcal{A}}(S^0)$ and a step $T \le p(|\mathcal{A}| + |\Psi|)$ such that $\mu, S^0 \models \Psi$, and $I_a^t = I_a^T$ for any $t > T$ and any $a \in \mathcal{A}$.*

Proof. First, we prove a few auxiliary propositions related to the truth of FO-*LTL* formulas on the trajectories with long sequences of repeated states. Recall that the formulas depend only on the states of the DB of a trajectory but not on the states of the corresponding message boxes.

Let $\mu = M^0, \ldots, M^i, \ldots$, and $\nu = N^0, \ldots, N^j, \ldots$ be two trajectories and $d \ge 0$ be an integer. We say that a pair (μ, M^i) is d-equivalent to a pair (ν, N^j) (the notation is $(\mu, M^i) \sim_d (\nu, N^j)$) if the equivalence $\mu, M^i \models \varphi \Leftrightarrow \nu, N^j \models \varphi$ holds for any FO-LTL formula φ of depth $d(\varphi) \le d$.

Proposition 1. *If the equivalences $(\mu, M^i) \sim_d (\nu, N^j)$ and $(\mu, M^{i+1}) \sim_{d+1} (\nu, N^{j+1})$ hold for certain $d \ge 0$, then the equivalence $(\mu, M^i) \sim_{d+1} (\nu, N^j)$ is also valid.*

Proof. Indeed, let $(\mu, M^i) \sim_d (\nu, N^j)$ and $(\mu, M^{i+1}) \sim_{d+1} (\nu, N^{j+1})$, and let φ be an arbitrary FO-LTL formula of depth $d + 1$. If $\varphi = \varphi_1 \oplus \varphi_2$ ($\oplus \in \{\wedge, \vee\}$) or $\varphi = \neg\varphi_1$, then $d(\varphi_1) \le d$ and $d(\varphi_2) \le d$. By the assumption, $\mu, M^i \models \varphi_k \Leftrightarrow \nu, N^j \models \varphi_k$ ($k = 1, 2$). Hence, $\mu, M^i \models \varphi \Leftrightarrow \nu, N^j \models \varphi$. If $\varphi = \mathbf{X}(\varphi_1)$, then, by the assumption, we have $\mu, M^{i+1} \models \varphi_1 \Leftrightarrow \nu, N^{j+1} \models \varphi_1$ and $\mu, M^i \models \varphi \Leftrightarrow \nu, N^j \models \varphi$. Now, suppose that $\mu, M^i \models \varphi$ for $\varphi = \varphi_1\mathbf{U}\varphi_2$. Then, by the definition of the operator $\mathbf{U}$, (i) $\mu, M^i \models \varphi_2$ or, otherwise (ii) $\mu, M^i \models \varphi_1$ and $\mu, M^{i+1} \models \varphi$. In case (i), taking into account that $d(\varphi_2) = d$, from the first assumption we obtain $\nu, N^j \models \varphi_2$ and, hence, $\nu, N^j \models \varphi$. Similarly, in case (ii) we can show that $\nu, N^j \models \varphi_1$. Moreover, from the second assumption we obtain $\nu, N^{j+1} \models \varphi$. Thus, in this case we also obtain $\nu, N^j \models \varphi$. Hence, in all cases we obtain $\nu, N^j \models \varphi$ and $(\mu, M^i) \sim_{d+1} (\nu, N^j)$. □

Proposition 1 immediately implies the following.

Proposition 2. *If $(\mu, M^i) \sim_0 (\nu, N^j)$ and $(\mu, M^{i+1}) \sim_d (\nu, N^{j+1})$ for some $d \ge 0$, then $(\mu, M^i) \sim_d (\nu, N^j)$.*

Proposition 3. *Let $\mu = M^0, \ldots$ be a certain trajectory and i be a step number such that $(\mu, M^i) \sim_d (\mu, M^{i+1}) \sim_d (\mu, M^{i+2})$. Then $(\mu, M^i) \sim_{d+1} (\mu, M^{i+1})$.*

Proof. Let φ be an FO-LTL formula of depth $d(\varphi) = d + 1$. If φ is a Boolean combination of formulas of depth $\le d$, then $\mu, M^i \models \varphi \Leftrightarrow \mu, M^{i+1} \models \varphi$ because $(\mu, M^i) \sim_d (\mu, M^{i+1})$. If $\varphi = \mathbf{X}(\varphi_1)$, then $d(\varphi_1) \le d$ and $\mu, M^i \models \varphi \Leftrightarrow \mu, M^{i+1} \models \varphi$ because $(\mu, M^{i+1}) \sim_d (\mu, M^{i+2})$. If $\varphi = \varphi_1\mathbf{U}\varphi_2$, then $d(\varphi_1) \le d$ and $d(\varphi_2) \le d$. Suppose that $\mu, M^i \models \varphi$. Then, by the definition of the operator $\mathbf{U}$, we have $\mu, M^i \models \varphi_2$, or $\mu, M^i \models \varphi_1$ and $\mu, M^{i+1} \models \varphi$. In both cases, it is clear that $\mu, M^{i+1} \models \varphi$ (in the first case we used the assumption $M^i \sim_d M^{i+1}$. On the other hand, suppose that $\mu, M^{i+1} \models \varphi$. Then, (i) $\mu, M^{i+1} \models \varphi_1$ or (ii) $\mu, M^{i+1} \models \varphi_2$. In case (i), from the assumption $(\mu, M^i) \sim_d (\mu, M^{i+1})$ we obtain $\mu, M^i \models \varphi_1$; combined with $\mu, M^i \models \varphi$, this yields $\mu, M^i \models \varphi$. Case (ii) is analogous. □

Proposition 3 immediately implies

Proposition 4. *Let $\mu = M^0, \ldots$ be a certain trajectory and i be a step number such that $M^i = M^{i+1} = M^{i+2} = \ldots = M^{i+2+K}$. Then $(\mu, M^i) \sim_K (\mu, M^{i+1})$.*

