On Feasible Cases of Checking Multi-Agent Systems 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

The complexity of multi-agent systems behavior properties is studied. The behavior properties are formulated using classical temporal logic languages and are checked relative to the transition system induced by the multi-agent system definition. We show that there are deterministic or nondeterministic polynomial time check algorithms under some realistic structural and semantic restrictions on agent programs and actions.

Key words: Multi-Agent System, Temporal logics, Model checking, Complexity

Dedicated to the 60th anniversary of Anatol Slissenko

1 Introduction

Emerging in the late 80ies, the terms 'Intelligent Agent' (IA) and 'Multi-Agent System' (MA-system) refer to a new and generic metaphor of an artificial intelligence based computing technology. The range of IA applications extends from

[1] This work was sponsored by the Russian Fundamental Studies Foundation (Grants 01-01-00278 and 00-01-00254).

operating system interfaces, processing of satellite imaging data and WEB navigation to air traffic control, business process management and electronic commerce. This is the reason why there is no unified reading of the terms. We address the reader to the book [26] and several publications [29,15,18,19,23,22] discussing their different readings and definitions. Meanwhile, the intuitive appeal of the term 'Intelligent Agent' is quite clear :

- an IA is *autonomous,* i.e. it can function by itself in predetermined environments;

- it is *reactive,* i.e. it is capable of perceiving and responding to the stimuli of other agents or of its medium;

- it is *intelligent,* i.e. its actions are determined by a certain logic and estimation of its environment;

- and it is *goal oriented,* i.e. even if its functioning is continuous, it is oriented on reaching states with some predetermined properties.

Concrete realizations of these properties determine particular agent architectures. Agent's intelligence capacity can vary from finite state control structures or IF-THEN rules to logic programs, non monotone belief based systems or deontic logics (see [26] for a discussion and references).

In this paper, we study the complexity of recognizing properties of behavior of MA-systems. The research of this kind is rather scarce (besides the cited book [26], see [9,10,14,28]). The reason for this lacuna is the difference in orientations of MA-system architecture definitions and of logical or complexity analysis of behavior properties. The former are oriented to higher expressivity and adequacy relative to applications. The latter, on the contrary, needs abstraction from details and substantial simplifications of analyzed models. So one of the problems is that of finding an adequate abstraction level. ¿From many known agent architectures, we have chosen the so called IMPACT architecture described in detail in the book [26]. This very elaborated architecture is neatly formalized in terms of state transition systems and carefully studied. In particular, the complexity bounds of several aspects of decision making are established in [26] (chapter 11), which partially characterize the complexity of one-step state transition. This complexity being rather high (from NP and FNP to Π_3^P-complete), we simplify this architecture leaving only the agent features concerning actions, decision policies and communication. We express behavior properties of deterministic and nondeterministic MA-systems as the properties of trajectories (i.e. finite or infinite paths) in the state transition systems they define. This allows the use of classical temporal logics as behavior properties description languages. The *"MA-BEHAVIOR"* problem we consider in this paper, consists in verifying that a temporal logic formula Φ holds on the tree of trajectories of a given MA-system. So it is a model checking type problem. Model checking has been extensively studied since the early 80ies (see [6,20,16,24,27,25,8,11,12,7]). There is however a substantial difference between the classical problem statement and that of this paper. Traditionally, the complexity results are established for explicitly presented transition diagrams or else for some their fixed representation (by a finite automata, by OBDD).

Meanwhile, we establish the complexity bounds with respect to MA-systems whose operational semantics is presented in the form of transition systems. MA-systems constitute a compact representation of the corresponding transition system. For example, even for a ground (i.e. variable-free) MA-system A, the transition system $T(A)$ describing its trajectories may have the size exponential in $|A|$, because it can occur that it has $\boldsymbol{O}(2^{|A|})$ states. So the lower bounds may be (and they are) more pessimistic as compared with the classical ones for the same classes of logics. This being so, we nevertheless establish in this paper interesting classes of MA-systems, in which the MA-BEHAVIOR problem turns out to be decidable in deterministic or nondeterministic polynomial time. And this is due to a new possibility of formulating natural constraints in terms of structural parameters of MA-systems.

2 Agent and MA-system architecture

An *intelligent agent (IA)* a, as it shows up in this paper, has its *internal state*, which is a finite set of ground atoms in the signature $\mathbf{P}^e_a$, communicates other agents through messages, which are ground message atoms in the signature $\mathbf{P}^m_a$, held in its message box, is capable of performing a number of parameterized actions in the signature $\mathbf{P}^{act}_a$, constituting its action base AB_a, is controlled by a program P_a, whose semantics determines the set of executable actions and uses its one-step semantics Act_a in order to select the actions to execute.

A **multi-agent system** (**MA-system**) $\mathcal{A} = \{a_1, ..., a_n\}$ serves as a common frame for interacting IA $a_1, ..., a_n$. It fixes some finite extensional signature $\mathbf{P}^e$, message signature $\mathbf{P}^m$, and intensional actions signature $\mathbf{P}^{act}$ such that $\mathbf{P}^e_a \subseteq \mathbf{P}^e$, $\mathbf{P}^m_a \subseteq \mathbf{P}^m$, $\mathbf{P}^{act}_a \subseteq \mathbf{P}^{act}$ for $a \in \mathcal{A}$.

We adopt a domain closure assumption fixing some finite set of constants $\mathbf{C}$ denoting the domain objects and considering a set $\mathbf{\Pi}$ of polynomial time computable built-in predicates and operations (e.g., the standard arithmetical operations over numbers).

Hereafter, by $\mathbf{A}^e, \mathbf{A}^m$, $\mathbf{A}^{act}$ and $\mathbf{L}^e, \mathbf{L}^m$, $\mathbf{L}^{act}$ we denote the sets of atoms and literals in the corresponding predicate signatures, using constants in $\mathbf{C}$ and variables in some countable set $\mathbf{V}$. By $\mathbf{B}^e, \mathbf{B}^m$, and $\mathbf{B}^{act}$ we denote the corresponding sets of ground atoms, and by $\mathbf{LB}^e, \mathbf{LB}^m$, and $\mathbf{LB}^{act}$ we denote the corresponding sets of ground literals.

The message box of an agent a, denoted $MsgBox_a$ can hold messages received from other agents, i.e. pairs of the form $(Sender\_agent, Message)$, where $Message$ is a ground atom in the signature $\mathbf{P}^m_a$. We call *local states* the pairs $IM_a = (I_a, MsgBox_a)$ consisting of the current individual state I_a and the current message box contents $MsgBox_a$.

Action base AB_a is a finite set of *actions* specified by expressions

$$(\alpha(X_1, ..., X_l), ADD(\alpha), DEL(\alpha), SEND(\alpha)),$$

in which the atom $\alpha(X_1, ..., X_l)$ (uniquely) determines the action's name and parameter list, $ADD(\alpha)$ and $DEL(\alpha)$ are lists of atoms to add to (respectively, remove from) the state, and $SEND(\alpha)$ is a set of messages to send to indicated agents. Atoms in $ADD(\alpha), DEL(\alpha)$ and messages in $SEND(\alpha)$) may share parameters $X_1, ..., X_l$. So each ground substitution τ binding the parameters fixes the corresponding action instance $\tau(\alpha)$. An action α is *expanding* if $DEL(\alpha) = \emptyset$. Agent a is *expanding* if it has only expanding actions.

