ПОДХОД К ПРОГРАММИРОВАНИЮ АГЕНТОВ В МУЛЬТИАГЕНТНЫХ СИСТЕМАХ

Дмитрий Черемисинов, Людмила Черемисинова

Аннотация. *Рассмотрены методы спецификации протоколов взаимодействия агентов в мультиагентных системах на языке ПРАЛУ параллельных алгоритмов логического управления, который обладает средствами для представления последовательности состояний диалога, приема и отправки сообщений. Показано, что описание поведения агентов на языке ПРАЛУ позволяет моделировать поведение мультиагентной системы целиком. Предложена методология программирования агентов ПРАЛУ, использующая двухблочную архитектуру: блок синхронизации и функциональный блок. Оригинальной компонентой этой методологии является средство автоматической трансляции блока синхронизации по описанию на ПРАЛУ.*

Ключевые слова*: протокол взаимодействия, BDI агент, онтология, параллельный алгоритм.*

ACM Classification Keywords*: I.2.11 [Computer Applications]; Distributed Artificial Intelligence, Multiagent systems; D.3.3 [Programming Languages]: Language Constructs and Features – Concurrent programming structures.*

Conference*: The paper is selected from XIVth International Conference "Knowledge-Dialogue-Solution" KDS 2008, Varna, Bulgaria, June-July 2008*

Введение

Мультиагентные системы (МАС) состоят из множества искусственных агентов, которые работают совместно, чтобы достигнуть некоторых целей [1]. Агент представляет собой открытую систему, помещенную в некоторую среду, причем агенты обладают собственным поведением, удовлетворяющим определенным правилам. Примерами искусственных агентов служат роботы. МАС можно рассматривать как организацию агентов (по аналогии с человеческой организацией) или, другими словами, как некоторое искусственное сообщество. Технология программирования на основе использования взаимодействующих агентов считается наиболее перспективным инструментом современного программирования. Наиболее известной промышленной системой, построенной на основе концепции агентов и предназначенной для управления процессом производства изделий на предприятии, является ARCHON [1].

Агенты МАС характеризуются процессами, которые происходят во время их работы и определяются описанием их потенциального поведения. Взаимодействие агентов предполагает обмен сообщениями между ними. Множество взаимосвязанных сообщений образует переговоры в МАС, которые основаны на протоколах взаимодействия, определяющих все возможные течения переговоров.

В большинстве моделей МАС поведение агентов описывается в терминах убеждений, желаний и намерений (beliefs, desires and intentions – BDI) [2], то есть на основе понятий из социологии, а коммуникация задается в терминах протоколов, которые не имеют прямой связи с первыми. Одна из проблем в связи таким подходом состоит в том, что чрезвычайно трудно разрабатывать и моделировать коммуникацию между агентами. В большинстве моделей МАС автономное поведение агентов описывается с использованием формализмов высокого уровня абстрактности, а коммуникация задается в понятиях, близких к реализации. Разница в уровнях описания не позволяет моделировать коммуникацию между агентами на том уровне, на котором описано их автономное поведение. Эта проблема возникает вследствие отсутствия модели агента, объединяющей аспекты внутреннего состояния и коммуникации. Главная причина отсутствия общей модели состоит в том, что отсутствует общее концептуальное основание, объединяющее все абстракции, связанные с агентами.

Для преодоления этих методологических трудностей используются такие языки представления теории агентов, в основе которых лежит некий формализм и система программирования, задающие семантику

языка программирования агентов. Хотя в литературе предлагаются все новые языки программирования агентов, но немногие из них полностью понятны с семантической точки зрения.

