

ФОРМАЛИЗОВАННОЕ ПРЕДСТАВЛЕНИЕ И АНАЛИЗ АГЕНТНО-ОРИЕНТИРОВАННЫХ ЗАДАЧ

ФИЛАТОВ В.А., УЗЛОВ Д.Ю.

Рассматривается класс мультиагентных систем управления информационными ресурсами. Приводится классификация типов задач, характерных для информационных систем. Предлагается спецификация агентно-ориентированного подхода. Обосновывается выбор математического аппарата на основе сетей Петри для моделирования и анализа рассмотренного класса задач. Исследуются варианты сетей Петри, их особенности при моделировании потоков агентно-ориентированных задач. Приводится пример решения агентно-ориентированной задачи.

Ключевые слова: мультиагентная система, база данных, сеть Петри, позиция, переход, маркер, поток задач, агентно-ориентированная задача, информационная система

Keywords: multi-agent system, database, Petri net, position, transition, marker, flow problems, agent-oriented tasks, information system

Введение

Рассматривается класс мультиагентных систем управления информационными ресурсами распределенных систем. Типичным представителем такого класса задач является планирование пути достижения нужной цели из некоторой фиксированной начальной ситуации. Результатом решения задачи должен быть план действий – частично упорядоченная совокупность действий. Таким планом является сценарий, в котором в качестве отношения между вершинами выступают отношения типа: «цель-подцель», «цель-действие», «действие-результат» и т. п. Любой путь в этом сценарии, ведущий от вершины, соответствующей текущей ситуации, в любую из целевых вершин, определяет план действий. Все задачи построения плана действий можно разбить на два типа, которым соответствуют различные модели: планирование в пространстве состояний и планирование в пространстве задач.

В первом случае считается заданным некоторое пространство ситуаций. Описание ситуаций включает состояние внешней среды и состояние информационной системы. Ситуации образуют некоторые обобщенные состояния, а действия или изменения во внешней среде приводят к изменению актуализированных в данный момент состояний. Среди обобщенных состояний выделяются начальные (обычно одно) и конечные (целевые) состояния.

Планирование по состояниям. Представление задач в пространстве состояний предполагает задание ряда описаний: состояний, множества операторов и их воздействий на переходы между состояниями, целевых состояний. Описания состояний могут представ-

лять собой строки символов, векторы, двумерные массивы, деревья, списки и т. п. Операторы переводят одно состояние в другое. Пространство состояний можно представить как граф, вершины которого помечены состояниями, а дуги – операторами.

Планирование по задачам. Метод планирования по состояниям можно рассматривать как частный случай метода планирования с помощью редукций, ибо каждое применение оператора в пространстве состояний означает сведение исходной задачи к двум более простым, из которых одна является элементарной. Поиск решения в пространстве задач заключается в последовательном сведении исходной задачи к все более простым до тех пор, пока не будут получены только элементарные задачи. Частично упорядоченная совокупность таких задач составит решение исходной задачи [1].

Целью проводимых исследований является разработка эффективных методов представления и анализа взаимосвязанных агентно-ориентированных задач на основе модифицированного аппарата сетей Петри.

Спецификация представления агентно-ориентированных задач

Определение агентно-ориентированной задачи: порядок выполнения каждой задачи определяется указанием множества состояний, которые могут быть наблюдаемы извне, и множества переходов между этими состояниями. Кроме того, могут быть определены те характеристики обрабатываемых объектов, которые могут влиять на требования к выполнению задач.

Требования координации агентно-ориентированных задач: такие требования обычно выражаются через межзадачные зависимости выполнения и зависимости потоков данных.

Требования к выполнению или требования корректности: требования к выполнению определяются для ограничения выполнения потока работ таким образом, чтобы удовлетворялись критерии корректности, зависящие от приложения. Они включают требования атомарности при сбоях (failure-atomicity), требования атомарности-выполнения (execution-atomicity) (включая правила видимости, указывающие, когда результаты зафиксированной задачи становятся видимыми для других параллельно выполняющихся потоков работ), а также требования к управлению синхронным выполнением потоков работ и требования восстановления.