Program P_a defines the agent's action policy. It is a logic program with the clauses of the form $H \leftarrow L_1, ..., L_n$, where $n \geq 0$, the head $H = \alpha(t_1, ..., t_l)$ is an (intensional) action atom in $\mathbf{A}^{act}$ such that $(\alpha(X_1, ..., X_l),\ ADD(\alpha),\ DEL(\alpha),\ SEND(\alpha)) \in AB_a$, literals L_i in its body are either action literals over $\mathbf{L}^{act}$, or (extensional) state literals in $\mathbf{LB}^e$, or message literals of the form $Received(Source\_agent, Message)$, or their negations $\neg Received(Source\_agent, Message)$ with $Message \in \mathbf{A}^m$, or atoms $q(\bar{t})$ with built-in predicates $q \in \mathbf{\Pi}$. An agent's program is *positive* if there are no negations in it's clauses. An agent with positive program is also called *positive*.

We suppose that the clauses are *safe* in the sense that all variables in the head H occur *positively* in the body $L_1, ..., L_n$, and that the program $P_a^{state} = P_a \cup \{p \leftarrow \ \mid p \in I_a\} \cup \{Received(Agent\_source, Message) \leftarrow \ \mid (Agent\_source, Message) \in MsgBox_a\}$ is *stratified* [2].

Program semantics determines the set of actions which in principle can be executed by the agent in its current local state. As it is well known (see [2]), stratified logic programs have a unique minimal model M_a^{state} computed by a standard polynomial time fixpoint computation procedure from the *grounding* $gr(P_a^{state})$ of the program P_a^{state} [2].

The semantics $Sem(P_a)(I_a, MsgBox_a)$ of P_a with respect to a local state $(I_a, MsgBox_a)$ is defined as $M_a^{act} = M_a^{state} \cap \mathbf{B}^{act}$. In other words, the semantics is the set of ground actions implied by the program P_a^{state}.

Agent's one-step semantics. Given the set $M = M_a^{act}$ of the actions available for execution, the role of an agent's one-step semantics Act_a is to choose (or guess) a set $Act_a(M) \subseteq M$ of the actions to execute. It is natural to suppose that a greater set of available actions leads to a greater set of chosen actions. So we assume the monotonicity of one-step semantics: $Act_a(M) \subseteq Act_a(M')$ for $M \subseteq M'$. We distinguish *deterministic* and *nondeterministic* semantics.

Deterministic one-step semantics is a function in the class $STEP^D = \{Act : M \rightarrow 2^M \mid Act(M)$ is computable in polynomial time$\}$. For instance, the *total deterministic* semantics defined by $Act^{td}(M) = M$ belongs to this

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

class. This semantics selects the whole M. We can also imagine other types of deterministic one-step semantics, e.g. priority driven deterministic semantics which presumes some partial order $\prec$ on the actions in $\mathbf{B}^{act}$ and is defined by $Act^{\prec d}(M) = \{m \in M \mid \neg\exists m' \in M\ (m' \prec m)\}$. *Deterministic agents* are those having a deterministic one-step semantics in $STEP^D$.

Nondeterministic one-step semantics is a relation Act in the class $STEP^N$ $= \{Act \subseteq M \times 2^M \mid Act$ is recognizable in polynomial time$\}$. The simplest nondeterministic one-step semantics in this class is the *unit choice* one-step semantics defined by $Act^{un}(M) = \{\{p\} \mid p \in M\}$. It guesses some available action in M. Another example is the *spontaneous* one-step semantics defined by $Act^{sn}(M) = \{M' \mid M' \subseteq M\}$. It guesses any subset of available actions in M. *Nondeterministic agents* are those having a nondeterministic one-step semantics in $STEP^N$.

Concurrent execution of actions. Let a set $AS = \{\sigma_1(\alpha_1), ..., \sigma_k(\alpha_k)\}$ $\subseteq \mathbf{B}^{act}$ of ground actions to execute be selected, each α_j being an action defined by an expression $(\alpha_j(X_1, ..., X_l), ADD(\alpha_j),\ DEL(\alpha_j), SEND(\alpha_j)) \in AB_a$ and σ_j being some ground substitution $(1 \leq j \leq k)$. Then AS defines the following concurrent local state change operator $\bigotimes\limits_a AS$.

The new internal state of a is defined by:

$$\bigotimes_a AS(I_a) = ((I_a \setminus \bigcup_{j=1}^{k} \sigma_j(DEL(\alpha_j))) \cup \bigcup_{j=1}^{k} \sigma_j(ADD(\alpha_j)))\ ^{3}.$$

The new message box states are defined for agents $b \neq a$ by:

$$\bigotimes_a AS(MsgBox_b) = MsgBox_b \cup \bigcup_{j=1}^{k} \{(a, \sigma_j(Msg)) \mid (b, Msg) \in SEND(\alpha_j)\}.$$

$\bigotimes\limits_a AS$ it computable in time polynomial with respect to $|AB_a| + |AS| + |I_a|$.

A global state of the MA-system $\mathcal{A}$ is defined as an n-tuple of local states of the agents $a_1, ..., a_n$, i.e. $S =< (I_{a_1}, MsgBox_{a_1}), \ldots, (I_{a_n}, MsgBox_{a_n}) >$. The set of all global states of $\mathcal{A}$ is finite and is denoted by $\mathcal{S}_{\mathcal{A}}$.

MA-system one step semantics. We have defined one step of each individual agent in $\mathcal{A}$. The one step semantics of the whole system $\mathcal{A}$ will define the one step transition $\Rightarrow_{\mathcal{A}}$ relation on $\mathcal{S}_{\mathcal{A}}$ as parallel execution of individual agents one step actions and message sending. Let $S =< (I_{a_1}, MsgBox_{a_1}),$ $..., (I_{a_n}, MsgBox_{a_n}) >$ and $S' = < (I'_{a_1}, MsgBox'_{a_1}), ..., (I'_{a_n}, MsgBox'_{a_n}) >$ be some global states in $\mathcal{S}_{\mathcal{A}}$. Then $S \Rightarrow_{\mathcal{A}} S'$ if S is transformed into S' as follows: for each $a_i \in \mathcal{A}$ its semantics $M_i^{act} = Sem(P_{a_i})(I_{a_i}, MsgBox_{a_i})$ is calculated and the agent's one-step semantics $AS_i = Act_{a_i}(M_{a_i}^{act})$ creates the action set to be executed concurrently. Then the message boxes of all agents in $\mathcal{A}$ are emptied (so the messages in S are forgotten). On the next stage for each $a_i \in \mathcal{A}$, in natural order, the actions AS_i are executed concurrently producing $I'_{a_i} = \bigotimes\limits_{a_i} AS_i(I_{a_i})$ and putting for each $j \neq i$ the corresponding messages of a_i

[3] So in the case where the same fact should be added and deleted, it will be added. Of course, other strategies of resolving such conflicts can also be used, e.g. the one, where adding and deleting annihilate each other.

into the message boxes of a_j: $MsgBox'_{a_j} := \bigotimes_{a_i} AS_i(MsgBox_{a_j})$.

