ПРОГРАММИРОВАНИЕ, 2009, № 4, с. 1–20

УДК 004.92+004.94

СИСТЕМЫ АГЕНТОВ, УПРАВЛЯЕМЫХ ЛОГИЧЕСКИМИ ПРОГРАММАМИ: СЛОЖНОСТЬ ВЕРИФИКАЦИИ

© **2009 г. М. К. Валиев*, М. И. Дехтярь**, А. Я. Диковский*****

**Институт прикладной математики им. М.В. Келдыша РАН*
125047 Москва, Миусская пл., 4
***Тверской государственный университет*
170013 Тверь, ул. Желябова, 3
****Нантский университет*
?
E-mail: valiev@spp.keldysh.ru, Michael.Dekhtyar@tversu.ru,
Alexandre.Dikovsky@irin.univ-nantes.fr

Поступила в редакцию

В статье рассматривается сложность проблемы верификации поведения (динамических свойств) систем взаимодействующих интеллектуальных агентов. Она продолжает наши публикации [1–3], в которых эта проблема в основном обсуждалась для детерминированных и недетерминированных систем, и больше концентрируется на асинхронных системах.

1. ВВЕДЕНИЕ

Понятия интеллектуального агента и системы взаимодействующих агентов (короче, мультиагентной системы, МА-системы или МАС) были введены более двух десятков лет назад, и их изучение и использование представляет собой одно из самых активно развивающихся направлений в искусственном интеллекте и прикладном программировании (см., например, [4–10]). Одним из существенных факторов, способствовавших проявлению такого интереса к агентам, явилось развитие сетевых технологий (в частности, связанных с Интернетом).

Понятие агента существенно обобщает понятие объекта из области объектно-ориентированного программирования за счет введения интеллектуальных компонент, определяющих политику выполнения действий агентом в зависимости от его состояния и сообщений, получаемых от других агентов системы и внешней среды. При этом состояние агента может иметь достаточно сложную структуру, например, включать в себя некоторую базу данных. Все это делает поведение таких систем достаточно сложным и приводит к необходимости проверки соответствия их поведения требуемым условиям (верификации их динамических свойств). Конечно, в общем случае такая проблема неразрешима, поэтому мы рассматриваем эту проблему (мы называем ее MA-BEHAVIOR) при разных ограничениях на вид агентов и класс проверяемых свойств. Заметим, что, хотя понятия агентов и МАС активно изучались в течение ряда лет, работы по верификации их поведения были начаты сравнительно недавно [1, 11–15].

Мультиагентные системы имеют широкую сферу применений, включающую такие разные области, как управление бизнес-процессами, электронная торговля, Интернет-навигация, социальные и военные приложения и т.д. Такое разнообразие приложений привело к появлению многочисленных различных подходов к определению понятий агента и МАС (использующих, к тому же, разные уровни абстракции от конкретных систем). При этом разные архитектуры агентов отличаются в основном устройством их интеллектуальных компонент, которые могут использовать деревья решений, логические программы, выводы в различных ви-

дах модальной логики (эпистемической, деонтической и др.) и т.д. Обзоры многих из этих подходов даны в вышеупомянутых ссылках.

Используемое нами определение агента в значительной степени базируется на архитектуре IMPACT, введенной и подробно описанной в [9][1]. Полное определение этой архитектуры весьма громоздко. Оно включает в себя различные довольно сложные механизмы и средства спецификации агентов, что делает их хорошо приспособленными к практическим приложениям. Однако, в то же время, такое обилие различных механизмов сильно усложняет формальное изучение свойств таких агентов. В частности, как показано в [9], семантика агентов архитектуры IMPACT, описанная в терминах переходов состояний, является в общем случае малоэффективной. Для достижения семантикой переходов полиномиальной временной сложности в [17] (см. также [9]) определены некоторые очень сложные ограничения на вид агентов. К сожалению, определение таких агентов становится совсем уж трудно воспринимаемым.

Мы предлагаем некоторые другие достаточно легко формулируемые ограничения на IMPACT-агенты, которые также ведут к семантике полиномиальной сложности. При этом мы фокусируемся на средствах, связанных с определением действий, политики принятия решений и коммуникации агентов. Таким образом, мы отвлекаемся от деталей IMPACT-архитектуры, связанных с агентизацией обычного программного кода, безопасностью, структурами метазнания, временными и вероятностными рассуждениями. Кроме того, в качестве состояний агентов мы рассматриваем базы фактов (аналог реляционных баз данных), в отличие от [9], где допускаются более общие структуры состояний. Заметим, что даже при этих ограничениях архитектура MAC остается достаточно представительной, и большая часть примеров из [9] укладывается в ограниченную таким образом архитектуру.

Агенты могут быть детерминированными или недетерминиронными, а взаимодействие агентов в MAC может быть синхронным или асинхронным. Синхронные системы детерминированных агентов для каждого своего начального состояния определяют некоторую траекторию вычисления. В остальных трех случаях система для начального состояния определяет некоторое дерево возможных траекторий вычисления.

В первом случае для описания динамических свойств системы естественно использовать логику, основанную на линейности времени, а в остальных случаях – те или иные варианты логики ветвящегося времени.

Рассматриваемая нами проблема MA-BEHAVIOR, которая состоит в проверке выполнимости некоторой формулы временной логики на траекториях данной MAC, фактически соответствует той проблеме, которая под названием model checking (проверка на моделях программ) изучается уже ряд лет для абстрактных систем (диаграмм) переходов (см. [18–24]). Однако имеется существенное различие между классической постановкой этого вопроса и рассматриваемым нами вариантом. Традиционно, сложностные результаты устанавливаются для диаграмм с неструктурированными состояниями и явно заданными переходами (или других аналогичных систем типа конечных автоматов или OBDD). В MAC же состояния могут иметь сложную структуру (у нас – это базы фактов), а переходы от одних состояний к другим определяются некоторыми программами. Это позволяет оценивать сложность проблемы верификации, основываясь на различных структурных и семантических свойствах систем.

MAC определяют компактное представление систем переходов. В частности, даже в случае базисных (т.е. не использующих переменных) MAC определяемая ею система переходов может иметь экспоненциальный размер относительно исходной MAC. Поэтому иногда наши нижние оценки выглядят более пессимистичными, чем для классического случая с теми же используемыми логиками. Что касается верхних оценок, иногда они более информативны и точны (в случаях, когда получены детерминированные или недетерминированные полиномиальные верхние оценки), или они получаются простой трансляцией классических оценок, принимая в расчет изменения размера при переходе от MAC к определяемой ею системе переходов.

[1] Заметим, что рассматривемые здесь агенты существенно отличаются от агентов в смысле [16].

Дальнейшая часть работы организована следующим образом. Раздел 2 содержит основные определения. В разделе 3 приводится некоторый пример, иллюстрирующий возможности рассматриваемых нами МАС. Раздел 4 содержит описание алгоритма верификации детерминированных МАС из [2], который используется в разделе 6 для построения недетерминированного алгоритма, решающего проблему верификации для некоторого класса асинхронных МАС за полиномиальное время. В разделе 5 дается обзор результатов из [2, 3], касающихся сложности верификации различных классов детерминированных и недетерминированных МАС (в виде таблиц 1 и 2). Раздел 6 содержит, кроме упомянутого выше результата, еще результаты о взаимном сведении проблемы MA-BEHAVIOR для недетерминированных и асинхронных МАС. Заключение содержит краткое резюме полученных результатов и некоторое обсуждение проблематики.

2. АГЕНТЫ И МНОГОАГЕНТНЫЕ СИСТЕМЫ

МА-система $\mathcal{A} = \{a_1, \ldots, a_n; P\}$ состоит из конечного множества $\{a_1, \ldots, a_n\}$ интеллектуальных агентов с общей предикатной сигнатурой и специального *почтового агента* P, моделирующего сеть связей между агентами a_i. У интеллектуального агента A имеется *внутренняя база данных (БД)* I_A, содержащая конечное множество базисных атомов (т.е. выражений вида $p(c_1, \ldots, c_k)$, где p – предикатный символ, $c_1, \ldots, c_k$ – константы, причем множество используемых данной системой констант ограничено), и почтовый ящик $MsgBox_A$.

Агенты общаются между собой посредством передачи сообщений вида $msg(Sender, Receiver, Msg)$, где $Sender$ и $Receiver$ – имена агентов (источника и адресата), а Msg – (передаваемый) базисный атом. Агент A посылает сообщения другим агентам системы, передавая их агенту P, и получает сообщения от агента P в свой почтовый ящик. При этом можно рассматривать два способа работы: 1) *синхронный*, когда предполагается, что P передает сообщение адресату сразу (в этом случае агент P фактически можно исключить из системы), и 2) *асинхронный*, когда передача сообщения от P к агентам может занять произвольное количество времени. Синхронные МА-системы больше подходят для описания практических систем, работающих в рамках локальной сети (или в отдельном компьютере), в асинхронных МА-системах лучше отражаются свойства сильно распределенных (например, в Интернете) систем.

Состояние агента P содержит все сообщения, полученные им и не отосланные до текущего момента. Текущие содержимые внутренней БД и почтового ящика агента A составляют его текущее локальное состояние $IM_A = <I_A, MsgBox_A>$. *Глобальное состояние* МА-системы $\mathcal{A}$ состоит из локальных состояний агентов a_i и состояния почтового агента.

С каждым агентом a связана его база AB_a *параметризованных действий* вида $< \alpha(X_1, \ldots, X_m)$, $ADD_\alpha(X_1, \ldots, X_m)$, $DEL_\alpha(X_1, \ldots, X_m)$, $SEND_\alpha(X_1, \ldots, X_m) >$, где $\alpha(X_1, \ldots, X_m)$ определяет *параметризованное имя действия*. $ADD_\alpha(X_1, \ldots, X_m)$ и $DEL_\alpha(X_1, \ldots, X_m)$ – списки атомов вида $p(t_1, \ldots, t_k)$, где p – k-местный предикат из сигнатуры внутренней БД, $t_1, \ldots, t_k$ – либо константы, либо параметры $X_1, \ldots, X_m$. Эти множества определяют изменения внутренней БД (добавления и удаления фактов) при выполнении соответствующего действия. $SEND_\alpha(X_1, \ldots, X_m)$ определяет аналогичным образом список сообщений вида $msg(a, b, p(t_1, \ldots, t_k))$, отправляемых агентом a другим агентам. Для сокращения записи мы иногда при определении $SEND_\alpha$, где α – действие агента a, будем вместо $msg(a, b, p(t_1, \ldots, t_k))$ писать просто $(b, p(t_1, \ldots, t_k))$.

Пусть $c_1, \ldots, c_m$ – константы. Обозначим через $ADD_a(c_1, \ldots, c_m)$ множество фактов, получаемых подстановкой $c_1, \ldots, c_m$ вместо $X_1, \ldots, X_m$ в атомы из $ADD_a(X_1, \ldots, X_m)$. Аналогично определяются $DEL_a(c_1, \ldots, c_m)$ и $SEND_a(c_1, \ldots, c_m)$. Базисные атомы вида $a(c_1, \ldots, c_m)$ назовем *базисными именами действий*.

Конкретный выбор этих действий для выполнения в зависимости от локального состояния агента определяется парой $< Pr_A, Sel_A >$. Здесь Pr_A – интеллектуальная компонента, определяющая совместно с фактами из локального состояния агента в текущий момент t набор

фактов о потенциально возможных в данный момент действиях. В качестве таких компонент могут выступать обычные логические программы [25], в сигнатуру которых включены атомы действий. Правила таких программ имеют вид

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

где H – атом действия; литералы L_i – либо литералы действия, либо (экстенсиональные) литералы с предикатами из сигнатуры внутренней БД, либо литералы сообщений вида $msg(Sender, A, Msg)$ или $\neg msg(Sender, A, Msg)$, либо некоторые вычислимые за полиномиальное время встроенные предикаты. Агент называется *позитивным,* если его программа не содержит отрицаний.

