Ярославский государственный университет им. П. Г. Демидова

На правах рукописи

Башкин Владимир Анатольевич

Некоторые методы ресурсного анализа сетей Петри

05.13.17 – Теоретические основы информатики

ДИССЕРТАЦИЯ
на соискание ученой степени
доктора физико-математических наук

Научный консультант
д. ф.-м. н., проф.
И. А. Ломазова

Ярославль – 2014

Содержание

Введение

Актуальность темы исследования. В последние десятилетия большой и устойчивый интерес проявляется к математическим средствам моделирования и анализа сложных параллельных и распределенных систем, к которым относятся, например, вычислительные машины и комплексы с параллельной и распределенной архитектурой, параллельные программы и алгоритмы, протоколы взаимодействия (коммуникационные, верифицирующие), модели технологических и бизнес–процессов. При разработке подобных систем, как правило, высок риск возникновения ошибки на стадии проектирования и чрезвычайно высока цена проявления ошибки на стадии эксплуатации. Все это обуславливает интерес к формальным математическим средствам анализа и верификации, позволяющим дать ответы на вопросы о свойствах модели, например, о наличии конфликтов, тупиков или неограниченных состояний (переполнений).

Одним из наиболее популярных формализмов для моделирования и анализа параллельных и распределенных систем являются сети Петри. Понятие сети Петри было введено в 1962 году Карлом-Адамом Петри. С тех пор теория сетей Петри сильно разрослась и продолжает активно развиваться. Причиной популярности сетей Петри является математическая строгость, простота и наглядность описания параллелизма, а также удобное графическое представление модели.

Среди отечественных исследований по сетям Петри и спецификации и анализу распределенных систем отметим работы Н. А. Анисимова, О. Л. Бандман, И. Б. Вирбицкайте, В. В. Воеводина, Н. В. Евтушенко, В. А. Захарова, Ю. Г. Карпова, В. Е. Котова, И. А. Ломазовой, В. А. Непомнящего, Р. И. Подловченко, Р. Л. Смелянского, В. А. Соколова, Л. Н. Столярова, Л. А. Черкасовой.

Сети Петри позволяют с достаточной степенью детализации моделировать вычислительные процессы, процессы управления в распределенных системах и протоколы взаимодействия. В них имеются простые конструкции для описания структур параллелизма: последовательная композиция, выбор, параллельное

слияние. Сети Петри представляют собой модель с бесконечным числом состояний, при этом они менее выразительны, чем универсальные вычислительные системы типа машин Тьюринга, что позволяет решать для них многие задачи анализа поведенческих свойств. В частности, для обыкновенных сетей Петри разрешимы проблемы достижимости, останова, живости, ограниченности, безопасности, покрытия. В то же время неразрешимыми остаются многие поведенческие эквивалентности, в частности, эквивалентность множеств достижимости и бисимуляционная эквивалентность.

Можно сказать, что сети Петри являются относительно сбалансированным инструментом, в котором имеются как интересные возможности моделирования, так и достаточно обширный набор алгоритмов анализа.

Одно из классических неформальных определений сети Петри — “асинхронная распределенная система с ресурсами”. Действительно, ключевыми характеристиками данного класса формальных моделей являются распределенность управления (возможность одновременного независимого функционирования различных частей системы) и локальная ресурсная синхронизация (возможность коммуникации между “соседними” частями системы посредством производства и потребления неких общих ресурсов).

Обычно неформальным термином “ресурс” обозначают содержимое позиции (её разметку) по отношению к использующим это содержимое переходам. То есть ресурс — это то, что может быть произведено или потреблено срабатываниями переходов сети. В данной работе предлагается более широкое толкование понятия ресурса сети Петри:

Глобальный ресурс — это разметка сети Петри, то есть состояние системы, выраженное в виде мультимножества фишек.

(Локальный) ресурс — это такая подразметка сети Петри (мультимножество фишек), которая каким-либо выраженным образом характеризует все содержащие её разметки.

Приведённое определение ресурса всё ещё неформально, однако позволяет по-новому взглянуть на многие аспекты теории сетей Петри. Это представляется особенно актуальным в связи с ростом интереса в последнее время к таким сферам теоретической информатики, как моделирование и анализ схем потоков работ (workflow), сервис-ориентированных архитектур, GRID-вычислений и т.п. Во всех перечисленных классах задач ресурсам (вычислительным, расходным, производимым и т.д.) и их свойствам (достаточности, эквивалентности, достижимости и т.д.) отводится первостепенное значение. Нам представляется, что разработка ресурсно-ориентированных подходов и алгоритмов позволит существенно расширить область применения формальных математических методов при создании процессно- и сервис-ориентированных распределенных систем.

Цели и задачи диссертационной работы.

Целью работы является создание новых методов моделирования и анализа параллельных и распределенных систем на основе понятия ресурса сети Петри.

Для достижения поставленной цели концепция ресурса может быть использована различными способами:

1. Как объект анализа. Рассматриваются свойства локальных ресурсов с точки зрения эквивалентности их воздействия на поведение системы.

2. Как инструмент классификации. Свойства ресурса (размерность) используются в качестве параметра при определении сужений класса сетей Петри, обладающих новыми конструктивными свойствами.

3. Как инструмент моделирования. Исследуются выразительные возможности концепции обобщенных (активных) ресурсов, допускающей двойственное или же более широкое толкование по сравнению с концепцией позиций/переходов классических сетей Петри.

Научная новизна. Основные результаты могут быть сгруппированы в соответствии со способом использования понятия ресурса следующим образом:

Методы анализа поведенческих эквивалентностей ресурсов.

Исследованы возможности бисимуляционного анализа поведения систем посредством выделения подобных ресурсов. Предложенные методы позволяют находить нетривиальные подмножества максимальной бисимуляции разметок, в общем случае невычислимой (П. Джанкар, 1994).

Базовые понятия теории эквивалентностей ресурсов сетей Петри были сформулированы в кандидатской диссертации соискателя. На защиту выносятся новые результаты, описывающие ключевые структурные свойства и алгоритмические приемы данной теории.

Введены и исследованы специальные виды отношения бисимуляции для случая ограниченных сетей, в том числе расширение бисимуляции достижимых разметок — отношение, учитывающее кроме достижимых разметок ещё и все бисимулярные достижимым (среди которых могут быть и неограниченные).

Предложены способы приближения отношения подобия снизу (при помощи ограниченного подобия) и сверху (при помощи расслоённого подобия). Разработаны методы адаптивного управления процессами на основе различных отношений эквивалентности ресурсов.

Представленные методы существенно мощнее известных подходов, основанных на отношении слияния позиций (Ф. Шнеблен, Н. С. Сидорова, 2000–2003), поскольку рассматривают ресурсы произвольной мощности.

Методы моделирования и анализа систем с одномерным ресурсом.

Разработана теория символьного анализа односчетчиковых сетей (сетей Петри с одной неограниченной позицией).

Доказано, что всякое полулинейное множество натуральных чисел может быть представлено при помощи однопериодического базиса: объединения конечного множества и конечного набора однопериодических линейных множеств с одинаковым периодом. Предложены оценки числовых характеристик бесконечной периодической части одномерного полулинейного множества, использующие наилучшие существующие (на текущий момент) приближения для обоб-

щенных чисел Фробениуса.

Введено понятие однопериодического базиса одномерного полулинейного множества. Показано, что такие базисы обладают нормальной формой и рядом конструктивных свойств, которые позволяют использовать их в качестве удобного и эффективного инструмента символьных вычислений. Данное представление в одномерном случае является удобной заменой известного способа символьного представления множеств достижимости полулинейных систем при помощи формул арифметики Пресбургера (Х. Комон, Ю. Юрский, 1998, и др.).

Доказано, что основной характеристикой диаграммы переходов управляющего автомата односчётчиковой сети, определяющей числовые свойства периодического базиса, является наибольший общий делитель эффектов (длин со знаком) всех простых циклов сильно связных компонент.

Определен и исследован класс односчетчиковых контуров — систем, представимых в виде односчетчиковых сетей с сильно-связными управляющими автоматами. Односчетчиковые контуры обладают удобным графическим представлением и рядом важных конструктивных свойств. Доказано, что произвольная односчетчиковая сеть представима в виде дерева односчетчиковых контуров.