Proposition 5. *Let $\mu = M^0, \ldots$ be a certain trajectory and i be a step number such that $M^i = M^{i+1} = M^{i+2} = \ldots = M^{i+2+K}$. Suppose that the trajectory $\mu' = M^0, \ldots, M^{i-1}, M^{i+1}, \ldots, M^{i+2+K}, \ldots$ is obtained from μ by removing M^i. Then $(\mu, M^0) \sim_K (\mu', M^0)$.*

Proof. It follows from Proposition 4 that $(\mu, M^i) \sim_K (\mu, M^{i+1})$. Then $(\mu, M^{i-1}) \sim_0 (\mu', M^{i-1})$ and $(\mu, M^i) \sim_K (\mu', M^{i+1})$. It follows from Proposition 2 that $(\mu, M^{i-1}) \sim_K (\mu, M^{i-1})$. Since $(\mu, M^{i-2}) \sim_0 (\mu', M^{i-2})$, we obtain $(\mu, M^{i-2}) \sim_K (\mu, M^{i-2})$ and so on, until we reach the equivalence $(\mu, M^0) \sim_K (\mu', M^0)$. □

Now, let us return to the proof of Lemma 4. Suppose that $\mathcal{T}_{\mathcal{A}}(S^0) \models \mathbf{E}\,\Psi$. This means that $\lambda, S^0 \models \Psi$ for a certain trajectory $\lambda = S^0, \ldots, S^t, \ldots, \in \mathcal{T}_{\mathcal{A}}(S^0)$. Let $t_1, t_2, \ldots, t_i, \ldots, t_k$ be the steps of the trajectory λ at which the states of the agents increase; i.e., $I_a^{t_i} \subset I_a^{t_i+1}$ for some $a \in \mathcal{A}$. Since $\mathcal{A}$ is an expanding system, k is bounded by the total number of actions of agents from $\mathcal{A}$. Hence, $k \le |\mathcal{A}|$. Due to the choice of steps of the form t_i, the states of DBs of all agents from $\mathcal{A}$ are not changed at steps $t_i + 1, t_i + 2, \ldots, t_{i+1}$ for any i. Let $k_i = t_{i+1} - t_i$ be the length of such a stable subsequence of states of the DB. Denote by N_m the number of all possible states of message boxes and of the mail agent P. The number of various messages in the message box of any agent and in the state P does not exceed rm. Then $N_m \le (2^{rm})^{m+1} = 2^{rm^2+rm} \le 2^{O(\log|\mathcal{A}|)} \le |\mathcal{A}|^c$ for some constant c.

Let $d = d(\Psi)$ and $k_i > d + 2 + N_m$. Then, there exist steps l and r, $t_i + d + 2 < l < r < t_{i+1}$ such that $S^l = S^r$. Then $\mathcal{T}_{\mathcal{A}}(S^0)$ has a trajectory $\mu = S^0, ..., S^l, S^{r+1}, ...$ that is obtained from λ by the removal of the states $S^{l+1}, ..., S^r$. It follows from Proposition 5 that $(\lambda, S^0) \sim_d (\mu, S^0)$, and, hence, $\mu, S^0 \models \Psi$. Then, there exits a trajectory $\mu \in \mathcal{T}_{\mathcal{A}}(S^0)$ such that $\mu, S^0 \models \Psi$, and the maximal length of the sequence of identical states of the DBs in μ for some constants c and c_1 is bounded by the number $d + 2 + N_m \le c_1|\mathcal{A}|^c + |\Psi|$). For this trajectory, the step T at which the above sequence is stabilized does not exceed $t_k + 1 \le k + k(d + 2 + N_m) \le pol(|\mathcal{A}| + |\Psi|)$. $\square$

The upper bound in Theorem 1 follows from Lemmas 1 and 4. Namely, let $\mathcal{A}$ be an asynchronous, ground, expanding system with at most r signals and m agents, and let $m^2 * r = \mathbf{O}(\log|\mathcal{A}|)$. Denote by S^0 its arbitrary initial state. Let $\mathcal{T} = \mathcal{T}_{\mathcal{A}}(S^0)$ be a tree of trajectories of $\mathcal{A}$ that start in S^0 and Ψ be an arbitrary LTL formula. Set $T = pol(|\mathcal{A}| + |\Psi|)$, where the polynomial pol is defined in Lemma 4. To find out if $\mathcal{T}_{\mathcal{A}}(S^0) \models \mathbf{E}\Psi$ holds, we apply the following nondeterministic algorithm NDetCh:

(1) guess a finite trajectory $\lambda = S^0, S^1, ..., S^T, ..., S^{2T}$ in the tree $\mathcal{T}_{\mathcal{A}}(S^0)$, in which $I_a^T = I_a^{T+1} = I_a^{T+2} = ... = I_a^{2T}$ for any $a \in \mathcal{A}$;

(2) using the algorithm DetCheck, verify if $\lambda', S^0 \models \Psi$ holds and give answer "Yes" if $\lambda, S^0 \models \Psi$.

It is clear that NdetCheck is run in nondeterministic polynomial time. To prove its correctness, note that if $\mathcal{T}_{\mathcal{A}}(S^0) \models \mathbf{E}\Psi$, then, by Lemma 4, there exists a trajectory $\lambda \in \mathcal{T}_{\mathcal{A}}(S^0)$, $\lambda, S^0 \models \Psi$, that has a finite prefix $\lambda' = S^0, ..., S^T$ such that $\lambda', S^0 \models \Psi$ and $I_a^j = I_a^T$ for any $a \in \mathcal{A}$ and any $j > T$. Thus, at step (1) of the algorithm, one can guess this short prefix and, at step (2), verification by DetCheck gives answer "Yes."