Мы предполагаем, что программы агентов *безопасны* в том смысле, что 1) любая переменная, встречающаяся в левой части некоторого правила, имеет положительное вхождение в правую часть соответствующего правила, и 2) программа Pr_A^{state}, полученная добавлением к Pr_A фактов из локального состояния $state$ (внутренней БД и почтового ящика) агента A, является *стратифицированной* [25].

При указанных условиях любая логическая программа определяет однозначно множество $Perm$, состоящее из базисных имен действий, которые выводятся по ее правилам из фактов локального состояния агента и разрешены к выполнению в данный момент. Оператор Sel_A выбирает (детерминированно или недетерминированно) из $Perm$ подмножество $Oblig$ фактически выполняемых на данном шаге действий. Простейший из таких операторов – это оператор единичного выбора: $Sel^{un}(S) = \{\{p\}|p \in S\}$.

Пусть $Oblig(= Oblig_t)$ – множество фактически выполняемых на данном шаге t действий, выбранное в соответствии с Sel_A. Тогда все действия из $Oblig$ выполняются параллельно следующим образом. Пусть ADD_{Oblig} равно объединению всех множеств $ADD_a(c_1, \ldots, c_m)$ таких, что базисное имя $a(c_1, \ldots, c_m)$ из $Oblig$ унифицируется с параметризованным именем $a(X_1, \ldots, X_m)$. Аналогично определяются множества DEL_{Oblig} и $SEND_{Oblig}$. Тогда следующее состояние внутренней БД агента A получается из текущего состояния удалением элементов из DEL_{Oblig} и добавлением элементов из ADD_{Oblig}. Кроме того, к базе почтового агента добавляются все элементы из $SEND_{Oblig}$, а почтовый ящик агента A опустошается. Конечно, можно рассматривать и другие стратегии добавлений и удалений из внутренней БД и почтового ящика.

Действие называется *расширяющим,* если удаляемое им множество пусто. Агент называется *расширяющим,* если все его действия расширяющие.

Вышеопределенная одношаговая семантика отдельных агентов естественным образом расширяется до семантики $Sem_{\mathcal{A}}$ всей МА-системы $\mathcal{A}$, определяющей преобразования ее глобальных состояний. А именно, в начале каждого шага работы системы из состояния почтового агента удаляются для синхронных систем – все, а для асинхронных – некоторые, сообщения вида $msg(sender, receiver, msg)$, которые сразу же помещаются в почтовые ящики соответствующих агентов-адресатов. Затем все агенты изменяют свои внутренние состояния в соответствии со своей одношаговой семантикой и передают свои сообщения почтовому агенту. После этого их почтовые ящики опустошаются. Из этого определения видно, что, если даже каждый агент имеет детерминированную семантику, в асинхронном случае семантика всей системы является недетерминированной.

Синхронные системы разделяются на два класса: если все агенты системы детерминированные, она называется детерминированной, в противном случае – недетерминированной.

2.1. *Поведение МА-систем*

В соответствии с вышеприведенными определениями любая МА-система $\mathcal{A} = \{a_1, \ldots, a_n, P\}$ определяет отношение $\Rightarrow_{\mathcal{A}}$ перехода за один шаг на множестве своих глобальных состояний.

Мы будем рассматривать поведение $\mathcal{A}$, которая начинает работать в некотором глобальном состоянии с пустыми почтовыми ящиками: $S^0 =< (I^0_{a_1}, MsgBox^0_{a_1}), ..., (I^0_{a_n}, MsgBox^0_{a_n}), I^0_P >$, в котором $MsgBox_{a_i} = \emptyset$, $1 \le i \le n$, $I^0_P = \emptyset$. Оно определяется деревом $T_{\mathcal{A}}(S^0)$ возможных бесконечных траекторий (т.е. последовательностей глобальных состояний) вида:

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

Для детерминированной МА-системы $\mathcal{A}$ дерево $\mathcal{T}$ состоит из единственной траектории, начинающейся в S^0. Если $\mathcal{A}$ – недетерминированная или асинхронная, то $\mathcal{T}$ – это бесконечное дерево траекторий с корнем S^0. Вершинами $\mathcal{T}$ являются глобальные состояния $S \in \mathcal{S}_{\mathcal{A}}$, достижимые из S^0 по рефлексивному и транзитивному замыканию отношения $\Rightarrow_{\mathcal{A}}$. Если S – это вершина $\mathcal{T}$, то состояния из $Next_{\mathcal{A}}(S)$ являются ее непосредственными потомками в $\mathcal{T}$. Конечную или бесконечную ветвь дерева $\mathcal{T}$, начинающуюся в его некоторой вершине, будем называть *траекторией* из $\mathcal{T}$.

Проблема верификации динамических свойств многоагентных систем (которую мы называем MA-BEHAVIOR) формулируется следующим образом. Пусть даны МА-система $\mathcal{A}$, ее начальное глобальное состояние S^0 и формула F, выражающая некоторое свойство деревьев траекторий, нужно проверить, выполняется ли F на дереве $T_{\mathcal{A}}(S^0)$.

Наличие реляционной структуры в состояниях агентов и определение переходов с помощью логических программ делает естественным рассмотрение предикатных расширений временных логик в качестве языков описания динамических свойств МА-систем.

Мы используем некоторые варианты предикатных расширений следующих пропозициональных логик (см. [22, 24, 26, 27]): логики линейного времени LTL, простых подмножеств **A**LTL и **E**LTL классической логики ветвящегося времени, существенно более богатого μ-исчисления Lμ (использующего, кроме обычных временных операторов, операторы построения наименьших и наибольших неподвижных точек) и его подмножеств Lμ_r ($r = 1, 2, \ldots$) с ограничениями на глубину альтернирования [22, 23][2]. Для предикатного расширения логики L будем использовать обозначение FO-L (FO – "first order"). Простое введение в теорию программных и временных логик можно найти в [28].

[2] В частности, определение глубины альтернирования в μ-исчислении можно найти в русском переводе книги [24] на стр. 145–146.

2.2. *Классы МА-систем*

Во всех трех классах МА-систем – детерминированных, недетерминированных и асинхронных – мы выделяем еще следующие подклассы, получаемые при различных естественных ограничениях на систему и ее агентов. МА-система $\mathcal{A} = \{a_1, \ldots, a_n\}$ является:

- *базисной*, если программы P_{a_i} всех ее агентов являются базисными[3];
- *k-параметрической*, если число аргументов (арность) всех предикатов действий агентов системы и всех предикатов в их сообщениях ограничено числом k. Очевидно, это свойство фиксирует максимальное число возможных параметров у действий и сообщений системы $\mathcal{A}$;
- *расширяющей*, если все ее агенты являются *расширяющими*;
- *позитивной*, если все ее агенты являются *позитивными*;
- *m-агентной*, если число агентов $n \leq m$.
- *r-сигнальной*, если число разных базисных (пропозициональных) предикатов у агентов системы (ее *сигналов*) не больше r.

Следующее простое предложение характеризует сложность[4] выполнения одного шага детерминированными, недетерминированными и асинхронными МА-системами.

Предложение 1.

(1) Для каждой детерминированной МА-системы $\mathcal{A}$ функция переходов $S \Rightarrow_{\mathcal{A}} S'$ вычислима за полиномиальное время относительно размера входа $|S| + |\mathcal{A}| + |S'|$ для подклассов базисных систем или систем с некоторым фиксированным числом параметров, а в общем случае она вычислима за (детерминированное) экспоненциальное время[5].

[3] Напомним, что это означает, что в программах отсутствуют переменные.

[4] Используемые в работе сложностные понятия можно найти, например, в [29, 30].

[5] Фактически за время, полиномиальное от размера базисной системы, эквивалентной исходной.

(2) Для каждой недетерминированной или асинхронной МА-системы $\mathcal{A}$ отношение (многозначная функция) переходов $S \Rightarrow_{\mathcal{A}} S'$ распознается за недетерминированное полиномиальное время относительно размера входа $|S|+|\mathcal{A}|+|S'|$ для подклассов базисных систем или систем с некоторым фиксированным числом параметров, а в общем случае она вычислима за недетерминированное экспоненциальное время.

3. ПРИМЕР МА-СИСТЕМЫ: РАСПРЕДЕЛЕНИЕ РЕСУРСОВ

Рассмотрим небольшую систему "распределения ресурсов" $\mathcal{RA}$, которая включает агента-менеджера m, владеющего некоторым (неограниченным) ресурсом, который он распределяет среди четырех агентов-пользователей u_1, u_2, u_3, u_4, получая от них заказы на этот ресурс. У каждого из агентов-пользователей имеется собственная стратегия запросов ресурса:

1) u_1 запрашивает ресурс в начальный момент, а затем повторяет запрос сразу же после очередного получения ресурса от m;

2) u_2 запрашивает ресурс сразу после того, как его заказал u_1;

3) u_3 запрашивает ресурс сразу после того, как u_1 получил ресурс от m;

4) u_4 запрашивает ресурс на каждом шаге.

Менеджер m ведет очередь заказов и, если она непуста, то выполняет первый из них, т.е. посылает ресурс агенту, заказ которого стоит в голове очереди, а сам этот заказ из очереди удаляет. В каждый момент времени в очереди может находиться не более одного заказа каждого агента-пользователя. Поэтому, если m получает заказ от некоторого агента u_i до того, как был выполнен его предыдущий заказ, то новый заказ игнорируется.

Видно, что все пять агентов из этого примера работают, обмениваясь некоторой информацией о заказах и их исполнении. Все они работают детерминированно в соответствии с достаточно простыми правилами. Впрочем, менеджер m должен иметь более сложную программу, позволяющую управлять очередью заказов. Агенты системы $\mathcal{RA}$ определяются следующим образом.

Состояния I_{u_1} агента u_1 могут содержать факт put_order и $receipt_1$. Состояния I_{u_i} $(i = 2,3,4)$ могут содержать факт $receipt_i$, означающий, что u_i получил ресурс на предыдущем шаге. Когда m выполняет заказ агента u_i, он посылает ему сообщение ok (т.е. передает почтовому агенту сообщение $msg(m, u_i, ok)$; здесь и ниже для упрощения записей мы при определении $SEND_\alpha$, где α – действие агента a, используем сокращения, определенные в разделе 2 при определении действий. Кроме того, мы опускаем некоторые подразумевающиеся индексы α при определении множеств $ADD_\alpha, DEL_\alpha, SEND_\alpha$). Агент u_1 посылает агенту u_2 сообщение $order$ (при посылании заказа) и агенту u_3 сообщение ok (при получении ресурса).

У каждого из агентов u_i $(i = 1,2,3,4)$ имеются два возможных действия:

$use\_resource_i$ – потребление полученного ресурса, которое задается для всех i множествами $DEL = \{receipt_i\}, ADD = SEND = \emptyset$, и

$receive_i$ – получение ресурса от m, которое задается для $i = 2,3,4$ множествами $ADD_i = \{receipt_i\}, DEL_i = SEND_i = \emptyset$, а при $i = 1$ действие $receive_1$ задается множествами $ADD_1 = \{receipt_1\}, DEL_1 = \{put\_order\}$, $SEND_1 = \{(u_3, ok)\}$.

Эти действия запускаются правилами:

$use\_resource_i \leftarrow receipt_i$ и

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

Кроме этого, у агентов имеется действие put, посылающее заказ менеджеру (сообщение $order$), со специфическим эффектом и правилами.

