On Complexity of Behavior Properties of Interacting Agents

Extended abstract *

Michael I. Dekhtyar
Dept. of CS, Tver St. Univ., Tver, Russia, 170000, *Michael.Dekhtyar@tversu.ru*
Alexander Ja. Dikovsky
IRIN, Université de Nantes, France, *Alexandre.Dikovsky@irin.univ-nantes.fr,*
Keldysh Inst. for Appl. Math., Moscow, Russia, 125047
Mars K. Valiev
Keldysh Inst. for Appl. Math. Moscow, Russia, 125047, *valiev@spp.keldysh.ru*

Abstract

The complexity is investigated of the properties of interacting intelligent agents' behavior, expressible in the first order logic of linear time. Some tight complexity bounds for such properties are established under certain restrictions on the classes of agent programs.

1 Introduction

Software agents are one of the most actively investigated and rapidly evolving areas in computer science. They have applications in artificial intelligence, Internet computing, networking and some other areas. There are various approaches to agents architecture and agents design. A survey of many (by no means all) of these approaches is included into the book of Subrahmanian et al. [12]. This book also contains a thorough presentation of a theoretical framework for the agent-based computing and of some implementation methods. This framework differs from many others by its orientation to a logical description of agents. In this paper, we use basically the concepts of [12], however, because of the space const6ain

in [4, 5, 6]. Our approach to the behavioristic analysis of agents systems is close to that developed in these papers.

Another logical framework for agents systems is presented in [1] where three different models are introduced : so-called I/O automata, a knowledge based Erdös environment, and an extension of Milner's π-calculus. In the Erdös environment, the behavior of agents systems is specified in the propositional temporal logic CTL. These temporal specifications are verified using model checking for CTL. This approach seems to be the closest to ours.

One more approach is related to the language Concurrent Metatem [2, 9]. Here the agents are described using linear temporal logic specifications, and the method of resolution adapted to temporal logic is used for verification of dynamic properties of agents systems.

We investigate the behavior of a system $\mathcal{A} = \{a_1, ..., a_n\}$ of n interacting agents. The agents in the system communicate via messages. At every step, each agent executes concurrently some subset of a predefined set of actions depending on its internal state and received messages. As a result, it changes its internal state and sends some messages to other agents. Informally, an architecture of an agent includes (i) *internal data structures (a database)* representing its internal state, (ii) a *message box* containing messages from other agents, (iii) *integrity constraints* on the states of the agent, (iv) an *action base consisting* of the list of actions which the agent can execute and (v) an *agent program* defining the action policy of the agent, i.e. the set of actions which the agent should execute at each step. Execution of an action consists of two steps: (a) the current state of the agent is updated, and (b) some messages are sent to other agents. The current content of an agent's message box consists of the messages received by the agent from other agents at the previous step. The global state of the system consists of internal states and contents of message boxes of all the agents in $\mathcal{A}$.

The semantics of $\mathcal{A}$ is described as a deterministic transition operator on global states composed of individual states of its agents. In this way, for any initial global state, $\mathcal{A}$ induces a unique trajectory, i.e. a sequence of global state transitions starting in this global state. The behavior of $\mathcal{A}$ is defined in terms of this trajectory.

We investigate the complexity of decision problems concerning the behavior analysis of systems of agents. One of the most important is the *safety* problem for $\mathcal{A}$. Safety means that when the initial global state S satisfies a given safety condition P, all the global states of $\mathcal{A}$ accessible from S do also satisfy P. Another interesting problem is that of *goal achievement* : to decide for a given agents system $\mathcal{A}$, an initial global state S and a goal statement G whether there will be a moment of time when the global state of $\mathcal{A}$ will satisfy G. In order to treat systematically these problems, we use as a behavior specification language a version of the First Order Linear Temporal Logic (FLTL) from [10, 8] without functional symbols. Similar logics are widely used by many authors as a means of formal specification and verification of parallel programs (cf.[8, 10]).