Conversely, if the algorithm NdetCheck gives answer "Yes," then there is a finite trajectory $\lambda = S^0, S^1, ..., S^T, ..., S^{2T}$ in the tree $\mathcal{T}_{\mathcal{A}}(S^0)$, such that $\lambda, S^0 \models \Psi$ and $I_a^T = I_a^{T+1} = I_a^{T+2} = ... = I_a^{2T}$ for any $a \in \mathcal{A}$. Since $T > N_m$, there exist i and j $(T < i < j < 2T)$ such that $S^i = S^j$. Then this prefix λ of length T can be extended to a certain infinite trajectory $\lambda' \in \mathcal{T}_{\mathcal{A}}(S^0)$ such that $\lambda' = S^0, ... S^T, S'^{T+1}, ...$ and $I_a^j = I_a^T$ for any $a \in \mathcal{A}$ and $j > T$. For this trajectory, we have $\lambda', \lambda', S^0 \models \Psi$ and $\mathcal{T}_{\mathcal{A}}(S^0) \models \mathbf{E}\Psi$.

Lower bound. Let us show that the satisiability of the Boolean formulas SAT is reduced in polynomial time to the MA-BEHAVIOR problem for ground, expanding, 1-signal, 2-agent asynchronous systems. Let α be a propositional formula and $V = \{x_1, ..., x_n\}$ be a set of variables in this formula. Consider an MA system $\mathcal{A}$ that has two agents a and b. At every step, the agent b sends message "1" to the agent a. The mail agent asynchronously forwards these messages to the agent a. If the agent a receives message "1" at the ith step, then it places the facts OK_i and x_i in its database, whereas, if a does nor receive a message, it places only the fact OK_i in its database. The extensional signature of a consists of V and $\{OK_i | 1 \le i \le n\}$. Its ground action AB_a includes $2n$ actions ac_{i0}, ac_{i1} $(i = 1, ..., n)$. Each action ac_{i1} adds facts OK_i and x_i to I_a, while the action ac_{i0} adds fact OK_i. The program P_a for each $i \in [1, n]$ contains two propositions:

$$ac_{i1} \longleftarrow OK_1, \ldots, OK_{i-1}, \neg OK_i, msg(b, 1) \text{ and}$$

$$ac_{i0} \longleftarrow OK_1, \ldots, OK_{i-1}, \neg OK_i, \neg msg(b, 1).$$

These definitions imply that, after n steps, all facts of the form OK_i $(1 \le i \le n)$ and a subset of facts from V appear to be in I_a; moreover, for any such subset, there is an appropriate trajectory of $\mathcal{A}$. Then, one can easily check that $\varphi \in SAT \Leftrightarrow \mathcal{T}_{\mathcal{A}}(S^0) \models \mathbf{E}(\varphi)$ holds for the empty initial state $S^0 = \emptyset$. $\square$

The following theorem shows that the complexity of the MA-BEHAVIOR problem rapidly increases if one relaxes the constraints on the class of MA systems. In particular, if we restrict only the number of agents or only the number of signals in the class of expanding systems, then the problem becomes EXPTIME-complete, just as it is in the general case of ground systems.

Theorem 2. (1) *The MA-BEHAVIOR problem is EXPTIME-complete for asynchronous, ground, expanding, m-agent systems and the behavior properties* Φ *from FO*-Lμ_1 (*for any fixed* $m \ge 2$).

(2) *The MA-BEHAVIOR problem is EXPTIME-complete for asynchronous, ground, expanding, r-signal systems and the behavior properties* Φ *from FO-L*μ_1 (*for any fixed r*).

(3) *In both cases, there exists a constant* $c > 1$ *such that appropriate versions of the MA-BEHAVIOR problem cannot be solved with deterministic time complexity less than* $c^{n/\log n}$.

(4) *The MA-BEHAVIOR problem for asynchronous ground MA systems*

(i) *is EXPTIME-complete for the behavior properties* Φ *from FO*-Lμ_r (*for any fixed r*).

(ii) *belongs to the class NEXPTIME* $\cap$ *coNEXPTIME for the behavior properties* Φ *from FO-L*μ.

The proof of this theorem is obtained by a slight modification of the proof of Theorem 7 in [3].

6.2. Mutual Reducibility of MA-BEHAVIOR Problems for Nondeterministic and Asynchronous MASs

The following theorem shows how nondeterministic MASs with unit choice operator can be modeled by asynchronous systems.

Theorem 3. *Suppose that the FO-*$L\mu_r$ *language is used to formulate the dynamical properties of MASs. Then the MA-BEHAVIOR problem for nondeterministic MASs with the unit choice operator* Sel^{un} *is reduced in polynomial time to the MA-BEHAVIOR problem for asynchronous ground MASs with deterministic agents.*

Proof. For simplicity, we give a proof only for ground MASs. The proof for nonground MASs is analogous.

Suppose that $\mathcal{A} = \{a_1, a_2, \ldots, a_n\}$ is a nondeterministic MAS each of whose agents $a \in \mathcal{A}$ uses the operator Sel^{un} for a nondeterministic choice of its actions. Let S^0 be the initial state of $\mathcal{A}$. Let us construct, by $\mathcal{A}$, S^0, and the FO-Lμ formula Φ, an asynchronous MAS $\mathcal{B}$, its initial state R^0, and an FO-Lμ formula Ψ so that $\mathcal{A}, S^0 \models \Phi \Leftrightarrow \mathcal{B}, R^0 \models \Psi$.

Let $Act_i = \{\alpha_{i1}, \ldots, \alpha_{im_i}\}$ be a set of ground action atoms of agent a_i.

The asynchronous MAS $\mathcal{B}$ consists of agents a_1', ..., a_n', two additional agents b and c, and a mail agent PA. At every step, the agent c sends three messages "1," "2," and "3" to each agent a_i' (we will focus on the trajectories on which a_i' receives these messages in the cyclic order 1–2–3).

