ПРОГРАММИРОВАНИЕ, 2010, № 4, с. 3–15

ТЕОРЕТИЧЕСКИЕ ВОПРОСЫ ПРОГРАММИРОВАНИЯ

УДК 681.3.06

ФОРМАЛИЗАЦИЯ СЕМАНТИКИ СИСТЕМ С НЕНАДЕЖНЫМИ АГЕНТАМИ ПРИ ПОМОЩИ СЕТЕЙ АКТИВНЫХ РЕСУРСОВ*

© 2010 г. В. А. Башкин

Ярославский государственный университет им. П.Г. Демидова
150000 Ярославль, ул. Советская, 14
E-mail: bas@uniyar.ac.ru

Поступила в редакцию 21.04.2008 г.

Представлен новый способ моделирования распределенных систем – сети активных ресурсов. Данный формализм обладает той же выразительной мощностью, что и обыкновенные сети Петри, однако в его синтаксисе заложен другой принцип моделирования: вместо разделения компонентов модели (вершин графа) на агенты и ресурсы (переходы и позиции) вводится разделение способов взаимодействия (дуг графа) на производство и потребление. Направление дуги определяет активную и пассивную стороны взаимодействия; один и тот же компонент при различных срабатываниях может выступать как в роли агента, так и в роли ресурса. В сетях активных ресурсов и в их синтаксических расширениях появляется возможность для удобной формализации таких семантических свойств, как одновременная работа агентов, блокировка агента, избыточность числа агентов, возможность замены надежного узла ненадежным и др.

Рассматривается метод построения распределенных приложений на основе динамически конфигурируемых наборов исполняющих модулей. Показано, что использование сетей активных ресурсов позволяет достаточно естественным образом задавать структуру, а также формулировать свойства подобных систем.

1. ВВЕДЕНИЕ

Описание поведения отдельного агента – одна из ключевых задач при моделировании сложной мультиагентной системы. Зачастую действие, выполняемое агентом, настолько неэлементарно, что для его описания требуется отдельная модель. Однако есть ряд достаточно распространенных типов агентов, которые удобно моделировать простыми математическими структурами.

Одним из таких типов агентов является агент – преобразователь ресурсов. Подобный агент получает на вход один фиксированный набор ресурсов и производит на выходе другой фиксированный набор ресурсов.

Системы с ресурсами, как правило, моделируются при помощи сетей Петри. Сеть Петри представляет собой ориентированный граф с двумя типами компонентов – позициями и переходами. Вершины-позиции моделируют состояние системы. Каждой позиции приписано целое неотрицательное число, показывающее количество доступных ресурсов данного типа (локальное состояние). Вершины-переходы моделируют действия, которые может выполнять система (срабатывания переходов). Каждый переход при срабатывании потребляет один набор ресурсов (предусловие) и производит другой (постусловие). Таким образом, понятию простого агента-преобразователя ресурсов в сетях Петри соответствует переход или набор переходов.

В данной работе представлен новый способ моделирования систем с более явным выделением понятия агента. В моделях, названных сетя-

*Работа выполнена при финансовой поддержке РФФИ (проект 09-01-00277) и ФЦП “Кадры” (проект 02.740.11.0207).

ми активных ресурсов (АР-сетями), используется тот же способ сохранения и преобразования информации, что и в обыкновенных сетях Петри – конечное мультимножество как хранилище данных и проверка наличия в нем данного конечного подмножества как условие запуска действия. Однако здесь убрано разбиение множества вершин графа сети на позиции и переходы – все они являются равноправными узлами. Объекты, хранящиеся в вершине A, могут быть использованы при срабатываниях объектов из других вершин в качестве потребляемых или производимых ресурсов. Точно так же объекты вершины A могут сработать, производя или потребляя какие-то другие (или даже такие же) объекты. Срабатывающий объект выступает в роли агента, потребляемый или производимый – в роли ресурса. Для определения вида связи между вершинами (производство или потребление) вводится тип дуги – производящая или потребляющая. Таким образом, на два класса разбивается не множество вершин, как в обыкновенных или супер-двойственных сетях Петри, а множество дуг графа. Доказывается, что класс сетей активных ресурсов равномощен классу обыкновенных сетей Петри.

Предлагаются способы расширения синтаксиса АР-сетей, позволяющие удобно формализовать семантику систем с более сложным поведением агента.

Вводится понятие работающего агента. Работающий агент уже потребил свои входные ресурсы, но еще не произвел выходные. Работающий агент не могут потребить другие агенты в качестве ресурса. В АР-сетях с работающими агентами формализуются такие семантические свойства, как одновременная работа нескольких агентов, блокировка ресурса (агента), постоянное наличие неактивных агентов данного типа (избыточность), возможность появления неограниченного количества работающих агентов и др. Важно отметить, что при введении понятия работающего агента структура графа сети (модель системы в целом) не меняется, усложняется только семантика фишки (модель агента/ресурса).

Вводится понятие ненадежного агента. Ненадежный агент может произвести не весь свой выходной ресурс. Ненадежность рассматривается не как свойство отдельного агента (фишки), а как свойство взаимодействия агента с конкретными видами ресурсов. В АР-сетях с ненадежными агентами формализуются такие семантические свойства, как возможность замены надежного узла ненадежным без ущерба для каких-то качеств системы, наилучшее и наихудшее поведение системы в целом (с точки зрения производства ресурсов) и др.

Доказывается, что представленные формализмы эквивалентны по выразительной мощности обыкновенным сетям Петри (предлагаются алгоритмы преобразования), то есть для них сохраняется разрешимость таких фундаментальных алгоритмических свойств, как завершаемость, достижимость разметки, ограниченность, безопасность и живость узла сети.