В работе предлагается методология программирования агентов МАС, в основе которой лежит язык ПРАЛУ описания параллельных алгоритмов логического управления [3]. ПРАЛУ как язык программирования агентов имеет то преимущество, что он имеет в качестве семантики логический формализм и допускает простую реализацию. ПРАЛУ обладает средствами для представления последовательности состояний диалога, приема и отправки сообщений. Компактность представления программ и простота синтаксиса являются факторами, значительно упрощающими реализацию языка. Показано, что описание поведения агентов на языке ПРАЛУ позволяет моделировать поведение МАС целиком. В основе предлагаемой методологии лежит двухблочная архитектура организации описания и реализации МАС, состоящая из блока синхронизации и функционального блока.

Формализмы задания протоколов взаимодействия агентов

Протокол – это набор правил, которым соответствует взаимодействие, имеющее место при координации работы нескольких агентов. Формальные модели протоколов изучались в рамках теории распределенных вычислений. Фундаментальный признаком, по которому отличаются эти модели, является степень синхронизации поведения участников взаимодействия. Если абстрагироваться от назначения агентов, то единственной целью взаимодействия является синхронизация поведения взаимодействующих агентов, так как ненадлежащая синхронизация полностью разрушает целесообразность совместной работы агентов. Достижение синхронизации требует специальной организации взаимодействующих процессов.

Причинно-следственные и временные отношения являются главными характеристиками протоколов. Анализ этих зависимостей позволяет установить возможные последовательности возникновения событий при функционировании протокола, что дает возможность выявить, реализуются ли при выполнении протокола желательные события и обнаружить ошибки, вызывающие нежелательные события.

Когда агенты вовлечены во взаимодействие, где параллелизм не допустим, протоколы традиционно задаются детерминированными конечными автоматами. Самым простым из других представлений протокола является диаграмма потока сообщений, такая как используется в стандарте FIPA [4]. Для более сложных протоколов лучшим представлением является диаграмма взаимодействия таких языков, как UML (Uniffied Modelling Language – универсальный язык моделирования) [5], AUML [6] и цветных сетей Петри (CPN – Coloured Petri Nets) [7]. UML – один из наиболее популярных в настоящее время графических языков проектирования, который де-факто является стандартом для описания систем программного обеспечения. Язык AUML – это расширение UML для представления асинхронного обмена сообщениями между агентами. Представление протоколов практической сложности на языках автоматов, UML или AUML, к сожалению, требует существенных усилий для реализации агента, его отладки и понимания правил поведения.

Модель системы в виде CPN описывает множество состояний, в которых может находиться система, и переходы между этими состояниями. Формализм CPN обеспечивает математический базис для описания, реализации и анализа распределенных и параллельных систем, может выражать взаимодействия в графической форме и обладает строгой семантикой, что позволяет автоматизировать формальный анализ и преобразования описаний [7]. Используя CPN, протокол взаимодействия агентов может быть задан сетью компонентов, задающей структуру протокола. В CPN состояния взаимодействия агентов представляются местами CPN, передаваемые при взаимодействии данные – символами, значения которых указываются их цветами. Последовательность взаимодействий задают переходы и связанные с ними дуги. Переход допустим, если все его входные места имеют символы и цвета этих символов удовлетворяют ограничениям, которые определены на дугах. Если переход допустим, то он может быть запущен, а определяемые им действия выполнены. После выполнения перехода, состояние (маркировка) сети изменяется, и работа протокола заканчивается, когда нет допустимых или запущенных переходов.

Имеется множество работ по использованию обычных [8] или цветных сетей Петри для представления протоколов взаимодействия агентов, считается [7], что CPN – это один из лучших способов представления протоколов взаимодействия агентов. Однако понятие выполнения действия агентом в сети Петри не имеет явного представления [9], каждую роль агента нужно задавать отдельной сетью,

некоторые ситуации в поведении агентов не могут быть выражены стандартной сетью Петри. Все это значительно усложняет проектирование всего протокола в целом.