For every action $\alpha \in Act_i$, agent a_i' has a new message α and a new duplicate of this action α' with the lists $ADD_{\alpha'} = DEL_{\alpha'} = \emptyset$ and $SEND_{\alpha'} = \{(a_i', b, \alpha)\}$. The program $P_{a_i'}$ includes

(i) all propositions P_{a_i} in which every occurrence of any atom of action α is replaced by α' and an atom $msg(c, a_i', 1)$ is added to the body of each proposition;

(ii) the proposition

$$\alpha \longleftarrow msg(b, a_i', \alpha'), msg(c, a_i', 3)$$

for every $\alpha \in Act_i$.

For every a_i' and $\alpha \in Act_i$, the agent b includes the action α^{a_i} with the lists $ADD_{\alpha^{a_i}} = DEL_{\alpha^{a_i}} = \emptyset$ and $SEND_{\alpha^{a_i}} = \{(b, a_i, \alpha)\}$. For any a_i' and $\alpha \in Act_i$, the program P_b contains the proposition $\alpha^{a_i} \longleftarrow msg(a_i, b, \alpha)$.

Some trajectories of $\mathcal{B}$ model the trajectories of $\mathcal{A}$; in this case, one step $S \Rightarrow_{\mathcal{A}} S'$ of the system $\mathcal{A}$ corresponds to three steps of the system $\mathcal{B}$.

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

$$\{msg(b, a_i, \alpha) | \alpha \in Perm_{a_i}\}$$

to the agent a_i', and one of these messages necessarily reaches a_i'.

At the third step, a_i' executes an action received from b (i.e., a_i' changes its internal DB and sends all the messages that should be sent according to this action to all the agents a_j'), while the PA sends the remaining part of $Perm_{a_i'}$ to the agent a_i', and these messages are lost at the next step.

The initial state $R^{(0)}$ of the system $\mathcal{B}$ is defined as follows: $I_{a_i'}^{(0)} = I_{a_i}^{(0)}$, $Msgbox_{a_i'}^{(0)} = Msgbox_{a_i}^{(0)} \cup \{msg(c, a_i, 1)\}$, $I_c^{(0)} = I_b^{(0)} = I_{PA}^{(0)} = Msgbox_b^{(0)} = Msgbox_c^{(0)} = \emptyset$.

The formula $\Psi = \Psi(\Phi, \mathcal{A})$ is obtained from Φ by inductive replacement of all subformulas of the form $\exists X\Theta$ by the formulas $\exists X(f_1 \wedge \exists X(f_2 \wedge \exists X(f_3 \wedge \Theta)))$, where f_1, f_2, and f_3 are given by

$$\bigwedge_{i=1}^{n}\Big(\bigwedge_{j=1}^{m_i} \neg msg(a_i', b, \alpha_{ij}) \wedge msg^i(c, a_i', 2)$$

$$\wedge \neg msg^i(c, a_i', 1) \wedge \neg msg^i(c, a_i', 3)\Big),$$

$$\bigwedge_{i=1}^{n}\Big(\bigwedge_{j=1}^{m_i} msg^i(b, a_i', \alpha_{ij})$$

$$\wedge \bigwedge_{k,l=1;\,k \neq l}^{m_i} (\neg msg(b, a_i', \alpha_{ik}) \vee \neg msg^i(b, a_i', \alpha_{il})$$

$$\wedge\, msg(c, a_i', 3) \wedge \neg msg(c, a_i', 1)$$

$$\wedge \neg msg^i(c, a_i', 2)\Big),$$

$$\bigwedge_{i=1}^{n}\Big(\bigwedge_{j=1}^{n}\Big(\bigwedge_{q^{(r)} \in \mathbf{P}_{a_i}^m} \forall Z_1, \ldots,$$

$$Z_r(\neg msg(a_i', a_j', q^{(r)}(Z_1, \ldots, Z_r))$$

$$\wedge\, msg^i(c, a_i', 1) \wedge \neg msg^i(c, a_i', 2)$$

$$\wedge \neg msg^i(c, a_i', 3)),$$

respectively.

The formula f_1 implies that "each agent a_i' received message 2 from c and PA does not contain any messages on actions, sent to the agent b" (i.e., all messages on actions from $Perm_{a_i}$ sent to the agents a_i are immediately forwarded to b).

The formula f_2 implies that "each agent a_i' received message 3 from c and exactly one message on a certain action α_{ij} from b" (this message belongs to the set $Perm_{a_i}$, which was sent by the agent a_i to the agent b at the previous step).

The formula f_3 implies that "each agent a_i' received message 1 from c and, for any i and j, PA does not contain any information messages sent by the agent a_i to the agent a_j" (i.e., all information messages sent by the agents a_i are immediately forwarded to their destinations).

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

Let us define the correspondence between the global states of $\mathcal{A}$ and $\mathcal{B}$. The global state

$$R = \langle (I_{a_1'}, MsgBox_{a_1'}), \ldots, (I_{a_n'}, MsgBox_{a_n'}),$$

$$(I_b, MsgBox_b), (I_c MsgBox_c), I_{PA}\rangle$$

of the system $\mathcal{B}$ is said to correspond to a global state

$$S = \langle (I_{a_1}, MsgBox_{a_1}), \ldots, (I_{a_n}, MsgBox_{a_n})\rangle$$

of the system $\mathcal{A}$ if and only if, for any i, $I_{a_i'} = I_{a_i}$ and $MsgBox_{a_i}$ is obtained from $MsgBox_{a_i'}$ after the removal of all messages received from the agents b and c.

This definition immediately implies that $R^{(0)}$ corresponds to $S^{(0)}$.

Define a mapping G of the set of vertices of $\mathcal{T}_{\mathcal{A}}(S^{(0)})$ to the set of vertices of $\mathcal{T}_{\mathcal{B}}(R^{(0)})$. Set $G(S^{(0)}) = R^{(0)}$. Suppose that, for a certain vertex S from $\mathcal{T}_{\mathcal{A}}(S^{(0)})$, $G(S) = R$ is already defined so that the message $msg(c, a_i', 1)$ is contained in $MsgBox_{a_i'}$. Let S' be the next vertex after S. Then $G(S') = R'$, where R' is obtained by the three steps of $\mathcal{B}$ that model one step of $S \Rightarrow_{\mathcal{A}} S'$, as described above.