Рассматривается схема построения распределенного приложения на основе сети активных ресурсов. Приложение представляет собой набор веб-сервисов, контролируемых единой управляющей системой. Управляющая система в качестве схемы процесса использует сеть активных ресурсов, при этом узел АР-сети выступает в качестве шаблона (спецификации) сервиса, а фишки – в качестве конкретных экземпляров соответствующих данному шаблону сервисов. Вебсервис является агентом – он может совершать действия над ресурсами (продолжительные по времени и не всегда успешные). Но при этом он же выступает и в качестве ресурса – производится и потребляется (точнее, система сама способна искать в интернете и регистрировать подходящие под шаблон сервисы, а также удалять из реестра сервисы, ставшие недоступными). Использование формализма сетей активных ресурсов позволяет достаточно естественным образом описывать структуру и формулировать свойства подобных систем.

2. ОСНОВНЫЕ ОПРЕДЕЛЕНИЯ

Через Nat обозначим множество неотрицательных целых чисел.

Пусть X – непустое множество.

Мультимножеством M над множеством X называется функция $M : X \to Nat$. *Мощность* мультимножества $|M| = \sum\limits_{x \in X} M(x)$. Числа $\{M(x) \mid x \in X\}$ называются *коэффициентами*

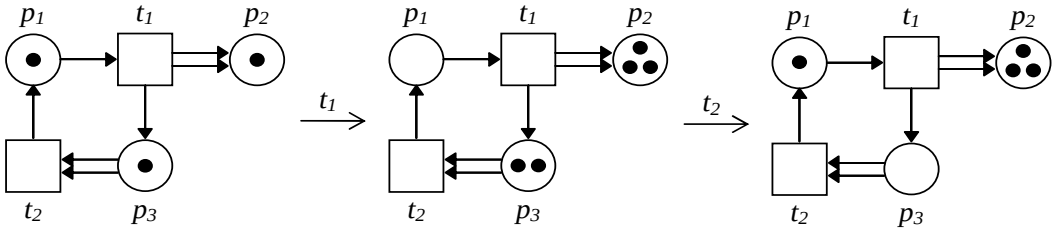


Рис. 1. Сеть Петри.

мультимножества, коэффициент $M(x)$ определяет число экземпляров элемента x в M. Мультимножество M *конечно*, если конечно множество $\{x \in X \mid M(x) > 0\}$. Множество всех конечных мультимножеств над данным множеством X обозначается как $\mathcal{M}(X)$.

Операции и отношения теории множеств естественно расширяются на конечные мультимножества.

Пусть $M_1, M_2, M_3 \in \mathcal{M}(X)$. Полагаем:

- $M_1 = M_2 + M_3 \Leftrightarrow \forall x \in X \; M_1(x) = M_2(x) + M_3(x)$ – операция сложения двух мультимножеств;
- $M_1 = M_2 - M_3 \Leftrightarrow \forall x \in X \; M_1(x) = M_2(x) \ominus M_3(x)$ – разность мультимножеств (где $\ominus$ – вычитание до нуля).

Сетью Петри называется набор $N = (P, T, F)$, где

- P – конечное множество позиций;
- T – конечное множество переходов, $P \cap T = \varnothing$;
- $F : (P \times T) \cup (T \times P) \to Nat$ – функция инцидентности (мультимножество дуг).

Разметкой (состоянием) сети N называется функция вида $M : P \to Nat$, сопоставляющая каждой позиции сети некоторое натуральное число (или ноль). Разметка может рассматриваться как мультимножество над множеством позиций сети, то есть элемент множества $\mathcal{M}(P)$.

Размеченной сетью Петри называется пара (N, M_0), где $N = (P, T, F)$ – сеть Петри, $M_0 \in \mathcal{M}(P)$ – начальная разметка (количество ресурса в наличии при запуске сети).

Графически сеть Петри изображается как двудольный ориентированный граф. Вершины-позиции изображаются кружками и характеризуют локальные состояния сети, вершины-переходы изображаются прямоугольниками и соответствуют действиям. Дуги соответствуют элементам F. Позиции могут содержать маркеры (фишки), изображаемые черными точками. При разметке M в каждую позицию p помещается $M(p)$ фишек.

Переход $t \in T$ *активен* при разметке M, если $\forall p \in P \;\; M(p) \geq F(p,t)$ (все входные позиции содержат достаточное количество фишек).

Активный при разметке M переход t может *сработать*, порождая новую разметку M', где $\forall p \in P \;\; M'(p) = M(p) - F(p,t) + F(t,p)$.

Множество всех разметок, достижимых от начальной разметки M за одно или несколько срабатываний переходов, обозначается как $\mathcal{R}(N, M)$.

Пример срабатывания переходов в обыкновенной сети Петри приведен на рис. 1. Последовательно срабатывают переходы t_1 и t_2.

В сети Петри модель системы явно разделена на пассивную и активную составляющие. Пассивная часть – это состояние системы, то есть мультимножество имеющихся в наличии ресурсов. Для хранения этих ресурсов (фишек) предназначен конечный набор вершин-позиций. Активная часть – конечный набор возможных преобразований состояния (множество вершин-переходов). Позиции могут быть связаны только с переходами, и наоборот. Таким образом, любая модель, основанная на сетях Петри, состоит из двух частей. Одна из них (множество позиций) меняет свое “содержимое” в ходе функционирования сети, другая (множество переходов) – жестко фиксирована и выступает только в качестве системы связей между позициями.

Такая “несмешиваемость” – одно из удобств использования сетей Петри, позволяющее детально структурировать модель системы с точки зрения разбиения ее на отдельные элементарные действия. Однако она же приводит к тому,

что модель получается весьма низкоуровневая, неудобная для дальнейшей композиции или декомпозиции. Кроме того, из-за простоты функционирования отдельных переходов для моделирования сколько-нибудь сложных действий приходится вводить в модель большое количество дополнительных (служебных) компонентов.

В качестве возможных путей решения этой проблемы предлагаются различные расширения синтаксиса формализма за счет введения высокоуровневых [1] и/или иерархических конструкций [2].

Отдельным и гораздо менее развитым направлением исследований являются модификации базового синтаксиса сетей Петри с целью изменения самого способа взаимодействия между активной и пассивной составляющими модели. При этом предлагается, в частности, более сбалансированное использование двудольности графа (в обзоре К. Петри [3] исследование двойственности прямо указано в качестве возможного дополнительного источника полезной информации о сетях).