Разработан ряд алгоритмов анализа односчетчиковых сетей Петри, использующих однопериодическое представление одномерного полулинейного множества достижимых состояний. В частности, для односчётчиковых сетей предложен алгоритм построения символьной свёртки пространства состояний, который позволяет получать более компактное (однопериодическое полулинейное) представление, чем известные для одно- и двухсчётчиковых сетей способы символьного описания при помощи деревьев линейных базисов множеств разметок (Дж. Хопкрофт, Ж.–Ж. Пансио, 1975) и полулинейных формул путей (Ж. Леру, Г. Сютре, 2003–2005).

Сети потоков работ (WF-сети) представляют собой специальный подкласс сетей Петри, используемый для формализации управления технологическими процессами, бизнес-процессами, web-сервисами, распределенными вычислени-

ями и т.д. Их основная особенность — структурные ограничения, накладываемые на граф сети и гарантирующие правильное завершение любого варианта исполнения процесса.

Предложено обобщение класса WF-сетей — сети с одним неограниченным ресурсом. Этот класс достаточно интересен, поскольку позволяет моделировать многие прикладные системы — например, WF-процессы с дискретным временем. Доказана разрешимость бездефектности как для размеченной, так и для неразмеченной одномерной сети (предложены алгоритмы). Предложен алгоритм определения минимального бездефектного ресурса. Эти результаты обобщают до неограниченного случая известные результаты о разрешимости бездефектности для сетей потоков работ с фиксированным ресурсом (К. ван Ней, Н. С. Сидорова и др., 2005–2013).

Методы моделирования и анализа систем с бесконечномерным ресурсом.

Предложено обобщение формализма сетей Петри на случай бесконечной регулярной системной сети — клеточные P-сети. Клеточные сети представляют собой объединение двух концепций — счетчиковых сетей (сетей Петри) и клеточных автоматов. Подобная двойственность позволяет моделировать как асинхронный параллелизм, так и пространственную динамику.

Построена иерархия классов одномерных клеточных сетей (цепочек), основанная на ограничении топологии системной сети. Исследована выразительная мощность ряда базовых классов данной иерархии. Доказано, что: сети с полным набором портов эквивалентны машинам Тьюринга; сети без выходных портов эквивалентны конечным автоматам; сети без входных портов бисимулярны сетям Петри без коммуникаций (communication-free PN).

Методы моделирования и анализа систем, основанные на обобщении понятия ресурса.

Исследованы возможности развития языка сетей Петри засчет явного синтаксического выделения ресурсов и наделения их расширенной семантикой.

Предложен новый формализм моделей распределенных систем, названный

сетями активных ресурсов (АР-сетями). В отличие от обыкновенных сетей Петри, в нём убрано разделение компонентов системы на активные и пассивные (переходы и позиции). Каждый объект (фишка) может выступать и в качестве пассивного ресурса, потребляемого или производимого другими агентами, и в качестве активного агента, потребляющего и производящего другие ресурсы. Это позволяет решить известную проблему неудобства моделирования сетями Петри систем с динамической распределенной структурой, и при этом избежать введения отдельных конструкций для ресурсов и агентов (как это делается в других известных формализмах, например, в сетях М. Кёлера и Х. Рольке, 2006).

Доказано, что АР-сети и АР-сети с простым срабатыванием равномощны обыкновенным сетям Петри. Показано, что они обладают достаточно простым и наглядным синтаксисом, позволяющим компактно формализовать ряд интересных семантических свойств систем со сложным поведением агентов.

Введено и исследовано понятие модуля в АР-сетях. Показано, что, в отличие от классических сетей Петри, однородная структура вершин графа АР-сети позволяет использовать некоторые специфические модульные методы разработки и реорганизации систем. Изучены свойства разбиений сети на модули и эквивалентных замен модулей.

Предложены новые формализмы, использующие концепцию АР-сетей для моделирования распределенных систем с динамической структурой и ненадежными компонентами. По сравнению с известными формализмами (раскрашенные сети К. Йенсена, объектные сети Р. Фалька, вложенные сети И. А. Ломазовой, гиперсети В. Павловского и др.) предложенные нами языки моделирования обладают дополнительными возможностями описания процессно- и сервис-ориентированных систем, обусловленными дуалистичностью поведения активного ресурса (агента/ресурса).

Теоретическая и практическая значимость. Полученные результаты имеют в основном теоретический характер. Они также могут быть использованы для решения практических задач, в частности, при построении программных ком-

плексов разработки, верификации и оптимизации программ и систем, а также языков визуализации и средств управления процессами.

Положения, выносимые на защиту.

1. Методы поиска бисимуляционно-эквивалентных ресурсов в сетях Петри. Обладающие конструктивными свойствами расширения и сужения отношения подобия ресурсов.

2. Методы бисимуляционной редукции систем и адаптивного управления процессами на основе отношений эквивалентности ресурсов.

3. Способ описания бесконечной части пространства состояний односчётчиковой сети Петри при помощи арифметических прогрессий, характеристики которых выражаются как числа Фробениуса от эффектов циклов односчётчиковых контуров (сильно связных компонент сети).

4. Методы анализа и оптимизации односчётчиковых сетей на основе символьных вычислений при помощи однопериодических базисов (глобальная достижимость, глобальная верификация темпоральной логики EF, аппроксимация бисимуляции, правильная организованность).

5. Методы проверки бездефектности сетей потоков работ с одномерным неограниченным ресурсом.

6. Формализм сетей активных ресурсов (AP-сети) — развитие языка сетей Петри засчет явного синтаксического выделения ресурсов и наделения их расширенной семантикой.

7. Композициональные методы анализа AP-сетей и методы нормализации их модульной структуры.

8. Расширения формализма AP-сетей для моделирования тех или иных аспектов распределенных систем: AP-сети с динамическими и ненадежными

компонентами, ингибиторные AP-сети, сети управляемых ресурсами автоматов (P-сети).

9. Формализм клеточных P-сетей — обобщение сетей Петри на случай бесконечной регулярной системной сети. Иерархия классов одномерных клеточных сетей (цепочек).

Степень достоверности и апробация результатов. Основные результаты диссертации докладывались на научных семинарах ЯрГУ им. П. Г. Демидова и ВЦ РАН им. А. А. Дородницына, а также на международных конференциях: “Интеллектуальное управление: новые интеллектуальные технологии в задачах управления” (Переславль–Залесский 1999); “Concurrency, Specification and Programming” (Берлин 2002, Руциане–Нида 2005, Берлин 2006, Краков 2009, Берлин 2010, Пултуск 2011, Берлин 2012); “Parallel Computing Technologies” (Нижний Новгород 2003, Москва 2005, Санкт-Петербург 2013); “Методы и средства обработки информации” (Москва 2005); “Интеллектуальные системы и компьютерные науки” (Москва 2006); “Программные системы: теория и приложения” (Переславль–Залесский 2006); “Дискретные модели в теории управляющих систем” (Москва 2006, 2009); “Параллельные вычисления и задачи управления” (Москва 2008); “Компьютерные науки и технологии” (Белгород 2009); “Информационные технологии в науке, образовании и производстве” (Орёл 2010); “Семантика, спецификация и верификация программ: теория и приложения” (Казань 2010, Санкт–Петербург 2011, Н.Новгород 2012, Екатеринбург 2013); “Petri Net Compositions (Petri Nets 2011)” (Ньюкасл–на–Тайне 2011); “Petri Nets and Software Engineering (Petri Nets 2013)” (Милан 2013).

Публикации. Материалы диссертации опубликованы в 55 печатных работах, из них 16 статей в изданиях, рекомендованных ВАК (8 статей в российских журналах из перечня ВАК [12, 18, 21, 22, 24, 25, 27, 31] и 8 статей в зарубежных изданиях, входящих в международные индексы цитирования из перечня ВАК [23, 34, 79–81, 86, 87, 89]), 10 статей в прочих рецензируемых журна-

лах [3, 4, 7, 10, 66, 68, 70, 73, 77, 85], 27 статей в сборниках трудов конференций и прочих сборниках [5, 8, 9, 11, 13, 14, 16, 17, 19, 20, 26, 28–30, 33, 67, 69, 71, 72, 74–76, 78, 82–84, 88], одна монография [32] и одно учебное пособие [15]).