Hence we find that, for every S from $\mathcal{T}_{\mathcal{A}}(S^{(0)})$, the state of $G(S)$ corresponds to the state of S.

Henceforth we will denote by $G(\mathcal{A})$ 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 $set_{\mathcal{B}}(\Psi) = \{R|\mathcal{T}_{\mathcal{B}}(R^{(0)}), R \models \Psi\}$.

Then, the assertion of the theorem follows immediately from the following lemma.

Lemma 5. *The equality* $set_{\mathcal{B}}(\Psi(\Phi, \mathcal{A})) \cap G(\mathcal{A}) = \{G(S)|S \in set_{\mathcal{A}}(\Phi)\}$ *holds for any formula* Φ *of the FO-L*μ *language.*

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

First, we prove the following proposition.

(#) *If the lemma is valid for formulas* Θ *and* Θ', *then it is also valid for the formulas* $\neg\Theta$, $\Theta \wedge \Theta'$, $\Theta \vee \Theta'$, $\exists X\Theta$, *and* $\forall X\Theta$.

For Boolean connectives, this proposition is obvious.

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

(*) *For any vertex S of the tree* $\mathcal{T}_{\mathcal{A}}(S^{(0)})$,

$$\mathcal{T}_{\mathcal{A}}(S^{(0)}), \quad S \models \exists X\Theta$$

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

Suppose that $\Phi = \exists X\Theta$ and $\mathcal{T}_{\mathcal{A}}(S^{(0)}), S \models \Phi$. Then, it follows from the definition of $\exists X$ that $\mathcal{T}_{\mathcal{A}}(S^{(0)}), S' \models \Theta$ for a vertex S' such that $S \Rightarrow_{\mathcal{A}} S'$. Then, by the induction hypothesis, we obtain $\mathcal{T}_{\mathcal{B}}(R^{(0)}), G(S') \models \Psi(\Theta, \mathcal{A})$. By the definition of Ψ, we have $\Psi(\Phi, \mathcal{A}) = \exists X(f_1 \wedge \exists X(f_2 \wedge \exists X(f_3 \wedge \Theta(\Phi, \mathcal{A}))))$. Note that the definition of G implies that there is 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 $\mathcal{T}_{\mathcal{B}}(R^{(0)}), R_i \models f_i$, $i = 1, 2, 3$. Hence we find 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 (i) $\mathcal{T}_{\mathcal{B}}(R^{(0)}), R_i \models f_i$, $i = 1, 2, 3$ and (ii) $\mathcal{T}_{\mathcal{B}}(R^{(0)}), R_3 \models \Psi(\Theta, \mathcal{A})$.

It follows from (i) that each agent a_i' in R_2 executes a certain action $\alpha_{ij} \in Perm_{a_i}(S)$. Then, $a_i \in \mathcal{A}$ can also execute the same action in the state S, because its selection operator $Sel_{a_i}^{un}$ can choose in $Perm_{a_i}(S)$ any action to execute. After that, the system $\mathcal{A}$ goes to a state S' such that $G(S') = R_3$. Then, by the induction hypothesis, $\mathcal{T}_{\mathcal{A}}(S^{(0)}), S' \models \Theta$ and, hence, $\mathcal{T}_{\mathcal{A}}(S^{(0)}), \models \exists X\Theta$. Thus, we have proved assertion (*).

For the formulas $\Phi = \forall X\Theta$, the proof follows from the equivalence of the formulas $\forall X\Theta$ and $\neg\exists X\neg\Theta$. Hence, proposition (#) is proved.

If Φ is a basic state formula, then the assertion of the lemma follows from the correspondence between S and $G(S)$. Thus, it follows from proposition (#) that the lemma holds for an arbitrary formula Φ from FO-Lμ_0 (i.e., for a formula that does not contain operators μ and ν).

If Φ has the form of $\mu Z.\Theta(Z)$ or $\nu Z.\Theta(Z)$, then the lemma is proved by a direct, but rather tedious, induction on calculation of the fixpoints of these formulas. Consider, as an example, the following simple case.

Suppose that Φ has the form of $\mu Z.\Theta(Z)$, where $\Theta(Z)$ does not contain μ and ν. Then, $\Psi(\Phi, \mathcal{A}) = \mu Z.\Psi(\Theta(Z), \mathcal{A})$. By the definition of μ, we have

$set_{\mathcal{A}}(\Phi) = \bigcup_{i=0}^{\infty} set_{\mathcal{A}}(\Theta^i(false))$ and $set_{\mathcal{B}}(\Psi(\Phi, \mathcal{A})) = \bigcup_{i=0}^{\infty} set_{\mathcal{B}}((\Psi(\Theta, \mathcal{A}))^i(false))$. Hence, $\Psi(\Theta^i(false), \mathcal{A}) = (\Psi(\Theta, \mathcal{A}))^i(false)$ and $set_{\mathcal{B}}((\Psi(\Theta, \mathcal{A}))^i(false))) \cap G(\mathcal{A}) = \{G(S) | S \in set_{\mathcal{A}}(\Theta^i(false))\}$. Then $set_{\mathcal{B}}(\Psi(\Phi, \mathcal{A})) \cap G(\mathcal{A}) = \{G(S) | S \in set_{\mathcal{A}}(\Phi)\}$.

For formulas Φ of the form $\nu Z.\Theta(Z)$, where $\Theta(Z)$ does not contain μ and ν, the proof is analogous. □

The following theorem shows how asynchronous MASs can be modeled by nondeterministic MASs.