В работе К. Лаутенбаха [4] вводится формализм двойственных P/T-сетей. В таких сетях переходы могут содержать специальные маркеры. При этом их влияние на срабатывание тоже в некотором смысле двойственное по отношению к обычным фишкам: если в переходе есть хотя бы один маркер, то этот переход не может сработать. Маркеры перемещаются по переходам во время срабатывания позиций, при этом для их перемещения используются те же дуги, что и при срабатывании переходов, но развернутые в обратную сторону. Перемещение маркеров в переходах на практике используется для моделирования взаимосвязанных отказов компонентов системы (cascading fails). Ошибка, возникая в каком-то одном переходе, может в дальнейшем распространиться по всей системе, используя схему связей переходов и позиций.

В работе М. Колера и Х. Рольке [5] предлагается формализм супер-двойственных сетей Петри. Здесь переходы также содержат специальные маркеры. Переход может сработать только тогда, когда в нем есть маркер. Маркеры перемещаются по переходам при помощи срабатывания позиций, причем для их перемещения используется отдельный набор дуг (G-дуги). Таким образом, сеть представляет собой две "склееные" в вершинах сети Петри: F-сеть с активными переходами и пассивными позициями и G-сеть с активными позициями и пассивными переходами. Супер-двойственные сети предполагается использовать для моделирования систем со сложным поведением компонентов (в частности, систем с иерархической структурой переходов).

В супер-двойственных сетях можно изменять наборы активных переходов, добавляя и удаляя маркеры, но при этом двудольность графа сохранена – переход может быть связан дугой только с позицией, позиция – только с переходом. Однако здесь это разделение выглядит несколько искусственным – ведь в силу симметричности определений позиции и переходы в таких сетях обладают одинаковыми свойствами. Возникает вопрос – что произойдет с формализмом сетей Петри при полном "слиянии" понятий позиции и перехода, то есть при отказе от требования двудольности графа? Какие дополнительные семантические свойства можно будет формализовать?

3. СЕТИ АКТИВНЫХ РЕСУРСОВ

Рассмотрим модель, в которой любая фишка может быть и агентом, и ресурсом. Для того, чтобы описать весь возможный набор взаимосвязей таких обобщенных "агентов/ресурсов", разделим дуги сети на два непересекающихся класса: по первым одна фишка будет уничтожаться другой (потребляться при срабатывании), по вторым одна фишка будет создавать другую (производить при срабатывании).

Сетью активных ресурсов (*AP-сетью*) назовем набор $AR = (V, I, O)$, где

- V – конечное множество вершин;
- $I \in \mathcal{M}(V \times V)$ – мультимножество потребляющих дуг;
- $O \in \mathcal{M}(V \times V)$ – мультимножество производящих дуг.

Графически вершины сети изображаются кружками, потребляющие дуги – пунктирными стрелками, производящие дуги – непрерывными стрелками.

Естественным образом вводятся следующие термины:

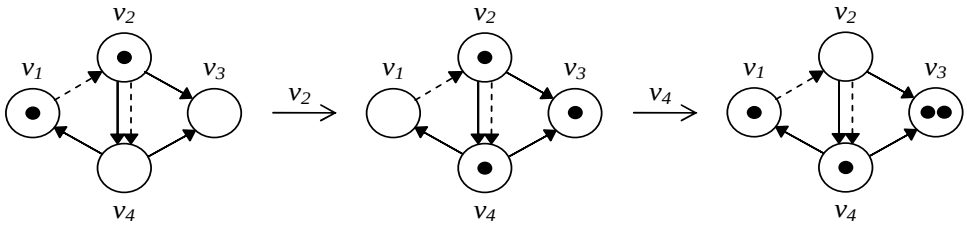


Рис. 2. Сеть активных ресурсов.

Пусть $i = (v_1, v_2)$ – потребляющая дуга. Тогда дуга i называется *входной* для вершины v_2 и *уничтожающей* для вершины v_1. Ресурс в вершине v_1 – *потребляемый* по дуге i, агент в вершине v_2 – *потребляющий* по дуге i.

Пусть $o = (v_1, v_2)$ – производящая дуга. Тогда дуга o называется *выходной* для вершины v_1 и *создающей* для вершины v_2. Агент в вершине v_1 – *производящий* по дуге o, ресурс в вершине v_2 – *производимый* по дуге o.

По определению не бывает выходных уничтожающих и входных создающих дуг. Один и тот же ресурс/агент может быть одновременно производящим, потребляющим, производимым и потребляемым (по разным инцидентным дугам).

Размеченной сетью активных ресурсов назовем пару (AR, M), где $AR = (V, I, O)$ – сеть активных ресурсов, $M \in \mathcal{M}(V)$ – начальная разметка.

На рисунках разметка изображается при помощи соответствующего количества фишек в вершинах.

Вершина $v \in V$ *активна* при разметке M, если

- $M(v) \geq 1$ (в сети есть агент v);
- $\forall w \in V \;\; M(w) \geq I(w, v)$ (потребляемых ресурсов достаточно).

Активная при разметке M вершина v может *сработать*, порождая новую разметку M', где $\forall w \in V \;\; M'(w) = M(w) - I(w, v) + O(v, w)$. Такое срабатывание обозначается как $M \xrightarrow{v} M'$.

Таким образом, в срабатывании вершины (в качестве агента) участвуют ее входные и выходные дуги, в изменении разметки вершины (трансформации соответствующего ресурса) – ее потребляющие и производящие дуги.

Множество всех разметок, достижимых от начальной разметки M за одно или несколько срабатываний вершин, обозначается как $\mathcal{R}(AR, M)$.

Пример срабатывания вершин в AP-сети приведен на рис. 2.