The general decision problem (*AGENT_BEHAVIOR*) we consider is defined as follows: for a given agents system $\mathcal{A}$, some its initial global state S and an FLTL-formula Φ to decide if the trajectory of $\mathcal{A}$ starting in S satisfies Φ.

We establish for this problem some tight complexity bounds under certain restrictions on the classes of agent programs.

This problem is a variant of the well known problem of model checking widely used

for verification of concurrent systems (see [8, 3]). Typically, the model checking is accomplished in two stages. First, the concurrent system C is transformed into a transition system T, and then the actual model checking is applied to T. The size of T is usually exponential with respect to the size of C, so the first stage of this process implies at least exponential complexity of model checking. Below we show that for deterministic synchronous agents systems, the model checking can be accomplished without preliminary complete generation of T: instead, T can be generated portionwise.

2 Interactive Intelligent Agents

The notion of agent we use is along the lines of [12]. Of course, the agent architecture IMPACT introduced in this book is too rich to be presented in a short paper. Moreover, some its features are not relevant for behavior analysis. So we simplify this architecture leaving only those agent features which are essential for the complexity analysis. In particular, we do not consider features related to the legacy code, security, metaknowledge structures, temporal and uncertainty reasoning. We fix an internal data structure of an agent to be a relational database (in IMPACT a more general structure is allowed), and consider conventional logic programs as a means of an agent action policy definition (IMPACT agent programs include deontic modalities "permitted action", "forbidden action", "action to be performed", etc.)

An agents system as it shows up in this paper, is a system $\mathcal{A} = \{a_1, ..., a_n\}$ of n interacting agents. The formal definition of $\mathcal{A}$ includes three finite predicate signatures: an extensional signature $\mathbf{P}_e$ used to represent facts of internal databases, a message signature $\mathbf{P}_m$, and an intensional signature $\mathbf{P}_{act}$ used to represent the actions of the agents. Besides this, some finite sets of variables $\mathbf{V}$ and constants $\mathbf{C}$ are associated with $\mathcal{A}$. By $\mathbf{A}_e, \mathbf{A}_m$, and $\mathbf{A}_{act}$, we denote the sets of atoms over the corresponding predicate signatures and $\mathbf{V}$ and $\mathbf{C}$. 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.

Agent Data. Each agent $a \in \mathcal{A}$ disposes of its own internal data structures. We represent these data in the form of a relational data base with the scheme denoted by $\mathbf{P}_e$ and current states denoted by I_a.

Messages. Agents in $\mathcal{A}$ communicate via *messages*. Messages are ground facts (atoms) in the message signature $\mathbf{P}_m$. Each agent a has a buffer $MsgBox_a$ which can hold messages received by a from other agents. Elements of $MsgBox_a$ are pairs of the form $(Source\_agent, Message)$, where $Source\_agent$ is a name of the agent sending the message. We call the pair $(I_a, MsgBox_a)$ *a local state* of the agent a.

Integrity Constraints. A set of *integrity constraints* IC_a is associated with each agent a, which defines a property of its DB states. Integrity constraints can be represented by a formula, or a logical program, or a conventional program computing this property. We presume that there is an algorithm which checks the integrity constraints IC_a in a state I_a in polynomial space with respect to $|I_a| + |IC_a|$.

Actions. Every agent has an associated set of actions. An action is represented by an atom $\alpha(X_1, ..., X_l)$, $\alpha^{(l)}$ being a predicate in $\mathbf{P}_{act}$ and X_i being different variables, and by three (eventually empty) sets of atoms with variables in $\{X_1, ..., X_l\}$:
$ADD(\alpha)$ – a list of facts in $\mathbf{A}_e$ to be added to I_a;

$DEL(\alpha)$ – a list of facts in $\mathbf{A}_e$ to be removed from I_a;
$SEND(\alpha)$ – a list of pairs of the form $(destination\_agent, Msg)$, where $Msg \in \mathbf{B}_m$.