Theorem 4. *Suppose that the FO-L*μ_r *language is used to formulate the dynamical properties of MASs. Then the MA-BEHAVIOR problem for asynchronous nondeterministic MASs is reduced in polynomial time to the MA-BEHAVIOR problem for (synchronous) nondeterministic MASs.*

Proof. Suppose that $\mathcal{A} = \{a_1, ..., a_n; PA\}$ is a nondeterministic asynchronous MAS and Φ is a formula to be verified. By $\mathcal{A}$, we construct a nondeterministic MAS $\mathcal{B} = \{a'_1, ..., a'_n, pa\}$ that models the operation of $\mathcal{A}$. A nondeterministic agent *pa* will model the operation of the mail agent *PA* by keeping messages in its database and then nondeterministically sending them to destinations.

For any predicate from the signature of messages $q \in \mathbf{P}^m_{a_i}$ and any $j \neq i$, we place a predicate q_{ij} into the signature $\mathbf{P}^m_{a'_i}$. The heads α of the propositions of the logical components of agents a'_i are the same as those in the programs of agents a_i, but each message of the form $msg(a_i, a_j, q)$ in the list $SEND_\alpha$ is replaced by $msg(a'_i, pa, q_{ij})$. In the bodies of propositions in the program P_{a_i}, each atom of the form $msg(a_j, a_i, q)$ is replaced by $msg(pa, a'_i, q_{ji})$. Moreover, a new atom $msg(pa, a'_i, 1)$ is added to the body of each proposition.

For every $q^{(k)}_{ij}$, the action basis AB_{pa} of a nondeterministic "mail" agent *pa* includes three actions $save(q_{ij})$, $resent(q_{ij})$, and $send(q_{ij})$ with the following lists:

$$ADD_{save(q_{ij})} = \{q_{ij}(X_1, \dots, X_k)\},$$
$$DEL_{save(q_{ij})} = SEND_{save(q_{ij})} = \emptyset,$$
$$ADD_{resend(q_{ij})} = DEL_{resend(q_{ij})} = \emptyset,$$
$$SEND_{resend(q_{ij})} = \{msg(pa, a_j, q_{ij}(X_1, \dots, X_k))\},$$
$$ADD_{send(q_{ij})} = \emptyset,$$
$$DEL_{send(q_{ij})} = \{q_{ij}(X_1, \dots, X_k)\},$$
$$SEND_{send(q_{ij})} = \{msg(pa, a_j, q_{ij}(X_1, \dots, X_k))\}.$$

AB_{pa} also includes two actions *add*_1 and *del*_1 for counting even and odd steps:

$$ADD_{add\_1} = \{1\}, \quad DEL_{add\_1} = \emptyset,$$
$$SEND_{add\_1} = \{msg(pa, a'_i, 1) | 1 \le i \le n\}.$$
$$ADD_{del\_1} = \emptyset, \quad DEL_{del\_1} = \{1\},$$
$$SEND_{del\_1} = \emptyset.$$

To start these actions, the following two propositions are included in the program P_{pa} of the agent *pa*:

$$add\_1 \longleftarrow \neg 1.$$
$$del\_1 \longleftarrow 1.$$

For each $q^{(k)}_{ij}$, the program P_{pa} contains three propositions:

$$save(q_{ij})(X_1, ..., X_k)$$
$$\longleftarrow msg(a_i, pa, q_{ij}(X_1, ..., X_k)).$$
$$resend(q_{ij})(X_1, ..., X_k)$$
$$\longleftarrow msg(a_i, pa, q_{ij}(X_1, ..., X_k)).$$
$$send(q_{ij})(X_1, ..., X_k) \longleftarrow q_{ij}(X_1, ..., X_k).$$

The selection operator Sel_{pa} chooses an arbitrary subset of actions of the form $send(q_{ij})(t_1, ..., t_k)$ in $Perm_{pa}$ and chooses one atom in each pair of action atoms of the form $\{save(q_{ij})(t_1, ..., t_k), resend(q_{ij})(t_1, ..., t_k)\} \subseteq Perm_{pa}$.

One can see from the construction of $\mathcal{B}$ that one step of system $\mathcal{A}$ is modeled by two steps of system $\mathcal{B}$.

The validity of the formula Φ in $\tau(\mathcal{A}, S^0)$ can be reduced to the validity of a certain formula Ψ of language FO-Lμ_r in $\tau(\mathcal{B}, S^0)$. The formula $\Psi = \Psi(\Phi, \mathcal{B})$ is constructed inductively, just as in the previous theorem, but a bit simpler.

According to the construction, this reduction can be performed in polynomial time.

These modeling theorems imply, in particular, the following propositions on the complexity of verification of asynchronous MASs.

Corollary 5. *The MA-BEHAVIOR problem for ground asynchronous MASs with deterministic agents*

(i) *is EXPTIME-hard for verifying formulas from FO-L*μ_r *for every fixed* $r \ge 1$ *and*

(ii) *belongs to the class EXPTIME coNEXPTIME for verifying formulas from FO-L*μ.

Corollary 6. *The MA-BEHAVIOR problem for nonground asynchronous MASs with deterministic agents*

(i) *is EXPTIME-hard for verifying formulas from FO-L*μ_r *for every fixed* $r \ge 1$ *and*

(ii) *belongs to the class EXPTIME coNEXPTIME for verifying formulas from FO-L*μ.

7. CONCLUSIONS

The MASs considered in this paper represent a class of general parallel and/or distributed software systems. Therefore, many well-known approaches to the verification of parallel programs can be applied to the analysis of the behavior of these systems. However, the architecture of MASs has a certain specific feature, which requires that these approaches should be significantly modified.

For MASs with their rich architecture, the adequacy of the analysis of behavior is closely related to the accurate choice of the specification level of the essential properties of this architecture and its parameters, as well as to the choice of natural constraints on these properties and parameters. We have chosen a special fragment of the IMPACT architecture [9] in which intelligent components of agents are described by logical programs. Depending on the operation conditions of the mail subsystem and on the one-step semantics implemented by agents, the MASs considered may be either synchronous or asynchronous, or deterministic or nondeterministic. Following known approaches to the verification of program models (see [22, 24]), we used different classes of temporal logics (linear and branching time) for representing the behavior properties of various versions of MASs.