Определение агентно-ориентированной задачи в потоке работ

Задача в потоке работ - это единица работы, которая может быть выполнена обрабатываемым объектом, таким как прикладная система или система управления базой данных (СУБД). Задача может быть определена независимо от обрабатываемого объекта, который может ее выполнить, или на основании возмож-

ностей и поведения этого объекта. Для управления потоком работ нет необходимости моделировать все аспекты задач. Рассмотрим пример транзакции, выполняемой СУБД. С точки зрения потока работ все подробности транзакции, описывающие ее последовательную обработку, не являются необходимыми.

Каждая задача выполняет некоторые операции над определенной системой (базой данных). Следовательно, задача является программой (транзакцией) и важно, чтобы она была «корректной». Однако, как и при обеспечении корректности традиционных транзакций, на уровне потока работ мы не моделируем внутренние операции задачи – мы имеем дело только с теми аспектами задачи, которые видимы извне.

Следовательно, структура задачи может быть определена заданием:

- множества состояний выполнения задачи;
- множества переходов между этими состояниями и условий, которые делают допустимыми эти переходы, (условия переходов могут быть использованы для указания межзадачных требований выполнения).

Основные свойства агентно-ориентированных задач

Абстрактной моделью задачи является машина (автомат) состояний, поведение которой может быть определено заданием для ее диаграммы смены состояний. В общем случае каждая задача (и соответствующий ей автомат) может иметь различную внутреннюю структуру для различных диаграмм смены состояний. Это в большой степени зависит от характеристик системы, в которой выполняется эта задача.

Некоторые из свойств обрабатываемых объектов, ответственных за выполнение задачи, таких как наличие или отсутствие интерфейса двухфазной фиксации, непосредственно влияют на структуру задачи и, таким образом, на определение потока работ. Переходы между различными состояниями задачи могут вызываться различными событиями планирования. Некоторые из этих переходов контролируются планировщиком, ответственным за соблюдение межзадачных зависимостей. Например, задача может быть запущена на выполнение, что вызывает переход из состояния «Начало» в состояние «Выполнение».

Другие переходы контролируются локальной системой, ответственной за выполнение задачи. Например, выполняющаяся задача может быть прервана системой (скажем, для того чтобы избежать тупика), что вызывает переход из состояния «Выполнение» в состояние «Прервана».

Одно или более состояний задачи могут быть определены как состояния завершения. Когда задача достигает такого состояния, никакие переходы больше не разрешены. И, наконец, задача может обладать различными свойствами изолированности. Например, результаты еще незавершенной задачи могут стать

видимыми для других параллельно выполняющихся задач, или эти результаты могут быть скрыты до тех пор, пока задача не будет зафиксирована. Эти и другие свойства влияют на механизмы асинхронного выполнения и восстановления, что может быть использовано планировщиком.

Частичный вывод задачи может быть доступным для других параллельно выполняющихся задач, или задача может запросить получение информации из других задач. Предполагается, что задачи потока работ могут взаимодействовать посредством переменных, локальных относительно потока работ. Эти переменные могут содержать параметры программ задачи. Различные начальные параметры могут приводить к различному выполнению задач. Поток данных между подтранзакциями определяется присваиванием значений их входных и выходных переменных. Выполнение подтранзакции влияет на состояние базы данных и выходную переменную подтранзакции.

В любой момент времени состояние потока работ может быть определено как набор состояний входящих в поток задач и значений всех переменных. Выполнение потока работ начинается в начальном состоянии, которое задает параметры инициализации. Различные начальные состояния потока работ могут приводить к различным его выполнениям.

Некоторые характеристики системы, выполняющей задачу, могут влиять на свойства задачи, не затрагивая ее структуру. Например, система, выполняющая задачу, может гарантировать аналогичный порядок выполнения и сериализации, что позволит планировщику потока работ влиять на локальный порядок сериализации задач путем управления порядком их фиксации (старт, окончание). Подобным образом система может гарантировать идемпотентность, обеспечивая безопасное повторение задачи, если не получено положительное подтверждение или истекло время.