Concurrent Execution of Actions. Let $AS = \{\alpha_1(\bar{T}_1), ..., \alpha_k(\bar{T}_k)\} \subseteq \mathbf{B}_{act}$ be a set of ground actions of an agent a and $\bar{\sigma} = (\sigma_1, ..., \sigma_k)$, be a list of ground substitutions such that $\sigma_j(\alpha_j(\bar{X}_j)) = \alpha_j(\bar{T}_j)$. The concurrent execution of $\bar{\sigma}(AS)$ in a local state $(I_a, MsgBox_a)$ is defined as follows:
a) the new internal state of a results from its current state by the assignment :

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

(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.);
b) every message produced by $\bar{\sigma}(AS)$ is placed into the $MsgBox$ of the destination agent. To be more precise, for every agent $b \neq a$,

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

An Agent Program P_a defines the action policy of an agent a. It is a logic program with the clauses of the form

$$H \leftarrow L_1, ..., L_n,$$

where the head H is an (intensional) action atom in $\mathbf{A}_{act}$ and literals L_i in its body are either action atoms in $\mathbf{A}_{act}$, or data base (extensional) literals in $\mathbf{LB}_e$, or message literals of the form $Received(Source\_agent, Message)$ or $\neg Received(Source\_agent, Message)$ with $Message \in \mathbf{A}_m$.

Program Semantics determines the set of actions executed by the agent in its current local state. The semantics $Sem(P_a)(I_a, MsgBox_a)$ of P_a with respect to a state I_a and a message box $MsgBox_a$ is defined as a minimal Herbrand model of the program $P_a \cup I_a \cup \{Received(Agent\_source, Message) \mid (Agent\_source, Message) \in MsgBox_a\}$.

Evidently, the logic programs we consider are normal programs without recursion through negation, a very special case of stratified logic programs. As it is well-known, every such program has a single minimal model computed in polynomial time by a straightforward fixpoint computation procedure. In fact, the complexity results we obtain are valid for any subclass of normal logic programs for which a unique minimal model exists and is computed in polynomial time, e.g. for the class of stratified logic programs.

A global state of the agents system $\mathcal{A}$ at the moment t is defined as the n-tuple of local states of the agents $a_1, ..., a_n$, i.e. $S^t = < (I^t_{a_1}, MsgBox^t_{a_1}), ..., (I^t_{a_n}, MsgBox^t_{a_n}) >$.

The deterministic one-step semantics of the agents system $\mathcal{A}$ is defined as a transition operator $\Rightarrow$ on global states of the system, which we present in the form of the following algorithm transforming a global state S^t into the next global state S^{t+1} :

(1) FOR EACH $a_i \in \mathcal{A}$ DO $Acts^t_i := Sem(P_{a_i})(I^t_{a_i}, MsgBox^t_{a_i})$;
(2) FOR EACH $a_i \in \mathcal{A}$ DO $MsgBox^{t+1}_{a_i} := \emptyset$;

(3) FOR EACH $a_i \in \mathcal{A}$ DO
IF IC_{a_i} is not violated by the execution of $Acts_i^t$
THEN execute $Acts_i^t$ concurrently in the local state $(I_{a_i}^t, MsgBox_{a_i}^{t+1})$.

Thus, the net content of the message box of an agent a at the moment $t+1$ consists of all messages sent to a by all other agents at the moment t.

Agents System Behavior. The behavior of an agents system $\mathcal{A} = \{a_1, ..., a_n\}$ in the initial global state $S^0 =< (I_{a_1}^0, MsgBox_{a_1}^0), ..., (I_{a_n}^0, MsgBox_{a_n}^0) >$ is described as the following infinite sequence of global states, $\tau = \tau(\mathcal{A}, S^0)$:
$< (I_{a_1}^0, MsgBox_{a_1}^0), ..., (I_{a_n}^0, MsgBox_{a_n}^0) > \Rightarrow ...$
$< (I_{a_1}^t, MsgBox_{a_1}^t), ..., (I_{a_n}^t, MsgBox_{a_n}^t) > \Rightarrow$
$< (I_{a_1}^{t+1}, MsgBox_{a_1}^{t+1}), ..., (I_{a_n}^{t+1}, MsgBox_{a_n}^{t+1}) > \Rightarrow ...$
where the global state at the step $t+1$ is obtained from the preceding global state by applying the transition operator. We call it a *trajectory* of $\mathcal{A}$ in the state S^0.