For each of the above-mentioned classes of MASs, we have distinguished some natural subclasses, using structural constraints on the number of agents, the number of messages, and on the dimension (arity) of actions and messages. We have also considered some important semantic conditions that restrict the expressiveness of the programs and the results of agents' actions: the use of variables and negations in logical programs and the possibility of removing facts from the states of agents.

Our goal has been to determine the complexity of an appropriate MA-BEHAVIOR problem for various combinations of these constraints. For synchronous deterministic and nondeterministic MASs, the main results obtained in [1–3] are given here in Tables 1 and 2. In this paper, we have mainly focused on the behavior of asynchronous MASs. For this class, we could translate part of the results from the nondeterministic case using Theorems 3 and 4 on the mutual reducibility of nondeterministic and asynchronous MASs. However, the constructions of these theorems do not preserve the properties of MASs such as expansibility and positiveness. Therefore, to obtain "low" complexity bounds for appropriate subclasses of MASs, we needed separate constructions (Theorems 1 and 2).

The results obtained show that the complexity of the MA-BEHAVIOR problem for "unbounded" classes of MASs is very high; in particular, it takes exponential memory or time bounded by a double exponential function. At the same time, for some naturally bounded subclasses of MASs, the MA-BEHAVIOR problem has relatively low complexity: it is solved in deterministic or nondeterministic polynomial time or with polynomial memory.

Although the architecture of MASs considered in this paper is much simpler than the original IMPACT architecture defined in [9], our results extend to a more general case. In particular, the intelligent components of agents can be extended by using deontic modalities of types P (permitted), F (forbidden), O (obliged), and so on in the logical programs [9]. Similar assertions can be made for a number of other means from [9] that have not been considered here. However, as pointed out in the Introduction, even the description of these means is rather cumbersome. Some of our results on probabilistic MASs are contained in [31].

Naturally, the problem of the complexity of behavior is also of interest for other architectures of MASs. One of the methods for studying this problem is modeling MASs of different architectures by the systems of the IMPACT architecture. In particular, in [32] the authors discuss the possibility of modeling in this architecture the MASs defined in the AgentSpeak system [33], which is based on the so-called BDI (belief–desire–intention) agents. These agents work out plans of their actions on the basis of some formally defined sets of beliefs (facts of internal DB) and intentions. The verification problem for AgentSpeak systems was considered in the literature (see [13]); however, the complexity of appropriate algorithms was not estimated. The possibility of modeling these systems on systems of the type considered allows us to obtain upper bounds for the complexity of verification of AgentSpeak systems.

An essentially different type of agents is represented by the agents defined by epistemic logics [16] without using program components (in particular, logical components, as in the present case). These agents allow one to express knowledge based on incomplete information. By varying the accessibility of information, one can check how the views of an agent about the world and other agents are changed. Some results of verification of systems with such agents, in which the logic of knowledge is additionally combined with temporal logics, were obtained in [34, 35]. These logics allow one to describe the time evolution of agents' knowledge.

In conclusion, note that we have considered here "naive" versions of verification algorithms on models, leaving out various optimization methods such as symbolic verification, abstraction, the use of symmetry properties of MASs, etc. (see [24]) for further application.

As regards the practical development of verification systems for MASs, the results obtained here show that it is impossible to fully automatize the verification of general systems; therefore, such a system should allow one to trace the operation of MASs and apply special techniques to the verification of systems from subclasses of these MASs.