Classes of MA-systems. We distinguish two main classes of MA-systems: deterministic and nondeterministic. A MA-system $\mathcal{A}$ is *deterministic* if all its agents are deterministic, otherwise it is *nondeterministic*.

In both classes of MA-systems, we consider the following subclasses induced by natural constraints imposed on agents' components. A MA-system $\mathcal{A} = \{a_1, ..., a_n\}$ is

- *ground* if each program P_{a_i} is ground [4];
- *k-dimensional* if the arities of the action predicates in $\mathbf{P}^{act}$ and of the message predicates in $\mathbf{P}^m$ are bounded by k (*dimension-bounded*, if *k-dimensional* for some k). In fact, this property fixes the maximal number of parameters involved in the actions and in the messages of $\mathcal{A}$;
- *expanding* if all its agents are *expanding*;
- *positive* if all its agents are *positive*;
- *m-agent* if $n \leq m$.
- *r-signal* if $\mathbf{P}^m$ consists of no more than r primitive symbols (*signals*).

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

Proposition 1
(1) For each deterministic MA-system $\mathcal{A}$, the transition function $S \Rightarrow_{\mathcal{A}} S'$ is computable in polynomial time with respect to $|S| + |\mathcal{A}| + |S'|$ if $\mathcal{A}$ is ground or dimension bounded, and is computable in deterministic exponential time in the general case.
(2) For each nondeterministic MA-system $\mathcal{A}$, the transition relation $S \Rightarrow_{\mathcal{A}} S'$ is recognizable in nondeterministic polynomial time with respect to $|S| + |\mathcal{A}| + |S'|$ if $\mathcal{A}$ is ground or dimension bounded, and is recognizable in nondeterministic exponential time in the general case.

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

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

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

Example 1 *"Resource-allocation"*
A resource allocation system consists of a manager-agent m owing some resource,

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

which it distributes on orders among four user-agents u_1, u_2, u_3, u_4. Each user has its own strategy of 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 next moment after u_1 has ordered;
3) u_3 orders the next moment after u_1 has received the resource from m;
4) u_4 orders every time.
The manager m maintains the list of orders and fulfills the first order on the list, one order at a time. Only one order of each user-agent can be held in the list. So if m receives an order from some user before the previous order of this user has been fulfilled, then the new order is discarded.

We implement this specification in the form of the following MA-system [5] $\mathcal{RA}$.

The agents of $\mathcal{RA}$ are defined as follows. The states I_{u_1} of u_1 can contain the fact put_order. The states I_{u_i} $(i = 2, 3, 4)$ are always empty. The states I_m of m include the facts of the form $order(X, I)$ ($order(u_i, j)$ means that the order of agent u_i is kept in the position j in the order list of m), $actual(X)$ (an order of agent X stands on the list of m), $num\_orders(I)$ (I is the number of unfulfilled orders). 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, he sends to u_i the message ok. u_1 sends to u_3 the message ok in order to inform him about the receipt of a resource.

Agent u_1.
Actions: $put: \ ADD = \{put\_order\}, SEND = \{(m, order), (u_2, order)\}$;
$receive: \ DEL = \{put\_order\}, SEND = \{(u_3, ok)\}$;
P_{u_1} : $put \leftarrow \neg put\_order$
$receive \leftarrow Received(m, ok)$
Agent u_2.
Actions: $put: \ SEND = \{(m, order)\}$;
P_{u_2} : $put \leftarrow Received(u_1, order)$
Agent u_3.
Actions: $put: \ SEND = \{(m, order)\}$;
P_{u_3} : $put \leftarrow Received(u_1, ok)$
Agent u_4.
Actions: $put: \ SEND = \{(m, order)\}$;
P_{u_4} : $put \leftarrow .$
Agent m.
Actions:
$place\_order(X, I): \ ADD = \{order(X, I), actual(X)\}$;
$fulfill\_order(X): \ DEL = \{order(X, 1), actual(X)\}, SEND = \{(X, ok)\}$;
$shift(X, I): \ ADD = \{order(X, I)\}, DEL = \{order(X, I + 1)\}$;
$new\_num(I, J) : ADD = \{num\_orders(J)\}, DEL = \{num\_orders(I)\}$
P_m :
$new\_order(X) \leftarrow Received(X, order), \neg actual(X)$

[5] Strictly speaking, this MA-system definition does not fit in the constraints above because the program *place_order* is not stratified. However, it can be easily transformed into an equivalent stratified program. We don't do it because the resulting program is greater and less clear.

$first\_free(I) \leftarrow num\_orders(I), I > 0$
$first\_free(1) \leftarrow num\_orders(0)$
$place\_order(X, I) \leftarrow new\_order(X), first\_free(I) \quad (X \in \{u_1, u_2\})$
$place\_order(u_3, I) \leftarrow new\_order(u_3), place\_order(X, I-1) \quad (X \in \{u_1, u_2\})$
$place\_order(u_3, I) \leftarrow new\_order(u_3), first\_free(I), \neg place\_order(u_1, I),$
$\neg place\_order(u_2, I)$
$place\_order(u_4, I) \leftarrow new\_order(u_4), place\_order(u_3, I-1)$
$place\_order(u_4, I) \leftarrow new\_order(u_4), \neg place\_order(u_3, I-1),$
$place\_order(X, I-1) \quad (X \in \{u_1, u_2\})$
$place\_order(u_4, I) \leftarrow new\_order(u_4), first\_free(I), \neg place\_order(u_1, I),$
$\neg place\_order(u_2, I), \neg place\_order(u_3, I-1)$
$fulfill\_order(X) \leftarrow order(X, 1)$
$shift(X, 1) \leftarrow fulfill\_order(Y), order(X, 2)$
$shift(X, I) \leftarrow shift(Y, I-1), order(X, I+1)$
$new\_num(I, J) \leftarrow num\_orders(I), num\_new\_orders(K), J = I + K$

The initial state of m consists of the fact $num\_orders(0)$. Then the fact $new\_order(X)$ indicates whether a new order of agent X should be placed in the list, the fact $first\_free(I)$ defines the position I in the list, where a new order should be placed, $place\_order$ places new orders at the end of the list in the predefined order $u_1 < u_2 < u_3 < u_4$, $fulfill\_order$ sends a resource to the first agent in the list, and $shift$ shifts the elements of the list one position to the left, $new\_num(I, J)$ changes the old value of $num\_orders(I)$ by adding the number K of new orders unregistered in $MessageBox_m$ before the step. K is computed by the predicate num_new_orders not defined here.