Личный вклад автора. Результаты, изложенные в диссертации, получены автором самостоятельно. В коллективных публикациях автору принадлежат те их части, которые составляют основу диссертации.

Гранты. Работа по теме диссертации поддерживалась следующими грантами: РФФИ №№ 99-01-00-309-а, 03-01-00804-а, 06-01-00106-а, 07-01-07038-д, 07-01-00702-а, 09-01-00277-а, 09-0109370-моб_з, 11-01-00737-а, 11-07-00549-а, 12-01-00281-а, 12-01-31508-а; федеральная целевая программа “Научные и научно-педагогические кадры инновационной России” (проекты 2009-1.1113-050--013 и 14.B37.21.0392).

Структура и объем. Работа состоит из введения, четырёх глав, заключения и списка литературы (209 пунктов). Общий объем работы 268 страниц, включая 92 рисунка.

Глава 1

Предварительные сведения

1.1. Множества и отношения

Через $\mathbf{Z}, \mathbf{Z}_-$ и $\mathbf{Z}_+$ обозначим, соответственно, множества целых, отрицательных целых и положительных целых чисел. Через *Nat* обозначим множество неотрицательных целых чисел.

Обозначение 1.1. *Пусть X и Y — два множества. Будем использовать обозначения:*

- $X \subseteq Y$, *если X является подмножеством Y;*
- $X \subset Y$, *если X является собственным подмножеством Y;*
- $|X|$ — *мощность множества X;*
- $X \cup Y$ — *объединение множеств X и Y;*
- $X \cap Y$ — *пересечение множеств X и Y;*
- $X \times Y = \{(x, y) \mid x \in X \ \wedge \ y \in Y\}$ — *декартово произведение множеств X и Y.*

Определение 1.1. *Пусть X и Y — некоторые множества. Отношение R между множествами X и Y — это подмножество множества $X \times Y$.*

Будем говорить, что отношение R определено на множестве X, если R — это отношение между X и X.

Обозначение 1.2. *Пусть X — некоторое множество, R — отношение, определенное на X. Будем использовать обозначения:*

- $Id(X) = \{(x, x) \mid x \in X\}$ — *отношение идентичности на X;*

- $R^{-1} = \{(y, x) \mid (x, y) \in R\}$ — *отношение, обратное R;*
- $R_1 \circ R_2 = \{(x, y) \mid \exists z : (x, z) \in R_2 \ \wedge \ (z, y) \in R_1\}$ — *композиция отношений* R_1 *и* R_2.

Отношение R на множестве X есть отношение эквивалентности, если оно рефлексивно, симметрично и транзитивно.

Обозначение 1.3. *Для стандартных отношений на X будем использовать следующие обозначения:*

- R^k, *где* $k \in Nat$, *задается рекурсивным определением:*

  $R^0 = Id(X)$,

  для $k \geq 1$, $R^k = R^{k-1} \circ R$.

- $R^+ = R^1 \cup R^2 \cup \ldots$ — *транзитивное замыкание R;*
- $R^* = Id(X) \cup R^+$ — *рефлексивное транзитивное замыкание;*
- $\widetilde{R}$ — *рефлексивно-симметрично-транзитивное замыкание R. Очевидно, что это — наименьшее отношение эквивалентности на X, содержащее R.*

Определение 1.2. *Пусть A — некоторое множество. Конечной последовательностью на A будем называть отображение множества* $\{1, 2, \ldots, n\}$ *в A. Отображение* $\varepsilon : \emptyset \to A$ *будем называть пустой последовательностью.*

Пусть $\sigma : 1, \ldots, n \to A : a_1 \ldots a_n$ — конечная последовательность, тогда ее длину n будем обозначать $|\sigma|$. Длина пустой последовательности ε равна 0.

Проекцией последовательности σ на множество $A' \subseteq A$ будем называть ее подпоследовательность $s_{|A'}$, состоящую из тех и только тех элементов, которые входят в A'.

1.1.1. Мультимножества

Мультимножество является естественным обобщением множества. В мультимножестве один объект может находиться в нескольких экземплярах.

Пусть X — непустое множество.

Определение 1.3. Мультимножеством M *над множеством* X *называется функция* $M : X \to Nat$.

Мощность *мультимножества* $|M| = \sum_{x \in X} M(x)$.

Числа $\{M(x) \mid x \in X\}$ называются коэффициентами мультимножества, коэффициент $M(x)$ определяет число экземпляров элемента x в M. Если $\forall x \in X\ M(x) \leq 1$, то M является обычным множеством.

Мультимножество M *конечно*, если конечно множество

$$\{x \in X \mid M(x) > 0\}.$$

Множество всех конечных мультимножеств над множеством X обозначается как $\mathcal{M}(X)$.

Операции и отношения теории множеств естественным образом расширяются на конечные мультимножества.

Определение 1.4. *Пусть* $M_1, M_2, M_3 \in \mathcal{M}(X)$. *Полагаем:*

- $M_1 = M_2 \ \Leftrightarrow\ \forall x \in X\ M_1(x) = M_2(x)$ — *отношение равенства;*
- $M_1 \subseteq M_2 \ \Leftrightarrow\ \forall x \in X\ M_1(x) \leq M_2(x)$ — *отношение включения;*
- $M_1 \subset M_2 \ \Leftrightarrow\ M_1 \subseteq M_2 \ \wedge\ \exists x \in X\ M_1(x) \leq M_2(x)$ — *отношение строгого включения;*
- $M_1 = M_2 + M_3 \ \Leftrightarrow\ \forall x \in X\ M_1(x) = M_2(x) + M_3(x)$ — *операция сложения двух мультимножеств;*

- $M_1 = M_2 \cap M_3 \Leftrightarrow \forall x \in X\ M_1(x) = min(M_2(x), M_3(x))$ — *операция пересечения двух мультимножеств;*

- $M_1 = M_2 - M_3 \Leftrightarrow \forall x \in X\ M_1(x) = M_2(x) \ominus M_3(x)$ — *разность двух мультимножеств (где* $\ominus$ *— вычитание до нуля);*

- $M_1 = kM_2,\ k \in Nat \Leftrightarrow \forall x \in X\ M_1(x) = kM_2(x)$ — *операция умножения мультимножества на скаляр.*

Пример 1.1. *Рассмотрим* M_1 *и* M_2 *— мультимножества над множеством* $X = \{a, b, c\}$*, такие, что* $M_1 = \{a, a, b\}$, $M_2 = \{b, c\}$.

Выполняется:

$M_1(a) = 2,\ M_1(b) = 1,\ M_1(c) = 0$;

$M_1 \not\subset M_2,\ M_1 \not\supset M_2$;

$M_1 + M_2 = \{a, a, b, b, c\}$;

$M_1 \cap M_2 = \{b\}$;

$M_1 - M_2 = \{a, a\}$;

$2M_1 = \{a, a, a, a, b, b\}$.

Одним из способов записи мультимножеств являются вектора над *Nat*. При этом различным координатам вектора сопоставляются различные элементы X.

Пример 1.2. *Мультимножества* M_1 *и* M_2 *из примера 1.1 могут быть записаны как*

$$M_1 = (2, 1, 0), \quad M_2 = (0, 1, 1).$$

Утверждение 1.1. *Множество* $\mathcal{M}(X)$ *мультимножеств над* X *изоморфно множеству* $Nat^{|X|}$ *векторов длины* $|X|$ *с целочисленными неотрицательными коэффициентами.*

Доказательство: Непосредственно из определений. □

Рассмотрим мультимножества, представленные в виде векторов.

Определение 1.5. *Множество векторов* $m \subseteq Nat^k$ *называется* линейным, *если*

$$m = \{v + n_1 w_1 + \ldots + n_l w_l \mid n_1, \ldots, n_l \in Nat\}, \text{ где } v, w_1, \ldots, w_l \in Nat^k.$$

Определение 1.6. *Множество называется* полулинейным, *если оно является объединением конечного числа линейных множеств.*

Определим также операции сдвига множеств векторов:

Определение 1.7. *Пусть* $M \subseteq \mathcal{M}(X)$ *и* $m \in \mathcal{M}(X)$. *Полагаем:*

- $M \triangleright m =_{def} \{m' + m \mid m' \in M\}$;
- $M \triangleleft m =_{def} \{m' - m \mid m' \in M \text{ и } m' \geq m\}$.