This definition guarantees that the agents system behavior is locally consistent.

Lemma 1 *Let τ be a trajectory of an agents system $\mathcal{A} = \{a_1, ..., a_n\}$ in the state S^0. Then if for some $j, 1 \leq j \leq n$, the initial local state $I_{a_j}^0$ of a_j satisfies IC_{a_j}, then the state $I_{a_j}^t$ satisfies IC_{a_j} for every $t > 0$ as well.*

Example. Let us consider the following toy resource allocation system $\mathcal{A}$ consisting of an agent-manager m which owns some resource and distributes it among four agents-users u_1, u_2, u_2, u_4. Each of the users has its own strategy of the resource ordering:
1) u_1 asks for a resource at the first moment and he repeats his order the next moment after he receives the resource;
2) u_2 asks just after u_1 has asked;
3) u_3 asks just after u_1 has received the resource from m;
4) u_4 asks all the time.
Manager m keeps a list of orders and fulfills one order at a time and the first in the list. Only one order of each agent-user can be included in the list. So if m receives the order from some user before the previous order of this user has been fulfilled, then the new order is discarded.

The agents of $\mathcal{A}$ are defined as follows. The state I_{u_1} of u_1 can include the fact put_order, the states I_{u_i} $(i = 2, 3, 4)$ are always empty, database I_m of m includes the facts of the form $order(X, I)$ ($order(u_i, j)$ means that the order of agent u_i is at the j-th place in the order list of m), $is(X)$ (the order of agent X is in the list of m), $num\_orders(I)$ (I is the number of unsatisfied orders). In order to let know m and other users that u_i asks for a resource, the agent sends the message $order$ to them. When m fulfills an order of u_i he sends the message ok to u_i. When u_1 wants to inform u_3 that he has received the resource, he sends the message ok to u_3.
Agent u_1. Actions:
$do\_order:\ ADD = \{put\_order\}, SEND = \{(m, order), (u_2, order)\}$;
$receive\_order:\ DEL = \{put\_order\}, SEND = \{(u_3, ok)\}$;
P_{u_1} :
$do\_order \leftarrow \neg put\_order$
$receive\_order \leftarrow Received(m, ok)$

Agent u_2. Actions:
$do\_order: \ SEND = \{(m, order)\}$;
P_{u_2} :
$do\_order \leftarrow Received(u_1, order)$

Agent u_3. Actions:
$do\_order: \ SEND = \{(m, order)\}$;
P_{u_3} :
$do\_order \leftarrow Received(u_1, ok)$
Agent u_4. Actions:
$do\_order: \ SEND = \{(m, order)\}$;
P_{u_4} :
$do\_order \leftarrow .$

Agent m. Actions:
$ins\_order(X, I): \ ADD = \{order(X, I), is(X)\}$;
$fulfill\_order(X): \ DEL = \{order(X, 1), is(X)\}, SEND = \{(X, ok)\}$;
$move(X, I): \ ADD = \{order(X, I)\}, DEL = \{order(X, I + 1)\}$;
$new\_num(I, J) : ADD = \{num\_orders(J)\}, DEL = \{num\_orders(I)\}$

