ТРУДЫ КОНФЕРЕНЦИИ,
ПОСВЯЩЕННОЙ 90-ЛЕТИЮ СО ДНЯ РОЖДЕНИЯ
Алексея Андреевича Ляпунова
Новосибирск, 8-12 октября 2001 г.

О сложности поведения систем взаимодействующих агентов *

Валиев М.К.
Институт прикладной математики им. М.В.Келдыша РАН
Дехтярь М.И.
Тверской государственный университет
Диковский А.Я.
Институт прикладной математики им. М.В.Келдыша,
Нантский университет

Аннотация

Рассматривается сложность распознавания свойств поведения систем взаимодействующих агентов, выражаемых в языке временной логики предикатов. Устанавливаются некоторые точные оценки сложности распознавания таких свойств при некоторых ограничениях на классы агентных программ.

The complexity is investig ted, of the properties of inter cting intelligent gents' beh vior, expressible in the first order logic of line r time. Some tight complexity bounds for such properties re est blished under cert in restrictions on the cl sses of gent progr ms.

1 Введение

При построении сложных программных систем все больше использ ются технологии, основанные на понятии программных агентов. Агенты использ ются при построении систем иск сственного интеллекта, работы в Интернете и т.д. Имеются различные подходы к архитект ре и построению агентов. Обзор многих из этих подходов содержится в книге В.С браманяна и др. [12]. Эта книга также содержит подробное рассмотрение некоторого теоретического подхода к определению и исследованию вычислений, основанных на использовании агентов, и некоторых методов их построения. Этот подход отличается от

*Эта работа выполнена при поддержке РФФИ (гранты 01-01-00278 и 00-01-00254)

многих др гих подходов своей ориентацией на логическое описание агентов. В нашей статье мы в основном след ем понятиям из [12], однако в связи с ограничениями на объем, использ ем их в несколько прощенном виде.

Многие важные теоретические проблемы, связанные с программными агентами, же были предметом интенсивных исследований, но, тем не менее, все еще остается ряд не менее с щественных проблем, не привлекших пока достаточного внимания исследователей. Одна из таких проблем - это сложностной анализ поведения систем взаимодейств ющих агентов. Подобные проблемы активно из чались для параллельных программ (см., например, [8, 10]). Рез льтаты этих исследований неприменимы непосредственно к системам агентов, рассматриваемым в данной работе, которые ближе с семантической точки зрения к динамическим дед ктивным базам данных (ДДБД). Сложностные проблемы для ДДБД рассматривались нами в [4, 5, 6], и наш подход здесь к анализ поведения систем агентов близок к подход , использованном в этих работах.

Др гой логический подход к описанию систем агентов содержится в [1], где представлены три различные модели: так называемые I/O-автоматы, среда программирования Er ös, основанная на понятиях из теории представления знаний, и некоторое расширение π-исчисления Милнера. В среде Er ös поведение систем агентов специфир ется в пропозициональной временной логике CTL, и эти спецификации верифицир ются с помощью проверки истинности форм л (mo el checking) на моделях CTL. Среди работ др гих авторов этот подход кажется наиболее близким к подход , использ емом в данной работе.

Еще один логический подход к системам агентов связан с языком Concurrent Metatem [2, 9]. В этом сл чае агенты описываются с помощью форм л линейной временной логики, и для верификации динамических свойств систем агентов использ ется метод резолюции, адаптированный к временной логике.

В настоящей работе мы рассматриваем поведение системы $\mathcal{A}$ из n агентов, общающихся межд собой с помощью сообщений. На каждом шаге работы каждый агент выполняет некоторое подмножество действий в зависимости от своего вн треннего состояния и пол ченных сообщений. В рез льтате этих действий он изменяет свое вн треннее состояние и посылает сообщения др гим агентам. Говоря неформально, архитект ра агента включает в себя (i) *внутренние структуры данных (базу данных)*, определяющие вн треннее состояние агента, (ii) а *почтовый ящик*, содержащий сообщения от др гих агентов, (iii) *ограничения целостности* на состояния агента, (iv) *базу действий*, которые агент может выполнить, и (v) *программу*, определяющ ю политик выполнения действий. Выполнение действия состоит из (a) изменения тек щего состояния гента и (b) посылки сообщений др гим агентам. Тек щее содержание почтового ящика состоит из сообщений, пол ченных агентом от др гих агентов на предыд щем шаге. Глобальное состояние системы состоит из вн тренних состояний и содержимых почтовых ящиков всех агентов системы.