Спецификация протоколов взаимодействия агентов на ПРАЛУ

Язык ПРАЛУ удобно использовать для описания систем, характеризующихся сложным взаимодействием, асинхронностью и параллелизмом. Он (основан на расширенных сетях свободного выбора [10]) объединяет возможности моделей «если–то» с возможностями сетей Петри и обладает средствами для представления последовательности текущих состояний диалога, приема и отправки сообщений. С помощью языка ПРАЛУ [3] возможно описание временной упорядоченности событий, возникающих при реализации протокола, абстрагируясь от всех деталей, кроме тех, что выражаются причинно-следственными и временными отношениями. Алгоритмы на ПРАЛУ представляются в виде причинно-временных зависимостей между событиями, происходящими в МАС.

Основными операциями языка ПРАЛУ являются *операции ожидания и действия*. Операция ожидания «$-p_i$» сводится к ожиданию наступления некоторого события p_i, представленного конъюнкцией логических переменных, и ограничивается проверкой условия его истинности, завершаясь после ее выполнения. Операция действия «$\rightarrow A_i$» приводит к наступлению некоторого события, представленного также конъюнкцией логических переменных, в описываемом объекте (каким-то изменениям его состояния) и выполняется в течение некоторого промежутка времени после ее инициализации.

Алгоритм управления представляется *неупорядоченной совокупностью предложений*. Каждое предложение состоит из одной или нескольких одинаково помеченных цепочек «μ_i: $l_i \rightarrow \nu_i$»; через l_i обозначен некоторый линейный алгоритм, составленный из операций языка; μ_i и ν_i – начальная и конечная метки, которыми служат непустые подмножества из множества $M = \{1, 2, \ldots, m\}$ целых чисел. Допускается $\nu_i = \varnothing$, что обозначается как «$\rightarrow$.» и является концом реализации алгоритма.

Порядок выполнения цепочек алгоритма управления в процессе его реализации определяется множеством N запуска, его текущие значения $N_t \subset M$. Среди предложений алгоритма выделяется одно – начальное, его метка заносится в N перед реализацией алгоритма.

В процессе реализации алгоритма управления цепочки запускаются независимо друг от друга. Если в некоторый момент времени для некоторой цепочки «μ_i: $l_i \rightarrow \nu_i$» выполняется условие $\mu_i \subseteq N_t$ и реализуется событие p_i, с ожидания которого начинается цепочка l_i, то она запускается. При этом N_t заменяется на $N_t \setminus \mu_i$, а после завершения цепочки N_t становится равным $(N_t \setminus \mu_i) \cup \nu_i$. Предложенный механизм достаточен для отображения *альтернативного ветвления и распараллеливания* процессов. Синтаксически параллельный алгоритм характеризуется наличием меток $|\mu_i| > 1$, $|\nu_i| > 1$. Альтернативное ветвление обеспечивается ограничением $(i \neq j) \wedge (\mu_i \cap \mu_j \neq \varnothing) \rightarrow (p_i \wedge p_j = 0)$.

Язык ПРАЛУ поддерживает *иерархическое описание алгоритмов*, которое является особенно важным в случае описания сложных систем. Для обеспечения реакции устройства управления на некоторые особые события, происходящие в системе, в язык введена операция гашения в трех модификациях: «$\rightarrow *$», «$\rightarrow *\gamma$» и «$\rightarrow *\gamma'$», где $\gamma \subseteq M$. Ее действие заключается в прекращении реализации всех или некоторых (из множества γ или $\gamma' = N_t \setminus \gamma$) активных цепочек алгоритма. Наряду с булевыми в ПРАЛУ допустимо использование и арифметических переменных, в частности введены операции: задержки "$-n$" – выдержки n единиц времени; счета событий: «$\rightarrow (x = n)$» – присвоения многозначной переменной x натурального значения n; «$\rightarrow (x +)$» и «$\rightarrow (x -)$» – единичного положительного и отрицательного «$\rightarrow (x -)$» приращения значения; «$-(x = n)$» – ожидания наступления события: значение x равно n.