У агента u_1 действие put задается множествами $ADD = \{put\_order\}$, $SEND = \{(m, order), (u_2, order)\}$, а запускается правилом:

$$put \leftarrow \neg put\_order.$$

У агента u_2 действие put задается множеством $SEND = \{(m, order)\}$, а запускается правилом:

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

У агента u_3 действие put задается множеством $SEND = \{(m, order)\}$, а запускается правилом:

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

У агента u_4 действие put задается множеством $SEND = \{(m, order)\}$, а запускается на каждом шаге правилом (фактом):

$$put \leftarrow .$$

Агент m управляет очередью заказов, представленной с помощью предикатов внутренней БД:

$first(A)$ – заказ агента A стоит в очереди первым;

$next(A, B)$ – вслед за заказом агента A в очереди идет заказ агента B;

$last(A)$ – заказ агента A стоит в очереди последним.

Он также использует вспомогательные предикаты (их можно считать действиями с пустыми множествами $ADD, DEL, SEND$): $empty\_queue$, который проверяет пустоту очереди, и $in\_queue(A)$, который проверяет, находится ли заказ A в очереди. Они определяются предложениями:

$$empty\_queue \leftarrow \neg first(u_1),$$
$$\neg first(u_2), \neg first(u_3), \neg first(u_4).$$
$$in\_queue(A) \leftarrow first(A).$$
$$in\_queue(A) \leftarrow in\_queue(B), next(B, A).$$

Когда m получает новые заказы, он помещает их в конец очереди, при этом, если одновременно нужно поместить несколько заказов, то они ставятся в очередь в предопределенном порядке: $u_1 < u_2 < u_3 < u_4$. Для каждой подпоследовательности $X_1 < \ldots < X_i$ последовательности (u_1, u_2, u_3, u_4) у m имеется действие $insert(F, S, L, X_1, \ldots, X_i)$:

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

Каждое такое действие запускается правилом:

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

(здесь $\{Y_1, \ldots, Y_j\} = \{u_1, u_2, u_3, u_4\} \setminus \{X_1, \ldots, X_i\}$).

$$new\_order(X) \leftarrow msg(X, m, order), \neg in\_queue(X).$$

Когда очередь пуста, m запускает одно из действий вида $insert_0(X_1, \ldots, X_i)$:

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

запускаемых предложениями вида:

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

(здесь снова $\{Y_1, \ldots, Y_j\} = \{u_1, u_2, u_3, u_4\} \setminus \{X_1, \ldots, X_i\}$).

Для определенной здесь системы $\mathcal{RA}$ формула $\mathbf{G}\ \mathbf{F} \bigwedge_{i=1}^{4} receipt_i$ истинна на траектории, начинающейся в пустом состоянии $S_0 = \{<\emptyset, \emptyset>, <\emptyset, \emptyset>, <\emptyset, \emptyset>, <\emptyset, \emptyset>, <\emptyset, \emptyset>\}$. Содержательно, эта формула означает, что каждый агент получает требуемый ресурс бесконечно часто[6].

В то же время, следующие две формулы не будут верны на этой траектории: $\mathbf{F}\ (receipt_1 \wedge \wedge\ \mathbf{X}\ receipt_1)$ (имеются два подряд идущих момента, в которые получает ресурс агент u_1) и $\mathbf{G}\ (first(S) \wedge next(S, u_i) \rightarrow \mathbf{X}\ \mathbf{X}\ \neg receipt_i)$ (агент, стоящий в очереди вторым, не получит ресурс через два шага).

Если эти же агенты поместить в асинхронную систему $\mathcal{RA}_{as}$, то такие простые формулы, как $\forall\mathbf{G}\ \forall\mathbf{F}\ receipt_1$ или даже $\forall\mathbf{F}\ receipt_1$[7], окажутся неверными для $\mathcal{RA}_{as}$. Это связано с тем, что, хотя u_1 и запрашивает ресурс сразу на первом шаге, существуют траектории, на которых этот запрос никогда не будет передан агенту m.

В то же время, формула $\forall\mathbf{G}\ \exists\mathbf{F}\ receipt_1$ верна для $\mathcal{RA}_{as}$. Действительно, пусть s – некоторая вершина дерева траекторий. Если в этом состоянии сообщение $msg(m, u_1, ok)$ находится в почтовом агенте, то оно уже на следующем шаге может быть передано в u_1 и вызовет действие

[6] Операторы **G**, **F**, **X** логики линейного времени означают, соответственно, "всегда в будущем", "когда-нибудь в будущем" и "в следующий момент".

[7] Операторы $\forall\mathbf{G}, \forall\mathbf{F}, \exists\mathbf{F}$ логики ветвящегося времени означают, соответственно, "во всех возможных будущих состояниях", "на всех траекториях найдется момент" и "найдется возможное будущее состояние".

$receive_1$. Если оно не в почтовом агенте, рассмотрим последнее сообщение $Msg = msg(u_1, m, order)$, выработанное агентом u_1 до попадания системы в состояние s. Легко понять, что Msg в момент попадания в состояние s находится либо в почтовом агенте, либо в очереди агента m (когда оно пропадает, в почтовом агенте обязательно появляется сообщение $msg(m, u_1, ok)$). Когда Msg в очереди, m при обработке своей очереди доберется до этого сообщения и передаст почтовому агенту сообщение $msg(m, u_1, ok)$ (попадем в предыдущий случай). Если Msg находится в почтовом агенте, то работа из состояния s может быть продолжена его передачей агенту m и установкой в очередь.

Аналогичными рассуждениями можно показать, что для $\mathcal{RA}_{as}$ выполняется и более сильная формула $\forall\mathbf{G}\ \exists\mathbf{F}\ \bigwedge_{i=1}^{4} receipt_i$. Некоторые более сложные свойства могут быть выражены с использованием формул модального μ-исчисления Lμ.

4. ПОВЕДЕНИЕ ДЕТЕРМИНИРОВАННЫХ МА-СИСТЕМ

Этот раздел, в котором обсуждаются детерминированные МА-системы, фактически является сжатым изложением содержания раздела 4.1 из [2]. Его результаты используются ниже в разделе 6.1.

Множество глобальных состояний любой рассматриваемой МА-системы $\mathcal{A}$ конечно. Поэтому в случае детерминированности ее траектория $\tau(\mathcal{A}, S^0)$ является периодической. Таким образом, несмотря на бесконечность $\tau(\mathcal{A}, S^0)$, ее можно свернуть в некоторую конечную структуру. Прямой алгоритм проверки FO-LTL-формулы на этой структуре потребовал бы ее явного представления и, следовательно, памяти не меньшей, чем общее число различных глобальных состояний. Однако можно предложить более аккуратный способ проверки формулы на модели, который будет работать с ее небольшими подмножествами поточечно. Это позволяет получить существенно лучшие оценки сложности для проблемы MA-BEHAVIOR.

Для периодической траектории $\tau = S^0, S^1, \ldots, S^t, \ldots$ обозначим через k и N наименьшие числа такие, что $S^t = S^{t+N}$ для всех $t \geq k$. В приводимом ниже алгоритме будут использоваться три вспомогательные функции. Первая – $move(t, i)$ – по моменту времени t и сдвигу i вычисляет такой момент $j < k + N$, для которого $S^j = S^{t+i}$:

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

Вторая функция F^τ играет роль оракула, который для каждого t выдает состояние $F^\tau(t) = S^t$ траектории в этот момент. Третья функция проверяет истинность формулы первого порядка Φ на состоянии S: FO_Check(S, Φ) возвращает логическое значение, равное $TRUE$, если $S \models \Phi$, и $FALSE$ в противном случае.

Пусть $\tau = \tau(\mathcal{A}, S^0)$ – это периодическая траектория с параметрами k и N, Φ – произвольная FO-LTL-формула, а t – момент времени. Положим $s_{max}(\tau) = \max\{|S^t| \mid 0 \leq t \leq k + N\}$ и обозначим через $s(F^\tau)$ и $t(F^\tau)$ максимальный размер памяти и времени, соответственно, требуемые для вычисления $F^\tau(t)$ при $0 \leq \leq t \leq k + N$. Обозначим также через $s_{FO}(\tau, n)$ и $t_{FO}(\tau, n)$ максимальный размер памяти и времени, соответственно, требуемые для проверки условия $S^t \models \Psi$ для $0 \leq t \leq k + N$ и любой формулы 1-го порядка Ψ длины, не большей n.

Следующий рекурсивный алгоритм проверяет свойство $\tau, S^t \models \Phi$.

Алгоритм 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 – базисная формула
(5) $S^t := F^\tau(t)$;
(6) return FO_Check(S^t, Φ);
(7) CASE $\Phi = \Phi_1 \oplus \Phi_2$ $(\oplus \in \{\wedge, \vee\})$
(8) $b_1 :=$ DetCheck(τ, k, N, Φ_1, t);
(9) $b_2 :=$ DetCheck(τ, k, N, Φ_2, t),
(10) return $b_1 \oplus b_2$;
(11) CASE $\Phi = \neg\Phi_1$
(12) return $\neg$DetCheck(τ, k, N, Φ_1, t);
(13) CASE $\Phi = \mathbf{X}(\Phi_1)$
(14) $t_1 :=$ move$(t, 1)$;
(15) return DetCheck$(\tau, k, N, \Phi_1, t_1)$;
(16) CASE $\Phi = \Phi_1\mathbf{U}\Phi_2$
(17) IF $t < k$ THEN $R := k + N - t$
(18) ELSE $R := N$ END_IF

Таблица 1. Сложность верификации детерминированных МА-систем

Баз.	Расш.	Параметры	Логика	Сложность	N
Да	Да	позитивные	LTL	P	d1
		$m^2 * r = O(\log N)$	LTL	P	d2
		фикс. $m \geq 2$	LTL	PSPACE	d3
		фикс. $r \geq 1$	LTL	PSPACE	d4
	Нет		FO-LTL	PSPACE	d5
Нет	Да	позит., фикс. k	LTL	P	d6
		позитивные	LTL	EXPTIME	d7
	Нет	фикс. k	FO-LTL	PSPACE	d8
			FO-LTL	EXPSPACE	d9

```
(19) FOR i = 0 TO R − 1 DO
(20)  r := move(t, i); p := i;
(21)  IF DetCheck(τ, k, N, Φ2, r)
(22)   THEN EXIT_FOR END_IF
(23) END_ DO
(24) IF p = R − 1
(25) THEN return TRUE
(26) ELSE
(27)  b := TRUE;
(28)  FOR j = 0 TO p − 1 DO
(29)  r′ := move(t, j);
(30)  IF ¬DetCheck(τ, k, N, Φ1, r′)
(31)  THEN b := FALSE; EXIT_FOR
(32)  END_IF END_ DO;
(33) return b END_IF
(34) END_SELECT
```

Лемма 1. *Для заданных чисел k, N и t и FO-LTL-формулы Φ алгоритм DetCheck проверяет условие $\tau^t \models \Phi$ на периодической траектории τ с параметрами k и N, используя функции F^τ и FO_Check в качестве оракулов. Память, используемая алгоритмом, ограничена величиной $\mathbf{O}(|t| + |\Phi| + d^*(\Phi)\log(k+N) + s_{max}(\tau) + s(F^\tau) + s_{FO}(\tau, |\Phi|))$, а время – $pol(|t| + |\Phi|(k+N)(t(F^\tau) + t_{FO}(\tau, |\Phi|))$ для некоторого полинома pol.*

Оракул F^τ, предоставляющий в алгоритме информацию о траектории τ, можно эффективно вычислять для траекторий τ, порожденных МА-системами.

Лемма 2. *Существует алгоритм, который по МА-системе $\mathcal{A}$, ее начальному состоянии S^0 и моменту времени $t \geq 0$ вычисляет состояние S^t траектории $\tau(\mathcal{A}, S^0)$ в момент t. Для некоторого полинома pol память, используемая для этого вычисления, ограничена величиной $pol(|\mathcal{A}| + \max\{|S^r| \mid 0 \leq r \leq t\})$.*

Следующая лемма устанавливает границы параметров периодичности для траекторий МА-систем.