Пошаговая семантика системы $\mathcal{A}$ описывается как детерминированный оператор переходов над глобальными состояниями. Таким образом, для каждого начального глобального состояния S поведение системы $\mathcal{A}$ задается в терминах единственной траектории, определяемой последовательностью переходов глобальных состояний, начиная с S.

В настоящей работе мы из чаем сложность алгоритмических проблем, связанных с анализом поведения систем агентов. Одна из наиболее важных проблем такого типа - это проблема инвариантности (safety problem), т.е. сохранения некоторого свойства начального состояния S на всех состояниях траектории, начинающейся с S. Др гая интересная проблема - это проблема достижимости цели, т.е. выяснения того, попадет ли система в состояние, довлетворяющее заданном свойств . Для систематического рассмотрения

динамических свойств систем агентов в качестве языка спецификации мы использ ем некотор ю версию первопорядковой логики линейного времени (FLTL) без ф нкциональных символов. Подобные логики широко использ ются как средство формальной спецификации и верификации параллельных программ (см. [8, 10]).

Общая алгоритмическая проблема ($AGENT\_BEHAVIOR$), котор ю мы рассматриваем, форм лир ется след ющим образом: для данной системы $\mathcal{A}$, ее начального глобального состояния S и FLTL-форм лы Φ выяснить, выполняется ли Φ на траектории системы $\mathcal{A}$, начинающейся с S. Для этой проблемы мы станавливаем некоторые точные оценки сложности при некоторых ограничениях на классы программ-агентов.

Эта проблема представляет собой вариант хорошо известной проблемы проверки на моделях, широко использ емой при верификации параллельных программ (см. [8, 3]). Обычно проверка на моделях выполняется в два этапа. Вначале параллельная система C преобраз ется в недедетерминированный автомат (transition system) T, и после этого фактическая проверка на моделях применяется к T. Размер T может с щественно (в общем сл чае экспоненциально) вырасти относительно размера C. В настоящей работе мы показываем, что в ряде сл чаев вместо явного полного построения T можно ограничиться ее построением по частям, что позволяет с щественно понизить верхние оценки сложности проблемы $AGENT\_BEHAVIOR$ по сравнению с обычными подходами к проверке на моделях.

2 Взаимодействующие интеллектуальные агенты

Определение понятия агента, использ емое в настоящей работе, в основном след ет работе [12]. Конечно, архитект ра IMPACT для систем агентов, введенная в этой книге, слишком сложна для представления в короткой статье. Кроме того, некоторые ее особенности не имеют значения при анализе сложности поведения этих систем. Поэтом мы представляем здесь несколько прощенное по сравнению с [12] определение агентов. В частности, мы не рассматриваем средств, связанных с включением в архитект р обычных программ (legacy co e), средств обеспечения безопасности, метазнания и расс ждений в словиях неопределенности. В качестве вн тренних стр кт р агентов мы использ ем реляционные базы данных (в архитект ре IMPACT доп скаются более общие стр кт ры) и для описания политики действий агентов использ ем обычные логические программы (в IMPACT программы агентов мог т включать деонтические модальности "разрешенное действие", "обязательное действие" и т.д.).