В работе [11] приведен пример описания на языке ПРАЛУ протокола мультиагентной системы, представляющей английский аукцион [4].

BDI агенты

Любая реальная МАС открыта – агенты работают в изменяющейся окружающей среде, которая воздействует на агентов и изменяется вследствие их работы. Система должна принимать решения, чтобы достичь поставленных перед ней целей, для этого она должна иметь модель среды, в которой развивается ее поведение. В области разработки моделей и архитектур агентов доминирующую роль

играет архитектура BDI, в которой агент рассматривается как социальное существо, общающееся с другими агентами посредством некоторого языка и играющее в обществе определенную роль, зависящую от убеждений (Beliefs), желаний (Desires) и намерений (Intentions) [2]. Считаются, что эти три компоненты полностью задают состояние «ума» социального агента.

В терминах программирования, убеждения BDI-агента представляют собой знания (информацию), которые имеет агент о состоянии окружающей среды и которые обновляются после каждого его действия. Желания обозначают цели, к которым стремится агент, включая их приоритеты. Намерения определяют действия, которые должны быть выполнены, чтобы достичь цели (образцы поведения). Протоколы взаимодействия позволяют агенту сократить пространство поиска возможных решений, определяя ограниченный диапазон ответов на сообщения, возможные для данной ситуации.

Онтология архитектуры BDI для языка ПРАЛУ

В практическом программировании агент – это оформленная в оболочку компьютерная система, которая расположена в некоторой окружающей среде и предназначена для гибких, автономных действий в этой среде с целью достижения заданных целей. Агенты отличаются от обычного программного обеспечения сложностью сценариев взаимодействия и коммуникации.

Онтологии – это явные формальные спецификации терминов предметной области и отношений между ними. Этот термин социальных наук, используемый в теории агентов, почти эквивалентен принятому в программировании понятию семантики языка в определенной предметной области. Задачей онтологии в нашем случае является определение терминов архитектуры BDI в понятиях языка ПРАЛУ.

Можно считать, что агент состоит из множества убеждений *B*, планов *P*, ситуаций *E*, действий *A* и намерений *I*. Когда агент замечает изменение в окружающей среде, он считает, что произошло событие, представляющее собой некоторую ситуацию внешней среды из *E*. Регистрация события агентом состоит в изменении состояния его «ума»: выбора некоторого убеждения из *B*. В соответствии с ним и желанием (определяемым некоторым планом из *P*) агент намеревается выполнить некоторые намерения из *I*, представляющие последовательности действий из *A*. Эти действия составляют план достижения поставленной цели. Таким образом, планируемое действие определяется выбранным планом из *P*. Затем планируемые действия осуществляются, изменяя текущую ситуацию в окружающей среде.

В традиционных параллельных языках программирования основными понятиями являются данные и управление вычислениями. Данные представляются значениями переменных, а управление задается набором процессов, которые трансформируют локальные состояния их памяти и задаются отображением переменных в их значения. Концепция интеллектуального агента вводит в параллельном программировании новые представления о манипуляции данными. Информационные состояния агента вместо отображений переменных в значения задаются более сложными информационными структурами логики предикатов первого порядка или модальной логики. Вычисления как трансформации локальных состояний памяти процессов представляют трансформации состояний «ума» агента. При программировании агентов это имеет два важных последствия. Во-первых, понятие присваивания значения переменной, на котором основаны традиционные языки программирования, нужно заменить операторами информационного обмена; во-вторых, механизм рассылки сообщений в MAC должен быть заменен механизмом, который обеспечивал бы обмен информацией между агентами в соответствии с некоторым протоколом. Таким образом, вычисления как трансформации состояний памяти заменяются протоколами коммуникации агентов.