P_m :
$new\_order(X) \leftarrow Received(X, order), \neg is(X)$
$first\_place(I) \leftarrow num\_orders(I), I > 0$
$first\_place(1) \leftarrow num\_orders(0)$
$ins\_order(X, I) \leftarrow new\_order(X), first\_place(I) \quad (X \in \{u_1, u_2\})$
$ins\_order(u_3, I) \leftarrow new\_order(u_3), ins\_order(X, I - 1) \quad (X \in \{u_1, u_2\}$
$ins\_order(u_3, I) \leftarrow new\_order(u_3), first\_place(I), \neg ins\_order(u_1, I),$
$\neg ins\_order(u_2, I)$
$ins\_order(u_4, I) \leftarrow new\_order(u_4), ins\_order(u_3, I - 1)$
$ins\_order(u_4, I) \leftarrow new\_order(u_3), \neg ins\_order(u_3, I - 1),$
$ins\_order(X, I - 1) \quad (X \in \{u_1, u_2\})$
$ins\_order(u_4, I) \leftarrow new\_order(u_3), first\_place(I), \neg ins\_order(u_1, I),$
$\neg ins\_order(u_2, I), \neg ins\_order(u_3, I - 1)$
$fulfill\_order(X) \leftarrow order(X, 1)$
$move(X, 1) \leftarrow fulfill\_order, order(X, 2)$
$move(X, I) \leftarrow move(Y, I - 1), order(X, I + 1)$
$new\_num(I, J) \leftarrow num\_orders(I), num\_new\_orders(K), J = I + K$

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

3 Properties of Agents System Behavior and Temporal Logic

We investigate the complexity of decision problems related with behavior analysis of systems of agents. We have chosen the language of temporal logic as a means of behavior specification. The variant of temporal logic we use, the First Order Linear Temporal Logic (FLTL), is a simplified version of the logic in [10, 8].

The language of FLTL is defined as the closure of the set of (first order) state formulas under boolean connectives and temporal operators "until" (U) and "nexttime" ($\circ$). The signature of the state formulas we use includes the equality predicate and the set $\bigcup_1^n \mathbf{P}_e^i \cup \{Received^{(3)}\}$, where $Received^{(3)}$ is the name of the predicate $Received(Source\_agent, Target\_agent, Message)$ and $\mathbf{P}_e^i$ denotes the set of predicate symbols in the extensional signature of $\mathcal{A}$ indexed by i. The terms in this logic are either variables or constants in a set $\mathbf{D}$. We assume that $\mathbf{D}$ includes all constants which occur in agent programs, states and message boxes.

The trajectories of the agents systems serve as models of FLTL. The satisfaction of a formula in a model is defined pointwise, i.e. on a fixed global state in the trajectory.

Let $\tau = \tau(\mathcal{A}, S^0)$ be a trajectory of $\mathcal{A}$, $S^t =< (I^t_{a_1}, MsgBox^t_{a_1}), ..., (I^t_{a_n}, MsgBox^t_{a_n}) >$ be a global system state in τ, p^i be a k-ary predicate symbol in $\mathbf{P}_e^i$, and $c_1, ...c_k$ be constants in $\mathbf{D}$. Then a ground atom $p^i(c_1, ..., c_k)$ is true in $< \tau, S^t >$ (denoted $\tau, S^t \models p^i(c_1, ..., c_k)$) iff $p^i(c_1, ..., c_k) \in I^t_{a_i}$.

Let m be a k-ary predicate symbol in $\mathbf{P}_m$. Then the atom $Received(a_i, a_j, m(c_1, ..., c_k))$ is true in $< \tau, S >$ iff $(a_i, m(c_1, ..., c_k)) \in MsgBox^t_{a_j}$. The semantics of the boolean connectives and temporal operators is defined as usual. E.g. $\tau, S^t \models \Phi U \Psi$ iff $\exists t' \geq t$ such that $\tau, S^t \models \Psi$, and for all $t''(t \leq t'' \leq t')$, $\tau, S^{t''} \models \Phi$. Below we will use the usual temporal operators $\Box$ ("always in the future") and $\Diamond$ ("somewhere in the future") which are expressible in this logic.

The general decision problem ($AGENT\_BEHAVIOR$) we consider in this paper is as follows : for a given agents system $\mathcal{A}$, its initial global state S^0, and a FLTL-formula Φ, decide whether $\tau(\mathcal{A}, S^0), S^0 \models \Phi$.

The abovementioned safety and goal achievement problems are special cases of $AGENT\_BEHAVIOR$, where Φ has a form $\Box\Psi$ (respectively $\Diamond\Psi$) for some state formula Ψ. We also consider some other instances of the general $AGENT\_BEHAVIOR$ problem, where some restrictions are imposed on agent programs (e.g. groundness), their actions (absence of deletions), or signatures (bounds of their size and arities).