Система агентов в настоящей статье - это система $\mathcal{A}$, состоящая из n взаимодейств - ющих агентов $a_1, ..., a_n$. Формальное определение $\mathcal{A}$ включает три конечные предикатные сигнат ры: экстенсиональная сигнат ра $\mathbf{P}_e$, использ емая для представления фактов вн тренней базы данных, сигнат ра сообщений $\mathbf{P}_m$ и интенсиональная сигнат ра $\mathbf{P}_{act}$, использ емая для представления действий агентов. Кроме этого, с $\mathcal{A}$ связаны конечные множества переменных $\mathbf{V}$ и констант $\mathbf{C}$. Через $\mathbf{A}_e, \mathbf{A}_m$ и $\mathbf{A}_{act}$ мы обозначаем множества атомов над соответств ющими предикатными сигнат рами, $\mathbf{V}$ и $\mathbf{C}$. Через $\mathbf{B}_e, \mathbf{B}_m$ и $\mathbf{B}_{act}$. Также мы обозначаем соответств ющие множества базисных атомов, через $\mathbf{LB}_e, \mathbf{LB}_m$ и $\mathbf{LB}_{act}$ соответств ющие множества базисных литералов.

Структуры данных у агентов. Каждый агент $a \in \mathcal{A}$ обладает собственными вн - тренними стр кт рами данных. Мы считаем, что эти данные имеют вид некоторой реляционной базы данных со схемой $\mathbf{P}_e$. Тек щее состояние обозначается I_a.

Сообщения. Агенты из $\mathcal{A}$ общаются с помощью *сообщений*. Сообщения - это базисные факты (атомы) в сигнат ре $\mathbf{P}_m$. Каждый агент a имеет б фер (почтовый ящик) $MsgBox_a$, содержащий сообщения, пол чаемые им от др гих агентов. Элементы в $MsgBox_a$ – это пары вида $(Source\_agent, Message)$, где $Source\_agent$ - имя агента, пославшего сообщение. Пар $(I_a, MsgBox_a)$ мы называем *локальным состоянием* агента a.

Ограничения целостности. С каждым агентом a ассоциировано некоторое множество *ограничений целостности*, задающее некоторое свойство состояний его базы данных. Ограничения целостности мог т быть представлены форм лами, логическими программами или обычными программами, распознающими это свойство. Мы предполагаем, что с - ществ ет алгоритм, проверяющий ограничение целостности IC_a в состоянии I_a, использ я полиномиальный относительно $|I_a| + |IC_a|$ объем памяти.

Действия. С каждым агентом ассоциировано некоторое множество действий. Действие представляется атомом $\alpha(X_1, ..., X_l)$, где $\alpha^{(l)}$ - предикат в сигнат ре $\mathbf{P}_{act}$, X_i - различные переменные. и тремя (возможно, п стыми) множествами атомов с переменными из $\{X_1, ..., X_l\}$:
$ADD(\alpha)$ – список фактов из $\mathbf{A}_e$, добавляемых к I_a;
$DEL(\alpha)$ – список фактов из $\mathbf{A}_e$, даляемых из I_a;
$SEND(\alpha)$ – список пар вида $(destination\_agent, Msg)$, где $Msg \in \mathbf{B}_m$.

Параллельное выполнение действий. П сть $AS = \{\alpha_1(\bar{T}_1), ..., \alpha_k(\bar{T}_k)\} \subseteq \mathbf{B}_{act}$ – множество базисных действий агента a и $\bar{\sigma} = (\sigma_1, ..., \sigma_k)$– список базисных подстановок таких, что $\sigma_j(\alpha_j(\bar{X}_j)) = \alpha_j(\bar{T}_j)$. Параллельное выполнение множества действий $\bar{\sigma}(AS)$ в локальном состоянии $(I_a, MsgBox_a)$ определяется след ющим образом:
a) новое вн треннее состояние a пол чается из тек щего состояния присваиванием

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

(Таким образом, когда некоторый факт должен быть дален и добавлен, он добавляется. Конечно, др гие стратегии разрешения конфликтов тоже возможны, например, когда даление и добавление взаимно аннигилир ют);
b) любое сообщение, порождаемое множеством $\bar{\sigma}(AS)$ помещается в б фер $MsgBox$ агента назначения Более точно, для любого агента $b \neq a$

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

Программа агента Программа P_a определяет политик выполнения действий агентом a и представляет собой логическ ю программ с предложениями вида

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

где голова H – (интенсиональный) атом действия из $\mathbf{A}_{act}$, а литералы L_i – либо атомы из $\mathbf{A}_{act}$, либо (экстенсиональные) литералы из $\mathbf{LB}_e$, либо литералы сообщений вида $Received(Source\_agent, Message)$ или $\neg Received(Source\_agent, Message)$, где $Message \in \mathbf{A}_m$.