Для полулинейных множеств $M, M' \subseteq Nat^k$ и вектора $m \in Nat^k$ множества $Nat^k \setminus M$, $M \cup M'$, $M \cap M'$, $M \triangleright m$ и $M \triangleleft m$ также полулинейны [129]. Полулинейные множества — в точности все множества, выразимые при помощи формул арифметики Пресбургера ([182], см., например, [36]).

1.1.2. Отношения на мультимножествах

В этом подразделе приведены некоторые результаты из диссертации [6], касающиеся возможности представления бесконечных бинарных отношений на мультимножествах при помощи конечных базисов.

Пусть $(M_1, M_2), (M'_1, M'_2) \in \mathcal{M}(X) \times \mathcal{M}(X)$ — пары мультимножеств над множеством X. Их сумма определяется как

$$(M_1, M_2) + (M'_1, M'_2) =_{def} (M_1 + M'_1, M_2 + M'_2).$$

Пусть $B \subseteq \mathcal{M}(X) \times \mathcal{M}(X)$ — бинарное отношение на множестве мультимножеств над X.

Определение 1.8. *Для данного отношения B его* аддитивным замыканием B^A *называется наименьшее (по вложению) подмножество множества $\mathcal{M}(X) \times \mathcal{M}(X)$, такое, что*

1. $B \subseteq B^A$;
2. $\forall (M_1, M_2),\ (M_1', M_2') \in B^A \quad (M_1 + M_1', M_2 + M_2') \in B^A$.

Определение 1.9. *Для данного отношения B его* транзитивным замыканием B^T *называется наименьшее (по вложению) подмножество множества $\mathcal{M}(X) \times \mathcal{M}(X)$, такое, что*

1. $B \subseteq B^T$;
2. $\forall (M_1, M_2),\ (M_2, M_3) \in B^T \quad (M_1, M_3) \in B^T$.

Определение 1.10. *Для данного отношения B его* аддитивным транзитивным замыканием B^{AT} *называется наименьшее (по вложению) подмножество множества $\mathcal{M}(X) \times \mathcal{M}(X)$, такое, что*

1. $B \subseteq B^{AT}$;
2. $\forall (M_1, M_2),\ (M_1', M_2') \in B^{AT} \quad (M_1 + M_1', M_2 + M_2') \in B^{AT}$;
3. $\forall (M_1, M_2),\ (M_2, M_3) \in B^{AT} \quad (M_1, M_3) \in B^{AT}$.

В дальнейшем будем использовать сокращения A-замыкание, T-замыкание и AT-замыкание соответственно.

В случае отношений на мультимножествах AT-замыкание отношения не всегда совпадает с аддитивным замыканием транзитивного замыкания отношения и с транзитивным замыканием аддитивного замыкания отношения. Строго говоря, имеет место следующая вложенность друг в друга различных видов замыканий отношений на мультимножествах (см. также рисунок 1.1):

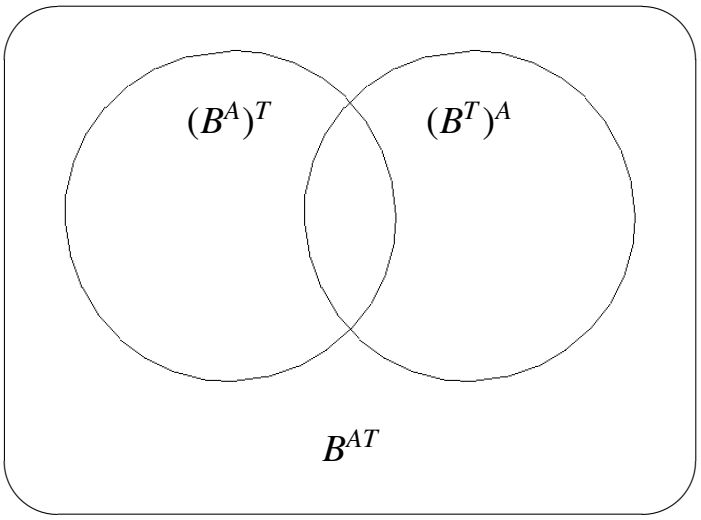


Рис. 1.1. Вложенность различных видов замыканий

Лемма 1.1. *[6]*

1. $\forall\ X,\ B \in \mathcal{M}(X)\times\mathcal{M}(X)$ *выполняется* $(B^A)^T \subseteq B^{AT}$;
2. $\forall\ X,\ B \in \mathcal{M}(X)\times\mathcal{M}(X)$ *выполняется* $(B^T)^A \subseteq B^{AT}$;
3. $\exists\ X,\ B \in \mathcal{M}(X)\times\mathcal{M}(X)\ :\ (B^A)^T \subset B^{AT}$;
4. $\exists\ X,\ B \in \mathcal{M}(X)\times\mathcal{M}(X)\ :\ (B^T)^A \subset B^{AT}$;
5. $\exists\ X,\ B \in \mathcal{M}(X)\times\mathcal{M}(X)\ :\ (B^A)^T \not\subseteq (B^T)^A$;
6. $\exists\ X,\ B \in \mathcal{M}(X)\times\mathcal{M}(X)\ :\ (B^T)^A \not\subseteq (B^A)^T$.

Таким образом, AT-замыкание — наиболее эффективный (из представленных) способ построения бесконечных отношений (получаемые с его помощью бесконечные отношения самые слабые).

На практике в теории формальных моделей мы почти всегда имеем дело с транзитивно замкнутыми отношениями на множестве состояний исследуемой системы (или даже отношениями эквивалентности). Накладывая дополнительное требование аддитивной замкнутости, мы получаем возможность выделять некоторые “структурированные” подмножества базового отношения (возможно,

представимые конечным базисом). Как показано в лемме 1.1, аддитивную замкнутость можно вводить *тремя* различными способами.

Определим, в каких случаях конечный базис существует всегда (у любого отношения B, удовлетворяющего заданным ограничениям).

Пусть $\alpha \in \{A, T, AT\}$.

Определение 1.11. *Отношение* $B' \subseteq \mathcal{M}(X) \times \mathcal{M}(X)$ *называется* α-базисом *отношения* B, *если* $(B')^{\alpha} = B^{\alpha}$.

Базис B' *называется* минимальным α-базисом отношения B, *если не существует* $B'' \subset B'$, *такого что* $(B'')^{\alpha} = B^{\alpha}$.

Лемма 1.2. *[6]*

1. *Все минимальные* α*-базисы отношения* B *либо конечны, либо бесконечны.*
2. *Если у* B *существует конечный* α*-базис, то все минимальные* α*-базисы отношения* B *конечны;*
3. *Если у* B *существует бесконечный минимальный* α*-базис, то все* α*-базисы отношения* B *бесконечны.*

Рассмотрим, при каких условиях α-замкнутое и в общем случае бесконечное отношение B обладает конечным α-базисом, то есть может быть представлено конечным числом элементов. В качестве возможных ограничений используются требования симметричности, рефлексивности и транзитивности отношения B.

Рассмотрим в качестве α аддитивные и транзитивные замыкания. Оказывается, даже если B — отношение эквивалентности, у него могут существовать бесконечные минимальные базисы.

Пример 1.3. *В качестве примера для обоих случаев рассмотрим множество* X *из одного элемента и отношение эквивалентности*

$$B = \{(i, i), (1, i + 2), (i + 2, 1) \mid i \in Nat\}$$

(пары мультимножеств представлены как пары чисел).

Рассмотрим случай A-замыкания.

Пусть $B' = \{(0,0),(1,1)\} \cup \{(1, i+2),(i+2,1) \mid i \in Nat\}$. *Отношение* B' *является A-базисом отношения* B^A. *Докажем минимальность* B'. *Предположим противное: пусть некоторая пара* $(x, y) \in B'$ *может быть получена аддитивным замыканием других пар из* B'. *Однако при любом сложении мультимножеств видов* $(1, i+2)$, $(i+2, 1)$ *или* $(1, 1)$ *(пустую пару можно не рассматривать) в итоговой паре оба коэффициента становятся больше единицы — противоречие.*

Рассмотрим случай T-замыкания.