Структура сетей активных ресурсов существенно отличается от структуры сетей Петри. Сеть активных ресурсов – это два ориентированных псевдографа на общем множестве вершин, тогда как сеть Петри – это двудольный ориентированный мультиграф. При этом они определяют один и тот же класс систем:

Теорема 1. Класс сетей активных ресурсов эквивалентен классу обыкновенных сетей Петри[1].

Доказательство см. в приложении.

Итак, мы построили всего лишь альтернативный способ представления сетей Петри. Его основное отличие состоит в появлении более четкого понятия агента. Кроме того, мы получаем достаточно широкие возможности модификации структуры системы (рассматривая агент в том числе и в качестве ресурса).

Изменение языка представления позволяет достаточно компактно и наглядно описывать ситуации, не всегда удобно моделируемые при использовании классических сетей Петри. В первую очередь, это поведение различных систем с нефиксированным набором агентов (сети сервисов, системы с динамическим распараллеливанием, адаптивные бизнес-процессы и т.п.).

В AP-сетях появляется возможность моделирования динамического изменения набора агентов, вплоть до самоуничтожения и самовоспроизводства агента (пример приведен на рис. 3).

Формализация ряда структурных семантических свойств системы (то есть свойств, не зависящих от ее начального состояния) приведена в таблице 1.

[1]Здесь и далее рассматривается эквивалентность с точки зрения равенства множеств достижимости.

Рис. 3. Самоуничтожающее и самовоспроизводящее срабатывания.

Таблица 1. Формализация некоторых структурных свойств вершины v

Определение	**Интерпретация**
$\forall w \in V\ I(w,v) \le O(v,w)$	Агент v самодостаточен (сам производит необходимые для него ресурсы).
$\forall w \in V\ I(v,w) \le O(w,v)$	Ресурс v не может быть израсходован системой ни при каком ее начальном состоянии.
$I(v,v) - O(v,v) = 1$	Агент v самоуничтожается.
$O(v,v) - I(v,v) = 1$	Агент v самовоспроизводится.

Таблица 2. Формализация некоторых свойств множества достижимости

Определение $(\forall M \in \mathcal{R}(AR, M_0))$	**Интерпретация**
$M(v) > 0$	В системе постоянно присутствует ресурс/агент v.
$M(v) > \max_{w \in V}(M(w) \div I(w,v))$	Система постоянно испытывает недостаток ресурсов, необходимых для агентов вида v.
$M(v) < \min_{w \in V}(M(w) \div I(w,v))$	В системе постоянно присутствует излишек ресурсов, необходимых для агентов вида v.
1) $M(v) < \min_{w \in V}(M(w) \div I(w,v))$ 2) $\forall w \in V\ I(v,w) < M(v)$	В системе постоянно присутствует избыточный ресурс/агент v (то есть компонент, не способный сработать сам и не нужный для срабатывания других компонентов).

Структурные свойства не учитывают конкретное начальное состояние системы, поэтому верны всегда. Однако в большинстве случаев анализируется система с заданным входом (или набором входов).

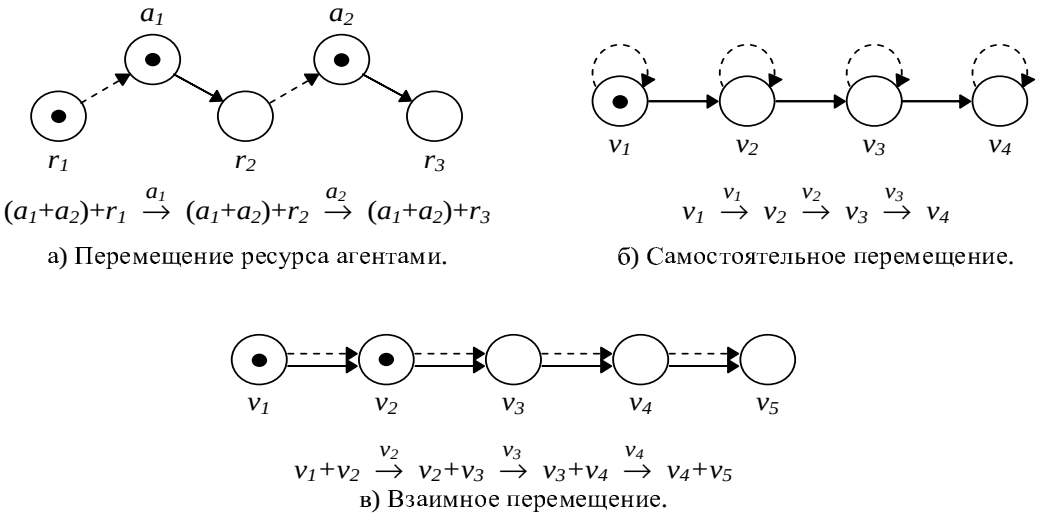
Формализация ряда семантических свойств множества возможных поведений системы в случае фиксированного начального состояния (то есть свойств множества достижимости) приведена в таблице 2.

На рис. 4 приводятся примеры моделирования АР-сетями различных способов перемещения ресурсов (что также можно рассматривать как смену состояний каких-то неэлементарных компонентов системы, представленных в виде наборов вершин).

Первый способ соответствует тому, как ресурсы (фишки) переносятся в обыкновенных сетях Петри. В модель вводятся специальные фиксированные агенты ("переходы"), которые занимаются исключительно переносом фишек и не могут сами ни возникать, ни пропадать.

Следующие два примера используют специфические возможности синтаксиса АР-сетей. При самостоятельном перемещении одна и та же фишка ("объект") выступает одновременно и в роли агента, и в роли ресурса, перенося "сама себя" из одной вершины в другую. При взаимном перемещении две фишки попеременно выступают в роли агентов, перенося друг друга. Таким образом, в частности, можно моделировать всевозможные коммуникации: самостоятельное перемещение объектов, эстафетную передачу данных, широковещательное распространение сообщений, сбоев, вирусов и т.п.

Важно заметить, что в силу эквивалентности формализмов подобные ситуации можно моделировать и при помощи сетей Петри. Однако



а) Перемещение ресурса агентами.

б) Самостоятельное перемещение.