Семантика программ определяет множество действий, выполняемых агентом в его тек щем локальном состоянии. Семантика $Sem(P_a)(I_a, MsgBox_a)$ программы P_a относительно состояния I_a и состояния б фера $MsgBox_a$ определяется как минимальная эрбранова модель программы $P_a \cup I_a \cup \{Received(Agent\_source, Message) \mid (Agent\_source,$

$Message) \in MsgBox_a\}$.

Очевидно, рассматриваемые здесь логические программы являются нормальными программами без рек рсии через отрицание, т.е. очень специальным сл чаем стратифицир - емых логических программ. Как хорошо известно, любая такая программа имеет единственн ю минимальн ю модель, которая может быть вычислена за полиномиальное время, использ я прост ю процед р вычисления неподвижной точки. Фактически, приводимые в этой работе сложностные рез льтаты верны для любых подклассов нормальных логических программ, для которых с ществ ет единственная минимальная модель, вычислимая в полиномиальное время, например, для стратифицир емых логических программ.

Глобальное состояние системы агентов $\mathcal{A}$ в момент t определяется как n-ка локальных состояний агентов $a_1, ..., a_n$, т.е. $S^t =< (I^t_{a_1}, MsgBox^t_{a_1}), ..., (I^t_{a_n}, MsgBox^t_{a_n}) >$.

Детерминированная одношаговая семантика системы агентов $\mathcal{A}$ определяется как оператор $\Rightarrow$ переходов глобальных состояний системы, который мы представляем в виде след ющего алгоритма, преобраз ющего глобальное состояние S^t в след ющее глобальное состояние S^{t+1} :

(1) ДЛЯ ЛЮБОГО $a_i \in \mathcal{A}$ ВЫПОЛНИ $Acts^t_i := Sem(P_{a_i})(I^t_{a_i}, MsgBox^t_{a_i})$;
(2) ДЛЯ ЛЮБОГО $a_i \in \mathcal{A}$ ВЫПОЛНИ $MsgBox^{t+1}_{a_i} := \emptyset$;
(3) ДЛЯ ЛЮБОГО $a_i \in \mathcal{A}$ ВЫПОЛНИ
ЕСЛИ IC_{a_i} не нар шено выполнением действий $Acts^t_i$
ТО выполни $Acts^t_i$ параллельно в локальном состоянии $(I^t_{a_i}, MsgBox^{t+1}_{a_i})$.

Таким образом, в итоге содержание почтового ящика агента a в момент $t+1$ состоит из всех сообщений, посланных агент a др гими агентами в момент t.

Поведение системы агентов $\mathcal{A}$ для начального глобального состояния $S^0 =$
$< (I^0_{a_1}, MsgBox^0_{a_1}), ..., (I^0_{a_n}, MsgBox^0_{a_n}) >$ описывается как след ющая бесконечная последовательность глобальных состояний, $\tau = \tau(\mathcal{A}, S^0)$:
$< (I^0_{a_1}, MsgBox^0_{a_1}), ..., (I^0_{a_n}, MsgBox^0_{a_n}) > \Rightarrow ...$
$< (I^t_{a_1}, MsgBox^t_{a_1}), ..., (I^t_{a_n}, MsgBox^t_{a_n}) > \Rightarrow$
$< (I^{t+1}_{a_1}, MsgBox^{t+1}_{a_1}), ..., (I^{t+1}_{a_n}, MsgBox^{t+1}_{a_n}) > \Rightarrow ...$
где глобальное состояние на шаге $t+1$ пол чается из предыд щего состояния применением оператора перехода. Эт последовательность мы называем *траекторией* системы $\mathcal{A}$ в состоянии S^0.

Это определение гарантир ет локальн ю совместность поведения системы агентов.