Пусть $B' = \{(0,0)\} \cup \{(1, i+2),(i+2,1) \mid i \in Nat\}$. *Очевидно, что* B' *является T-базисом отношения* B^T *(любая пара вида* (i, i) *может быть получена транзитивным замыканием двух симметричных пар). Докажем минимальность* B'. *Предположим противное: некая пара* $(x, y) \in B'$, *где* $x \neq y$, *может быть получена транзитивным замыканием других пар из* B'. *Однако при любом транзитивном замыкании мультимножеств видов* $(1, i+2)$ *и* $(i+2, 1)$ *может получиться только рефлексивная пара — противоречие.* □

Итак, ни аддитивное, ни транзитивное замыкание не могут быть использованы для генерации бесконечных отношений на основе конечных базисов, даже если B — отношение эквивалентности. Кроме того, легко показать, что существуют аддитивные и транзитивные замыкания отношений эквивалентности, которые сами не являются отношениями эквивалентности.

Рассмотрим в качестве α аддитивное транзитивное замыкание.

Потребуется одно техническое определение:

Определение 1.12. *Отношение* $R \in \mathcal{M}(X) \times \mathcal{M}(X)$ *называется* 1-рефлексивным, *если* R *содержит все пары вида* (x, x), *где* $|x| = 1$.

В случае аддитивных транзитивных замыканий вместо полной рефлексивности отношения B достаточно требовать только 1-рефлексивности, так как аддитивное замыкание 1-рефлексивного отношения рефлексивно (с точностью до

пустой пары $(\emptyset, \emptyset)$). Использовать же полную рефлексивность неудобно чисто технически, так как рефлексивные отношения на мультимножествах бесконечны a priori.

Во-первых, заметим, что существуют симметричные отношения с бесконечными минимальными AT-базисами.

Пример 1.4. *В качестве примера рассмотрим множество X из двух элементов и симметричное отношение*

$$B = \{((i, i+1), (i, i+1)) \mid i \in Nat, i > 0\}$$

(пары мультимножеств представлены как пары векторов длины 2).

Покажем, что базис B – минимальный. Пусть

$$B_k = \{((i, i+1), (i, i+1)) \mid i, k \in Nat, k \geq i > 0\}$$

– первые k элементов базиса B. Нам достаточно показать, что $(k+1)$*-й элемент не может быть выражен через предыдущие, то есть*

$$((k+1, k+2), (k+1, k+2)) \notin B_k^{AT}$$

Заметим, что и в левой, и в правой части всех пар векторов из B второй коэффициент на единицу больше первого. При сложении любых двух пар в левой и правой части получатся вектора, где второй коэффициент больше первого на 2. Только при транзитивном замыкании таких пар разность коэффициентов не меняется. Однако, очевидно, что одним транзитивным замыканием пару $((k+1, k+2), (k+1, k+2))$ *из* B_k *не получить.* □

Во-вторых, существуют 1-рефлексивные отношения с бесконечными минимальными AT-базисами.

Пример 1.5. *В качестве примера рассмотрим множество X из двух элементов и 1-рефлексивное отношение*

$$B = \{((1,0),(1,0)), ((0,1),(0,1))\} \cup \{((1,0),(i,1)) \mid i \in Nat, i \geq 2\}$$

(пары мультимножеств представлены как пары векторов длины 2).

Покажем, что базис B – минимальный. Пусть

$$B_k = \{((1,0),(1,0)),((0,1),(0,1))\} \cup \{((1,0),(i,1)) \mid i,k \in Nat, k \geq i \geq 2\}$$

– первые $k+2$ элементов базиса B. Достаточно показать, что $(k+3)$-й элемент не может быть выражен через предыдущие, то есть

$$((1,0),(k+1,1)) \notin B_k^{AT}$$

Рассмотрим все нерефлексивные элементы отношения B_k. Они имеют вид $((1,0),(i,1))$, где $i \geq 2$. Первый коэффициент в правой части пары на $i-1$ больше, чем первый коэффициент в левой. Второй коэффициент в правой части пары на 1 больше, чем второй коэффициент в левой.

Рассмотрим рефлексивные пары $((1,0),(1,0))$ и $((0,1),(0,1))$. В них левые и правые части совпадают.

Если сложить любые две нерефлексивные пары, то в полученной паре разность между вторым коэффициентом в правой и левой части увеличится и станет равной двум. При любом транзитивном замыкании эта разность уменьшиться не сможет, поскольку в B_k во всех парах "правая" координата не меньше "левой". В то же время для построения пары $((1,0),(k+1,1))$ нам необходимо по крайней мере одно суммирование нерефлексивных пар – чтобы увеличить на единицу разность по первому коэффициенту. □

Итак, по отдельности симметричность и 1-рефлексивность не гарантируют существования конечного AT-базиса. Использование обоих ограничений[1] приводит к полностью конечному случаю.

Определение 1.13. *[6] Определим частичный порядок $\sqsubseteq$ на множестве $B \subseteq \mathcal{M}(X) \times \mathcal{M}(X)$ пар мультимножеств:*

[1] Заметим, что при рассмотрении конкретных классов отношений требование симметричности и 1-рефлексивности, как правило, возникает естественным образом и не является существенным ограничением. Например, максимальная бисимуляция разметок сети Петри симметрична и 1-рефлексивна по построению.

для рефлексивных пар (кроме $(\emptyset,\emptyset)$*)*

$$(r_1, r_1) \sqsubseteq (r_2, r_2) \overset{def}{\Leftrightarrow} r_1 \subseteq r_2 \ \wedge \ r_1 \neq \emptyset;$$

для нерефлексивных пар рефлексивная часть и нерефлексивный остаток сравниваются отдельно

$$(r_1 + o_1, r_1 + o_1') \sqsubseteq (r_2 + o_2, r_2 + o_2') \overset{def}{\Leftrightarrow}$$

$$\overset{def}{\Leftrightarrow} o_1 \cap o_1' = \emptyset \ \wedge \ o_2 \cap o_2' = \emptyset \ \wedge \ r_1 \subseteq r_2 \ \wedge \ o_1 \subseteq o_2 \ \wedge \ o_1' \subseteq o_2'.$$

Заметим, что рефлексивные и нерефлексивные пары отношения B не сравнимы по $\sqsubseteq$.

Через B_s обозначим множество минимальных (относительно $\sqsubseteq$) элементов B^{AT}. В случае, когда $(\emptyset,\emptyset) \in B$, к B_s также добавляется пара $(\emptyset,\emptyset)$.

Теорема 1.1. *[6] Пусть отношение B симметрично и 1-рефлексивно. Тогда отношение* B_s *является его базисом и* B_s *конечно.*

Базис B_s называется *основным базисом* отношения B.

Из теоремы 1.1 и второго утверждения леммы 1.2 вытекает:

Теорема 1.2. *[6] Если отношение B симметрично и 1-рефлексивно, то все его минимальные AT-базисы конечны.*

Это утверждение (хотя и в несколько другой формулировке и в терминах конгруэнтностей в коммутативных полугруппах) было доказано Л.Редеи [185], а позднее (независимо) Й.Хиршфельдом [135]. В нашей работе [6] было использовано новое, конструктивное доказательство, описывающее структуру элементов конечного базиса.

Приведем пример основного базиса.

Пример 1.6. *Рассмотрим множество X из одного элемента и симметричное 1-рефлексивное отношение*

$$B = \{(1,1),\ (1,2),\ (2,1),\ (1,8),\ (8,1),\ (3,3),\ (4,5),\ (5,4)\}$$

(пары мультимножеств представлены как пары чисел).

Основной базис — $B_s = \{(1,1),\ (1,2),\ (2,1)\}$.

Основной базис отношения не всегда является минимальным. Например, в базисе из примера 1.6 избыточной является пара $(1,1)$, которая может быть получена транзитивным замыканием двух других пар. Однако важным достоинством основного базиса является хорошая структурированность. К тому же он по определению единственнен для любого B, тогда как минимальных AT-базисов может существовать бесконечно много [6].

Другим важным достоинством основного базиса (кроме его конечности) является то, что при помощи процедуры разложения, использованной в доказательстве теоремы 1.1, мы можем за конечное число шагов проверить принадлежность произвольной пары ресурсов замыканию B^{AT}. В частности, в [6] представлен алгоритм трудоёмкости $O(|X||B_s|^2)$.