в) Взаимное перемещение.

Рис. 4. Моделирование перемещения (смены состояния) объекта.

там для этого придется вводить дополнительные позиции и переходы, серьезно усложняющие модель.

Из теоремы 1 следует, что для АР-сетей разрешимы все те же свойства, которые разрешимы для обыкновенных сетей Петри: достижимость, ограниченность, безопасность, живость и др. Очевидно, что АР-сети также обладают свойством монотонности [6], поэтому для их анализа мы можем использовать метод построения полного покрывающего дерева сети.

Итак, мы рассмотрели некоторые особенности представления мультиагентных систем при помощи сетей активных ресурсов. Кроме того, модель АР-сетей можно расширять, вводя новые синтаксические конструкции, отражающие те или иные особенности работы более сложных агентов и ресурсов.

Однако существует опасность при расширении синтаксиса получить расширение класса моделируемых систем и в результате прийти к неразрешимости тех или иных алгоритмических свойств. В частности, если добавить в модель специальный вид вершины, которая может срабатывать не тогда, когда в ней есть фишка (агент), а, наоборот, только тогда, когда в ней нет ни одной фишки, то получится формализм, эквивалентный по мощности машинам Тьюринга.

4. СЕТИ С РАСШИРЕННЫМИ СРАБАТЫВАНИЯМИ

Рассмотрим модель системы, в которой некоторые агенты срабатывают не моментально, а в течение какого-то интервала времени. При этом мы считаем, что работающий агент на время своего срабатывания блокируется, то есть другие агенты не могут его потребить в качестве входного ресурса. Сам же этот агент потребляет свои входные ресурсы в момент запуска, а выходные ресурсы производит в момент остановки.

Сетью активных ресурсов с расширенными срабатываниями (РС-сетью) назовем набор $E = (V, V_e, I, O)$, где

- V – конечное множество обычных вершин;
- V_e – конечное множество расширенных вершин, где $V \cap V_e = \varnothing$;
- $I \in \mathcal{M}((V \cup V_e) \times (V \cup V_e))$ – мультимножество потребляющих дуг;
- $O \in \mathcal{M}((V \cup V_e) \times (V \cup V_e))$ – мультимножество производящих дуг.

На рисунках расширенные вершины изображаются овалами.

Размеченной сетью активных ресурсов с расширенными срабатываниями назовем тройку (E, M, M_e), где $E = (V, V_e, I, O)$ – РС-сеть, $M \in \in \mathcal{M}(V \cup V_e)$ – обычная разметка (ожидающие ресурсы), $M_e \in \mathcal{M}(V_e)$ – расширенная разметка (работающие расширенные ресурсы).

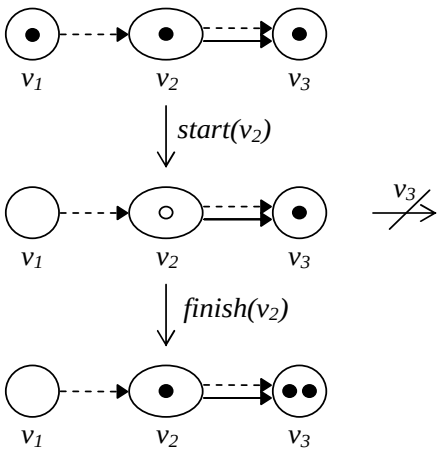


Рис. 5. Работающий агент.

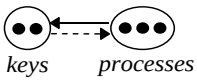


Рис. 6. Модель разделяемого доступа (mutex) с тремя процессами и двумя ключами.

На рисунках обычная разметка изображается при помощи обычных черных (закрашенных) фишек, расширенная – при помощи белых (незакрашенных) фишек. Черные фишки могут появляться в любых вершинах, белые – только в расширенных.

Правила срабатывания определяются по отдельности для обычных и для расширенных вершин. Для обычных вершин правило практически такое же, как и в обычных АР-сетях:

Обычная вершина $v \in V$ *активна* при разметке (M, M_e), если

- $M(v) \geq 1$;
- $\forall w \in (V \cup V_e) \;\; M(w) \geq I(w, v)$.

Активная при разметке (M, M_e) обычная вершина v может *сработать*, порождая новую разметку (M', M_e), где $\forall w \in (V \cup V_e) \;\; M'(w) = M(w) - I(w, v) + O(v, w)$. Такое срабатывание обозначается как $(M, M_e) \xrightarrow{v} (M', M_e)$.

Заметим, что работающие расширенные ресурсы (элементы разметки M_e) не могут быть потреблены при срабатывании.

Расширенная вершина $v \in V_e$ *готова начать работу* при разметке (M, M_e), если

- $M(v) \geq I(v, v) + 1$;
- $\forall w \in (V \cup V_e) \;\; M(w) \geq I(w, v)$.

Готовая начать работу при разметке (M, M_e) расширенная вершина v может *начать работу* (*заработать*), порождая новую разметку (M', M'_e), где

- $\forall w \in (V \cup V_e) \setminus \{v\} \;\; M'(w) = M(w) - I(w, v), \;\; M'_e(w) = M_e(w)$;
- $M'(v) = M(v) - I(v, v) - 1$;
- $M'_e(v) = M_e(v) + 1$.

Начало работы расширенной вершины v обозначается как $(M, M_e) \overset{start(v)}{\longrightarrow} (M', M'_e)$.

Расширенная вершина $v \in V_e$ *готова завершить работу* при разметке (M, M_e), если $M_e(v) > 0$.

Готовая завершить работу при разметке (M, M_e) расширенная вершина v может *завершить работу* (*отработать*), порождая новую разметку (M', M'_e), где

- $\forall w \in (V \cup V_e) \setminus \{v\} \;\; M'(w) = M(w) + O(v, w), \;\; M'_e(w) = M_e(w)$;
- $M'(v) = M(v) + 1 + O(v, v)$;
- $M'_e(v) = M_e(v) - 1$.