Лемма 1 *Пусть τ – траектория системы $\mathcal{A}$ в состоянии S^0. Тогда, если для некоторого $j, 1 \le j \le n$, начальное локальное состояние $I^0_{a_j}$ агента a_j удовлетворяет ограничению IC_{a_j}, то состояние $I^t_{a_j}$ также удовлетворяет ограничению IC_{a_j} для любого $t > 0$.*

Пример. Рассмотрим след ющ ю игр шечн ю систем распределения рес рсов $\mathcal{A}$, состоящ ю из агента-менеджера m, который владеет некоторыми рес рсами и распределяет их среди четырех агентов-пользователей u_1, u_2, u_3 и u_4. У каждого из пользователей имеется собственная стратегия запроса рес рсов:

1) u_1 запрашивает рес рс в первый момент и затем всякий раз повторяет свой заказ после того, как б дет выполнен предыд щий;
2) u_2 запрашивает рес рс непосредственно после того, как запрашивает u_1 ;
3) u_3 запрашивает рес рс непосредственно после того, как u_1 пол чает рес рс ;
4) u_4 запрашивает рес рс в каждый момент времени.

Менеджер m ведет список заказов и в каждый момент времени выполняет первый заказ

из этого списка. Только один заказ каждого пользователя может прис тствовать в списке заказов m. Поэтом , если m пол чает заказ некоторого пользователя до того, как был выполнен его предыд щий заказ, то этот новый заказ игнорир ется.

Система $\mathcal{A}$ задается след ющим образом. Состояние I_{u_1} агента u_1 может содержать факт put_order, а состояния I_{u_i} $(i = 2, 3, 4)$ остальных пользователей всегда п сты. База данных I_m агента m содержит факты вида:
1) $order(X, I)$ – $order(u_i, j)$ означает, что заказ агента u_i находится на j-ом месте в списке заказов m,
2) $is(X)$ – заказ пользователя X содержится в списке m,
3) $num\_orders(I)$ – число невыполненных заказов равно I.

Чтобы сообщить m и др гим пользователям о своем заказе u_i посылает им сообщение $order$. Когда m выполняет заказ u_i, он посылает u_i сообщение ok. Когда u_1 хочет информировать u_3 об очередном пол чении рес рса от m, он посылает u_3 сообщение ok.
Агент u_1. Действия:
$do\_order: \ ADD = \{put\_order\}, SEND = \{(m, order), (u_2, order)\}$;
$receive\_order: \ DEL = \{put\_order\}, SEND = \{(u_3, ok)\}$;
P_{u_1} :
$do\_order \leftarrow \neg put\_order$
$receive\_order \leftarrow Received(m, ok)$
Агент u_2. Действия:
$do\_order: \ SEND = \{(m, order)\}$;
P_{u_2} :
$do\_order \leftarrow Received(u_1, order)$

Агент u_3. Действия:
$do\_order: \ SEND = \{(m, order)\}$;
P_{u_3} :
$do\_order \leftarrow Received(u_1, ok)$
Агент u_4. Действия:
$do\_order: \ SEND = \{(m, order)\}$;
P_{u_4} :
$do\_order \leftarrow .$
Агент m. Действия:
$ins\_order(X, I): \ ADD = \{order(X, I), is(X)\}$;
$fulfill\_order(X): \ DEL = \{order(X, 1), is(X)\}, SEND = \{(X, ok)\}$;
$move(X, I): \ ADD = \{order(X, I)\}, DEL = \{order(X, I + 1)\}$;
$new\_num(I, J) : ADD = \{num\_orders(J)\}, DEL = \{num\_orders(I)\}$
P_m :
$new\_order(X) \leftarrow Received(X, order), \neg is(X)$
$first\_place(I) \leftarrow num\_orders(I), I > 0$
$first\_place(1) \leftarrow num\_orders(0)$
$ins\_order(X, I) \leftarrow new\_order(X), first\_place(I) \quad (X \in \{u_1, u_2\})$
$ins\_order(u_3, I) \leftarrow new\_order(u_3), ins\_order(X, I - 1) \quad (X \in \{u_1, u_2\}$
$ins\_order(u_3, I) \leftarrow new\_order(u_3), first\_place(I), \neg ins\_order(u_1, I),$
$\neg ins\_order(u_2, I)$
$ins\_order(u_4, I) \leftarrow new\_order(u_4), ins\_order(u_3, I - 1)$
$ins\_order(u_4, I) \leftarrow new\_order(u_3), \neg ins\_order(u_3, I - 1),$