3 Logics for MA-System Behavior Properties

We follow the tradition of using temporal logic languages of discrete time for expressing the properties of MA-system trajectories. In this paper, we use first order extensions of PTL [11] whith the first order sentences on states (called *basic state formulas*) in the place of propositional letters [6]. We call $FLTL$ the following minimal first order extension of PTL using the standard linear time operators $\mathbf{X}$ ("nexttime") and $\mathbf{U}$ ("until") :
(p1) Each basic state formula ϕ is a $FLTL$ formula.
(p2) If ϕ and ψ are formulas, then $\neg\phi$, $\phi \wedge \psi$ and $\phi \vee \psi$ are formulas.
(p3) If ψ_1 and ψ_2 are formulas, then $\mathbf{X}(\psi_1)$ and $\psi_1 \mathbf{U} \psi_2$ are formulas. □

The validity of a $FLTL$ formula ϕ on a trajectory $\tau = S_1, S_2, ...$ in the trajectory tree $\mathcal{T} = \mathcal{T}_{\mathcal{A}}(S_0)$ of a MA-system $\mathcal{A}$ is defined as follows. Let $S_1 =< (I_{a_1}, MsgBox_{a_1}), ..., (I_{a_n}, MsgBox_{a_n}) >$.
1. For a basic state formula ϕ,

[6] So these sentences do not depend on agents' message boxes. This constraint does not lead to the loss of generality (having in mind the possibility for each agent to copy its messages into its internal state).

$$\tau \models \phi \text{ iff } \bigcup_{i=1}^{n} I_{a_i} \models_{FO} \phi$$

($\models_{FO}$ corresponds to the standard first order validity).
2. $\tau \models \neg\phi$ iff $\tau \not\models \phi$.
3. $\tau \models \phi_1 \wedge \phi_2$ iff $\tau \models \phi_1$ and $\tau \models \phi_2$.
4. $\tau \models \phi_1 \vee \phi_2$ iff $\tau \models \phi_1$ or $\tau \models \phi_2$.
5. $\tau \models \mathbf{X}\ \psi$ iff $\tau^2 \models \psi$ (where τ^k denotes the suffix $S_k, S_{k+1}, ...,$).
6. $\tau \models \psi_1 \mathbf{U} \psi_2$ iff there exists $k > 0$ such that $\tau^k \models \psi_2$ and $\tau^j \models \psi_1$ for all $1 \leq j < k$. □

We may also use in $FLTL$ several other linear time operators easily expressible throuhg the standard ones: $\mathbf{F}$ ("sometimes"): $\mathbf{F}\phi =$ **true** $\mathbf{U}\phi$, and its dual $\mathbf{G}$ ("always"): $\mathbf{G}\phi = \neg(\mathbf{F}\neg\phi)$, and $\mathbf{V}$ ("unless"): $\phi_1 \mathbf{V} \phi_2 = \neg(\neg\phi_1 \mathbf{U} \neg\phi_2)$.

In this paper, $FLTL$ is used in the case of deterministic $\mathcal{A}$, i.e. the case where for any starting global state S_0, the trajectory tree $\mathcal{T} = \mathcal{T}_{\mathcal{A}}(S_0)$ has the single trajectory $\mathcal{T} = S_0, S_1, ...$ So the validity of a $FLTL$ formula ϕ on this tree $\mathcal{T}_{\mathcal{A}}(S_0) \models \phi$ is defined as $\mathcal{T} \models \phi$.

In the case of nondeterministic MA-systems, we consider two simple extensions of $FLTL$ by branching quantifiers: $\exists LTL$ and $\forall LTL$ consisting respectively of all formulas of the form $\mathbf{E}(\phi)$ and $\mathbf{A}(\phi)$, where $\phi \in FLTL$. For example, the validity of a formula $\mathbf{E}(\phi) \in \exists LTL$ on the tree $\mathcal{T}_{\mathcal{A}}(S_0)$ is defined as: $\mathcal{T}_{\mathcal{A}}(S_0) \models \mathbf{E}(\phi)$ iff $\tau \models \phi$ for some trajectory τ in $\mathcal{T}_{\mathcal{A}}(S_0)$ starting in S_0.

In the case where the basic state first order formulas are quantifier (and object variable) free, we do not distinguish $FLTL$ from its propositional counterpart LTL because their model checking and validity problems have the same complexity modulo polynomial time.

For example, for a state formula Ψ, the formula $\mathbf{G}\ \Psi$ expresses the classical *safety* property, and $\mathbf{F}\ \Psi$ expresses the so called *accessibility*.

Example 2 *(example 1 continued)*
For the MA-system $\mathcal{RA}$ above, one may check that the following formulas are valid on the trajectory generated by $\mathcal{RA}$:

$$\mathbf{G}\ \mathbf{F}\ \mathit{Received}(m, u_i, ok)$$

(every agent receives the resource infinitely often),

$$\mathbf{G}\ \forall I \forall X \forall Y (\mathit{order}(X, I)\ \wedge\ \mathit{order}(Y, I) \rightarrow X = Y)$$

(at each moment, only one order can be placed in any position of the list), whereas the following formulas are not valid on this trajectory:

$$\mathbf{F}\ (\mathit{Received}(m, u_1, ok)\ \wedge\ \mathbf{X}\ \mathit{Received}(m, u_1, ok))$$

(there are two consecutive moments when u_1 receives a resource) and

$$\mathbf{G}\ (\mathit{order}(u_i, 2) \rightarrow \mathbf{X}\ \mathbf{X}\ \neg \mathit{Received}(m, u_i, ok)).$$

4 Behavior of deterministic MA-systems

The *"MA-BEHAVIOR"* problem we consider in this paper applies to deterministic MA-systems as well as to nondeterministic ones. Given such a system $\mathcal{A}$, an initial global state S_0 and a formula Φ expressing a property

of trajectories, the MA-BEHAVIOR problem $\mathcal{A}, S_0, \Phi$ has a positive solution if Φ holds on the tree $\mathcal{T}_{\mathcal{A}}(S_0)$ of trajectories of $\mathcal{A}$ starting in S_0 (denoted $\mathcal{T}_{\mathcal{A}}(S_0), S_0 \models \Phi$). We see that it is of the kind of model checking, though applied to MA-systems in the role of transition systems specification. We consider some instances of the MA-BEHAVIOR problem under restrictions imposed on semantics (deterministic, nondeterministic), on agent programs (e.g. groundness restriction), on action bases (e.g. deletion absence), or signatures (e.g. m-agent or k-dimensional). We first consider general deterministic MA-systems.

4.1 A check algorithm for deterministic MA-systems

The set of global states of any MS-system $\mathcal{A}$ is finite. So when it 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 of checking a $FLTL$-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 looks-up the structure portionwise. It allows to obtain essentially better complexity upper bounds for the MA-BEHAVIOR problem.

For a periodic trajectory $\tau = S^0, S^1, ..., S^t, ...$, let k and N be the least 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)*, for any time point t and a shift i, returns such time point $j < k + N$ that $S^j = S^{t+i}$:

move(t,i)= IF $t + i < k + N$ THEN $t + i$ ELSE $(t + i - k)$ **mod** $N + k$.

The second function F^τ serves as the oracle, which returns the state $F^\tau(t) = S^t$ of trajectory τ at any time point t. The third is the boolean-valued function FO_Check(S, Φ), which given a global state S and a closed first-order formula Φ, 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. The following recursive algorithm checks the property $\tau^t \models \Phi$.