Лемма 3. *Для любой МА-системы $\mathcal{A}$ и ее начального состояния S^0 траектория $\tau(\mathcal{A}, S^0)$ является периодической с некоторыми параметрами $k(\mathcal{A}, S^0)$ и $N(\mathcal{A}, S^0)$. При этом, если $\mathcal{A}$ базисная, то $k(\mathcal{A}, S^0) + N(\mathcal{A}, S^0) \leq 2^{pol(|\mathcal{A}|+|S^0|)}$. В общем случае справедливо неравенство $k(\mathcal{A}, S^0) + N(\mathcal{A}, S^0) \leq 2^{2^{pol(|\mathcal{A}|+|S^0|)}}$.*

Из лемм 1, 2 и 3 можно получить верхние границы сложности верификации свойств МА-систем, выразимых в FO-LTL.

Предложение 2. *Пусть заданы МА-система $\mathcal{A}$ и ее начальное состояние S^0, и пусть $k = k(\mathcal{A}, S^0)$, $N = N(\mathcal{A}, S^0)$. Тогда для некоторого полинома pol проверку истинности FO-LTL-формулы Φ на траектории $\tau(\mathcal{A}, S^0)$ можно выполнить, используя память размера $2^{pol(|\Phi|+|\mathcal{A}|)}$ в общем случае и память размера $pol(|\Phi| + |\mathcal{A}|)$ в случае базисных систем.*

5. СЛОЖНОСТЬ ВЕРИФИКАЦИИ ДЕТЕРМИНИРОВАННЫХ И НЕДЕТЕРМИНИРОВАННЫХ МА-СИСТЕМ

В наших предыдущих работах [1–3] задача о сложности проблемы MA-BEHAVIOR решалась для классов детерминированных и недетерминированных МА-систем и ряда их подклассов. Основные полученные в этих работах результаты суммированы в таблицах 1 и 2.

Первые три столбца в них определяют классы рассматриваемых МА-систем. Столбцы “Баз.” и “Расш.” позволяют различить базисные систе-

Таблица 2. Сложность верификации недетерминированных МА-систем

Баз.	Расш.	Параметры	Логика	Сложность	N
Да	Да	$m^2 * r = O(\log N)$	$\exists$ LTL	NP	n1
			$\forall$ LTL	co-NP	n2
		фикс. $m \geq 2$	FO-Lμ_1	EXPTIME	n3
		фикс. $r \geq 1$	FO-Lμ_1	EXPTIME	n4
	Нет		FO-Lμ_l, фикс. l	EXPTIME	n5
			FO-Lμ	в NEXPTIME $\cap$ $\cap$ co-NEXPTIME	n6
Нет	Да	фикс. $r \geq 1$	$\exists$ LTL	NEXPTIME	n7
			$\forall$ LTL	co-NEXPTIME	n8
		позитивные	$\exists$ LTL	EXPSPACE-hard	n9
	Нет	фикс. k	FO-Lμ	EXPTIME	n10
			FO-Lμ_l, фикс. l	EXPEXPTIME	n11
			FO-Lμ	в NEXPEXPTIME $\cap$ $\cap$ co-NEXPEXP-TIME	n12

мы и небазисные системы, а также расширяющие (без удалений) системы и системы с удалениями. В столбце "Параметры" указаны некоторые ограничения на параметры архитектуры МА-систем, приводящие к меньшей сложности верификации. В частности, N обозначает размер всего описания системы, m – число ее агентов, k – это максимальная размерность атомов действий и сообщений (в пропозициональном случае $k = 0$), r – это число различных сигналов (т.е. сообщений). В столбце "Логика" указан логический язык, в котором формулируются свойства поведения соответствующего класса МА-систем. В столбце "Сложность" указана оценка сложности соответствующей проблемы MA-BEHAVIOR.

Во всех строках, кроме n6, n9 и n12, соответствующая проблема MA-BEHAVIOR является полной в указанном сложностном классе. В строках n6 и n12 указаны только верхние оценки сложности, а в строке n9 – только нижняя оценка.

Неудивительно, что в общем случае проблема MA-BEHAVIOR является очень сложной: EXPSPACE-полной для детерминированных систем (d9), EXPEXPTIME-полной для недетерминированных (n11). Однако нам удалось выделить и некоторые подклассы систем, для которых она решается эффективно. Эти системы либо обладают свойствами сильной монотонности (d1, d6), либо не удаляют никаких фактов из своих баз данных и имеют ограничения на структуру: $m^2 * r = O(\log N)$ (d2). Строки d3, d4, d7 показывают, что устранение любого из ограничений приводит к немедленному росту сложности.

Что касается недетерминированных систем, то было бы трудно ожидать эффективного решения проблемы верификации даже для весьма ограниченных их подклассов. Утверждения n1 и n2 описывают случаи, когда эта проблема разрешима с (ко-)недетерминированным полиномиальным временем. А оценки n3, n4, n5 показывают, что ослабление условий приводит к усложнению проблемы. Отметим, что во всех этих случаях можно получить более точную нижнюю оценку вида $\mathbf{o}(c^{n/log\ n})$. Отметим также, что верхние оценки для языков FO-Lμ, FO-Lμ_1, FO-Lμ_l получаются путем обобщения известных общих результатов из [23] о сложности проверки на моделях для μ-исчисления.

6. ПОВЕДЕНИЕ АСИНХРОННЫХ СИСТЕМ

Для верификации асинхронных МАС важно принимать в расчет внутреннее состояние почтового агента, которое состоит из атомов ви-

да $msg(a_i, a_j, p)$. Но такие атомы встречаются также в почтовых ящиках агентов. Поэтому с целью корректного упоминания этих атомов в формулах мы примем следующие обозначения: $msg^j(a_i, a_j, p)$ в формулах будет обозначать атом, значение истинности которого оценивается по отношению к почтовому ящику агента a_j, а $msg(a_i, a_j, p)$ обозначает атом, оцениваемый над почтовым агентом.

Недетерминированные и асинхронные системы довольно близки друг к другу. Это формально подтверждается результатами подраздела 6.2 о взаимной сводимости проблемы MA-BEHAVIOR для недетерминированных и асинхронных МАС. Это позволяет непосредственно перенести результаты о сложности верификации для недетерминированных МАС на асинхронные МАС, но только для систем общего вида. Это связано с тем, что для моделирования расширяющих недетерминированных систем используются немонотонные асинхронные, и наоборот. Тем не менее, в разделе 6.1 показывается, что результат о NP-полноте проблемы MA-BEHAVIOR для некоторого подкласса недетерминированных расширяющих МАС переносится на аналогичный класс асинхронных МАС.

6.1. Подкласс асинхронных МАС с NP-полной проблемой MA-BEHAVIOR

Теорема 1. *Проблема MA-BEHAVIOR со свойствами поведения, выраженными формулами* $\Phi \in \exists LTL$ $(\forall LTL)$, *не содержащими атомов-сообщений, для класса базисных, расширяющих, r-сигнальных, m-агентных асинхронных систем* $\mathcal{A}$, *таких, что* $m^2 * r = \mathbf{O}(\log|\mathcal{A}|)$, *является* NP-*полной* (*соответственно, coNP-полной*).

Доказательство. Мы приведем доказательство для $\exists LTL$-формул. Случай $\forall LTL$-формул получается с помощью эквивалентности: $\mathcal{T}_{\mathcal{A}}(S^0) \models \mathbf{E}(\Psi) \Leftrightarrow \mathcal{T}_{\mathcal{A}}(S^0) \not\models \mathbf{A}(\neg\Psi)$.

Верхняя оценка. Пусть $\mathcal{A}$ – базисная, расширяющая, r-сигнальная, m-агентная асинхронная система, для которой $m^2 * r = \mathbf{O}(\log|\mathcal{A}|)$. Недетерминированный алгоритм, решающий проблему MA-BEHAVIOR за полиномиальное время, основан на следующем свойстве стабилизации некоторых траекторий из $\mathcal{T}_{\mathcal{A}}(S^0)$.

Лемма 4. *Существует такой полином* $p(n)$, *что* $\mathcal{T}_{\mathcal{A}}(S^0) \models \mathbf{E}\,\Psi$ *тогда и только тогда, когда существуют траектория* $\mu = S^0, \ldots, S^t, \ldots \in \mathcal{T}_{\mathcal{A}}(S^0)$ *и шаг* $T \le p(|\mathcal{A}| + |\Psi|)$ *такие, что* μ, $S^0 \models \Psi$ *и* $I^t_a = I^T_a$ *для всех* $t > T$ *и каждого* $a \in \mathcal{A}$.

Доказательство. Мы установим вначале несколько вспомогательных утверждений, относящихся к истинности FO-LTL-формул на траекториях с длинными последовательностями повторяющихся состояний. Напомним, что формулы зависят только от состояний БД траектории, но не от состояний соответствующих почтовых ящиков.

Пусть $\mu = M^0, \ldots, M^i, \ldots$ и $\nu = N^0, \ldots, N^j, \ldots$ – это две траектории, а $d \ge 0$ – некоторое целое число. Скажем, что пара (μ, M^i) *d-эквивалентна* паре (ν, N^j) (обозначение: $(\mu, M^i) \sim_d \sim_d (\nu, N^j)$), если эквивалентность $\mu, M^i \models \varphi \Leftrightarrow \Leftrightarrow \nu, N^j \models \varphi$ имеет место для любой FO-LTL-формулы φ глубины $d(\varphi) \le d$.

Утверждение 1. *Если для некоторого* $d \ge 0$ *имеют место эквивалентности* $(\mu, M^i) \sim_d \sim_d (\nu, N^j)$ *и* $(\mu, M^{i+1}) \sim_{d+1} (\nu, N^{j+1})$, *то имеет место и эквивалентность* $(\mu, M^i) \sim_{d+1} (\nu, N^j)$.

Доказательство. Действительно, пусть $(\mu, M^i) \sim_d (\nu, N^j)$ и $(\mu, M^{i+1}) \sim_{d+1} (\nu, N^{j+1})$, и пусть φ – произвольная FO-LTL-формула глубины $d+1$. Если $\varphi = \varphi_1 \oplus \varphi_2$ $(\oplus \in \{\wedge, \vee\})$ или $\varphi = \neg\varphi_1$, то $d(\varphi_1) \le d$ и $d(\varphi_2) \le d$. По предположению, $\mu, M^i \models \varphi_k \Leftrightarrow \nu, N^j \models \varphi_k$ $(k = 1, 2)$. Следовательно, $\mu, M^i \models \varphi \Leftrightarrow \nu, N^j \models \varphi$. если $\varphi = \mathbf{X}(\varphi_1)$, то, по предположению, имеем $\mu, M^{i+1} \models \varphi_1 \Leftrightarrow \nu, N^{j+1} \models \varphi_1$ и $\mu, M^i \models \varphi \Leftrightarrow \Leftrightarrow \nu, N^j \models \varphi$. Предположим теперь, что $\mu, M^i \models \varphi$ для $\varphi = \varphi_1 \mathbf{U} \varphi_2$. Тогда по определению оператора $\mathbf{U}$ (i) $\mu, M^i \models \varphi_2$ или, в противном случае, (ii) $\mu, M^i \models \varphi_1$ и $\mu, M^{i+1} \models \varphi$. В случае (i), учитывая $d(\varphi_2) = d$, из первого предположения получаем, что $\nu, N^j \models \varphi_2$ и, следовательно, $\nu, N^j \models \varphi$. Аналогично, в случае (ii) можно показать, что $\nu, N^j \models \varphi_1$. Более того, из второго предположения выводим, что $\nu, N^{j+1} \models \varphi$. Таким образом, и в этом случае получаем, что $\nu, N^j \models \varphi$. Следовательно, во всех случаях имеет место $\nu, N^j \models \varphi$ и $(\mu, M^i) \sim_{d+1} (\nu, N^j)$. □

Из утверждения 1 непосредственно вытекает следующее

Утверждение 2. *Если для некоторого* $d \geq 0$ $(\mu, M^i) \sim_0 (\nu, N^j)$ и $(\mu, M^{i+1}) \sim_d (\nu, N^{j+1})$, *то* $(\mu, M^i) \sim_d (\nu, N^j)$.