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

Вначале БД менеджера m содержит только фактt $num\_orders(0)$. Затем на каждом шаге предикат $new\_order(X)$ проверяет, н жно ли включать в список заказ агента X, Предикат $first\_place(I)$ определяет то место I в списке, на которое должен быть помещен очередной заказ, ins_order вставляет новые заказы в конец списка (при конк ренции использ ется порядок: $u_1 < u_2 < u_3 < u_4$), $fulfill\_order$ посылает рес рс первом агент из списка, $move$ сдвигает все элементы списка на l "влево", а $new\_num(I, J)$ определяет новое значение I для $num\_orders(I)$, добавляя к прежнем число K новых незарегистрированных ранее сообщений из $MessageBox_m$. Это число K вычисляется не определяемым здесь предикатом num_new_orders.

3 Свойства поведения агентов и временна́я логика

Мы исслед ем сложность алгоритмических проблем, связанных с анализом поведения систем агентов. В качестве средства для спецификации свойств поведения мы выбираем линейн ю временн ю логик первого порядка (FLTL), которая является прощенной версией логики, использ емой, например, в работах [10, 8].

Язык FLTL определяется как замыкание множества форм л состояний (первого порядка) относительно б левых связок и временных операторов "пока" ("until", U) и "в след ющий момент'("nexttime", $\circ$). Сигнат ра форм л состояний включает предикат равенства и множество $\bigcup_1^n \mathbf{P}_e^i \cup \{Received^{(3)}\}$, где $Received^{(3)}$ – это имя предиката $Received(Source\_agent,\ Target\_agent, Message)$, а $\mathbf{P}_e^i$ обозначает множество символов предикатов экстенсиональной сигнат ры $\mathcal{A}$, проиндексированной номерами агентов i. Термами этого языка являются либо переменные, либо константы из множества $\mathbf{D}$. Мы предполагаем, что $\mathbf{D}$ включает все константы, встречающиеся в программах агентов, их состояниях и почтовых ящиках.

Моделями FLTL сл жат траектории систем агентов. Истинность форм лы в модели определяется поточечно, т.е. на каждом глобальном состоянии траектории.

П сть $\tau = \tau(\mathcal{A}, S^0)$ – некоторая траектория $\mathcal{A}$, $S^t =< (I^t_{a_1}, MsgBox^t_{a_1}), ...,$
$(I^t_{a_n}, MsgBox^t_{a_n}) >$ – некоторое глобальное состояние траектории τ, p^i – k-местный предикатный символ из $\mathbf{P}_e^i$ и $c_1, ...c_k$ – некоторые константы из $\mathbf{D}$. Тогда базисный атом $p^i(c_1, ..., c_k)$ истинен в $< \tau, S^t >$ (обозначение: $\tau, S^t \models p^i(c_1, ..., c_k)$) тогда и только тогда, когда $p^i(c_1, ..., c_k) \in I^t_{a_i}$.

П сть m – это k-местный предикатный символ из $\mathbf{P}_m$. Тогда атом $Received(a_i, a_j,$ $m(c_1, ..., c_k))$ истинен в $< \tau, S >$ тогда и только тогда, когда $(a_i, m(c_1, ..., c_k)) \in MsgBox^t_{a_j}$. Семантика б левых связок, кванторов по объектным переменным и временных операторов определяется обычным образом. Например, $\tau, S^t \models \Phi U \Psi$, $\exists t' \geq t$ такое, что $\tau, S^t \models \Psi$ и для всех $t''(t \leq t'' \leq t')$, $\tau, S^{t''} \models \Phi$. Ниже в примере мы б дем также использовать обычные временные операторы $\Box$ ("всегда в б д щем") и $\Diamond$ ("когда-ниб дь в б д щем"),