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),
(18) ELSE $R := N$ END_IF
(19) FOR $i = 0$ TO $R - 1$ DO
(20) $r :=$ move(t, i); $p := i$;
(21) IF DetCheck(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

(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 $R := k + N - t$
(27) $b :=$ *TRUE*;
(28) FOR $j = 0$ TO $p - 1$ DO
(29) $r' :=$ move(t, j);
(30) IF $\neg$DetCheck(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

Let $s_{max}(\tau) = \max\{|S^t| \mid 0 \leq t \leq k + N\}$, $s(F^\tau)$ and $t(F^\tau)$ be the maximal space and time required for computing $F^\tau(t)$ for $0 \leq t \leq k + N$ and $s_{FO}(\tau, n)$ and $t_{FO}(\tau, n)$ be 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.

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

Proof. It is easy to see that the first four cases of the algorithm (lines 4, 7, 11, 13) follow directly the definition of semantics of $FLTL$-formulas. In the case $\Phi = \Phi_1 \mathbf{U} \Phi_2$ (line 16) an integer R is defined (lines 17, 18) such that $S^{t+i} \in \tau' = \{S^t, S^{t+1}, \ldots, S^{R-1}\}$ for any $i \geq 0$, and $\tau^t \models \Phi \Leftrightarrow \tau'^t \models \Phi$. Then the loop in lines 19-23 searches for a minimal $i \geq 0$ such that $\tau^{t+i} \models \Phi_2$. If such i does not exist then *TRUE* is returned (line 25). Otherwise, the loop in lines 28-32 searches for an integer $j < i$ such that $\tau^{t+j} \not\models \Phi_1$. If such j is found, then $\tau^t \not\models \Phi$ and *FALSE* is returned (line 31). Otherwise the algorithm returns *TRUE* (line 33).

Let us evaluate the complexity of DetCheck with input parameters k, N, Φ and t. The trajectory τ is represented implicitly by the oracle F^τ. Each recursive call of DetCheck applies to some subformula Φ' of Φ and to some time point $t' < k + N$. For every such pair (Φ', t'), the corresponding call of DetCheck is effected at most once (we consider two different occurrences of the same subformula as two different subformulas). Hence, the number of recursive calls of DetCheck does not exceed $|\Phi|(k + N)$. The case of basic state formula Φ' (lines 3-6) takes time $pol_1(t_{max}(F^\tau) + t_{FO}(\tau, |\Phi'|))$ for some polynomial pol_1. So the total time in this case does not exceed $pol_1(|\Phi|(k + N)t_{max}(F^\tau) + t_{FO}(\tau, |\Phi'|))$. Since the number of operators in all other cases (lines 7-34) is linearly bounded by the number of recursive calls of DetCheck in lines 8,9,12,15,21 and 30, the total time required for these operators does not exceed $pol_2(|\Phi|(k + N))$ for some polynomial pol_2. Therefore, DetCheck takes time bounded by $pol(|t| + |\Phi|(k + N)(t(F^\tau) + t_{FO}(\tau, |\Phi|))$ for some polynomial *pol*.

In order to evaluate the space needed to execute *DetCheck*, we should

consider some implementation details. At each given moment of the computation, the needed space is the space taken by the call stack, whose call frames keep input parameters and local variable values of DetCheck recursive calls invoked and not finished by the moment. Since the input parameters k and N never change, they should not be doubled while recursive calls of DetCheck. Now let $(\Phi_1, t_1), \ldots, (\Phi_i, t_i), \ldots, (\Phi_m, t_m)$ be the sequence of call frames in the call stack. Only the top subformula Φ_m can be a basic state formula. If this is the case, this call can be executed in space $\boldsymbol{O}(s_{max}(\tau) + s(F^\tau) + s_{FO}(\tau, |\Phi|))$. Any other call (Φ_i, t_i) in the stack being recursive, its frame keeps several boolean variables and integer variables with the values in $[0, k+N)$ (e.g., when $\Phi_i = \Phi_i^1 \mathbf{U} \Phi_i^2$, the frame keeps integer variables t, r, R, r', i, j, the boolean variable b and a fixed number of auxiliary variables). Hence, the size of one frame does not exceed $c \log(k+N) + |\Phi_i|$ for some constant c. It is not difficult to implement the computation of DetCheck in such way that the total size of all frames on the stack $|\Phi_1| + \ldots + |\Phi_m|$ does not exceed $|\Phi|$. In order to bound the call stack depth m, let us notice that in the cases of unary operators: $\Phi_i = \neg\Phi'$ or $\Phi_i = \mathbf{X}\Phi'$, our algorithm is tail recursive, so they can appear at the last place only, i.e. when $i = m$. As it concerns the binary operators Φ_i, $i > 1$, we remark that $d^*(\Phi_i) > max\{d^*(\Phi_{i-1}), d^*(\Phi_{i-2})\}$. Therefore, $m \leq 2d^*(\Phi)$ and the total size of the call stack is $\boldsymbol{O}(|t| + |\Phi| + d^*(\Phi) \log(k+N) + s_{max}(\tau) + s(F^\tau) + s_{FO}(\tau, |\Phi|))$. $\square$

Proposition 1 shows that the oracle F^τ in the lemma can be efficiently computed along the trajectories τ generated by MA-systems:

Lemma 2 *There is a polynomial pol and an algorithm, which for a MS-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 $pol(|\mathcal{A}| + \max\{|S^r| \mid 0 \leq r \leq t\})$.*

The next assertion provides upper bounds on the parameters of the periodic trajectories of deterministic MA-systems.

Lemma 3 *The trajectory $\tau(\mathcal{A}, S^0)$ of a deterministic MA-system $\mathcal{A}$ in initial state 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, $k(\mathcal{A}, S^0) + N(\mathcal{A}, S^0) \leq 2^{2^{pol(|\mathcal{A}| + |S^0|)}}$.*

Proof. Since $\mathcal{A}$ is deterministic and the set $GS_\mathcal{A}$ of its global states is finite, the trajectory $\tau(\mathcal{A}, S^0)$ is periodic. The sum of its parameters $k(\mathcal{A}, S^0) + N(\mathcal{A}, S^0)$ is bounded by the size of $GS_\mathcal{A}$. When $\mathcal{A}$ is ground, the atoms in global states are those found in $\mathcal{A} \cup S^0$. If $M = |\mathcal{A}| + |S^0|$ and n is the number of agents in $\mathcal{A}$, then $|GS_\mathcal{A}| \leq 2^{Mn}$.

In the general case, the number of ground atoms in global states of $\mathcal{A}$ is bounded by $a = 2^{pol(M)}$ for some polynomial pol. So $|GS_\mathcal{A}| \leq 2^{an}$. $\square$

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

Proposition 2 *Let a MA-system $\mathcal{A}$ and an initial state S^0 be given, and $k = k(\mathcal{A}, S^0)$, $N = N(\mathcal{A}, S^0)$. 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.*

5 Ground deterministic MA-systems

By Proposition 2, the MA-BEHAVIOR problem for ground MA-systems belongs to PSPACE. We point out two interesting cases, where it is 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 MA-systems for the behavior properties $\Phi \in LTL$.*
(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 $m^2 * r = \boldsymbol{O}(\log |\mathcal{A}|)$, for the behavior properties $\Phi \in LTL$.

Proof. (1) We show that in this case, the time complexity of the algorithm DetCheck can be bounded by a polynomial in $|\mathcal{A}|+|\Phi|$. In the lemma to follow we establish a monotonicity property of trajectories.