Задачи моделируются их состояниями вместе со значимыми событиями, соответствующими переходам между состояниями – старт, фиксация, откат, которые могут быть форсированы, отвергнуты или задержаны. Межзадачные зависимости, такие как зависимости порядка $E_1 < E_2$ и зависимости существования $E_1 \rightarrow E_2$ между значимыми событиями задач, формально определяются с использованием логики деревьев вычисления (Computation Tree Logic – CTL) и обладают соответствующими автоматами зависимостей, которые могут быть построены автоматически. Эти зависимости могут поддерживаться путем проверки нужных автоматов зависимостей [2].

Требования атомарности по сбоям в потоке работ

Используя понимание семантики потока работ и мультисистемных ограничений согласованности, проектировщик потока работ определяет требования *атомар-*

ности по сбоям для потока. Традиционное понятие атомарности отказов состоит в том, что неудача при выполнении любой задачи приводит к неудачному выполнению всего потока. Тем не менее, во многих случаях поток работ может выдержать неудачное завершение одной из задач, выполнив, например, функционально эквивалентную задачу на другой установке. Система, выполняющая поток работ (планировщик потока работ), должна гарантировать, что любое выполнение будет завершено в состоянии, удовлетворяющем требованиям атомарности по сбоям, сформулированным разработчиком. Мы будем называть такие состояния *приемлемыми состояниями завершения* потока работ. Все остальные состояния выполнения входят во множество *неприемлемых состояний завершения*, в которых атомарность по сбоям может быть нарушена.

Приемлемое состояние завершения может быть зафиксированным или прерванным. *Зафиксированное приемлемое состояние завершения* – это состояние выполнения, при котором достигнуты цели потока работ. В противоположность этому *прерванное приемлемое состояние завершения* – это правильное состояние завершения, при котором поток работ претерпел неудачу в достижении своих целей. Если достигнуто прерванное приемлемое состояние завершения, все нежелательные эффекты частичного выполнения потока работ должны быть ликвидированы в соответствии с требованиями атомарности по сбоям.

В общем случае задача может быть зафиксирована и должна освободить свои ресурсы, прежде чем поток работ достигнет состояния завершения. Однако если поток работ позднее будет прерван, атомарность по сбоям может потребовать откатки результатов уже зафиксированных задач (т.е. зафиксированных подтранзакций) посредством выполнения *компенсирующих задач*. Понятия приемлемых состояний завершения и зависимостей планирования могут быть использованы для выражения семантики компенсации без применения специальных конструкций, требуемых в других моделях транзакций. Семантика компенсации требует, чтобы компенсирующие транзакции, в конечном счете, завершили свое выполнение успешно, возможно, после нескольких попыток выполнения. В описываемой здесь модели это свойство компенсирующих транзакций может быть определено соответствующим указанием их предусловий планирования.

Аналогично требованиям атомарности по сбоям проектировщик может указать требования *атомарности выполнения* для потока работ. Традиционная модель транзакций требует, чтобы весь поток работ состоял из целиком выполняющихся единиц. Тем самым чередуемое выполнение потоков работ должно было бы иметь тот же результат, что и последовательное выполнение в некотором порядке. Ослабление свойства атомарности выполнения транзакций в централизованных базах данных обсуждается в работах. Тран-

закция разбивается на атомарные шаги и чередование с другими параллельно выполняющимися транзакциями допустимо только между этими шагами. В контексте потока работ задачи обычно являются естественными атомарными элементами, поскольку выполняются различными обрабатываемыми объектами.

Растущий интерес к исследованиям по управлению потоками работ и возрастающая популярность продуктов автоматизации потоков работ привели к созданию ряда языков спецификации потоков работ. В общем случае можно разделить технологии спецификации потоков работ на две большие категории:

- графические средства спецификации потоков работ;
- языки спецификации потоков работ.

Неудивительно, что графические средства реализуются в основном в коммерческих системах, ориентированных на конечного пользователя, в то время как исследовательские прототипы потоков работ используют более формальные и мощные языки спецификаций.

Практический метод спецификации потоков работ (графическое средство или язык) должен поддерживать:

- описание различных типов задач и устройств обработки, которые выполняют эти задачи;
- спецификации потока управления в потоке работ, т.е. зависимости планирования для задач;
- спецификации потока данных между задачами и, возможно, различные преобразования форматов данных.