которые выразимы в этой логике. Временная логика FLTL, котор ю мы рассматриваем в этой работе, отличается от обычных пропозициональных временных логик тем, что в ней использ ются базисные форм лы состояний первого порядка вместо пропозициональных символов. Это может, конечно, приводить к величению сложности проверки истинности форм л. Хорошо известно, например, что проблема проверки истинности замкн той форм лы первого порядка на модели является PSPACE-полной. Поэтом имеет смысл выделить в FLTL естественный специальный сл чай, когда базисные форм лы состояний не содержат объектных переменных. При таком ограничении мы, фактически, пол чаем логик , эквивалентн ю пропозициональной линейной временной логике LTL, для которой б дем использовать такое же обозначение.

В общем виде проблема $AGENT\_BEHAVIOR$ проверки свойств поведения системы агентов, котор ю мы рассматриваем в этой работе, форм лир ется след ющим образом: для заданной системы агентов $\mathcal{A}$, ее начального глобального состояния S^0 и FLTL-форм лы Φ выяснить верно ли, что $\tau(\mathcal{A}, S^0), S^0 \models \Phi$.

В частности, выше казанные проблемы безопасности и достижимости цели являются частными сл чаями $AGENT\_BEHAVIOR$, в которых Φ имеет форм $\Box\Psi$ (соответственно, $\Diamond\Psi$) для соответств ющей форм лы состояний Ψ. Мы рассматриваем также некоторые др гие подклассы общей проблемы $AGENT\_BEHAVIOR$, которые пол чаются при некоторых ограничениях, накладываемых на программы агентов (например, отс тствие переменных), их действия (отс тствие далений), сигнат ры (ограниченность количества и арности предикатов).

Пример (продолжение). Для определенной выше системы $\mathcal{A}$ можно становить истсн-

задаваемых периодическими траекториями.

Лемма 2 *Существует алгоритм, который по заданным числам k и N и FLTL-формуле Φ проверяет условие $\tau, S^0 \models \Phi$ для периодической траектории τ с параметрами k и N, используя F^τ в качестве оракула, с емкостью памяти $pol(|\Phi|+\log(k+N)+\max\{|S^t| \mid 0 \leq t \leq k+N\})$ для некоторого полинома pol.*

Оракул F^τ из этой леммы можно эффективно вычислять для траекторий τ, порождаемых системами агентов.

Лемма 3 *Существует алгоритм, который по системе агентов $\mathcal{A}$, начальному состоянию S^0 и моменту времени $t \geq 0$ вычисляет состояние S^t траектории $\tau(\mathcal{A}, S^0)$ с емкостью памяти $pol(|\mathcal{A}| + \max\{|S^r| \mid 0 \leq r \leq t\})$ для некоторого полинома pol.*

Следующая лемма показывает, что траектории систем агентов являются периодическими и дает оценки параметров этих траекторий.

Лемма 4 *Для всякой системы агентов $\mathcal{A}$ и ее начального состояния S^0 траектория $\tau(\mathcal{A}, S^0)$ является периодической с параметрами $k(\mathcal{A}, S^0)$ и $N(\mathcal{A}, S^0)$. Если $\mathcal{A}$ базисная, то $k(\mathcal{A}, S^0) + N(\mathcal{A}, S^0) \leq 2^{|\mathcal{A}|+|S^0|}$. В общем случае $k(\mathcal{A}, S^0) + N(\mathcal{A}, S^0) \leq 2^{2^{|\mathcal{A}|+|S^0|}}$.*

Из лемм 2, 3 и 4 можно извлечь верхние оценки сложности проверки свойств поведения систем агентов, выразимых в FLTL.