Поведение агента (т.е. его взаимодействие с окружающей средой как внешнее поведение и процессы как внутреннее поведение) диктуется программой агента. Убеждения, желания и намерения агента в программе на ПРАЛУ не представляются явно как модальные формулы. Вместо этого проектировщик программы должен описать эти понятия теории агента внутри этой программы. Текущее состояние агента, которое объединяет состояние его «ума», окружающей среды и состояния других агентов, можно рассматривать как текущее состояние его убеждений. Состояния, которые агент собирается осуществить, основываясь на внешних или внутренних стимулах, можно рассматривать как его желания. Выбор плана действий для достижения поставленных целей можно рассматривать как намерения агента. Это изменение во взгляде на конструкции языка ПРАЛУ делает его не только языком спецификации, но и обеспечивает выполнимость описания поведения агента. Кроме того, мышление о поведении агента в

терминах убеждений, желаний и намерений, вероятно, дает хороший шанс для применения ПРАЛУ для программирования агентов, объединяя понятия теории МАС и практики проектирования параллельных алгоритмов управления.

Цель действия агента – это состояние окружающей среды, которое он хочет вызвать. На ПРАЛУ намерение достичь цели описывается операцией действия «$\rightarrow g$» (*DONE g*). Действия – основные единицы программы на ПРАЛУ, их выполнение является средством достижения цели. Действие можно выполнить, если некоторое убеждение оказалось верным. На ПРАЛУ проверка такого условия описывается операцией ожидания «– *a*» (*HAPPENS a*) события *a*. Предполагается, что агент выполняет действие, следующее за операцией «– *a*», только после того момента, когда «убеждение *a*» становится верным.

Программа агента является набором планов, определяющих средства, с помощью которых агент должен достичь конечной цели функционирования. План состоит из головных меток, тела и хвостовых меток. Тело плана – это последовательность действий, которая определяет цели, которые агент должен достигнуть, и условия, которые агент должен проверить. Головные и хвостовые метки плана символизируют намерения. Критическим понятием для задания поведения агента является понятие активного намерения. Предполагается, что агент имеет предопределенный список намерений. В каждый момент, когда агент выбирает некоторый план выполнения, намерение может быть активным или несущественным. В начале выполнения активно специальное начальное намерение *Init*. План будет выполняться только тогда, когда все намерения из числа его головных будут активны. Агент переводит эти намерения в несущественные, когда план выполнен, и делает активными все хвостовые намерения плана. Текущий набор активных намерений всегда не пуст. Тело плана и набор хвостовых намерений могут быть пустыми. В теории агентов существует понятие рациональности поведения агента (относительно достижения своих целей). Агент, заданный на языке ПРАЛУ, рационален в пределах заложенных в него планов и намерений.

Методология программирования агентов на ПРАЛУ

Программируя на языке ПРАЛУ, проектировщик агента указывает наборы его убеждений, планов и намерений. Такой стиль описания напоминает логическое программирование, в котором программа представляет собой спецификацию фактов и правил. Можно выделить следующие главные различия между логической программой и программой агента на языке ПРАЛУ.

– В логической программе нет различия между целью в теле правила и локальной переменной, описывающей условие ее применения. В программе агента на ПРАЛУ условия применения плана состоят из намерений, а не целей. Это обеспечивает планам на ПРАЛУ более выразительную форму, допуская управление выполнением планов как данными (используя добавления/удаления убеждений), так и намерениями.

– Правила в логической программе не контекстно-зависимы в отличие от планов на ПРАЛУ.

– Правила в логической программе указывают связывание переменных; в то время как выполнение планов имеет результатом последовательность действий, которые затрагивают окружающую среду.

– Если цель в логической программе становится не актуальной, выполнение доказательства не может быть прервано. Выполнение плана в программе агента на языке ПРАЛУ может быть остановлено операцией гашения этого языка, если при выполнении параллельных планов выяснилась неактуальность цели данного плана.

Процесс проектирования программы агента состоит из следующих этапов.