Завершение работы расширенной вершины v обозначается как $(M, M_e) \overset{finish(v)}{\longrightarrow} (M', M'_e)$.

Пример расширенного срабатывания агента приведен на рис. 5.

Введение расширенного срабатывания не меняет выразительности формализма:

Теорема 2. Класс сетей активных ресурсов с расширенными срабатываниями эквивалентен классу обыкновенных сетей Петри.

Доказательство см. в приложении.

Синтаксическая конструкция расширенного срабатывания более удобна для моделирования систем с неэлементарным поведением базовых компонентов. Так, классическая схема разделяемого доступа нескольких процессов к общему множеству ресурсов (например, к общей памяти) моделируется сетью с всего лишь двумя вершинами. На рис. 6 приведен пример системы с двумя ключами и тремя процессами. Для моделирования системы с другими наборами ключей и процессов достаточно поменять начальную разметку, не меняя сам граф. Заметим, что в сетях

Таблица 3. Формализация свойств одновременной работы агентов в РС-сетях

Определение $(v, w \in V_e,$ $\forall (M, M_e) \in \mathcal{R}(E, M_0, M_{e0}))$	**Интерпретация**
$M(v) = 0$ или $M(w) = 0$	Агенты видов v и w не могут работать одновременно.
$M_e(v) < n$	Степень параллелизма в узле v не превышает n.
$M(v) > 0$	Ресурсов в системе недостаточно для одновременной работы всех готовых к этому агентов вида v.

Таблица 4. Формализация свойств взаимных блокировок агентов в РС-сетях

Определение $(v, w \in V_e,$ $\exists (M, M_e) \in \mathcal{R}(E, M_0, M_{e0}))$	**Интерпретация**
$M_e(v) * I(w, v) > M(w),$ $M_e(w) * I(v, w) > M(v)$	Агенты видов v и w могут блокировать друг друга.
$\forall (M', M'_e) \in \mathcal{R}(E, M, M_e)$ $M'_e(v) * I(w, v) > M'(w),$ $M'_e(w) * I(v, w) > M'(v)$	Агенты видов v и w могут блокировать друг друга навсегда.
$M_e(v) > 0, I(w, v) > M(w),$ $M_e(w) > 0,\ I(v, w) > M(v)$	Все агенты видов v и w могут блокировать друг друга.
$\forall (M', M'_e) \in \mathcal{R}(E, M, M_e)$ $M'_e(v) > 0, I(w, v) > M'(w),$ $M'_e(w) > 0, I(v, w) > M'(v)$	Все агенты видов v и w могут блокировать друг друга навсегда.

Петри подобная гибкость определения достигается только при введении высокоуровневых конструкций.

Благодаря появлению понятия “работающего” агента мы можем формализовать свойства одновременной работы двух и более агентов (таблица 3). Также достаточно компактно формализуются свойства взаимных блокировок (таблица 4).

5. СЕТИ С НЕНАДЕЖНЫМИ СРАБАТЫВАНИЯМИ

Рассмотрим модель системы, в которой некоторые агенты могут срабатывать “ненадежно”, то есть не производя/не потребляя всех своих выходных/входных ресурсов. Очевидно, что в данном случае ненадежность – это свойство не столько самого агента, сколько его взаимодействия с конкретными ресурсами. Поэтому ненадежными мы объявляем не вершины графа, а дуги.

Сетью активных ресурсов с ненадежными срабатываниями (НС-сетью) назовем набор $U = (V, I, I_u, O, O_u)$, где

- V – конечное множество вершин (ресурсов);
- $I \in \mathcal{M}(V \times V)$ – мультимножество потребляющих дуг;
- $I_u \in \mathcal{M}(V \times V)$ – мультимножество ненадежных потребляющих дуг;
- $O \in \mathcal{M}(V \times V)$ – мультимножество производящих дуг;
- $O_u \in \mathcal{M}(V \times V)$ – мультимножество ненадежных производящих дуг.

На рисунках ненадежные дуги изображаются стрелками с незакрашенными наконечниками.

Определение разметки такое же, как и в случае обыкновенных АР-сетей.

Вершина $v \in V$ *активна* при разметке M, если

- $M(v) \geq 1$;
- $\forall w \in V \quad M(w) \geq I(w, v) + I_u(w, v)$.

Активная при разметке M вершина v может *сработать*, порождая любую разметку M' из ко-

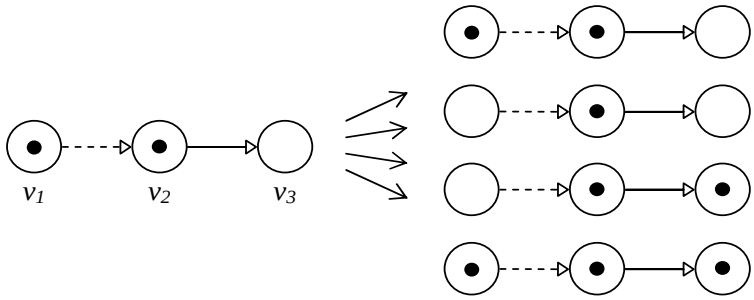


Рис. 7. Ненадежное срабатывание (в смысле потребления и в смысле производства ресурсов).

нечного множества

$$Res(M, v) = \{R \in \mathcal{M}(V) \mid \forall w \in V$$
$$R(w) = M(w) - I(w, v) - x + O(v, w) + y,$$
$$\text{где } 0 \le x \le I_u(w, v), 0 \le y \le O_u(v, w)\}.$$

Пример ненадежного срабатывания приведен на рис. 7. Здесь агент может сработать в четырех различных режимах.

Введение ненадежного срабатывания также не увеличивает выразительность (и не уменьшает разрешимость):

Теорема 3. Класс сетей активных ресурсов с ненадежными срабатываниями эквивалентен классу обыкновенных сетей Петри.

Доказательство см. в приложении.