Предложение 1 *Пусть для системы агентов $\mathcal{A}$ и ее начального состояния S^0 $k = k(\mathcal{A}, S^0)$ и $N = N(\mathcal{A}, S^0)$. Тогда существует алгоритм, проверяющий истинность произвольной FLTL-формулы Φ на траектории $\tau(\mathcal{A}, S^0)$ с емкостью памяти $pol(|\Phi|+\log(k+N)) + |\mathcal{A}| + \max\{|S^t| \mid 0 \leq t \leq k+N\})$ для некоторого полинома pol.*

4 Сложность разрешения свойств поведения

В этом разделе мы формулируем ряд результатов, характеризующих сложность проверки FLTL-свойств на траекториях, задаваемых системами агентов. Эта сложность существенно зависит от типа рассматриваемой системы. Выделим некоторые естественные подклассы систем. Система агентов $\mathcal{A}$ является
- *базисной*, если программы ее агентов не содержат переменных;
- *расширяющей*, если в действиях ее агентов не допускаются удаления из баз данных;
- *позитивной*, если в программах ее агентов нет отрицаний;
- *m-агентной*, если число ее агентов не превосходит m;
- *k-сигнальной*, если сигнатура сообщений $\mathbf{P}_m$ содержит не более k пропозициональных символов (*сигналов*).

Укажем вначале два случая, в которых проблема AGENT_BEHAVIOR допускает решение за полиномиальное время. В обоих случаях проверяемые свойства принадлежат LTL, а входящие в системы базисные агенты могут лишь аккумулировать информацию в своих внутренних базах данных.

Теорема 1 *Проблема AGENT_BEHAVIOR разрешима за полиномиальное время для выразимых в LTL свойств*

1) в классе базисных, расширяющих, позитивных систем агентов;

2) в классе базисных, расширяющих, m-агентных, k-сигнальных систем агентов $\mathcal{A}$ таких, что $m * k = O(\log |\mathcal{A}|)$.

Если ослабить ограничения в классе из п нкта (2) этой теоремы, то временная сложность величивается, но треб емая память остается линейной.

Теорема 2 *Проблема AGENT_BEHAVIOR является LIN-SPACE-полной (относительно сводимости за полиномиальное время и с линейным растяжением) для свойств, выразимых в LTL,*

1) в классе базисных, расширяющих, m-агентных систем агентов (для каждого фиксированного m);

2) в классе базисных, расширяющих, k-сигнальных систем агентов (для каждого фиксированного k).

След ющая теорема показывает, что в базисном сл чае использование далений не величивает сложность.

Теорема 3 *В классе базисных, расширяющих систем агентов, как и в классе базисных систем агентов, проблема AGENT_BEHAVIOR является PSPACE-полной.*

При наличии переменных в программах агентов сложность проверки свойств поведения, естественно, величивается. Приводимая ниже теорема 4 показывает, что в общем сл чае эта проверка треб ет экспоненциальной памяти. Вместе с тем, при вполне реалистичном ограничении арностей предикатов для нее достаточно полиномиальной памяти.

Теорема 4 *1) В общем случае проблема AGENT_BEHAVIOR является EXPSPACE-полной.*

2) В классе расширяющих, позитивных систем агентов проблема AGENT_BEHAVIOR является EXPTIME-полной.

3) В случае, когда арности всех предикатов в сигнатурах $\mathbf{P}_e$ *и* $\mathbf{P}_m$ *ограничены некоторой (произвольной) константой, проблема AGENT_BEHAVIOR является PSPACE-полной .*

5 Заключение

Мы становили в этой работе некоторые достаточно точные границы сложности для роблемы верификации свойств поведения систем интерактивных интеллект альных агентов.

Хотя использ емая нами модель агентов несколько проще, чем модель предложенная в [12], наши основные рез льтаты мог т быть легко распространены и на эт более общ ю модель, поскольк различные семантики для выполнения одного шага, предложенные в этой монографии, мог т быть вычислены с использованием памяти полиномиального размера (см. [7]).

В этой работе рассмотрен класс автономных, синхронных и детерминированных систем агентов. В б д щем мы предполагаем из чить также поведение недетерминированных, асинхронных и реактивных систем агентов.

Список литературы

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

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

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

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

[5] *Dekhtyar M.I., Dikovsky A.Ja.* On Homeostatic Behavior of Dynamic De uctive Data Bases. In: Proc. 2n 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 an Complexity. Artificial Intelligence, 108(1-2), 1999, 257-307.

[8] *Emerson E.A.* Temporal an mo al logic. In "Han book of Theor. Comput. Sci.", E . 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 an 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.