Lemma 4 *Let $\mathcal{A}$ be an expanding and positive MA-system (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 $S^t =< (I^t_{a_1}, MsgBox^t_{a_1}), \ldots, (I^t_{a_n}, MsgBox^t_{a_n}) >$ and $S^{t+1} =< (I^{t+1}_{a_1}, MsgBox^{t+1}_{a_1}), \ldots, (I^{t+1}_{a_n}, MsgBox^{t+1}_{a_n}) >$ of τ, the following inclusions hold for every $1 \leq i \leq n$: $I^t_{a_i} \subseteq I^{t+1}_{a_i}$ and $MsgBox^t_{a_i} \subseteq MsgBox^{t+1}_{a_i}$.*

Proof. The inclusions of DB-states is ensured by the absence of deletions in the actions of the agents of $\mathcal{A}$. The message boxes inclusion is proved by induction. It is evident for $t = 0$, since $MsgBox^0_{a_i} = \emptyset$ for every $0 \leq i \leq n$. Now, let us suppose that for any $a \in \mathcal{A}$, $MsgBox^t_a \subseteq MsgBox^{t+1}_a$. Since P_a is positive, $M^t_a = Sem(P_a)(I^t_a, MsgBox^t_a) \subseteq M^{t+1}_a = Sem(P_a)(I^{t+1}_a, MsgBox^{t+1}_a)$. Then, due to monotonicity of one-step semantics, the inclusion $Act^t_a \subseteq Act^{t+1}_a$ holds. Therefore, the set of messages which a sends at the step $t+1$, includes all the messages it has sent at the step t. These messages arrive at the message boxes of the agents $a_i \neq a$ at the step $t+2$, so $MsgBox^{t+1}_{a_i} \subseteq MsgBox^{t+2}_{a_i}$ and the assumption is valid for $t+1$. □

Turning back to the proof of the theorem, let us notice that from the lemma 4 it follows that at each moment t, when $S^t \neq S^{t+1}$, at least one new action $\alpha \in AB_a$ of some agent $a \in \mathcal{A}$ should be fired, i.e. $\alpha \in Act^t_a \setminus Act^{t-1}_a$. Let

$N_{\mathcal{A}} = \Sigma_{a\in\mathcal{A}}|AB_a|$. Then after $N_{\mathcal{A}}$ steps the trajectory τ cannot be changed, i.e. $S^t = S^{t+1}$ for any $t > N_{\mathcal{A}}$. Therefore, for this trajectory τ, the sum of its parameters $k + N$ does not exceed $N_{\mathcal{A}} \leq |\mathcal{A}|$. By lemma 1, we obtain for algorithm DetCheck the time bound $pol_1(|\Phi||\mathcal{A}|(t(F^\tau) + t_{FO}(\tau, |\Phi|))$ for some polynomial pol_1. Since we are interested only in states S^t of τ with $t \leq k + N$ and $|S^t| \leq |\mathcal{A}|$, it follows from lemma 2 that $t(F^\tau) \leq pol_2(|\mathcal{A}|)$ for some polynomial pol_2. The validity of a ground first-order formula on a state S^t of τ can also be recognized in time polynomial in $|S^t|$, i.e. $t_{FO}(\tau, |\Phi|) \leq pol_3(|\mathcal{A}| + |\Phi|)$ for some polynomial pol_3. Therefore, the algorithm DetCheck checks whether $\tau \models \Phi$ in time bounded by $pol(|\mathcal{A}| + |\Phi|)$ for some polynomial pol.

(2) In this case we also show that time of the algorithm DetCheck can be bounded by a polynomial in $|\mathcal{A}| + |\Phi|$. Let $\tau = \tau(\mathcal{A}, S^0) = S^0, S^1, \ldots S^t, \ldots$ be a trajectory of $\mathcal{A}$ and $T = k + N$ be such minimal step that $S^T = S^{T-N}$ for some $N > 0$. Since $\mathcal{A}$ is expanding, then for every $a \in \mathcal{A}$ and for every $t \geq 0$, the local state I_a^t is not reduced: $I_a^t \subseteq I_a^{t+1}$. Let N_m be the total number of all possible global states of message boxes of $\mathcal{A}$. Then for any $t \geq 0$ such that $t + N_m + 1 < T$, there is a pair $S^{t'}, S^{t'+1}$ in the subsequence $S^t, \ldots, S^{t+N_m+1}$ such that for some $a \in \mathcal{A}$, its local state increases at the step t' : $I_a^{t'} \subset I_a^{t'+1}$. But since $\mathcal{A}$ is ground, all the atoms added to $I_a^{t'}$ at this step are present in $\mathcal{A}$ itself. So the number of steps t' at which $I_a^{t'}$ increases is bounded by $|\mathcal{A}|$. Then the total number of steps at which the DB-state of some agent $a \in \mathcal{A}$ increases does not exceed $m|\mathcal{A}|$. Therefore, $T = k + N \leq N_m m|\mathcal{A}|$. Since the number of different messages in the message box of a sent by a particular agent is bounded by r and the number of agents of $\mathcal{A}$ is bounded by m, it is evident that $N_m \leq 2^{rm^2} \leq c|\mathcal{A}|$ for some constant c, and therefore, τ $k+N \leq cm|\mathcal{A}|^2$. This will allow to obtain a polynomial time bound for algorithm DetCheck along the same lines as in the case (1). □

For reasons of space, we do not consider in this paper the cases resulting from weakening the constraints imposed on the MA-systems by Theorem 1. In general, it causes a substantial increase of complexity of the MA-BEHAVIOR problem, especially if the constraint of groundness is lifted. There is however, an interesting particular case.

Corollary 1 *The MA-BEHAVIOR problem is decidable in polynomial time in the class of nonground expanding, positive k-dimensional MA-systems, for behavior properties $\Phi \in LTL$ and for any fixed k.*

Proof. Let $\mathcal{A}$ be an expanding, positive and k-dimensional MA-system, 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 extensional 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 $\alpha \in Act_a^t \backslash Act_a^{t-1}$ 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 assertion.

Lemma 5 *For all k, there is a polynomial pol such that for any k-dimensional MA-system $\mathcal{A}$ 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 $\mathcal{A}$.*

Proof. Since the total number of action names and of predicates in the extensional and message signatures of agents in $\mathcal{A}$ does not exceed $|\mathcal{A}|$, and the total number of constants in ground terms of τ is bounded by $|S^0| + |\mathcal{A}|$, the lemma follows from the evident inequalities $n_{act} \leq (|S^0| + |\mathcal{A}|)^k$ and $n_g \leq (|S^0| + |\mathcal{A}|)^k$. □

6 Ground nondeterministic MA-systems

This class of MA-systems has an interesting subclass, where the MA-BEHAVIOR problem is solvable in nondeterministic polynomial time.

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

Proof. We present a proof for $\exists LTL$ formulas. The case of $\forall LTL$ formulas is treated using the equivalence: $\tau(\mathcal{A}, S^0) \models \mathbf{E}(\Psi) \Leftrightarrow \tau(\mathcal{A}, S^0) \not\models \mathbf{A}(\neg\Psi)$.