Отношение B может задаваться произвольным конечным базисом (не обязательно минимальным и не обязательно основным). Однако любой конечный симметричный 1-рефлексивный базис можно преобразовать в основной базис при помощи эффективной процедуры — в [6] представлен алгоритм такой трансформации, имеющий трудоёмкость $O(|X||B|^4)$. Однако необходимо заметить, что на практике мощность B часто находится в экспоненциальной зависимости от мощности основного множества X.

1.2. Эквивалентность поведений

Понятие эквивалентности поведений — важнейшее понятие теории формальных систем. Поведенческие эквивалентности позволяют сравнивать парал-

лельные и распределенные системы с учетом тех или иных аспектов их функционирования, а также абстрагироваться от излишней информации. Эквивалентностные отношения используются также для сохраняющей поведение редукции систем и в процессе верификации, когда сравнивается ожидаемое и реальное поведения систем. Проблемы проверки эквивалентности поведений занимают важное место в теории автоматов и теории схем программ (см., например, работы А. А. Летичевского, Р. И. Подловченко, В. А. Захарова и др. [39, 40, 42, 50–52, 181, 209]).

1.2.1. Системы помеченных переходов

Системы помеченных переходов — абстрактная низкоуровневая модель описания поведения дискретных систем.

Система переходов — это помеченный ориентированный граф, описывающий все возможные состояния моделируемой системы. При этом одна из вершин графа соответствует начальному состоянию системы, а дуги — возможным переходам системы из одного состояния в другое.

Определение 1.14. Системой помеченных переходов *(Labelled Transition System) называется набор* $LTS = (S, Act, \rightarrow, s_0)$*, где*

- S *— множество* состояний *с элементами* $s_0, s_1, s_2, \ldots$*;*
- Act *— некоторый алфавит (*множество имен действий*);*
- $\rightarrow \subseteq (S \times Act \times S)$ — отношение переходов *между состояниями (с пометками из Act);*
- $s_0 \in S$ *— выделенное состояние, называемое* начальным состоянием *системы переходов.*

Переход (s, a, s') из $\rightarrow$ обычно обозначается как $s \xrightarrow{a} s'$, что означает, что переход с меткой a переводит систему из состояния s в состояние s'. Состояние s'

в этом случае называется *последующим* для s, а состояние s — *предыдущим* для s'. Состояния, не имеющие последующих состояний называются *финальными*. Если некоторый переход переводит состояние s в состояние s', то пишем $s \to s'$. Через $Succ(s)$ будем обозначать множество последующих состояний для s, через $Pred(s)$ — множество его предыдущих состояний. Мы рассматриваем только системы переходов с *конечным ветвлением* (finitely branching), то есть такие, в которых для любого s множество $Succ(s)$ конечно.

Бесконечность (в общем случае) множества S позволяет использовать помеченные системы переходов для моделирования любых классов систем с конечным ветвлением (даже универсальных, таких, как машины Тьюринга). Фактически, LTS — это формализованный способ записи всех возможных вариантов функционирования системы. Простота записи и универсальность моделирования делает системы помеченных переходов базовым языком для представления различных поведенческих свойств.

Определение 1.15. Последовательное исполнение *для* LTS *есть конечная или бесконечная цепочка переходов* $s_0 \to s_1 \to s_2 \to \cdots$, *где* s_0 — *начальное состояние системы. Каждому исполнению* LTS *соответствует некоторая строка в алфавите Act, составленная из меток сработавших переходов, называемая* распознанной строкой *или* трассой LTS.

Запись $s \xrightarrow{*} s'$ означает, что имеется конечная последовательность переходов, переводящая состояние s в состояние s'.

Графически система переходов LTS изображается как помеченный ориентированный граф, в котором вершинами являются элементы множества состояний S, а дуги определяются отношением переходов так, что дуга, помеченная a, соединяет вершину s с вершиной s' в том и только том случае, когда $s \xrightarrow{a} s'$.

Каждому последовательному исполнению в LTS соответствует ориентированный путь с началом в вершине s_0 в графе LTS.

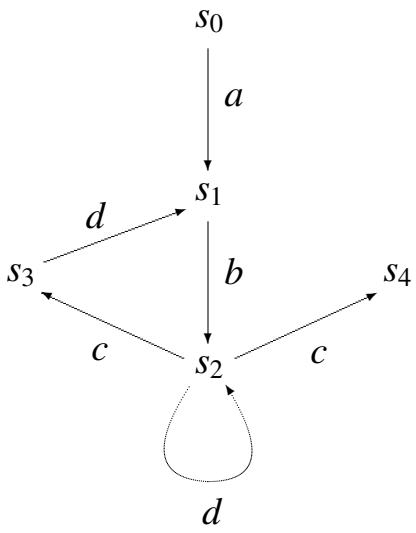


Рис. 1.2. Пример *LTS*

Одним из важнейших инструментов анализа динамики функционирования систем переходов является эквивалентность языков (трасс) [146].

Определение 1.16. Языком *системы помеченных переходов LTS (языком,* распознаваемым *системой LTS) называется множество строк из алфавита Act, соответствующих всем последовательным исполнениям для LTS*.

Языковая эквивалентность учитывает так называемую *интерливинговую* (последовательную) операционную семантику. Другими словами, анализируются "моментальные проекции" поведения системы, представляющие строки в некотором алфавите, без учета ветвлений.

Пример 1.7. *Рассмотрим системы переходов, изображенные на рисунке 1.3. Здесь моделируются автоматы, продающие кофе и чай. Множество Act соответствует действиям автомата, действия клиента выражаются в выборе возможных сценариев действия автомата (выборе пути в графе).*

В первом автомате клиент сначала опускает монету (действие "монета" — получение автоматом монеты), а затем выбирает либо кофе, либо чай — автомат отвечает действиями "кофе" или "чай" соответственно. Во втором автомате выбор кофе/чай происходит уже в самом начале, до опускания монеты. Далее от клиента уже ничего не зависит — автомат отрабатывает

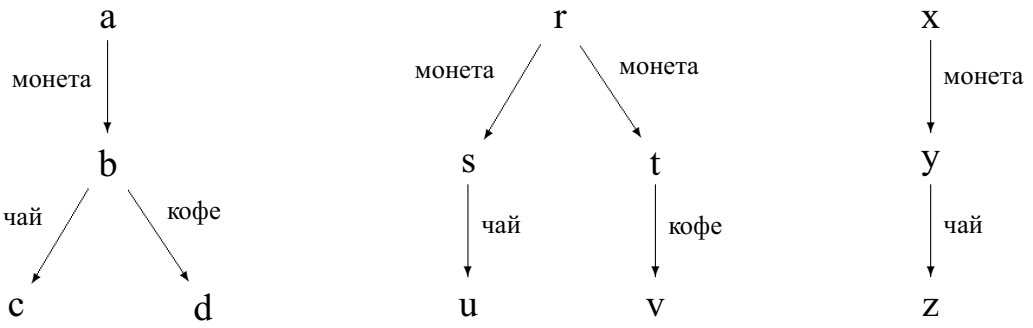


Рис. 1.3. Системы помеченных переходов

либо программу "монета кофе", либо "монета чай", не задавая дополнительных вопросов.

Языком и для первой и для второй системы является множество

{*монета, монета чай, монета кофе*},

то есть эти системы эквивалентны с точки зрения языковой эквивалентности. В то же время очевидно, что системы все-таки существенно различны. Если в первой после срабатывания "монета" можно выбрать и "чай", и "кофе", то во второй реальный выбор уже произошел, и изменить что-либо клиент уже не в состоянии.

Операционная семантика, учитывающая ветвления, называется семантикой *ветвящегося времени* (branching time). Если в последовательной семантике система полностью описывается распознаваемым языком (множеством трасс), то в семантике ветвящегося времени рассматривается полный граф срабатываний.

Для анализа динамики функционирования систем в семантике ветвящегося времени используется понятие бисимуляций.

1.2.2. Бисимуляции

Бисимуляционная эквивалентность [171, 176] — фундаментальное понятие в теории параллельных и распределенных систем. Бисимуляция обладает четкой математической трактовкой и более тонко отслеживает ветвления в дереве срабатываний системы по сравнению с языковой эквивалентностью. Два состояния

системы бисимулярны (симулируют друг друга), если внешний наблюдатель по наблюдаемому поведению системы не может определить, с какого из этих двух состояний она начала работу.