Утверждение 3. *Пусть* $\mu = M^0, \ldots$ – *некоторая траектория и* i – *такой номер шага, что* $(\mu, M^i) \sim_d (\mu, M^{i+1}) \sim_d (\mu, M^{i+2})$. *Тогда* $(\mu, M^i) \sim_{d+1} (\mu, M^{i+1})$.

Доказательство. Пусть φ – некоторая FO-LTL-формула глубины $d(\varphi) = d + 1$. Если φ является булевой комбинацией формул глубины $\leq d$, то $\mu, M^i \models \varphi \Leftrightarrow \mu, M^{i+1} \models \varphi$, так как $(\mu, M^i) \sim_d (\mu, M^{i+1})$. Если $\varphi = \mathbf{X}(\varphi_1)$, то $d(\varphi_1) \leq d$ и $\mu, M^i \models \varphi \Leftrightarrow \mu, M^{i+1} \models \varphi$, так как $(\mu, M^{i+1}) \sim_d (\mu, M^{i+2})$. Если $\varphi = \varphi_1 \mathbf{U} \varphi_2$, то $d(\varphi_1) \leq d$ и $d(\varphi_2) \leq d$. Предположим, что $\mu, M^i \models \varphi$. Тогда по определению оператора $\mathbf{U}$ имеем $\mu, M^i \models \varphi_2$ или $\mu, M^i \models \varphi_1$ и $\mu, M^{i+1} \models \varphi$. В обоих случаях ясно, что $\mu, M^{i+1} \models \varphi$ (в первом случае использовано предположение $M^i \sim_d \sim_d M^{i+1}$). С другой стороны, предположим, что $\mu, M^{i+1} \models \varphi$. Тогда (i) $\mu, M^{i+1} \models \varphi_1$ или (ii) $\mu, M^{i+1} \models \varphi_2$. В случае (i) из предположения $(\mu, M^i) \sim_d (\mu, M^{i+1})$ следует, что $\mu, M^i \models \varphi_1$ и вместе с $\mu, M^{i+1} \models \varphi$ это дает, что $\mu, M^i \models \varphi$. Случай (ii) аналогичен. □

Из утверждения 3 непосредственно следует

Утверждение 4. *Пусть* $\mu = M^0, \ldots$ – *некоторая траектория, а* i – *номер такого шага, что* $M^i = M^{i+1} = M^{i+2} = \ldots = M^{i+2+K}$. *Тогда* $(\mu, M^i) \sim_K (\mu, M^{i+1})$.

Утверждение 5. *Пусть* $\mu = M^0, \ldots$ – *траектория и* i – *номер шага такие, что* $M^i = M^{i+1} = M^{i+2} = \ldots = M^{i+2+K}$. *Пусть траектория* $\mu' = M^0, \ldots, M^{i-1}, M^{i+1}, \ldots, M^{i+2+K}, \ldots$ *получена из* μ *с помощью удаления* M^i. *Тогда* $(\mu, M^0) \sim_K (\mu', M^0)$.

Доказательство. Из утверждения 4 следует, что $(\mu, M^i) \sim_K (\mu, M^{i+1})$. Тогда $(\mu, M^{i-1}) \sim_0 \sim_0 (\mu', M^{i-1})$ и $(\mu, M^i) \sim_K (\mu', M^{i+1})$. Из утверждения 2 тогда следует, что $(\mu, M^{i-1}) \sim_K \sim_K (\mu', M^{i-1})$. Так как $(\mu, M^{i-2}) \sim_0 (\mu', M^{i-2})$, то мы отсюда получаем, что $(\mu, M^{i-2}) \sim_K \sim_K (\mu', M^{i-2})$, и так далее, пока не дойдем до эквивалентности $(\mu, M^0) \sim_K (\mu', M^0)$. □

Вернемся теперь к доказательству леммы 4. Предположим, что $\mathcal{T}_\mathcal{A}(S^0) \models \mathbf{E}\ \Psi$. Это означает, что $\lambda, S^0 \models \Psi$ для некоторой траектории $\lambda = S^0, \ldots, S^t, \ldots \in \mathcal{T}_\mathcal{A}(S^0)$. Пусть $t_1, t_2, \ldots, t_i, \ldots, t_k$ – это те шаги траектории λ, на которых увеличиваются состояния агентов, т.е. $I_a^{t_i} \subset I_a^{t_i+1}$ для некоторого $a \in \mathcal{A}$. Так как $\mathcal{A}$ расширяющая, то k ограничено общим числом действий агентов из $\mathcal{A}$. Отсюда, $k \leq |\mathcal{A}|$. По выбору шагов вида t_i состояния БД всех агентов из $\mathcal{A}$ не меняются на шагах $t_i+1, t_i+2, \ldots, t_{i+1}$ для всех i. Пусть $k_i = t_{i+1} - t_i$ – длина такой стабильной подпоследовательности состояний БД. Обозначим через N_m число всех возможных состояний почтовых ящиков и почтового агента $\mathcal{A}$. Число различных сообщений в почтовом ящике каждого агента в состоянии P не превосходит rm. Тогда $N_m \leq (2^{rm})^{m+1} = 2^{rm^2+rm} \leq \leq 2^{O(\log|\mathcal{A}|)} \leq |\mathcal{A}|^c$ для некоторой константы c.

Пусть $d = d(\Psi)$, $k_i > d + 2 + N_m$, тогда существуют шаги l и r, $t_i + d + 2 < l < r < t_{i+1}$, такие, что $S^l = S^r$. Тогда у $\mathcal{T}_\mathcal{A}(S^0)$ имеется траектория $\mu = S^0, \ldots, S^l, S^{r+1}, \ldots$, получающаяся из λ с помощью удаления состояний $S^{l+1}, \ldots, S^r$. Из утверждения 5 следует, что $(\lambda, S^0) \sim_d (\mu, S^0)$, и, следовательно, $\mu, S^0 \models \Psi$. Тогда существует траектория $\mu \in \mathcal{T}_\mathcal{A}(S^0)$ такая, что $\mu, S^0 \models \Psi$ и максимальная длина последовательности одинаковых состояний БД в μ для некоторых констант c и c_1 ограничена числом $d+2+N_m \leq c_1(|\mathcal{A}|^c + |\Psi|)$. Для этой траектории шаг T, на котором она стабилизируется, не превосходит $t_k+1 \leq k+k(d+2+N_m) \leq pol(|\mathcal{A}|+|\Psi|)$. □

Верхняя оценка в теореме 1 следует из лемм 1 и 4. Именно, пусть $\mathcal{A}$ – асинхронная, базисная, расширяющая система с не более чем r сигналами и m агентами, и пусть $m^2 * r = \mathbf{O}(\log|\mathcal{A}|)$. Обозначим через S^0 ее произвольное начальное состояние. Пусть $\mathcal{T} = \mathcal{T}_\mathcal{A}(S^0)$ – это дерево траекторий $\mathcal{A}$, начинающихся в S^0, а Ψ – произвольная LTL-формула. Положим $T = pol(|\mathcal{A}| + |\Psi|)$, где полином pol определен в лемме 4. Чтобы узнать, верно ли, что $\mathcal{T}_\mathcal{A}(S^0) \models \mathbf{E}\ \Psi$, применим следующий недетерминированный алгоритм NdetCh:

(1) угадать в дереве $\mathcal{T}_\mathcal{A}(S^0)$ конечную траекторию $\lambda = S^0, S^1, \ldots, S^T, \ldots, S^{2T}$, в которой $I_a^T = I_a^{T+1} = I_a^{T+2} = \ldots = I_a^{2T}$ для всех $a \in \mathcal{A}$;

(2) проверить, верно ли, что $\lambda', S^0 \models \Psi$, используя алгоритм DetCheck, и возвратить ответ "Да", если $\lambda, S^0 \models \Psi$.

Ясно, что NdetCh выполняется за недетерминированное полиномиальное время. Чтобы

доказать его корректность, заметим, что, если $\mathcal{T}_{\mathcal{A}}(S^0) \models \mathbf{E}\ \Psi$, то по лемме 4 существует траектория $\lambda \in \mathcal{T}_{\mathcal{A}}(S^0)$ такая, что $\lambda, S^0 \models \Psi$, имеющая конечный префикс $\lambda' = S^0, \ldots, S^T$ такой, что $\lambda', S^0 \models \Psi$ и $I_a^j = I_a^T$ для всех $a \in \mathcal{A}$ и каждого $j > T$. Таким образом, на шаге (1) алгоритма можно угадать этот короткий префикс, а на шаге (2) проверка с помощью DetCheck вернет ответ "Да".

Обратно, если алгоритм NdetCh дает ответ "Да", то в дереве $\mathcal{T}_{\mathcal{A}}(S^0)$ имеется конечная траектория $\lambda = S^0, S^1, \ldots, S^T, \ldots, S^{2T}$ такая, что $\lambda, S^0 \models \Psi$ и $I_a^T = I_a^{T+1} = I_a^{T+2} = \ldots = I_a^{2T}$ для всех $a \in \mathcal{A}$. Так как $T > N_m$, то имеются такие i и j $(T < i < j < 2T)$, что $S^i = S^j$. Тогда этот префикс λ длины T можно расширить до некоторой бесконечной траектории $\lambda' \in \mathcal{T}_{\mathcal{A}}(S^0)$ такой, что $\lambda' = S^0, \ldots, S^T, S'^{T+1}, S'^{T+2}, \ldots$ и $I_a'^j = I_a^T$ для всех $a \in \mathcal{A}$ и $j > T$. Для этой траектории λ', $\lambda', S^0 \models \Psi$ и $\mathcal{T}_{\mathcal{A}}(S^0) \models \mathbf{E}\ \Psi$.

Нижняя оценка. Нетрудно показать, что проблема SAT сводится за полиномиальное время к проблеме MA-BEHAVIOR для базисных, расширяющих, 1-сигнальных, 2-агентных асинхронных систем. Действительно, пусть α – пропозициональная формула и $V = \{x_1, \ldots, x_n\}$ – множество входящих в нее переменных. Рассмотрим МА-систему $\mathcal{A}$, у которой имеется два агента a и b. Агент b на каждом шаге посылает a сообщение "1". Почтовый агент асинхронно пересылает эти сообщения агенту a. Если на i-ом шаге a получает сообщение "1", то он помещает в свою базу факты OK_i и x_i, а если сообщение не получено, то в базу помещается лишь факт OK_i. Экстенсиональная сигнатура a состоит из V и $\{OK_i \mid 1 \le i \le n\}$. Его база действий AB_a включает $2n$ действий ac_{i0}, ac_{i1} $(i = 1, \ldots, n)$. Каждое действие ac_{i1} добавляет к I_a факты OK_i и x_i, а действие ac_{i0} добавляет факт OK_i. Программа P_a для каждого $i \in [1, n]$ содержит два предложения:

$$ac_{i1} \leftarrow\ OK_1, \ldots, OK_{i-1}, \neg OK_i, msg(b, 1) \text{ и}$$
$$ac_{i0} \leftarrow\ OK_1, \ldots, OK_{i-1}, \neg OK_i, \neg msg(b, 1).$$

Из этих определений следует, что после n шагов в I_a окажутся все n фактов вида OK_i $(1 \le \le i \le n)$ и некоторое подмножество фактов из V, причем для каждого такого подмножества имеется соответствующая траектория $\mathcal{A}$. Тогда легко проверить, что для пустого начального состояния $S^0 = \emptyset$ выполняется $\varphi \in SAT \Leftrightarrow$ $\Leftrightarrow \mathcal{T}_{\mathcal{A}}(S^0) \models \mathbf{E}\ (\varphi)$. □

Следующая теорема показывает, что сложность проблемы MA-BEHAVIOR быстро возрастает, если ослабить ограничения на класс рассматриваемых МА-систем. В частности, если в классе расширяющих систем мы ограничим только число агентов или только число сигналов, то проблема станет EXPTIME-полной, как и в общем случае для базисных систем.