Upper bound. Let $\mathcal{A}$ be a ground, expanding, and r-signal m-agent system such that $m^2 * r = \boldsymbol{O}(\log |\mathcal{A}|)$. A nondeterministic polynomial time algorithm for the MA-BEHAVIOR problem is based on the following length upper bound for some trajectories of $\tau(\mathcal{A}, S^0)$.

Lemma 6 *There is a polynomial $p(n)$ such that $\tau(\mathcal{A}, S^0) \models \mathbf{E}\ \Psi$ iff there is a trajectory $\mu = S^0, \ldots, S^t, \ldots \in \tau(\mathcal{A}, S^0)$ with $S^i =< (I^i_a, MsgBox^i_a) \mid a \in \mathcal{A} >$ and a step $T \leq p(|\mathcal{A}| + |\Psi|)$ such that $\mu \models \Psi$ and $I^t_a = I^T_a$ for all $t > T$ and every $a \in \mathcal{A}$.*

Proof. We will establish several auxiliary assertions concerning the validity of $FLTL$-formulas on trajectories with long series of subsequent repetitions of states. We remind that the formulas depend only on DB-states (and not on the message boxes) in the trajectories. Let $\mu = M^0, \ldots, M^i, \ldots$ and $\nu = N^0, \ldots, N^j, \ldots$ be two trajectories and $d \geq 0$ be an integer. We say that a pair (μ, M^i) is *d-equivalent* to a pair (ν, N^j) (denoted: $(\mu, M^i) \sim_d (\nu, N^j)$) iff $\mu^i \models \varphi \Leftrightarrow \nu^j \models \varphi$ is true for any $FLTL$-formula φ of depth $d(\varphi) \leq d$.

Assertion 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 some* $d \geq 0$, *then* $(\mu, M^i) \sim_{d+1} (\nu, N^j)$ *is true.*

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 a $FLTL$-formula of depth $d+1$. If $\varphi = \varphi_1 \oplus \varphi_2$ ($\oplus \in \{\wedge, \vee\}$) or $\varphi = \neg\varphi_1$, then $d(\varphi_1) \leq d$ and $d(\varphi_2) \leq d$. By the assumption, $\mu^i \models \varphi_k \Leftrightarrow \nu^j \models \varphi_k$ ($k = 1, 2$). Hence, $\mu^i \models \varphi \Leftrightarrow \nu^j \models \varphi$. If $\varphi = \mathbf{X}(\varphi_1)$, then by the assumption, $\mu^{i+1} \models \varphi_1 \Leftrightarrow \nu^{j+1} \models \varphi_1$ and $\mu^i \models \varphi \Leftrightarrow \nu^j \models \varphi$. Now suppose that $\mu^i \models \varphi$ for $\varphi = \varphi_1 \mathbf{U} \varphi_2$. Then by definition of operator $\mathbf{U}$, (i) $\mu^i \models \varphi_2$ or else (ii) $\mu^i \models \varphi_1$ and $\mu^{i+1} \models \varphi$. In the case (i), since $d(\varphi_2) = d$, by the first assumption, we obtain $\nu^j \models \varphi_2$ and consequently, $\nu^j \models \varphi$. In a similar manner, in the case (ii), we show that $\nu^j \models \varphi_1$. Moreover, from the second assumption, we deduce $\nu^{j+1} \models \varphi$. So in this case too, we establish $\nu^j \models \varphi$. Therefore, in all the cases, $\nu^j \models \varphi$ and $(\mu, M^i) \sim_{d+1} (\nu, N^j)$.

This assertion directly implies the next one:

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

Assertion 3 *Let* $\mu = M^0, \ldots$ *be a 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* $(\mu, M^i) \sim_{d+1} (\mu, M^{i+1})$.

Let φ be some $FLTL$-formula of depth $d(\varphi) = d + 1$. If φ is a boolean combination of formulas of depth $\leq d$, then $\mu^i \models \varphi \Leftrightarrow \mu^{i+1} \models \varphi$, since $(\mu, M^i) \sim_d (\mu, M^{i+1})$. If $\varphi = \mathbf{X}(\varphi_1)$, then $d(\varphi_1) \leq d$ and $\mu^i \models \varphi \Leftrightarrow \mu^{i+1} \models \varphi$, since $(\mu, M^{i+1}) \sim_d (\mu, M^{i+2})$. If $\varphi = \varphi_1 \mathbf{U} \varphi_2$, then $d(\varphi_1) \leq d$ and $d(\varphi_2) \leq d$. Suppose that $\mu^i \models \varphi$. Then by definition of operator $\mathbf{U}$, we have $\mu^i \models \varphi_2$ or else $\mu^i \models \varphi_1$ and $\mu^{i+1} \models \varphi$. In both cases, it is evident that $\mu^{i+1} \models \varphi$ (in the first case the assumption $M^i \sim_d M^{i+1}$ is used). On the other hand, assume $\mu^{i+1} \models \varphi$. Then (i) $\mu^{i+1} \models \varphi_1$ or (ii) $\mu^{i+1} \models \varphi_2$. In the case (i), the assumption $(\mu, M^i) \sim_d (\mu, M^{i+1})$ implies $\mu^i \models \varphi_1$ and, together with $\mu^{i+1} \models \varphi$, it implies $\mu^i \models \varphi$. The case (ii) is similar.

Assertion 3 directly implies

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

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

¿From assertion 4, it follows that $(\mu, M^i) \sim_K (\mu, M^{i+1})$. Thus, $(\mu, M^{i-1}) \sim_0 (\mu', M^{i-1})$ and $(\mu, M^i) \sim_K (\mu', M^{i+1})$. By assertion 2, this implies $(\mu, M^{i-1}) \sim_K (\mu', M^{i-1})$. Then since $(\mu, M^{i-2}) \sim_0 (\mu', M^{i-2})$, we get further $(\mu, M^{i-2}) \sim_K (\mu', M^{i-2})$ and so on until we get $(\mu, M^0) \sim_K (\mu', M^0)$.

Now, returning to the proof of lemma 6, let us suppose that $\tau(\mathcal{A}, S^0) \models$

$\mathbf{E}\ \Psi$. This means that $\lambda \models \Psi$ for some trajectory $\lambda = S^0, \ldots, S^t, \ldots \in \tau(\mathcal{A}, S^0)$. Let $t_1, t_2, \ldots, t_i, \ldots, t_k$ be those steps of λ at which agents' DB-states grow, i.e. $I_a^{t_i} \subset I_a^{t_i+1}$ for some $a \in \mathcal{A}$. Since $\mathcal{A}$ is expanding, k is bounded by the total number of actions of agents in $\mathcal{A}$. So $k \leq |\mathcal{A}|$. By the choice of steps t_i, the DB-states of all agents in $\mathcal{A}$ do not change at steps $t_i + 1, t_i + 2, \ldots, t_{i+1}$ for all i. Let $k_i = t_{i+1} - t_i$ be the length of such stable subsequence of DB-states. Let N_m denote the number of all possible states of message boxes of $\mathcal{A}$. Then, as it was shown in the proof of theorem 1 (2), $N_m \leq c|\mathcal{A}|$ for some constant c. Let $d = d(\Psi)$. If $k_i > d + 2 + N_m$, then there are steps l and r, $t_i + d + 2 < l < r < t_{i+1}$, such that $S^l = S^r$. Then $\tau(\mathcal{A}, S^0)$ has also the trajectory $\mu = S^0, \ldots, S^l, S^{r+1}, \ldots$ obtained from λ by deleting states $S^{l+1}, \ldots, S^r$. Assertion 5 ensures that $(\lambda, S^0) \sim_d (\mu, S^0)$ and therefore, $\mu \models \Psi$. Thus, there is a trajectory $\mu \in \tau(\mathcal{A}, S^0)$ such that $\mu \models \Psi$ and the length of longest subsequence of equal DB-states in μ is bounded by $d + 2 + N_m \leq c(|\mathcal{A}| + |\Psi|)$. For this trajectory, we have a stabilization step $T \leq t_k + 1 \leq k + k(d + 2 + N_m) \leq pol(|\mathcal{A}| + |\Psi|)$.