Ненадежное срабатывание – дополнительный способ добавления недетерминизма в модель. Он может быть использован для моделирования различных неоднозначных и нечетких поведений. Например, замена в модели канала передачи данных части дуг на ненадежные приводит к построению канала связи с шумами (рис. 8). Здесь второе звено цепи ненадежно в том смысле, что может “забыть” о том, что пакет уже передан, и передать его еще раз. Третье звено может просто потерять пакет.

При помощи синтаксических средств НС-сетей можно формализовать такое важное семантическое свойство, как возможность замены надежного агента ненадежным без ущерба для системы в целом. Рассмотрим исходную НС-сеть U и сеть U', полученную из исходной путем замены надежной дуги (v, w) на ненадежную. Такая замена допустима при начальной разметке M_0, если $\mathcal{M}(U, M_0) = \mathcal{M}(U', M_0)$.

Концепция ненадежного срабатывания немного напоминает способ функционирования перехода в высокоуровневых сетях Петри [1]. Там также один переход может работать в нескольких режимах. Однако ненадежное срабатывание – это более частный случай, поскольку здесь мы не имеем возможности настолько точно задавать конкретные способы преобразования ресурсов. Но при этом и синтаксис модели здесь проще, чем в высокоуровневых сетях.

6. РАЗРАБОТКА РАСПРЕДЕЛЕННЫХ ПРИЛОЖЕНИЙ

В качестве примера использования моделей на базе АР-сетей рассмотрим систему управления распределенным приложением. Ядро системы представляет собой схему процесса (алгоритма), представленную в виде АР-сети. Каждой фишке соответствует объект, обладающий конечным набором атрибутов. Список атрибутов соответствует типу объекта, то есть вершине АР-сети, в которой находится данная фишка. Часть фишек представляют собой чистые ресурсы, то есть они не могут обладать собственным поведением, часть фишек – чистые агенты, то есть их количество фиксировано и не может меняться, но есть и такие фишки, которые могут выполнять действия, но при этом также могут пропадать и возникать (типичные примеры – веб-сервис и временный сотрудник).

Управляющая система проверяет (в соответствии с правилами АР-сетей) готовность к работе агента (сервиса) и при наличии в сети всех необходимых ресурсов запускает его на выполнение, передавая в качестве входных данных атрибуты входных фишек. После завершения работы сервиса система получает от него выходные данные и присваивает их значения соответ-

ствующим атрибутам произведенных выходных фишек (соответствие определяется по имени атрибута).

Одним из видов действия агента в такой системе может быть поиск и регистрация другого агента. Например, это может быть поиск веб-сервиса в интернете или прием на работу нового сотрудника (подобные ситуации легко могут быть промоделированы при помощи AP-сетей). То есть непосредственно в саму схему процесса распределенного приложения мы можем заложить принцип динамического изменения набора исполняющих модулей (агентов). Например, при помощи данной технологии можно организовывать приложения в виде динамических композиций веб-сервисов.

Рассмотрим простой пример подобной системы, приведенный на рис. 9 (здесь использована сеть с расширенными и ненадежными срабатываниями). Схема процесса представляет собой простой поток работ с одним входом и одним выходом.

Система получает на вход данные как атрибуты объектов-фишек в вершине in и выдает на выходе данные в виде значений атрибутов фишек в вершине out. Обработка делится на две части, которые могут выполняться параллельно. Распараллеливание и синхронизацию процесса вычислений осуществляют агенты preproc и postproc соответственно. Они же сохраняют служебную информацию о ходе обработки в системном журнале (вершина log).

Первый вид вычислений над данными производят внешние веб-сервисы (агенты в вершине wsrvs). Созданием этих агентов занимается агент alloc (он ищет в интернете подходящие веб-сервисы и регистрирует их в системном реестре). Агент dealloc занимается отслеживанием доступности зарегистрированных веб-сервисов и удаляет из реестра ссылки на недоступные. В ходе выполнения действия веб-сервис использует один из двух общесистемных ресурсов (забирая перед этим один из ключей доступа, хранящихся в вершине keys).

Второй вид вычислений над данными производит один из трех фиксированных внутренних сервисов системы (агенты в вершине srvs). Этим сервисам для работы также требуется доступ к общесистемным ресурсам (ключам в вершине keys).

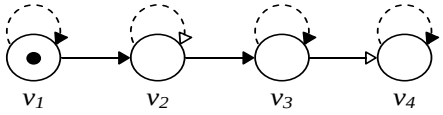


Рис. 8. Модель канала с шумами.

И веб-сервисы, и внутренние системные сервисы обладают расширенным срабатыванием, то есть для их выполнения требуется какое-то время. Временем срабатывания агентов preproc, postproc, alloc и dealloc можно пренебречь.

Агент alloc ненадежен в том смысле, что ему не при каждом срабатывании удается найти новый подходящий веб-сервис (а срабатывать он может в синхронном режиме, например, по таймеру). Аналогично, агент dealloc не всегда может удалить какой-нибудь из зарегистрированных веб-сервисов.

Простейший анализ семантических свойств модели средствами AP-сетей позволяет выяснить следующую полезную информацию о системе:

1. возможен избыток агентов вида wsrvs;
2. один агент вида srvs избыточен;
3. степень параллелизма в узлах wsrvs и srvs не превышает 2;
4. замена надежных узлов ненадежными невозможна.