Теорема 2. *(1) Проблема MA-BEHAVIOR является EXPTIME-полной для асинхронных, базисных, расширяющих, m-агентных систем и свойств поведения* Φ *из FO-L*μ_1 *(при каждом фиксированном* $m \ge 2$*).*

(2) Проблема MA-BEHAVIOR является EXPTIME-полной для асинхронных, базисных, расширяющих, r-сигнальных систем и свойств поведения Φ *из FO-L*μ_1 *(при каждом фиксированном* $m \ge 2$*).*

(3) В обоих случаях существует такая константа $c > 1$*, что соответствующие версии проблемы MA-BEHAVIOR нельзя решить с детерминированной временной сложностью, меньшей, чем* $c^{n/\log n}$.

(4) Проблема MA-BEHAVIOR для асинхронных, базисных МА-систем:

- *(i) является EXPTIME-полной для свойств поведения* Φ *из FO-L*μ_r *(для любого фиксированного* r*);*
- *(ii) принадлежит классу* $NEXPTIME \cap$ $\cap\, coNEXPTIME$ *для свойств поведения* Φ *из FO-L*μ.

Доказательство этой теоремы получается некоторой модификацией доказательства теоремы 7 из [3].

6.2. Взаимная сводимость проблемы MA-BEHAVIOR для недетерминированных и асинхронных МАС

Следующая теорема показывает возможность моделирования недетерминированных МАС с оператором единичного выбора асинхронными.

Теорема 3. *Пусть для формулировки динамических свойств МАС используется язык FO-L*μ_r*. Тогда проблема MA-BEHAVIOR для недетерминированных МАС с оператором выбора*

Sel^{un} за полиномиальное время сводится к проблеме MA-BEHAVIOR для асинхронных MAC с детерминированными агентами.

Доказательство. Для простоты мы приведем доказательство только для базисных MAC. Доказательство для небазисных MAC аналогично.

Пусть $\mathcal{A} = \{a_1, a_2, \ldots, a_n\}$ – недетерминированная MAC, каждый агент которой $a \in \mathcal{A}$ при недетерминированном выборе исполняемых действий использует оператор Sel^{un}. Пусть S^0 – начальное состояние $\mathcal{A}$. Построим по $\mathcal{A}$, S^0 и FO-Lμ-формуле Φ асинхронную MAC $\mathcal{B}$, ее начальное состояние R^0 и FO-Lμ-формулу Ψ так, что $\mathcal{A}, S^0 \models \Phi \Leftrightarrow \mathcal{B}, R^0 \models \Psi$.

Пусть $Act_i = \{\alpha_{i1}, \ldots, \alpha_{im_i}\}$ – это множество базисных атомов действий агента a_i.

Асинхронная MAC $\mathcal{B}$ состоит из агентов $a'_1, \ldots, a'_n$, двух дополнительных агентов b и c и почтового агента PA. На каждом шаге агент c посылает каждому агенту a'_i три сообщения "1", "2" и "3" (нас будут интересовать траектории, на которых a'_i будет получать эти сообщения в циклическом порядке 1–2–3).

Для каждого действия $\alpha \in Act_i$ у агента a'_i имеется новое сообщение α и новый дубликат этого действия α' со списками $ADD_{\alpha'} = DEL_{\alpha'} = \emptyset$ и $SEND_{\alpha'} = \{(a'_i, b, \alpha)\}$. Программа $P_{a'_i}$ включает:

(i) все предложения P_{a_i}, в которых каждое вхождение всякого атома действия α заменено на α' и в тело каждого предложения добавлен атом $msg(c, a'_i, 1)$;

(ii) предложение

$$\alpha \leftarrow msg(b, a'_i, \alpha'), msg(c, a'_i, 3)$$

для каждого $\alpha \in Act_i$.

Агент b для каждых a'_i и $\alpha \in Act_i$ включает действие α^{a_i} со списками $ADD_{\alpha^{a_i}} = DEL_{\alpha^{a_i}} = \emptyset$ и $SEND_{\alpha^{a_i}} = \{(b, a_i, \alpha)\}$. Программа P_b для всех a'_i и $\alpha \in Act_i$ содержит предложение $\alpha^{a_i} \leftarrow \leftarrow msg(a_i, b, \alpha)$.

Некоторые траектории $\mathcal{B}$ моделируют траектории $\mathcal{A}$, при этом одному шагу $S \Rightarrow_{\mathcal{A}} S'$ системы $\mathcal{A}$ соответствуют три шага системы $\mathcal{B}$.

На первом из этих трех шагов a'_i определяет множество $Perm_{a_i} = Perm_{a_i}(S)$ действий, допустимых в состоянии S, и посылает агенту b множество сообщений $\{msg(a_i, b, \alpha) |\ \alpha \in Perm_{a_i}\}$. На втором шаге агент b посылает множество сообщений

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

агенту a'_i, и одно из этих сообщений обязательно доходит до a'_i.

На третьем шаге a'_i выполняет действие, полученное от b (т.е. изменяет свою внутреннюю БД и посылает всем агентам a'_j все сообщения, которые должны быть посланы в соответствии с этим действием), а PA пересылает агенту a'_i остаток $Perm_{a'_i}$, и эти сообщения "теряются" на следующем шаге.

Начальное состояние $R^{(0)}$ системы $\mathcal{B}$ задается следующим образом: $I^{(0)}_{a'_i} = I^{(0)}_{a_i}$, $Msgbox^{(0)}_{a'_i} = = Msgbox^{(0)}_{a_i} \cup \{msg(c, a_i, 1)\}$, $I^{(0)}_c = I^{(0)}_b = I^{(0)}_{PA} = = Msgbox^{(0)}_b = Msgbox^{(0)}_c = \emptyset$.

Формула $\Psi = \Psi(\Phi, \mathcal{A})$ получается из Φ индуктивной заменой всех подформул вида $\exists X \Theta$ на формулы $\exists X(f_1 \wedge \exists X(f_2 \wedge \exists X(f_3 \wedge \Theta)))$, где f_1, f_2, f_3 имеют вид

$$\begin{array}{l}
\bigwedge_{i=1}^{n}(\bigwedge_{j=1}^{m_i} \neg msg(a'_i, b, \alpha_{ij}) \ \wedge msg^i(c, a'_i, 2) \wedge \\
\wedge \neg msg^i(c, a'_i, 1) \wedge \neg msg^i(c, a'_i, 3)), \\
\bigwedge_{i=1}^{n}(\bigvee_{j=1}^{m_i} msg^i(b, a'_i, \alpha_{ij}) \ \wedge \\
\wedge \bigwedge_{k,l=1;k\neq l}^{m_i}(\neg msg^i(b, a'_i, \alpha_{ik}) \vee \neg msg^i(b, a'_i, \alpha_{il})) \wedge \\
\wedge \ msg^i(c, a'_i, 3) \wedge \neg msg(c, a'_i, 1) \wedge \\
\wedge \ \neg msg^i(c, a'_i, 2)), \\
\bigwedge_{i=1}^{n}(\bigwedge_{j=1}^{n}(\\
\bigwedge_{q^{(r)} \in \mathbf{P}^m_{a_i}} \forall Z_1, ..., Z_r(\neg msg(a'_i, a'_j, q^{(r)}(Z_1, ..., Z_r)) \wedge \\
\wedge \ msg^i(c, a'_i, 1) \wedge \neg msg^i(c, a'_i, 2) \wedge \\
\wedge \neg msg^i(c, a'_i, 3)),
\end{array}$$

соответственно.

Формула f_1 означает, что "каждый агент a'_i получил сообщение 2 от c, и PA не содержит никакого сообщения о действиях, посланного агенту b" (т.е. все сообщения о действиях из $Perm_{a_i}$, посланные агентами a_i, немедленно передаются b).

Формула f_2 означает, что "каждый агент a'_i получил сообщение 3 от c и получил в точности одно сообщение о некотором действии α_{ij} от b" (это сообщение α_{ij} принадлежит множеству $Perm_{a_i}$, которое было послано агенту b агентом a_i на предыдущем шаге).

Формула f_3 означает, что "каждый агент a'_i получил сообщение 1 от c, и ни для какой па-

ры i, j PA не содержит никаких информационных сообщений, посланных агентом a_i агенту a_j" (т.е. все информационные сообщения, посланные агентами a_i, передаются их получателям немедленно).

Из приведенных определений следует, что система $\mathcal{B}$, состояние $R^{(0)}$ и формула Ψ строятся за полиномиальное от размеров $\mathcal{A}, S^{(0)}$ и Φ время.

Определим отношение соответствия между глобальными состояниями $\mathcal{A}$ и $\mathcal{B}$. Скажем, что глобальное состояние

$$R =< (I_{a'_1}, MsgBox_{a'_1}), \ldots, (I_{a'_n}, MsgBox_{a'_n}), (I_b, MsgBox_b), (I_c, MsgBox_c), I_{PA} >$$

системы $\mathcal{B}$ соответствует глобальному состоянию

$$S =< (I_{a_1}, MsgBox_{a_1}), \ldots, (I_{a_n}, MsgBox_{a_n}) >$$

системы $\mathcal{A}$ тогда и только тогда, когда для всех i $I_{a'_i} = I_{a_i}$, и $MsgBox_{a_i}$ получается из $MsgBox_{a'_i}$ после удаления всех сообщений, полученных от агентов b и c.

Из этого определения непосредственно следует, что $R^{(0)}$ соответствует $S^{(0)}$.

Определим отображение G множества вершин $\mathcal{T}_{\mathcal{A}}(S^{(0)})$ во множество вершин $\mathcal{T}_{\mathcal{B}}(R^{(0)})$. Положим $G(S^{(0)}) = R^{(0)}$. Пусть для некоторой вершины S из $\mathcal{T}_{\mathcal{A}}(S^{(0)})$ $G(S) = R$ уже определено так, что сообщение $msg(c, a'_i, 1)$ содержится в $MsgBox_{a'_i}$. Пусть S' – это следующая за S вершина. Тогда $G(S') = R'$, где R' получено с помощью трех шагов $\mathcal{B}$, моделирующих один шаг $S \Rightarrow_{\mathcal{A}} S'$, как это было описано выше.

Отсюда получаем, что для каждого S из $\mathcal{T}_{\mathcal{A}}(S^{(0)})$ состояние $G(S)$ соответствует состоянию S.

Далее мы через $G(\mathcal{A})$ будем обозначать множество $\{G(S) | S \in \mathcal{T}_{\mathcal{A}}(S^{(0)})\}$. Пусть $set_{\mathcal{A}}(\Phi) = \{S | \mathcal{T}_{\mathcal{A}}(S^{(0)}), S \models \Phi\}$, и $set_{\mathcal{B}}(\Psi) = \{R | \mathcal{T}_{\mathcal{B}}(R^{(0)}), R \models \Psi\}$.

Тогда утверждение теоремы непосредственно следует из следующей леммы.

Лемма 5. *Для всякой формулы Φ языка FO-Lμ имеет место равенство $set_{\mathcal{B}}(\Psi(\Phi, \mathcal{A})) \cap G(\mathcal{A}) = \{G(S) | S \in set_{\mathcal{A}}(\Phi)\}$.*

Лемма доказывается индукцией по структуре формулы Φ.