The upper bound follows from lemma 6 and theorem 1(2). Let $\mathcal{A}$ be a ground, expanding, and r-signal nondeterministic m-agent system verifying the condition $m^2 * r = \boldsymbol{O}(\log |\mathcal{A}|)$, S^0 be its initial state, $\tau = \tau(\mathcal{A}, S^0)$ be the tree of trajectories of $\mathcal{A}$ starting in S^0 and Ψ be an LTL formula. Set $T = pol(|\mathcal{A}| + |\Psi|)$, where pol is the polynomial defined in lemma 6. In order to check whether $\tau(\mathcal{A}, S^0) \models \mathbf{E}\ \Psi$, we use the following nondeterministic algorithm NdetCh:
(1) guess in tree $\tau(\mathcal{A}, S^0)$ a finite trajectory $\lambda = S^0, S^1, \ldots, S^T, \ldots, S^{2T}$, in which $I_a^T = I_a^{T+1} = I_a^{T+2} = \ldots = I_a^{2T}$ for all $a \in \mathcal{A}$;
(2) check whether $\lambda' \models \Psi$ using algorithm DetCheck as it is implemented in theorem 1 (2) and return answer "Yes" if $\lambda \models \Psi$.

It is evident that NdetCh works in nondeterministic polynomial time. In order to prove its correctness, we remark that if $\tau(\mathcal{A}, S^0) \models \mathbf{E}\ \Psi$, then by lemma 6, there is a trajectory $\lambda \in \tau(\mathcal{A}, S^0)$ verifying $\lambda \models \Psi$ and having a finite prefix $\lambda' = S^0, \ldots, S^T$ such that $\lambda' \models \Psi$ and $I_a^j = I_a^T$ for all $a \in \mathcal{A}$ and every $j > T$. So at the step (1), the algorithm can guess this short finite prefix and return the answer "Yes". Conversely, if NdetCh returns the answer "Yes", then in $\tau(\mathcal{A}, S^0)$, there is a finite trajectory $\lambda = S^0, S^1, \ldots, S^T, \ldots, S^{2T}$ such that $\lambda \models \Psi$ and $I_a^T = I_a^{T+1} = I_a^{T+2} = \ldots = I_a^{2T}$ for all $a \in \mathcal{A}$. Since $T > N_m$, there are such i and j $(T < i < j < 2T)$ that $S^i = S^j$. Then the prefix of λ of length T can be extended to some infinite trajectory $\lambda' \in \tau(\mathcal{A}, S^0)$ such that $\lambda' = S^0, \ldots, S^T, S'^{T+1}, S'^{T+2}, \ldots$ and $I'^j_a = I_a^T$ for all $a \in \mathcal{A}$ and $j > T$. For this trajectory λ', $\lambda' \models \Psi$ and $\tau(\mathcal{A}, S^0) \models \mathbf{E}\ \Psi$.

Lower bound. It is easy to show that the problem SAT is reducible in polynomial time to MA-BEHAVIOR problem for nondeterministic ground, expanding, and 0-signal 1-agent systems. Indeed, let α be a propositional formula and let $V = \{x_1, \ldots, n\}$ be the set of all its propositional letters. Let us consider the MA-system $\mathcal{A}$ having a single agent a with extensional signature V, with action base AB_a consisting of n actions ac_i $(i = 1, \ldots, n)$, each

action ac_i adding x_i to I_a and with program P_a consisting of facts $ac_i \leftarrow$ $(i = 1, \ldots, n)$. Then it is easy to check that for $S^0 = \emptyset$, $\varphi \in SAT \Leftrightarrow \tau(\mathcal{A}, S^0) \models \mathbf{E}(\varphi)$ if we choose the unit-choice nondeterministic one-step semantics and $\varphi \in SAT \Leftrightarrow \tau(\mathcal{A}, S^0) \models \mathbf{EX}(\varphi)$ if we choose the spontaneous nondeterministic one-step semantics. □

The complexity of MA-BEHAVIOR problem increases substantially if we weaken requirements to MA-systems. We do not consider this case for space reasons.

Conclusion

IA and MA-system architectures published within the past few years are dissimilar and diversified because they represent various application domains of this new software technology. Technically, our study concerns one such specific architecture. However, it illustrates the way in which penetrating deeply in a complex MA-system architecture permits in some cases to understand more deeply the behavior properties and in this way, to discover interesting classes of MA-systems with with efficiently checked behavior properties.

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] 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).

[5] Bradfield, J., Stirling, C., Modal logics and mu-calculi. In: J. Bergstra, A. Ponse and S. Smolka (Eds.) *Handbook of Process Algebra.* 2001, Elsevier, North-Holland, 293-332.

[6] Clarke, E. M., Emerson, E. A. Design and synthesis of synchronization skeletons using branching time temporal logic. In: *Proc.of Workshop on Logics of Programs, LNCS,* N. 181, 1981, 52-71.

[7] Clarke, E. M., Grumberg, O. and Long, D., Model Checking, *NATO ASI series F,* v. 152, 1996.

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

[9] Dekhtyar, M. I., Dikovsky, A. Ja., On Homeostatic Behavior of Dynamic Deductive Data Bases. In: *Proc. 2nd Int. A.P.Ershov Memorial Conference "Perspective of Systems Informatics", LNCS,* N. 1181, 1996, 420-432.

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

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

[12] 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.

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

[14] Fisher, M., Wooldridge, M., Specifying and Verifying Distributed Intelligent Systems. In: M. Filgueiras and L. Damas (Eds.) *Progress in Artificial Intelligence – Sixth Portuguese Conf. on Artificial Intelligence. LNAI*, N. 727, 1993, 13-28.

[15] 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.

[16] 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.

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

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

[19] Petrie, C., What is an agent? In: J.P. Müller, M. J. Wooldridge, and N. R. Jennings (Eds.) *Intelligent Agents III – Proc. of the Third Intern. Workshop on Agent Theories, Architectures, and Languages, LNAI,* N. 1193, 41-43.

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

[21] 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.

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

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

[24] Sistla, A. P., Clarke, E. M., The complexity of propositional linear temporal logic. J.ACM, 1985, 32(3): 733-749.

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

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

[27] 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.

[28] Wooldridge, M., The Computational Complexity of Agent Design Problem. In: E. Durfee, (Ed.) *Proc. of the Fourth Intern. Conf. on Multi-Agent Systems (ICMAS 2000)*, IEEE Press, 2000.

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