1. Анализ и разработка ПРАЛУ-описания агента, основываясь на неформальной спецификаци агентов МАС на языке ПРАЛУ.
2. Верификация логической непротиворечивости поведения МАС путем проверки корректности и моделирования ПРАЛУ-описания. Устранение обнаруженных ошибок.
3. Разработка программ, реализующих поведение каждого агента МАС, путем трансляции ПРАЛУ-описания поведения каждого агента в программу на языке программирования.
4. Тестирование сгенерированных программ.

Благодаря использованию языка ПРАЛУ как средства представления спецификаций агентов, а также выполнимости этих спецификаций, процесс проектирования становится структурированным. В нем разрешимы все формальные задачи, и для этих задач имеются программные средства их решения.

Предлагаемая стратегия проектирования естественным образом диктует разбиение программы на два блока: блок *синхронизации* и функциональную часть – блок *сопряжения*. Блок синхронизации координирует совместное выполнение параллельных процессов программы агента. Функциональная часть управляет данными и выполняет требуемые программой вычисления.

Это разбиение выполняется на уровне исходного описания: в спецификации на ПРАЛУ функциональная часть представлена предикатами, описывающими состояния памяти программы и внешней среды или предписывающими выполнение определенных действий. В ПРАЛУ-описании каждому предикату соответствует своя логическая переменная, установка единичного значения которой запускает процесс вычисления предиката. На этапе проверки логической непротиворечивости удобно интерпретировать предикаты как независимые логические переменные.

Такой подход позволяет отделить разработку синхронизирующей части ПРАЛУ-описания от функциональной. Отодвинув разработку функциональной части на более поздние стадии проектирования, можно добиться значительного упрощения проверки логической непротиворечивости поведения агентов.

Для трансляции ПРАЛУ-описания используется транслятор ПРАЛУ на промежуточный язык, применяемый в программе моделирования [12]. Этот транслятор строит символические представления программ вычисления реакций и управления датчиками и исполнительными механизмами агента. Управляющую структуру программы вычисления реакций составляет бесконечный цикл, заключающийся во вводе в оперативную память сигналов датчиков агента, формирующих его убеждения, вычислении по значениям этих сигналов реакции агента и выводе сигналов на его исполнительные механизмы.

Все связи по управлению и по данным между блоком вычисления реакций и блоком управления датчиками и исполнительными механизмами заложены в сгенерированном представлении ПРАЛУ-описания на промежуточном языке.

Чтобы реализовать параллельный алгоритм на одном процессоре [12], нужно упорядочить операции промежуточного языка с помощью планировщика промежуточного языка ПРАЛУ, который имеет свойства и методы. Свойства планировщика – две очереди: ждущих и готовых ветвей. Методы планировщика составляют: запуск, остановка и приостановка подпроцесса. Программа автоматически упорядочивает выполнение операций в процессе выполнения, и в предварительном планировании нет необходимости. Начальное состояние планировщика – очередь готовых содержит первую операцию алгоритма, а очередь ждущих пуста. Если планировщик обнаруживает, что очередь готовых пуста, он переносит содержимое очереди ждущих в очередь готовых, очередь ждущих становится пустой. Информационный обмен с внешней средой выполняется, когда очередь готовых пуста. Это гарантирует, что параллельные операции исходного алгоритма имеют равную продолжительность. Таким образом, программная реализация ПРАЛУ имеет семантику измеряемого времени с выполнением условия рандеву [13].

В этой модели программы предикаты функциональной части рассматриваются как «дополнительное оборудование» агента, формирующее логические сигналы или запускаемое по сигналу. Если при управлении требуется учитывать результаты выполнения расчетных операций, то «дополнительное оборудование» должно специально для этой цели вырабатывать логические сигналы. В программе управления датчиками и исполнительными механизмами это «дополнительное оборудование» описывается в виде выражений обычного языка программирования разработчиком программы агента.