Отношение бисимуляции может быть использовано для определения эквивалентности различных моделей (например, для выявления соответствия системы её спецификации). Кроме того, выявление сходных структур в множестве состояний позволяет существенно упрощать систему без изменения ее наблюдаемого поведения (бисимуляционная редукция). Проблема поиска эквивалентных состояний также важна для поддержки методологии адаптивного управления, согласно которой структура системы может изменяться непосредственно в ходе ее функционирования, например, в ответ на изменения внешних условий или же при возникновении каких-то внутрисистемных событий (болезнь сотрудника, отказ оборудования, внедрение новых элементов системы и т.п.).

Бисимуляцию определяют с помощью так называемого свойства переноса:

Определение 1.17. *Пусть $R \subseteq S \times S$ — отношение на множестве состояний системы помеченных переходов. Отношение R обладает* свойством переноса*, если для любой пары $(s,t) \in R$ и любого перехода $s \xrightarrow{a} s'$ найдется имитирующий переход $t \xrightarrow{a} t'$, такой что $(s',t') \in R$.*

Свойство переноса можно проиллюстрировать диаграммой:

$$\begin{array}{ccc} s & R & t \\ a \downarrow & & \downarrow (\exists)\ a \\ s' & R & t' \end{array}$$

Определение 1.18. *Отношение $R \subseteq S \times S$ на множестве состояний системы помеченных переходов называется* отношением бисимуляции*, если R и R^{-1} обладают свойством переноса.*

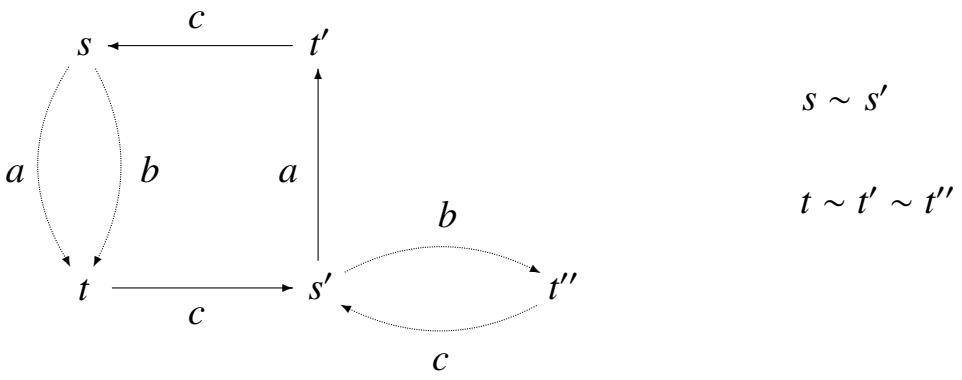


Рис. 1.4. Бисимулярные состояния в LTS.

Простейшими примерами бисимуляций являются отношение идентичности $Id(S)$ и пустое отношение.

Определение 1.19. *Состояния s и t системы помеченных переходов называются* бисимуляционно эквивалентными *(или* бисимулярными*), что обозначается как* $s \sim t$*, если существует отношение бисимуляции R, такое, что* $(s,t) \in R$.

Пример бисимулярных состояний приведен на рисунке 1.4. У изображенной системы помеченных переходов бисимулярны состояния s и s', а также t, t' и t''.

Кроме свойства переноса, существует ещё один классический способ определения бисимуляции — с использованием так называемых бисимуляционных игр.

В качестве “игровой доски” берется система помеченных переходов, в которой выделяются два состояния — E_0 и F_0 (в этом случае для игры используется обозначение $G(E_0, F_0)$). В игре участвуют два игрока, Алиса (подразумевается “Attacker”) и Боб (“Bisimulator”), которые являются наблюдателями, выбирающими переходы системы. Алиса стремится доказать, что состояния E_0 и F_0 в некотором смысле “различны”, Боб — что они в том же самом смысле “эквивалентны”. Результатом игры является конечная или бесконечная последовательность вида

$$(E_0, F_0), (E_1, F_1), \ldots, (E_k, F_k), \ldots, \text{ где } E_i, F_i \in S,$$

причем каждая следующая пара последовательности получена из предыдущей по правилу:

1. Алиса выбирает срабатывание (дугу в графе LTS) $E_i \xrightarrow{a} E_{i+1}$ или $F_i \xrightarrow{a} F_{i+1}$.
2. Боб выбирает некоторое имитирующее срабатывание $F_i \xrightarrow{a} F_{i+1}$ или $E_i \xrightarrow{a} E_{i+1}$ (обязательно с той же меткой a, что и у срабатывания, выбранного Алисой).

Алиса выигрывает игру, если на некотором ходе Боб не может ответить на ее ход своим (имитирующих переходов не осталось); у Боба два варианта выигрыша:

1. Алиса попала в тупик и не может сделать ни одного хода (из вершин E_i и F_i не ведет ни одной дуги).
2. Игра бесконечна, то есть последовательность ходов не закончилась ни победой Алисы, ни первым вариантом победы Боба.

Защищая свой тезис, Алиса должна выбирать срабатывания таким образом, чтобы у Боба не осталось возможности найти имитирующий переход. Боб, в свою очередь, должен отвечать на ходы Алисы так, чтобы требуемая эквивалентность состояний поддерживалась (или чтобы Алиса первой попала в тупик).

Стратегия игрока — это набор правил, которые позволяют определить следующий ход в зависимости от того, что произошло перед этим. Стратегия называется выигрывающей, если игрок побеждает в каждой игре при использовании этой стратегии (независимо от действий противника).

Определение 1.20. *Состояния E_0 и F_0 называются* бисимулярными*, если у Боба существует выигрывающая стратегия для игры $G(E_0, F_0)$.*

Пример 1.8. *Рассмотрим игру $G(s, s')$ в системе переходов, изображенной на рисунке 1.4. У Боба существует очевидная выигрывающая стратегия — отвечать на ходы Алисы ходами с той же меткой срабатывания. В состояниях s и s'*

возможны срабатывания с метками a и b (причем только по одному на каждую метку!), в состояниях t, t' и t'' — только единственное срабатывание с меткой c. Кроме того, как легко заметить, при любом последовательном исполнений состояния видов s^* и t^* чередуются.

Утверждение 1.2. *Определения бисимулярности 1.19 и 1.20 эквивалентны.*

Основные свойства бисимуляции:

Утверждение 1.3. *Пусть $R, R_1, R_2 \subseteq S \times S$ — отношения бисимуляции на множестве состояний системы помеченных переходов. Тогда:*

1. *Отношение $Id(S)$ есть бисимуляция.*
2. *Отношение R^{-1} есть бисимуляция.*
3. *Отношение $R_1 \cup R_2$ есть бисимуляция.*
4. *Отношение $\widetilde{R}$ есть бисимуляция.*
5. *Отношение $R_2 \circ R_1$ есть бисимуляция.*
6. *Отношение $\sim =_{def} \bigcup\{R \mid R \subseteq S \times S$ — бисимуляция $\}$ есть наибольшая бисимуляция на множестве S (относительно вложения).*
7. *Отношение $\sim$ является отношением эквивалентности на множестве S.*

Доказательство: Свойства 1, 2 и 3 непосредственно следуют из определения бисимуляции.

Свойство 5: Покажем, что $R_2 \circ R_1$ обладает свойством переноса, и, следовательно, является бисимуляцией.

Пусть $(s_1, s_3) \in R_2 \circ R_1$, тогда существует состояние s_2 такое, что выполняется $(s_1, s_2) \in R_1$ и $(s_2, s_3) \in R_2$.

Рассмотрим шаг $s_1 \xrightarrow{a} s_1'$. Тогда существует шаг $s_2 \xrightarrow{a} s_2'$, такой что $(s_1', s_2') \in R_1$, и, следовательно, существует шаг $s_2 \xrightarrow{a} s_2'$, такой что $(s_1', s_2') \in R_2$, то есть $(s_1', s_3') \in R_2 \circ R_1$. Аналогично можно показать, что отношение $R_2 \circ R_1$ обладает свойством переноса в обратном направлении. Таким образом, $R_2 \circ R_1$ — бисимуляция.

Свойство 4: Поскольку свойства 2 и 3 могут быть обобщены для объединения (композиции) бесконечного числа множеств, а

$$\widetilde{R} = (R \cup R^{-1})^* = Id(S) \cup (R \cup R^{-1}) \cup (R \cup R^{-1})^2 \cup \cdots \cup (R \cup R^{-1})^k \cup \ldots,$$

то очевидно, что $\widetilde{R}$ — бисимуляция.