Example (continuation). For the system $\mathcal{A}$ above, one may check that the following formulas are valid on the trajectory generated by $\mathcal{A}$: $\Box\Diamond Received(m, u_i, ok)$ (every agent receives the resource infinitely often), $\Box\forall I \forall X \forall Y (orders(X, I)\ \&\ orders(Y, I) \rightarrow X = Y)$ (at each moment only one order can be placed at any place in the list), whereas the following formulas are not valid on this trajectory: $\Diamond(Received(m, u_1, ok) \& \circ Received(m, u_1, ok))$ (there are two consecutive moments when u_1 receives the resource) and $\Box(orders(u_i, 2) \rightarrow \circ\circ\neg Received(m, u_i, ok))$.

The following proposition shows that the $AGENT\_BEHAVIOR$ problem can be efficiently reduced to the tautology problem for FLTL.

Proposition 1 *There exists a polynomial time algorithm which for an input pair* $< \mathcal{A}, S_0 >$ *constructs an FLTL-formula* Ψ *such that for any FLTL-formula* Φ, $< \mathcal{A}, S^0, \Phi >$

$\in$ *AGENT_BEHAVIOR iff* $\Psi \to \Phi$ *is valid.*
If $\mathcal{A}$ *is ground, then* Ψ *is ground too.*

Since the tautology problem for the propositional (ground) version of FLTL (even with a more powerful temporal connective "until") is decidable in polynomial space [11], Proposition 1 implies that the *AGENT_BEHAVIOR* problem for ground agents systems $\mathcal{A}$ and ground properties Φ belongs to PSPACE.

The tautology problem for the general FLTL is undecidable. So, Proposition 1 does not help to decide the *AGENT_BEHAVIOR* problem if there are variables in $\mathcal{A}$ or in Φ. We remark however, that the *AGENT_BEHAVIOR* problem is in fact a variant of the model checking problem for a model $\tau(\mathcal{A}, S^0)$ implicitly defined by an agents system $\mathcal{A}$ and an initial state S^0.

Since $\mathcal{A}$ is deterministic and the set of its global states is finite, the trajectory $\tau(\mathcal{A}, S^0)$ is periodic. Hence, even though $\tau(\mathcal{A}, S^0)$ defines an infinite model, it can be folded into a finite pseudomodel. Model checking on such pseudomodels is decidable for FLTL-formulas. A straightforward implementation of model-checking would require an explicit representation of this pseudomodel, and consequently, the space at least equal to the total size of its global states. However, in our situation, as we show below, there exists a more intelligent way of model checking which looks-up the pseudomodel portionwise. It allows to obtain essentially better complexity upper bounds for the *AGENT_BEHAVIOR* problem.

We call a trajectory $\tau = S^0, S^1, ..., S^t, ...$ *periodic* if there exist two numbers k and N such that for any $t \geq k$, $\quad S^t = S^{t+N}$. We denote by F^τ the function-oracle which returns the state $F^\tau(t) = S^t$ of trajectory τ at any time point t.

The following lemma estimates the complexity of model checking for periodic trajectories.

Lemma 2 *There is an algorithm which for given numbers* k *and* N *and FLTL formula* Φ, *checks whether* $\tau, S^0 \models \Phi$ *for a periodic trajectory* τ *with parameters* k *and* N, *using* F^τ *as oracle, in the space* $pol(|\Phi| + \log(k+N) + \max\{|S^t| \mid 0 \leq t \leq k+N\})$, *for some polynomial* pol.

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

Lemma 3 *There is an algorithm which for an agents 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* $pol(|\mathcal{A}| + \max\{|S^r| \mid 0 \leq r \leq t\})$ *for some polynomial* pol.

The following lemma shows that trajectories of the agents systems are periodic and provides some bounds on the parameters of these trajectories.