Вначале мы установим следующее предложение.

(#) *Если лемма справедлива для формул Θ и Θ', то она справедлива и для формул $\neg\Theta$, $\Theta \wedge \Theta'$, $\Theta \vee \Theta'$, $\exists X \Theta$, $\forall X \Theta$.*

Для булевых связок оно очевидно.

Для формул вида $\exists X \Theta$ это предложение следует из утверждения:

(*) *Для всякой вершины S дерева $\mathcal{T}_{\mathcal{A}}(S^{(0)})$*

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

тогда и только тогда, когда $\mathcal{T}_{\mathcal{B}}(R^{(0)}), G(S) \models \models \Psi(\exists X \Theta, \mathcal{A})$.

Предположим, что $\Phi = \exists X \Theta$ и $\mathcal{T}_{\mathcal{A}}(S^{(0)}), S \models \models \Phi$. Тогда из определения $\exists X$ следует, что $\mathcal{T}_{\mathcal{A}}(S^{(0)}), S' \models \Theta$ для некоторой вершины S' такой, что $S \Rightarrow_{\mathcal{A}} S'$. Тогда, по индукционному предположению, имеем $\mathcal{T}_{\mathcal{B}}(R^{(0)}), G(S') \models \models \Psi(\Theta, \mathcal{A})$. По определению Ψ имеем: $\Psi(\Phi, \mathcal{A}) = = \exists X(f_1 \wedge \exists X(f_2 \wedge \exists X(f_3 \wedge \Theta(\Phi, \mathcal{A}))))$. Отметим, что из определения G следует, что в $\mathcal{T}_{\mathcal{B}}(R^{(0)})$ имеется путь $G(S) = R \Rightarrow_{\mathcal{B}} R_1 \Rightarrow_{\mathcal{B}} \Rightarrow_{\mathcal{B}} R_2 \Rightarrow_{\mathcal{B}} R_3 = G(S')$.

Ясно, что $\mathcal{T}_{\mathcal{B}}(R^{(0)}), R_i \models f_i, i = 1, 2, 3$. Откуда получаем, что $\mathcal{T}_{\mathcal{B}}(R^{(0)}), G(S) \models \Psi(\Phi, \mathcal{A})$.

Обратно, предположим, что $\mathcal{T}_{\mathcal{B}}(R^{(0)}), G(S) \models \models \Psi(\Phi, \mathcal{A})$. Тогда существует путь $G(S) = R \Rightarrow_{\mathcal{B}} \Rightarrow_{\mathcal{B}} R_1 \Rightarrow_{\mathcal{B}} R_2 \Rightarrow_{\mathcal{B}} R_3$ такой, что (1) $\mathcal{T}_{\mathcal{B}}(R^{(0)})$, $R_i \models f_i, i = 1, 2, 3$, и (2) $\mathcal{T}_{\mathcal{B}}(R^{(0)}), R_3 \models \Psi(\Theta, \mathcal{A})$.

Из (1) следует, что каждый агент a'_i в R_2 выполняет некоторое действие $\alpha_{ij} \in Perm_{a_i}(S)$. Тогда и $a_i \in \mathcal{A}$ может выполнить то же действие в состоянии S, так как его оператор выбора $Sel^{un}_{a_i}$ может в $Perm_{a_i}(S)$ выбрать любое действие для исполнения. После этого система $\mathcal{A}$ переходит в состояние S' такое, что $G(S') = = R_3$. Тогда по индукционному предположению $\mathcal{T}_{\mathcal{A}}(S^{(0)}), S' \models \Theta$ и, следовательно, $\mathcal{T}_{\mathcal{A}}(S^{(0)}), \models \models \exists X \Theta$. Таким образом, мы доказали утверждение (*).

Для формул $\Phi = \forall X \Theta$ доказательство следует из эквивалентности $\forall X \Theta$ и $\neg\exists X \neg\Theta$.

Следовательно, доказано предложение (#).

Если формула Φ – базисная, то утверждение леммы следует из соответствия S и $G(S)$. Таким образом, из предложения (#) мы заключаем, что лемма верна для произвольной формулы Φ из FO-Lμ_0 (т.е. для формулы, не содержащей операторов μ и ν).

Если Φ имеет вид $\mu Z.\Theta(Z)$ или $\nu Z.\Theta(Z)$, то лемма доказывается прямой, но достаточно громоздкой индукцией по вычислению неподвижных точек этих формул. В качестве примера мы рассмотрим только следующий простой случай.

Предположим, что Φ имеет вид $\mu Z.\Theta(Z)$, где $\Theta(Z)$ не содержит μ и ν. Тогда $\Psi(\Phi,\mathcal{A}) = \mu Z.\Psi(\Theta(Z),\mathcal{A})$. По определению μ имеем: $set_{\mathcal{A}}(\Phi) = \bigcup_{i=0}^{\infty} set_{\mathcal{A}}(\Theta^i(false))$, и $set_{\mathcal{B}}(\Psi(\Phi,\mathcal{A})) = \bigcup_{i=0}^{\infty} set_{\mathcal{B}}((\Psi(\Theta,\mathcal{A}))^i(false))$. Отсюда получаем, что $\Psi(\Theta^i(false),\mathcal{A}) = (\Psi(\Theta,\mathcal{A}))^i(false)$, и $set_{\mathcal{B}}((\Psi(\Theta,\mathcal{A}))^i(false)) \cap G(\mathcal{A}) = \{G(S) | S \in set_{\mathcal{A}}(\Theta^i(false))\}$. Тогда $set_{\mathcal{B}}(\Psi(\Phi,\mathcal{A})) \cap G(\mathcal{A}) = \{G(S) | S \in set_{\mathcal{A}}(\Phi)\}$.

Для формул Φ вида $\nu Z.\Theta(Z)$, где $\Theta(Z)$ не содержит μ и ν, доказательство аналогично. □

Следующая теорема показывает, как асинхронные МАС можно моделировать недетерминированными МАС.

Теорема 4. *Пусть для формулировки динамических свойств МАС используется язык FO=Lμ_r. Тогда проблема MA-BEHAVIOR для асинхронных недетерминированных МАС за полиномиальное время сводится к проблеме MA-BEHAVIOR для (синхронных) недетерминированных МАС.*

Доказательство. Пусть $\mathcal{A} = \{a_1,\ldots,a_n; PA\}$ – недетерминированная асинхронная МАС, а Φ – проверяемая формула. Мы построим по $\mathcal{A}$ недетерминированную МАС $\mathcal{B} = \{a'_1, \ldots, a'_n, pa\}$, которая моделирует работу $\mathcal{A}$. Недетерминированный агент pa будет моделировать работу почтового агента PA, сохраняя сообщения в своей базе данных, а затем недетерминированно пересылая их получателям.

Для каждого предиката из сигнатуры сообщений $q \in \mathbf{P}^m_{a_i}$ и каждого $j \neq i$ мы поместим в сигнатуру $\mathbf{P}^m_{a'_i}$ предикат q_{ij}. Головы α предложений логических компонент агентов a'_i те же, что и в программах агентов a_i, но каждое сообщение вида $msg(a_i,a_j,q)$ в списке $SEND_\alpha$ заменяется на $msg(a'_i,pa,q_{ij})$. В телах предложений программ P_{a_i} каждый атом вида $msg(a_j,a_i,q)$ заменяется на $msg(pa,a'_i,q_{ji})$. Кроме того, к телу каждого предложения добавляется новый атом $msg(pa,a'_i,1)$.

База действий AB_{pa} недетерминированного "почтового" агента pa для каждого $q^{(k)}_{ij}$ включает три действия $save(q_{ij}), resend(q_{ij})$ и $send(q_{ij})$ со следующими списками:

$$
\begin{aligned}
&ADD_{save(q_{ij})} = \{q_{ij}(X_1,\ldots,X_k)\},\\
&DEL_{save(q_{ij})} = SEND_{save(q_{ij})} = \emptyset,\\
&ADD_{resend(q_{ij})} = DEL_{resend(q_{ij})} = \emptyset,\\
&SEND_{resend(q_{ij})} =\\
&= \{msg(pa,a_j,q_{ij}(X_1,\ldots,X_k)\},\\
&ADD_{send(q_{ij})} = \emptyset,\\
&DEL_{send(q_{ij})} = \{q_{ij}(X_1,\ldots,X_k)\},\\
&SEND_{send(q_{ij})} =\\
&= \{msg(pa,a_j,q_{ij}(X_1,\ldots,X_k))\}.
\end{aligned}
$$

AB_{pa} также содержит два действия add_1 и del_1 для счета четных и нечетных шагов:

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

Для запуска этих действий в программу P_{pa} агента pa включены два предложения:

$$
\begin{aligned}
&add\_1 \leftarrow \neg 1.\\
&del\_1 \leftarrow 1.
\end{aligned}
$$

Для каждого $q^{(k)}_{ij}$ программа P_{pa} содержит три предложения:

$$
\begin{aligned}
&save(q_{ij})(X_1,\ldots,X_k) \leftarrow\\
&\leftarrow msg(a_i,pa,q_{ij}(X_1,\ldots,X_k).\\
&resend(q_{ij})(X_1,\ldots,X_k) \leftarrow\\
&\leftarrow msg(a_i,pa,q_{ij}(X_1,\ldots,X_k).\\
&send(q_{ij})(X_1,\ldots,X_k) \leftarrow q_{ij}(X_1,\ldots,X_k).
\end{aligned}
$$

Оператор выбора Sel_{pa} выбирает из $Perm_{pa}$ произвольное подмножество действий вида $send(q_{ij})(t_1,\ldots,t_k)$ и из каждой пары атомов действий вида $\{save(q_{ij})(t_1,\ldots,t_k), resend(q_{ij})(t_1,\ldots,t_k)\} \subseteq Perm_{pa}$ выбирает один атом.

По конструкции $\mathcal{B}$ видно, что один шаг системы $\mathcal{A}$ моделируется двумя шагами системы $\mathcal{B}$.

Выполнимость формулы Φ в $\tau(\mathcal{A},S^0)$ можно свести к выполнимости некоторой формулы Ψ языка FO-Lμ_r в $\tau(\mathcal{B},S^0)$. Формула $\Psi = \Psi(\Phi,\mathcal{B})$ строится индуктивно, как и в предыдущей теореме, но несколько проще.

Это сведение, как следует из конструкции, можно выполнить за полиномиальное время. □

Из этих теорем о моделировании можно, в частности, получить следующие утверждения о сложности верификации асинхронных МАС.

Следствие 5. *Проблема MA-BEHAVIOR для базисных асинхронных MAC с детерминированными агентами*:

1) *является $EXPTIME$-трудной при верификации формул из FO-Lμ_r, для каждого фиксированного $r \geq 1$,*

2) *принадлжит классу $EXPTIME$ при верификации формул из FO-Lμ.*

Следствие 6. *Проблема MA-BEHAVIOR для небазисных асинхронных MAC с детерминированными агентами*:

1) *является $EXPEXPTIME$-трудной при верификации формул из FO-Lμ_r, для каждого фиксированного $r \geq 1$, и*

2) *принадлежит классу $EXPEXPTIME$ при верификации формул из FO-Lμ.*

7. ЗАКЛЮЧЕНИЕ

Рассмотренные в этой статье MAC представляют один из классов общих параллельных и/или распределенных программных систем. Поэтому к анализу их поведения приложимы многие из известных общих подходов к верификации параллельных программ. Однако архитектура MAC обладает определенной спецификой, что приводит к необходимости существенного видоизменения этих подходов.