Проектировщик потока работ может задать дополнительные свойства потока работ, например, критерии корректности выполнения задач.

Важной функцией средства спецификации потока работ является поддержка автоматического (или полуавтоматического) анализа корректности спецификаций. В общем случае автоматическая верификация спецификаций представляет собой такую же трудную проблему, что и верификация традиционных программ. Тем не менее, к некоторым аспектам координации задач можно относиться, как к протоколам обмена сообщениями, и они таким же образом могут быть проанализированы.

Методы анализа взаимосвязанных агентно-ориентированных задач на основе сетей высокого уровня

Проектирование и эксплуатация автоматизированных систем поддержки принятия решений, функционирующих в условиях жесткого ограничения на ресурсы, вызывает трудности, связанные с необходимостью обеспечения заданных функциональных и эксплуатационных свойств системы. Существующие средства диагностики и тестирования, как правило, обеспечи-

вают лишь контроль состояния системы и, в меньшей степени, контроль выполнения заданий конкретными приложениями. Функционирование системы на основе серийных программно-аппаратных средств в значительной мере определяется качеством приложений. Положение усложняется распределенным характером обработки данных и значительным влиянием субъективного фактора на этапе проектирования.

Другим известным формализмом системного, ситуационного анализа, который позволяет генерировать и изучать разнообразные сценарии работы управляющих систем, моделировать различные (в том числе конфликтные и нежелательные) ситуации, изучать и оценивать последствия решений на поведение управляемой системы, являются сети Петри. Они позволяют моделировать параллельную и распределенную работу алгоритма. В частности, многие параллельные алгоритмы управления и процессы обработки информации естественно моделируются граф-схемами. Граф-схема алгоритма представляет собой ориентированный граф: $G = \langle V, E \rangle$, где V – множество вершин, E – множество дуг.

Множество вершин состоит из вершин следующих трех видов:

Операторные вершины соответствуют действиям. Среди них выделяются две обязательные вершины начала и конца алгоритма.

Вершины альтернативного выбора соответствуют решению конфликтов в зависимости от некоторых условий. Такие вершины имеют одну входную и две выходных дуги, помеченные словами «да» и «нет» соответственно. Если условие выполнено, то процесс продолжается по дуге «да», в противном случае – по дуге «нет».

Вспомогательные вершины, которые в свою очередь подразделяются на три группы:

- вершины параллельного ветвления имеют одну входную и несколько выходных дуг и обозначают переход к параллельному выполнению нескольких независимых действий;
- вершины параллельного слияния имеют несколько входных и одну выходную дугу и означают переход к следующему действию при условии, что все предыдущие выполнены;
- вершины условного слияния также имеют несколько входных и одну выходную дугу и означают переход к следующему действию при условии выполнения одного (любого) из предыдущих.

Важным достоинством граф-схем является то, что граф-схема алгоритма может быть формально преобразована в сеть Петри специального вида – сеть Петри со свободным выбором. При этом обоснование корректности граф-схемы алгоритма сводится к проверке корректности и безопасности соответствующей сети Петри. В сетях Петри со свободным выбором

каждая позиция (условие), обуславливающая срабатывание более чем одного перехода, является единственной обуславливающей эти переходы позицией. Структурные особенности сетей Петри со свободным выбором позволили разработать эффективные алгоритмы их семантического анализа непосредственно по структуре сети.

Однако моделирование алгоритмов управления граф-схемами алгоритмов (сетями Петри со свободным выбором) имеет ряд существенных недостатков. В граф-схеме структура алгоритма задана изначально и не допускает, в частности, динамического распараллеливания задачи. Также граф-схема алгоритма не обладает модульной структурой, что делает моделирование и анализ больших задач весьма затруднительными. Для устранения этого ограничения можно использовать расширение стандартного формализма сетей Петри – вложенные сети Петри.

Методы анализа сложных взаимосвязанных процессов на основе вложенных сетей Петри

Во вложенных сетях Петри фишки, помечающие позиции, рассматриваются как объекты, имеющие самостоятельное поведение, которое в свою очередь описывается также некоторыми сетями Петри. Название «вложенные сети» указывает на то, что элементы сетей в них сами являются сетями, подобно тому, как в системе вложенных множеств элементами некоторого множества могут быть множества. Вложенные сети Петри являются удобным и мощным средством для моделирования и анализа иерархических мульти-агентных распределенных систем. Они обладают естественным механизмом модульности. Модульные сети Петри можно рассматривать как специальный случай вложенных сетей [3].