REFERENCES

1. Valiev, M.K., Dekhtyar, M.I., and Dikovsky, A.Ya., On the Complexity of Behavior of Systems of Interacting Agents, *Trudy konferentsii, posvyashchennoi 90-letiyu so dnya rozhdeniya A.A. Lyapunova* (Proc. of the Conf. Dedicated to the 90th Birthday of A.A. Lyapunov), Novosibirsk: Nauka, 2001, pp. 18–28.
2. Dekhtyar, M., Dikovsky, A., and Valiev, M., On Feasible Cases of Checking Multi-Agent Systems Behavior, *Theor. Comput. Sci.,* 2003, vol. 303, no. 1, pp. 63–81.
3. Dekhtyar, M.K., Dikovsky, A.Ja, and Valiev, M.K., On Complexity of Verification of Interacting Agents' Behavior, *Ann. Pure Appl. Logic*, 2006, vol. 141, no. 3, pp. 336–362.
4. Barringer, H., Fisher, M., Gabbay, D., Gough, G., and Owens, R., METATEM: An Introduction, in *Formal Aspects of Computing*, 1995, vol. 7, pp. 533–549.
5. Georgeff, M. and Lansky, A., Reactive Reasoning and Planning, *Proc. of the Conf. of the American Association of Artificial Intelligence*, Seattle, 1987, pp. 677–682.
6. van der Hoek, W. and Wooldridge, M., Multi-Agent Systems, in *Handbook of Knowledge Representation,* van Harmelen, F., Lifschitz, V., and Porter, B., Eds., Amsterdam: Elsevier, 2008, pp. 887–928.
7. Reiter, R., *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*, Cambridge: MIT, 2001.
8. Shoham, Y., Agent Oriented Programming, *Artif. Intell.,* 1993, no. 60, pp. 51–92.
9. Subrahmanian, V.S., Bonatti, P., Dix, J., et. al., *Heterogeneous Agent Systems*, Cambridge: MIT, 2000.
10. Tarasov, V.B., *Ot mnogoagentnykh system k intellektual'nym organizatsiyam* (From Multiagent Systems to Intellectual Organizations), Moscow: Editorial URSS, 2002.
11. Araragi, T., Attie, P., Keidar, I., Kogure, K., Luchangco, V., Lynch, N., and Mano, K., On Formal Modeling of Agent Computations, *NASA Workshop on Formal Approaches to Agent-Based Systems,* 2000.
12. Benerecetti, M., Guinchiglia, F., and Serafini, L., Model Checking Multiagent Systems, *Technical Report no. 9708-07*, Instituto Trentino di Cultura, 1998.
13. Bordini, R.H., Fisher, M., Pardavila, C., and Wooldridge, M., Model Checking AgentSpeak, *Proc. of the Second Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS-03)*, Melbourne, 2003.
14. Wooldridge, M. and Dunne, P.E., The Computational Complexity of Agent Verification, in *Intelligent Agents VIII*, Meyer, J.J. and Tambe, M., Eds., Lecture Notes in AI, vol. 2333, Berlin: Springer, 2002.
15. Wooldridge, M., Huget, M.P., Fisher, M., and Parsons, S., Model Checking Multi-Agent Systems: The MABLE Language and Its Applications, *Int. J. Artif. Intell. Tools*, 2006, vol. 5, no. 2, pp. 195–225. (Preliminary version: Wooldridge, M., Huget, M.P., Fisher, M., and Parsons, S., Model Checking Multiagent Systems with MABLE, *Proc. of the First Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS-02)*, Bologna, 2002.
16. Fagin, R., Halpern, J.Y., Moses, Y., and Vardy, M.Y., *Reasoning about Knowledge*, Cambridge: MIT, 1995.
17. Eiter, T. and Subrahmanian, V.S., Heterogeneous Active Agents. III: Polynomially Implementable Agents, *Tech. Rep. INFSYS RR-1843-99-07*, Inst. für Informationssysteme, Technische Universität Wien, A-10-40,Vienna, 1999.
18. Clarke, E.M. and Emerson, E.A., Design and Synthesis of Synchronization Skeletons Using Branching Time Temporal Logic, *Proc. of Workshop on Logics of Programs*, Lecture Notes in Computer Science, 1981, no. 181, pp. 52–71.
19. Vardi, M. and Wolper, P., An Automata-Theoretic Approach to Automatic Program Verification, *Proc. of the IEEE Symposium on Logic in Computer Science*, 1986, pp. 332–344.
20. Queille, J.P. and Sifakis, J., Specification and Verification of Concurrent Programs in CESAR, *Proc. of the 5th Int. Symposium on Programming,* Lecture Notes in Computer Science, 1982, no. 137, pp. 195–220.
21. Sistla, A.P. and Clarke, E.M., The Complexity of Propositional Linear Temporal Logic, *J. Assoc. Comput. Mach.*, 1985, vol. 32, no. 3, pp. 733–749.
22. Emerson, E.A., Temporal and Modal Logic, in *Handbook of Theoretical Computer Science*, Leeuwen, J., Ed., Amsterdam: Elsevier, 1990.
23. Emerson, E.A., Model Checking and the Mu-Calculus, in *Descriptive Complexity and Finite Models, Proc. of the DIMACS Workshop*, Immerman, N. and Kolaitis, P.H., Eds., Amsterdam: Elseiver, 1996, pp. 185–214.
24. Clarke, E.M., Grumberg, O., and Peled, D., *Model Checking*, Cambridge: MIT, 2000. Translated under the title *Verifikatsiya modelei programm: Model Checking*, Moscow: MTSNMO, 2002.
25. Apt, K.R., Logic Programming, in *Handbook of Theoretical Computer Science*, Vol. B: *Formal Models and Semantics, Chapter 10*, Leeuwen, J., Ed., Amsterdam: Elsevier, 1990, pp. 493–574.
26. Kozen, D., Results on the Propositional μ-Calculus, *Theor. Comput. Sci.*, 1983, vol. 27, pp. 333–354.
27. Manna, Z. and Pnueli, A., *The Temporal Logic of Reactive and Concurrent Systems: Specification*, Berlin: Springer, 1991.
28. Yi, K., Shilov, N.V., and Bodin E.V., Program Logics Made Simple, in *Sistemnaya informatika* (System Informatics), Novosibirsk, 2002, issue 8, pp. 206–249.
29. Garey, M. and Johnson, D.S., *Computers and Intractability—A Guide to the Theory of NP-Completeness,* New York: Freeman, 1979.
30. Papadimitriou, C.H., *Computational Complexity*, Reading: Addison-Wesley, 1994.
31. Dekhtyar, M.I., Dikovsky, A.Ja., and Valiev, M.K., Temporal Verification of Probabilistic Multi-Agent Systems, in *Pillars of Computer Science: Essays Dedicated to Boris (Boaz) Trakhtenbrot on the Occasion of His 85th Birthday*, Avron, A., Dershowitz, N., and

Rabinovich, A., Eds., Lecture Notes in Computer Science, Berlin: Springer, 2008, vol. 4800.

32. Burmistrov, M.Yu., Valiev, M.K., Dekhtyar, M.I., and Dikovsky, A.Ya., On the Verification of Dynamical Properties of Systems of Interacting Agents, *Trudy X natsional'noi konferentsii po iskusstvennomu intellektu s mezdunarodnym uchastiem* (Proc. of the X National Conf. on Artificial Intelligence with International Participation), Obninsk: Fizmatlit, 2006, pp. 908–915.

33. Rao, A.S., AgentSpeak (L): BDI Agents Speak out in a Logical Computable Language, in *Lecture Notes in AI*, Berlin: Springer, 1996, vol. 1038.

34. Garanina, N.O. and Shilov, N.V., Verification of Combinatorial Logics of Knowledge, Actions, and Time in Models, in *Sistemnaya informatika* (System Informatics), Novosibirsk: Sib. Otd. RAN, 2006, issue 10, pp. 114–173.

35. Shilov, N.V., Garanina, N.O., and Choe, K.-M., Update and Abstraction in Model Checking of Knowledge and Branching Time, *Fund. Inform.*, 2006, vol. 72, no. 1-3, pp. 347–361.

36. Emerson, E.A. and Lei, C.L., Efficient Model Checking in Fragments of the μ-Calculus, *Proc. of the IEEE Symposium on Logic in Computer Science*, 1986.