Заключение

Язык ПРАЛУ как язык программирования агентов имеет то преимущество, что он имеет в качестве семантики логический формализм и допускает простую реализацию. Компактность представления программ и простота синтаксиса являются факторами, значительно упрощающими реализацию языка.

Для языка ПРАЛУ имеются программные инструменты, выполняющие имитационное моделирование системы по описанию ее поведения на ПРАЛУ. Имитационное моделирование позволяет объединить высокое быстродействие ЭВМ с гибкостью человеческого мышления. Проводя эксперимент и интерпретируя его результаты, проектировщик системы может контролировать неформальные проектные

операции. Использование ПРАЛУ для моделирования поведения протоколов позволяет получать информацию о логической корректности непосредственно, а не по результатам оценки статистики сеансов имитационного моделирования. При моделировании с целью анализа логической корректности важно рассматривать таймауты, т.е. учитывать параллелизм и асинхронность выполнения операций. Без учета таймаутов опасность ошибочных ситуаций значительно увеличивается.

Поддержка

Работа поддержана РФФИ НАН Беларуси (Проект F07-125).

Библиография

1. Lesser V. Cooperative Multiagent Systems: A Personal View of the State of the Art // IEEE Trans. Knowledge and Data Engineering, 1999. – Vol. 11. –No 1. – P. 133–142.
2. Burmeister B., Sundermeyer K. Cooperative problem-solving guided by intensions and perception, edited by E. Werner and Y. Demazean, Decentralized A.I. 3. – Amsterdam, The Netherlands, North Holland, 1992.
3. Закревский А.Д. Параллельные алгоритмы логического управления. – Мн.: Ин-т техн. кибернетики НАН Беларуси, 1999. – 202 с.
4. Foundation for Intelligent Physical Agents (FIPA). Communicative Act Library Specification, 2002. Available at http://www.fipa.org/specs/fipa00037/.
5. Booch G., Rumbaugh J., Jacobson I. The Unified Modeling Language User Guide – Addison Wesley, 1999.
6 Bauer B., Müller J.P., Odell J. Agent UML: A Formalism for Specifying Multiagent Interaction // Agent-Oriented Software Engineering , edited by P. Ciancarini and M. Wooldridge, Springer-Verlag, Berlin, 2001. – P – 91–103.
7 Quan Bai, Minjie Zhang Khin, Than Win A Colored Petri Net Based Approach for Multi-agent Interactions // 2nd Intern. Conf. on Autonomous Robots and Agents, Palmerston North, New Zealand, Dec. 13–15, 2004. – P. 152–157.
8 Nelson R.A., Haibt L.M., Sheridan P.T. Casting Petri nets into programs // IEEE Trans. Software Eng., 1983. – V. 9. – No 5. – P. 590–602.
9. Paurobally S., Cunningham J. Achieving Common Interaction Protocols in Open Agent Environments // AAMAS '02, Melbourne, Australia, 2002.
10. Hack M. Analysis of production schemata by Petri nets – Project MAC-94, Cambridge, 1972.
11. Cheremisinov D., Cheremisinova L. Developing Agent Interaction Protocols with PRALU // Information Theories & Applications (IJ ITA), 2006. – V. 13. – No 3. – P. 239–246.
12. Черемисинов Д.И. Реализация параллельных алгоритмов управления на одном микропроцессоре // Программирование, 1986. – № 1. – С. 37–45.
13. Cheremisinov D., Cheremisinova L. The specification of agent interaction in multi-agent systems // Proc. of the 5th Intern. Conf. «Information research and application (i.tech)», June 26–30, 2007, Varna, v. 2, Sofia: ITHEA, p. 428–434.

Информация об авторах

Дмитрий Иванович Черемисинов, Людмила Дмитриевна Черемисинова *– Объединенный институт проблем информатики Национальной академии наук Беларуси, ул. Сурганова, 6, Минск, 220012, Беларусь, e-mail: cher@newman.bas-net.by, cld@newman.bas-net.by*