По сравнению с другими расширениями формализма сетей Петри за счет понятия и конструкции объекта вложенные сети Петри сохраняют такие важные свойства стандартных сетей Петри, как простота и выразительность модели и разрешимость некоторых важных для верификации свойств.

Вложенная сеть Петри состоит из системной (корневой) сети и множества элементных сетей, представляющих фишки системной сети. В простейшем случае двухуровневой вложенной сети элементные сети являются обыкновенными сетями Петри, в которых фишки, как обычно, изображаются черными точками, не имеют собственной структуры и не различимы между собой.

Поведение вложенной сети Петри включает четыре типа шагов.

Шаг переноса – это срабатывание перехода системной сети в соответствии с обычными правилами для сетей Петри высокого уровня, при этом элементные сети рассматриваются как фишки, не имеющие собственной структуры. Шаг переноса может перемес-

титель, породить или убрать объекты, но не может изменить их внутреннее состояние.

Элементно-автономный шаг меняет только внутреннее состояние (маркировку) элементной сети, не меняя ее местонахождения в системной сети. Этот шаг выполняется также в соответствии с обычными правилами срабатывания перехода для сети Петри.

Шаг горизонтальной синхронизации есть одновременное срабатывание двух переходов в двух элементных сетях, находящихся в одной позиции системной сети. При этом переходы, которые должны срабатывать синхронно, помечаются взаимно дополнительными метками из некоторого специального множества меток для горизонтальной синхронизации.

Шаг вертикальной синхронизации используется для синхронизации перехода в системной сети с некоторыми переходами элементных сетей. Переходы, которые должны срабатывать синхронно, помечаются метками из некоторого специального множества меток для вертикальной синхронизации. При этом метка перехода в системной сети и метка соответствующего перехода в элементной сети должны быть взаимно дополнительными. Вертикальная синхронизация означает одновременное срабатывание перехода системной сети и переходов (помеченных дополнительной меткой) в задействованных в этом срабатывании элементных сетях. В том случае, когда решаются задачи управления распределенными мультиагентными системами, сложность алгоритмов управления существенно возрастает, что повышает значение моделирования различных сценариев поведения системы при различных стратегиях управления. При этом вложенные сети Петри сохраняют важные для возможности их семантического анализа свойства (являются хорошо структурированными системами переходов). В частности, доказано, что свойство останова (завершаемости алгоритма) и свойство поддержания некоторого заданного состояния управления (control-state maintainability) разрешимы для вложенных сетей Петри [4].

Методы анализа сложных взаимосвязанных процессов на основе предикатных сетей Петри

Пусть задана структура инструментальных средств принятия решений в виде некоторой совокупности последовательно-параллельно взаимодействующих задач $\{z_\gamma\}, \gamma \in H$. В процессе их выполнения некоторое подмножество задач:

$$\{z_{\gamma_1}\} \subset \{z_\alpha\}, \gamma_1 \in H_1, H_1 \subset H \quad (1)$$

может быть выполнено с привлечением дополнительно подчиненных им задач:

$$\{z_{\gamma_1'}\}, \gamma_1 \in H_1, H_1 \subset H. \quad (2)$$

Расширение подмножества задач $\{z_{\gamma_1'}\} \subset \{z_\alpha\}$ на $\{z_{\gamma_1}\}, \gamma_1 \in H_1, H_1 \subset H$ может носить как постоян-

ный, так и временный характер, что приводит к сложной, многоуровневой иерархии подзадач. Это ведет к дополнительным затратам вычислительных ресурсов.

Задача проводимых исследований может быть сформулирована следующим образом: разработать эффективные процедуры динамического анализа и синтеза иерархии взаимосвязанных подзадач (1) и (2). Методы решения должны быть ориентированы на современные информационные технологии.