7. ЗАКЛЮЧЕНИЕ

В работе рассмотрен новый способ моделирования систем – сети активных ресурсов. Этот формализм обладает достаточно простым и наглядным синтаксисом, позволяющим компактно формализовать ряд интересных семантических свойств систем со сложным поведением агентов. По выразительной мощности класс AP-сетей совпадает с классом обыкновенных сетей Петри, то есть в нем разрешимы все те же проблемы, что и в сетях Петри: достижимость, ограниченность, живость и др. При этом в сетях активных ресурсов появляется возможность “двойного” использования ресурса (фишки) – не только в качестве пассивного ресурса, но еще и в качестве активного агента. Это чисто синтаксическое

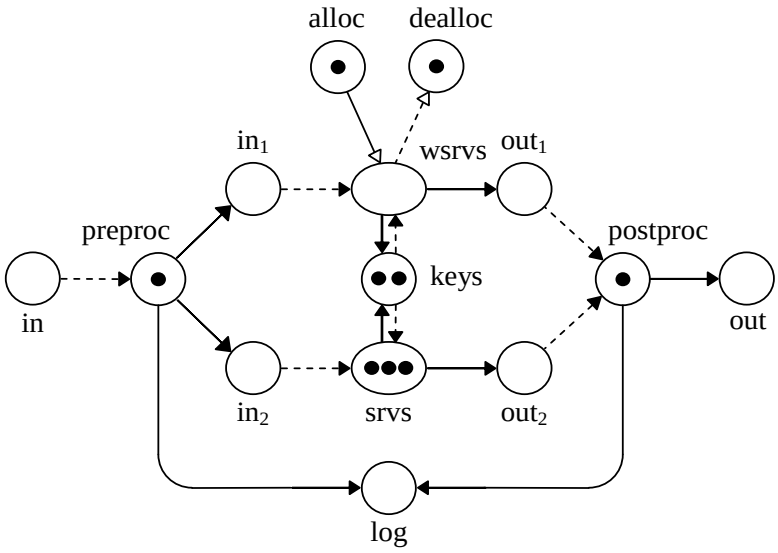


Рис. 9. Управляющий процесс распределенного приложения.

расширение может оказаться полезным на практике при моделировании и разработке систем с ненадежными агентами.

СПИСОК ЛИТЕРАТУРЫ

1. *Jensen K.* Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use. Springer, 1994.
2. *Ломазова И.А.* Вложенные сети Петри: моделирование и анализ распределенных систем с объектной структурой. М.: Научный мир, 2004.
3. *Petri C.A.* "Forgotten Topics" of Net Theory. Proc. of ATPN'1987. P. 500–514.
4. *Lautenbach K.* Duality of Marked Place/Transition Nets. Universität Koblenz-Landau. Institut für Informatik. Research Report № 18. 2003.
5. *Kohler M., Rolke H.* Super-Dual Nets. Proc. of CS&P'2005. Warsaw University, 2005. P. 271–280.
6. *Котов В.Е.* Сети Петри. М.: Наука, 1984.
7. *Kohler M., Rolke H.* Properties of Super-Dual Nets // Fundamenta Informaticae. 2006. V. 72. P. 245–254.

ПРИЛОЖЕНИЕ

Доказательство теоремы 1.

1) Докажем, что для любой сети Петри существует эквивалентная ей AP-сеть.

Действительно, мы можем переделать сеть Петри в AP-сеть, преобразовав переходы и позиции в вершины, дуги от позиций к переходам – в потребляющие дуги, дуги от переходов к позициям – в производящие дуги и добавив в начальной разметке по одной фишке в каждую вершину, получившуюся из перехода. В AP-сети множество достижимых разметок вершин, получившихся из позиций, будет совпадать с множеством достижимых разметок исходной сети Петри. Вершины, получившиеся из переходов, всегда будут размечены единицами.

2) Докажем, что для любой AP-сети существует эквивалентная ей сеть Петри.

Здесь можно применить способ моделирования, аналогичный предложенному в [5] для моделирования супер-двойственной сети Петри при помощи обыкновенной сети Петри.

Каждой вершине v в AP-сети сопоставим позицию p_v и переход t_v в сети Петри, связанные парой разнонаправленных дуг (p_v, t_v) и (t_v, p_v). Каждой потребляющей дуге (v, w) сопоставим дугу (p_v, t_w), а каждой производящей дуге (v, w) – дугу (t_v, p_w). Фишки из вершины v перенесем в позицию p_v. Очевидно, что множество достижимых разметок сети Петри будет совпадать с множеством достижимых разметок исходной AP-сети.

Доказательство теоремы 2.

В силу теоремы 1 нам достаточно доказать, что PC-сети эквивалентны AP-сетям.

Поскольку PC-сети являются синтаксическим расширением AP-сетей, нам достаточно доказать, что для любой PC-сети существует эквивалентная ей AP-сеть.

Расширенная вершина v преобразуется в две вершины – v_p и v_a (хранилища для пассивных ожидающих и активных работающих ресурсов). Все входные, создающие и уничтожающие дуги вершины v перенаправляются на v_p, все выходные – на v_a. Добавляются две новые потребляющие дуги (v_p, v_p) и (v_a, v_a), а также две новые производящие дуги (v_p, v_a) и (v_a, v_p).

Обычные вершины остаются такими же, как в исходной PC-сети. Если рассматривать в качестве аналога расширенной разметки вершины v исходной PC-сети разметку вершины v_a построенной AP-сети, то множества достижимых разметок будут совпадать.

Доказательство теоремы 3.

В силу теоремы 1 нам достаточно доказать, что HC-сети эквивалентны AP-сетям.

Поскольку HC-сети являются синтаксическим расширением AP-сетей, нам достаточно доказать, что для любой HC-сети существует эквивалентная ей AP-сеть.

Количество различных возможных режимов срабатывания (способов изменения разметки) у вершины v с ненадежными входными и/или выходными дугами конечно. Заменим вершину v на соответствующее множество вершин $Repl(v)$, заменяя часть ненадежных входных и выходных дуг на надежные и убирая все остальные. Все создающие, уничтожающие и надежные входные и выходные дуги просто размножим. Повторим эту процедуру для всех вершин.

В качестве начальной разметки поместим в каждую вершину из $Repl(v)$ ровно $M_0(v)$ фишек (где M_0 – начальная разметка исходной HC-сети). Заметим, что по построению у всех вершин из $Repl(v)$ мультимножества создающих и уничтожающих дуг одни и те же (то есть дуги ведут к одним и тем же вершинам). Следовательно, в любых достижимых разметках наборы фишек во всех вершинах из $Repl(v)$ будут совпадать, то есть мы можем рассматривать $Repl(v)$ как одну вершину. Определенное таким образом множество достижимых разметок совпадает с множеством достижимых разметок исходной HC-сети.