Для MAC, с их богатой архитектурой, адекватность анализа поведения тесно связана с точным выбором уровня детализации существенных свойств этой архитектуры и ее параметров, а также с выбором естественных ограничений на эти свойства и параметры. Мы выбрали для исследования специальный фрагмент архитектуры IMPACT [9], в котором интеллектуальные компоненты агентов описываются логическими программами. Рассматриваемые MAC могут быть синхронными или асинхронными и детерминированными или недетерминированными в зависимости от режима работы почтовой подсистемы и от семантики выполнения агентами одного шага. Следуя известным подходам к верификации моделей программ (см. [22, 24]), для представления свойств поведения различных вариантов MAC мы использовали различные классы временных логик (линейного или ветвящегося времени).

Для каждого из основных указанных выше классов MAC мы выделили некоторые естественные подклассы, используя структурные ограничения: на число агентов, число сообщений, на размерность (арность) действий и сообщений. Мы рассмотрели также некоторые существенные семантические условия, ограничивающие выразительность программ и результаты действий агентов: использование переменных и отрицаний в логических программах, возможность удаления фактов из состояний агентов.

Нашей целью было определение сложности соответствующей проблемы MA-BEHAVIOR для различных комбинаций этих ограничений. Для синхронных детерминированных и недетерминированных MAC основные результаты, полученные в работах [1–3], приведены здесь в таблицах 1 и 2. В данной работе основное внимание уделено поведению асинхронных MAC. Часть результатов для этого класса удалось перенести с недетерминированного случая за счет теорем 3 и 4 о взаимном моделировании недетерминированных и асинхронных MAC. Но конструкции этих теорем не сохраняют таких свойств MAC, как расширяемость и позитивность. Поэтому для получения “низких” оценок сложности для соответствующих подклассов MAC потребовались отдельные конструкции (теоремы 1 и 2).

Полученные нами результаты показывают, что сложность проблемы MA-BEHAVIOR для “неограниченных” классов MAC весьма велика, в частности, требует экспоненциальной памяти или времени, ограниченного двойной экспонентой. Вместе с тем, для некоторых естественным образом ограниченных подклассов MAC проблема MA-BEHAVIOR имеет сравнительно небольшую сложность – решается за детерминированное или недетерминированное полиномиальное время или с полиномиальной памятью.

Хотя рассмотренная в работе архитектура MAC существенно проще исходной архитектуры IMPACT, определенной в [9], наши результаты распространяются и на более общий слу-

чай. В частности. это касается возможности расширения интеллектуальных компонент агентов за счет использования в логических программах деонтических модальностей типа P ("permitted"), F ("forbidden"), O ("obliged") и т.д. (см. [9]). Аналогичное утверждение можно сделать для ряда других средств из [9], оставшихся здесь нерассмотренными. Но, как было отмечено во введении, само описание этих средств уже достаточно громоздко. Некоторые наши результаты о вероятностных MAC содержатся в [31].

Естественно, представляет интерес и вопрос о сложности поведения для других архитектур MAC. Одним из способов изучения этого вопроса может быть моделирование MAC различных архитектур системами архитектуры IMPACT. В частности, в работе [32] обсуждается возможность моделирования в этой архитектуре MAC, определенных в системе AgentSpeak [33], основанной на использовании так называемых BDI-агентов (Belief-Desire-Intention). Эти агенты вырабатывают планы своих действий, основываясь на некоторых формально определенных наборах убеждений (фактов внутренней БД) и намерений. Проблема верификации для AgentSpeak-систем рассматривалась в литературе (см. [13]), однако без оценки сложности соответствующих алгоритмов. Возможность их моделирования на рассмотренном нами типе систем позволяет получить верхние оценки сложности верификации AgentSpeak-систем.

Существенно другой тип представляют агенты, определяемые с помощью эпистемических логик [16] без использования программных компонент (в частности, и логических, как у нас), которые позволяют выражать знания, основанные на неполной информации. При этом можно, меняя доступность той или иной информации, проверять, как меняется у агента представление о мире и о других агентах. Некоторые результаты о верификации систем с такого рода агентами, в которых логика знаний дополнительно комбинируется с временными логиками, получены в [34, 35]. Такие логики позволяют описывать эволюцию знаний агентов во времени.

В заключение отметим, что мы рассмотрели здесь "наивные" варианты алгоритмов проверки формул на моделях, оставив на дальнейшее применение различных оптимизационных методов, таких, как символическая проверка, абстракция, использование свойств симметрии MAC и др. (см. [24]).

Что касается практической разработки систем верификации MAC, то полученные нами результаты показывают, что полностью автоматизировать верификацию для систем общего вида нельзя, поэтому такая система должна давать возможность трассировать работу MAC и применять специальные приемы для верификации систем из их подклассов.

СПИСОК ЛИТЕРАТУРЫ

1. *Валиев М.К., Дехтярь М.И., Диковский А.Я.* О сложности поведения систем взимодействующих агентов. Труды конференции, посвященной 90-летию со дня рождения А.А. Ляпунова. Новосибирск: Наука, 2001. С. 18–28.
2. *Dekhtyar M., Dikovsky A., Valiev M.* On Feasible Cases of Checking Multi-Agent Systems Behavior // Theoretical Computer Science. 2003. V. 303. №1. P. 63–81.
3. *Dekhtyar M.I., Dikovsky A.Ja., Valiev M.K.* On complexity of verification of interacting agents' behavior // Annals of Pure and Applied Logic. 2006. V. 141. №3. P. 336–362.
4. *Barringer H., Fisher M., Gabbay D., Gough G., Owens R.* METATEM: An Introduction // Formal Aspects of Computing. 1995. V. 7. P. 533–549.
5. *Georgeff M., Lansky A.* Reactive Reasoning and Planning. Proc. of the Conf. of the American Assoc. of Artificial Intelligence. 1987. Seattle, WA. P. 677–682.
6. *van der Hoek W., Wooldridge M.* Multi-Agent Systems / van Harmelen F., Lifschitz V., Porter B. (eds.) Handbook of Knowledge Representation. Elsevier, 2008. P. 887–928.
7. *Reiter R.* Knowledge in action: Logical Foundations for Specifying and Implementing Dynamical Systems. MIT Press, 2001.
8. *Shoham Y.* Agent oriented programming // Artificial Intelligence. 1993. №60. P. 51–92.
9. *Subrahmanian V.S., Bonatti P., Dix J.* et al. Heterogeneous Agent Systems. MIT Press, 2000.

10. *Тарасов В.Б.* От многоагентных систем к интеллектуальным организациям. М.: Эдиториал УРСС, 2002.

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

12. *Benerecetti M., Guinchiglia F., Serafini L.* Model Checking Multiagent Systems. Technical Report №9708-07. Instituto Trentino di Cultura, 1998.

13. *Bordini R.H., Fisher M., Pardavila C., Wooldridge M.* Model Checking AgentSpeak. Proceedings of the Second International Conference on Autonomous Agents and Multiagent Systems (AAMAS-03). Melbourne, Australia. July 2003.

14. *Wooldridge M., Dunne P.E.* The Computational complexity of Agent Verification / Meyer J.-J., Tambe M. (eds.) Intelligent Agents VIII // Lecture Notes in AI. Volume ? Springer-Verlag, March 2002.

15. *Wooldridge M., Huget M.-P., Fisher M., Parsons S.* Model Checking Multi-Agent Systems: The MABLE Language and Its Applications // International Journal on Artificial Intelligence Tools. April 2006. V. 15. №2. P. 195–225. (Предварительная версия: *Wooldridge M., Fisher M., Huget M.-P., Parsons S.* Model Checking Multiagent systems with MABLE. Proceedings of the First International Conference on Autonomous Agents and Multiagent Systems (AAMAS-02). Bologna, Italy. July 2002).

16. *Fagin R., Halpern J.Y., Moses Y., Vardi M.Y.* Reasoning about Knowledge. MIT Press, 1995. 477 p.

17. *Eiter T., Subrahmanian V.S.* Heterogeneous active agents. III: Polynomially implementable agents. Techn. Rept. INFSYS RR-1843-99-07. 1999. Inst. für Informationssysteme, Technische Universität Wien. A-10-40, Vienna, Austria.

18. *Clarke E.M., Emerson E.A.* Design and synthesis of synchronization skeletons using branching time temporal logic. Proc. of Workshop on Logics of Programs // Lecture Notes in Computer Science. 1981. №181. P. 52–71.

19. *Vardi M., Wolper P.* An automata-theoretic approach to automatic program verification. Proc. of the IEEE Symposium on Logic in Computer Science. 1986. P. 332–344.

20. *Queille J.P., Sifakis J.* Specification and verification of concurrent programs in CESAR. Proc. of the 5th International Symposium on Programming // Lecture Notes in Computer Science. 1982. №137. P. 195–220.

21. *Sistla A.P., Clarke E.M.* The complexity of propositional linear temporal logic // J. ACM. 1985. V. 32. №3. P. 733–749.

22. *Emerson E.A.* Temporal and modal logic / van Leeuwen J. (ed.) Handbook of Theor. Comput. Sci. Elsevier Sci. Publishers, 1990.

23. *Emerson E.A.* Model checking and the mu-calculus / Immerman N., Kolaitis P.H. (eds.) Descriptive Complexity and Finite Models. Proc. of the DIMACS Workshop. 1996. P. 185–214.

24. *Clarke E.M., Grumberg O., Peled D.* Model Checking. MIT Press, 2000. (Русский перевод: *Кларк Э.М., Грамберг О., Пелед Д.* Верификация моделей программ: Model Checking. М.: МЦНМО, 2002).

25. *Apt K.R.* Logic Programming / van Leeuwen J. (ed.) Handbook of Theoretical Computer Science. Volume B. Formal Models and Semantics. Chapter 10. Elsevier Science Publishers B.V., 1990. P. 493–574.

26. *Kozen D.* Results on the Propositional μ-calculus // Theoretical Computer Science. 1983. V. 27. P. 333–354.

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

28. *Ии К., Шилов Н.В., Бодин Е.В.* О программных логиках – просто / Системная информатика. Выпуск 8. Новосибирск: Наука, 2002. С. 206–249.

29. *Garey M., Johnson D.S.* Computers and intractability – a guide to the theory of NP-completeness. New York: W.H. Freeman, 1979.

30. *Papadimitriou C.H.* Computational complexity. Addison Wesley, 1994.

31. *Dekhtyar M.I., Dikovsky A.Ja., Valiev M.K.* Temporal Verification of Probabilistic Multi-Agent Systems. Pillars of Computer Science: Essays Dedicated to Boris (Boaz) Trakhtenbrot on the Occasion of His 85th Birthday. Avron A., Dershowitz N., Rabinovich A. (eds.) // Lecture Notes in Computer Science. Berlin: Springer-Verlag, 2008. V. 4800.

32. *Бурмистров М.Ю., Валиев М.К., Дехтярь М.И., Диковский А.Я.* О верификации динамических свойств систем взаимодействующих агентов. Труды X национальной конференции по искусственному интеллекту с международным участием. Обнинск, Москва: Физматлит, 2006. С. 908–915.

33. *Rao A.S.* AgentSpeak (L): BDI agents speak out in a logical computable language // Lect. Notes in AI. Berlin: Springer-Verlag, 1996. №1038.

34. *Гаранина Н.О., Шилов Н.В.* Верификация комбинированных логик знаний, действий и времени в моделях / Системная информатика. Выпуск 10. Новосибирск: Издательство СО РАН, 2006. С. 114–173.

35. *Shilov N.V., Garanina N.O., Choe K.-M.* Update and Abstraction in Model Checking of Knowledge and Branching Time // Fundamenta Informaticae. 2006. V. 72. №1–3. P. 347–361.

36. *Emerson E.A., Lei C.L.* Efficient model checking in fragments of the mu-calculus. Proc. of the IEEE Symposium on Logic in Computer Science. 1986.