Перспективным направлением комплексного решения поставленных задач является моделирование взаимодействующих процессов моделями на основе расширений предикатных сетей Петри и оценочных Е-сетей. Достоинства сетей основаны на возможностях учета множества условий, определяемых предметной областью, а также наличием механизмов управления пространством состояний и структурой сети.

Рассмотрим возможности управления пространством состояний и структурой сетей в условиях иерархии взаимодействующих задач. С этой целью введем понятие интерпретированных метапозиций

$\{p_\alpha^{(IM)}\}, \alpha \in A$ и интерпретированных метапереходов $\{t_\beta^{(IM)}\}, \beta \in B$ сети S .

Построение иерархии сетей взаимосвязанных подзадач с переменной структурой

Рассмотрим особенности построения сетей и управление динамикой моделирования процессов обработки данных с использованием модифицированных предикатных сетей и модифицированных Е-сетей.

Модифицированные предикатные сети. Учитывая известные сложности при интерпретации и моделировании процессов обработки данных предикатными сетями, признано целесообразным применение следующей модификации сетей:

$$SPR = \langle P, T, F, A_\tau, C, \{V_S\}, K, M_0 \rangle, \quad (3)$$

где P – множество позиций; T – множество переходов; F – функция инцидентностей позиций и переходов; C – функция цвета маркера; A_τ – параметр времени, отнесенный ко всем компонентам сети; $P, T, F, M_0; V_S$ – условие выполнения переходов $t_k \in T, k \in K$, отнесенных к компонентам сети: $V_{P_k}^I$ – условие выполнения $t_k \in T, k \in K$, отнесенное к его входным позициям; $V_{P_k}^O$ – условие выполнения $t_k \in T, k \in K$, отнесенное к его выходным позициям; $V_{t_k}^{P_k}$ – условие выполнения $t_k \in T, k \in K$, отнесенное непосредственно к этому переходу; $V_{M_{P_l}}$ – условие выполнения $t_k \in T, k \in K$, отнесенное к маркированию ему инцидентных позиций, $l \in L$; V_F – условие выполнения $t_k \in T, k \in K$, отнесенное к его входным и выходным дугам инцидентных позиций; K – ем-

кость маркеров в позициях с учетом C ; M_0 – вектор начальной маркировки.

Условие выполнения перехода V_S включает также такие свойства, как надежность, сложность, стоимость, степень достоверности конкретного события.

Анализ возможных решений показал, что для сети (3), в задачах моделирования процессов подчиненных задач на основе сетей S' удобно применение интерпретированной метапозиции

$$p_\alpha^{(IM)} \in \{p_\alpha^{(IM)}\}, \alpha \in A.$$

Это определяется тем, что собственно позиции в (3) интерпретируют условия выполнения действий моделируемых событий, а также определяют пространство состояний модели.

Модифицированные E-сети. Существующие ограничения E-сетей не позволяют эффективно использовать их в практических приложениях. В связи с этим предлагается следующее развитие E-сетей:

$$E^{(m)} = \langle P, T, I(T), O(T), Q, \Theta, M_0 \rangle, \quad (4)$$

где P – конечное множество позиций; T – конечное множество переходов; $I(T), O(T)$ – соответственно связи входных и выходных позиций переходов T ;

Q – конечное множество вычислительных процедур, которые управляют выполнением переходов и исключают конфликты на переходах сети;

Θ – конечное множество условий выполнения компонент сети, пространства состояний сети и процедур перехода, которые управляют сменой цвета (атрибутов) маркера, определяют приоритеты входных и выходных позиций перехода, функцию времени A_s выполнения переходов; M_0 – вектор начальной маркировки E-сети [5].