Lemma 4 *For any agents system* $\mathcal{A}$ *and initial state* S^0, *the trajectory* $\tau(\mathcal{A}, S^0)$ *is periodic with parameters* $k(\mathcal{A}, S^0)$ *and* $N(\mathcal{A}, S^0)$. *If* $\mathcal{A}$ *is ground, then* $k(\mathcal{A}, S^0) + N(\mathcal{A}, S^0) \leq 2^{|\mathcal{A}|+|S^0|}$. *In the general case,* $k(\mathcal{A}, S^0) + N(\mathcal{A}, S^0) \leq 2^{2^{|\mathcal{A}|+|S^0|}}$.

From Lemmas 2, 3 and 4 we obtain upper complexity bounds of verification of FLTL expressible properties of agents systems behavior.

Proposition 2 *Let an agents system $\mathcal{A}$ and initial state S^0 be given, and $k = k(\mathcal{A}, S^0)$, $N = N(\mathcal{A}, S^0)$. Then the model checking for a FLTL formula Φ over the trajectory $\tau(\mathcal{A}, S^0)$ can be executed within the space complexity $pol(|\Phi| + \log(k+N)) + |\mathcal{A}| + \max\{|S^t| \mid 0 \leq t \leq k+N\}$).*

4 Complexity of Deciding Behavior Properties

In this section, we present some complexity results characterizing verification of FLTL-properties of trajectories of agents systems. The following theorem estimates this complexity in the case where ground agents can only accumulate information in their internal data bases. Its statement 2) shows that in the case where the agents exchange a bounded number of messages, the problem is tractable.

Theorem 1 *(The case of ground programs and no deletions). Let the agent programs in $\mathcal{A}$ be ground, and there be no deletions connected with agents' actions. Then*
1) AGENT_BEHAVIOR problem is PSPACE-complete.
2) Under the additional conditions that $\mathbf{P}_m$ and the property Φ are propositional, the size of $\mathbf{P}_m$ is bounded by a constant, and all integrity constraints are polynomial time computable, the AGENT_BEHAVIOR problem is decidable in polynomial time.

The next theorem shows that in the ground case the deletions do not increase the complexity.

Theorem 2 *(The case of ground programs). If all the agent programs in $\mathcal{A}$ are ground, then AGENT_BEHAVIOR problem is PSPACE-complete.*

Theorem 3 formulated below shows that verification of the agents systems properties we consider is untractable (exponential space complete) in the case of unlimited arity of predicates. But in the rather realistic case of bounded arity and bounded number of messages, its complexity decreases to polynomial space.

Theorem 3 *(The case of non-ground programs).*
1) AGENT_BEHAVIOR problem is EXPSPACE-complete in the general case.
2) Under the additional condition that the arities of all predicates in the signatures $\mathbf{P}_e$ and $\mathbf{P}_m$ are uniformly bounded by a constant, the AGENT_BEHAVIOR problem is PSPACE-complete.

5 Conclusion

We have established some tight complexity bounds for the verification complexity of behavior properties of systems of interacting intelligent agents.

Despite the fact that our model of agents is somewhat simpler than that in [12], our main results can be extended to the general model, since various one-step semantics of agents considered in this book can be computed in polynomial space, as it is shown in [7].

In this paper, we consider only autonomous, synchronous and deterministic agents systems. We intend to study the behavior of nondeterministic, asynchronous and reactive agents systems in the future.

References

[1] Araragi T., Attie P., Keidar I., et al., *On Formal Modeling of Agent Computations.* In NASA Workshop on Formal Approaches to Agent-Based Systems. April, 2000.

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

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

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

[5] 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", Lect. Notes in CS, N 1181, 1996, 420-432.

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

[7] Eiter T., Subrahmanian V.S. Heterogeneous Active Agents, II: Algorithms and Complexity. Artificial Intelligence, 108(1-2), 1999, 257-307.

[8] Emerson E.A. Temporal and modal logic. In "Handbook of Theor. Comput. Sci.", Ed. J. van Leeuwen, Elsewier Sci. Publishers, 1990.

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

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

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

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