Пример

Ресурс типа A_1 обеспечен информационной поддержкой и хранится в файлах баз данных, например $B_{11}, B_{12}, \dots, B_{1n}, n \in N$, где N – количество баз данных. Условие поиска для рассматриваемого фрагмента системы принятия решений формулируется следующим образом: определить заданное количество ресурса типа $A_1 = A_1^*$ на множестве $\{B_{11}, B_{12}, \dots, B_{1n}\}, n \in N$. Поиск решения осуществляется двумя типами действий – на первом шаге ПОИСК по базе данных B_1 , если условие не выполняется, осуществляется ОБЪЕДИНЕНИЕ баз данных $B_1 + B_2$ и поиск повторяется. Процедура поиска и объединения баз данных может осуществляться в терминах структурированного языка запросов SQL с использованием опции SELECT и INSERT:

```
Select sum([A1]), from b1.dbf;
```

```
Insert into b1...Select b2...From b2.
```

Представим фрагмент рассмотренной выше задачи в виде *модифицированной предикатной сети Петри S* (рис. 1). Множество позиций $\{p_0, p_1\}$ интерпретируют соответственно входное и выходное условие выполнения действия, что представлено на сети переходом t_0 , множество позиций $\{p_1, p_2\}$ интерпретируют соответственно входное и выходное условие выполнения действия t_1 . Вектор начальной маркировки $M_0 = (1, 0, 0)$ определяет начальное пространство состояний сети.

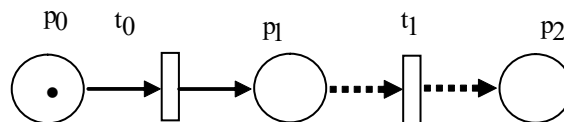


Рис. 1. Сеть Петри S

Описание основных ограничений, представленных на рис. 1: $V_{p_k}^i$ – входное условие выполнения перехода

$t_1 \in T, k \in K$; $V_{p_k}^o$ – выходное условие выполнения

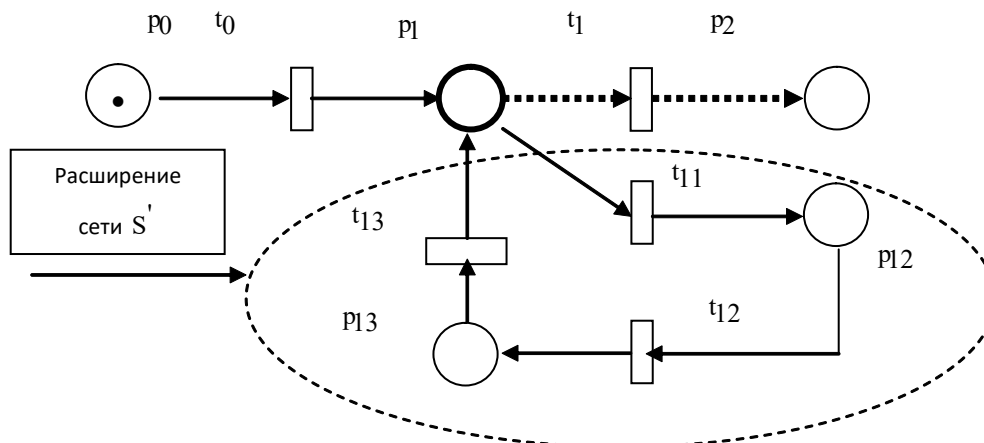


Рис. 2. Сеть Петри S с расширением S'

перехода t_1 . Если входное условие V_{p_k} выполнения перехода t_1 не удовлетворено: $V_{p_k} \neq V_{t_k}$, то для данного перехода $t_1 \in T, k \in K$ генерируется подчиненная сеть S' (рис.2), где позиция p_1 является метапозицией. Вектор начальной маркировки $M_0 = (1, 0, 0, 0, 0)$ определяет начальное пространство состояний сети.

Маркирование позиции p_{13} и выполнение перехода t_{13} маркирует метапозицию p_1 , что запускает выполнение сети S . Если искомое решение невыполнимо средствами $S \cup S'$, то осуществляем иерархию задач на основе поиска решений из множества $\{B_{11}, B_{12}, \dots, B_{1n}\}, n \in N$ путем объединения баз данных и расширения сети S' .

Выводы

В результате анализа методов управления информационными ресурсами в корпоративных системах рассмотрен подход к представлению агентно-ориентированных задач на основе анализа состояния внешней среды и состояния информационной системы. Исследованы два основных метода представления агентно-ориентированных задач: планирование по состояниям и планирование по задачам. Рассмотрены особенности управления пространством состояний и структурой модифицированных предикатных сетей в условиях иерархии взаимодействующих задач. Полученные результаты могут быть распространены на некоторую иерархию подчиненных сетей с учетом их особенностей и необходимости генерации метапозиций и метапереходов. Предложенный подход позволяет снизить временные и вычислительные ресурсы за счет подключения подчиненных задач, определения устойчивых связей и исключения из рассмотрения нерациональных и неудовлетворительных решений.

Литература: 1. Пономаренко Л.А., Цыбульник Е.Е., Филатов В.А. Агентные технологии в задачах поиска информации и принятия решений // УСИМ: Управляющие системы и машины. 2003. № 1. С. 36-41. 2. Кучеренко Е.И., Филатов В.А. Модели принятия решений в задачах анализа сложных систем на основе сетей высокого уровня // Системы обработки информации. Харьков: ХВУ. 2004. Вып. 5. С. 139-148. 3. Мурата Т. Сети Петри: Свойства, анализ, приложения // ТИИЭР. 1989. Т. 77. №4. С. 41-85. 4. Girault C., Volk R. Petri nets for systems engineering – A guide to modelling, verification and applications, Springer-Verlag,

2003. 5. Филатов В.А., Кучеренко В.Е. О моделях обработки данных на основе иерархии сетей высокого уровня // Радиоэлектроника, информатика, управление. Научный журнал, Запорожье: ЗНТУ. 2004. №1(11). С. 95-100.

Транслитерированный список литературы: 1. Ponomarenko L.A., Cybul'nik E.E., Filatov V.A. Agentnye tehnologii v zadachah poiska informacii i prinjatija reshenij // USiM: Upravljajushhie sistemy i mashiny – № 1, 2003. – S. 36-41. 2. Kucherenko E.I., Filatov V.A. Modeli prinjatija reshenij v zadachah analiza slozhnyh sistem na osnove setej vysokogo urovnja. // Sistemi obrobki informacii. – Har'kov: HVU. 2004. Vip.5. S.139-148. 3. Murata T. Seti Petri: Svoystva, analiz, prilozhenija // ТИИЭР. 1989. Т. 77. №4. Р. 41-85. 4. Girault C., Volk R. Petri nets for systems engineering – A guide to modelling, verification and applications, Springer-Verlag, 2003. 5. Filatov V.A., Kucherenko V.E. O modeljah obrabotki dannyh na osnove ierarhii setej vysokogo urovnja. // Radiojelektronika, informatika, upravlenie. Nauchnyj zhurnal, Zaporozh'e: ZNTU. 2004. №1(11). S. 95-100.

Поступила в редколлегию 14.12.2015

Рецензент: д-р техн. наук, проф. Рубан И.В.

Филатов Валентин Александрович, д-р техн. наук, профессор, заведующий кафедрой искусственного интеллекта ХНУРЭ. Научные интересы: базы данных и знаний, агентные технологии, мультиагентные системы, извлечение знаний из данных. Увлечения и хобби: велосипед, авто, мото. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел: +3 8(057) 7021337, e-mail: valentin.filatov@nure.ua

Узлов Дмитрий Юрьевич, канд.техн. наук, полковник полиции, начальник Управления информационного обеспечения ГУНП в Харьковской области. Научные интересы: интеллектуальные информационные системы, извлечение данных из знаний, интеллектуальный анализ данных, системы поддержки принятия решений. Увлечения и хобби: путешествия, фантастика. Адрес: Украина, 61000, Харьков, ул. Совнаркомовская, 5, тел: +3 8(057)7059775.

Valentin Filatov, Doctor of Sciences, Professor, Head of the Department of Artificial Intelligence in the Harkiv National University of Radioelectronics. Research interests: databases and knowledge, agent technology multi-agent systems, data mining. Interests and hobbies: bike, car, moto. Address: Harkiv, 61166, Lenin Ave, 14, Phone: +3 8 (057) 7021337, e-mail: valentin.filatov@nure.ua

Uzlov Dmitry, PhD, the colonel of police, chief the Information Department of National Policy in Harkiv region. Research interests: intelligent information systems, data mining, and decisions support systems. Interests and hobbies: traveling, fantasy. Address: Kharkiv, str. Sovnarkomovskaya, 5. Phone: +3 8(057)7059775.