Свойство 6 является очевидным следствием свойства 3, а свойство 7 — следствием свойств 3 и 4. □

Бисимуляция — достаточно тонкая эквивалентность на множестве состояний, адекватно отражающая свойства системы в семантике ветвящегося времени. Однако, в силу своей универсальности, для многих классов систем отношение бисимуляции неразрешимо, то есть не существует алгоритма, отвечающего на вопрос, являются ли данные два состояния бисимулярными или нет.

Бисимуляционная эквивалентность изучалась для различных классов формальных моделей [56, 63, 105, 136, 139–142, 145, 173, 194, 196]. Был получен ряд результатов по ее разрешимости. В частности, бисимуляция разрешима для всех классов систем с конечным числом состояний (конечных автоматов), так как в них для проверки бисимулярности достаточно просто перебрать множество состояний. Бисимуляция разрешима также для таких классов моделей с бесконечным множеством состояний, как:

- базовые параллельные процессы (BPP, Basic Parallel Processes) [105, 106, 137],

- базовые алгебры процессов (BPA, Basic Process Algebra) [63, 107],

- нормированные алгебры процессов (normed PA, normed Process Algebra) [136],

- автоматы с одним счетчиком (one-counter machines) [141],

- нормированные магазинные автоматы (normed PDA, normed Pushdown Automata) [194].

Бисимуляция неразрешима для следующих классов моделей (упорядочены по возрастанию выразительной мощности и в обратном хронологическом порядке по времени доказательства неразрешимости бисимуляции):

- автоматы мультимножеств (MSA, Multiset Automata) [173],

- помеченные сети Петри (labelled PN, labelled Petri Nets) [140],

- универсальные модели (машины Минского, машины Тьюринга, сети Петри с ингибиторными дугами, . . .).

Важным примером расширения бисимуляции является так называемая расслоенная (stratified) бисимуляция [145] (обозначается $\sim_n$). Она определяется индуктивно:

Во-первых, полагаем $s_1 \sim_0 s_2$ для любых $s_1, s_2 \in S$. Далее, для любого $n \in Nat$ полагаем $s_1 \sim_{n+1} s_2$, если для любого $a \in Act$:

- если $s_1 \xrightarrow{a} s_1'$, то $s_2 \xrightarrow{a} s_2'$, причем $s_1' \sim_n s_2'$; и
- если $s_2 \xrightarrow{a} s_2'$, то $s_1 \xrightarrow{a} s_1'$, причем $s_1' \sim_n s_2'$.

Другими словами, n-расслоенная бисимуляция — это бисимуляция в том случае, когда нам не важно, что будет происходить с системой после n-го шага. Все последующее поведение сети просто игнорируется.

Известно, что для любого n отношение $\sim_n$ является эквивалентностью, кроме того, $(\sim_{n+1}) \subseteq (\sim_n)$. Также выполняется $s_1 \sim s_2 \Leftrightarrow s_1 \sim_n s_2$ для любого $n \in Nat$. Наибольшая бисимуляция является пределом последовательности расслоенных бисимуляций: $(\sim) = (\sim_\infty)$.

Известно [145], что проблема n-расслоенной бисимулярности разрешима для любого n.

1.3. Сети Петри

Определение 1.21. Обыкновенной сетью Петри *(ordinary Petri Net) называется набор* $N = (P, T, F)$*, где*

P *— конечное множество позиций;*

T *— конечное множество переходов,* $P \cap T = \emptyset$*;*

$F : (P \times T) \cup (T \times P) \to Nat$ *— функция инцидентности.*

Графически сеть Петри изображается как двудольный ориентированный граф. Вершины-позиции изображаются кружками и характеризуют локальные состояния сети, вершины-переходы изображаются прямоугольниками и соответствуют действиям моделируемой системы. Дуги в графе соответствуют элементам F.

Определение 1.22. *Пусть* $N = (P, T, F)$ *— обыкновенная сеть Петри.* 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.

Графически разметка изображается при помощи маркеров (называемых “фишками”) — черных точек внутри позиций. При разметке M в каждую позицию p помещается ровно $M(p)$ фишек. Если не хватает места на рисунке, то вместо точек рисуется число $M(p)$.

Определение 1.23. Маркированной (размеченной) сетью Петри *называется пара* (N, M_0) *— сеть Петри N вместе с некоторой выделенной разметкой* M_0*, называемой начальной разметкой.*

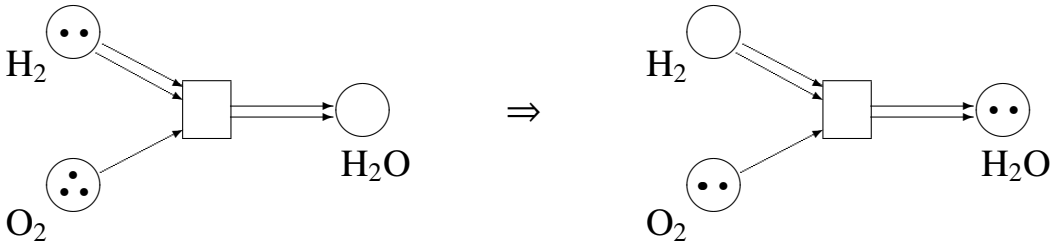


Рис. 1.5. Сеть Петри, моделирующая химическую реакцию

Определим поведение сети Петри.

Определение 1.24. *Пусть $N = (P, T, F)$ — обыкновенная сеть Петри.*

- *Для перехода $t \in T$ через ${}^\bullet t$ и $t^\bullet$ обозначим мультимножества его входных и выходных позиций, такие, что*

$$\forall p \in P \quad {}^\bullet t(p) =_{def} F(p,t),\ t^\bullet(p) =_{def} F(t,p).$$

- *Переход $t \in T$* готов к срабатыванию *при разметке M, если ${}^\bullet t \subseteq M$ (все входные позиции содержат достаточное количество фишек).*

- *Готовый к срабатыванию переход t может* сработать*, порождая новую разметку $M' =_{def} M - {}^\bullet t + t^\bullet$ (используется обозначение $M \xrightarrow{t} M'$).*

Фишки, находящиеся в той или иной позиции, моделируют наличие в системе того или иного ресурса, используемого или порождаемого при срабатывании переходов. Например, в сети на рисунке 1.5 фишки изображают молекулы водорода, кислорода и воды до и после химической реакции синтеза воды. Сама реакция моделируется переходом.

Определение 1.25. *Пусть (N, M_0) — маркированная сеть Петри. Разметка M сети N называется* достижимой*, если существует последовательность переходов $\sigma \in T^*$, переводящая сеть из начального состояния M_0 в состояние M:*

$$M_0 \xrightarrow{t_1} M_1 \xrightarrow{t_2} \ldots \xrightarrow{t_k} M,\ t_1.t_2.\ldots.t_k = \sigma,$$

что обозначается как $M_0 \xrightarrow{\sigma} M$ или просто как $M_0 \to M$.

Множество всех достижимых разметок сети обозначается как $\mathcal{R}(N, M_0)$

Определение 1.26. *Пусть* (N, M_0) *— маркированная сеть Петри. Позиция* $p \in P$ *называется* ограниченной, *если* $\exists n \in Nat$, *такое, что* $\forall M \in \mathcal{R}(N, M_0)$ *выполняется* $M(p) \leq n$.

Маркированная сеть Петри называется ограниченной, *если все её позиции ограничены. Очевидно, что множество состояний ограниченной сети Петри конечно.*

На рисунках 1.6–1.8 приведены примеры того, как при помощи обыкновенных сетей Петри можно моделировать некоторые элементы реальных систем.

На рисунке 1.6 изображен буфер ограниченной емкости. Независимо от поведения сети, количество фишек в позиции *буфер* не превысит двух. При этом количество фишек в служебной позиции *свободно* показывает, сколько еще “места” осталось в буфере. Предложенная структура сети универсальна — мы можем изменять моделируемую емкость буфера, просто изменяя количество фишек в начальной разметке позиции *свободно*. В начальном состоянии буфер пуст, на входе имеются три фишки “данных”.

Сеть 1.7 моделирует систему разделения доступа для двух различных процессов к общей ячейке памяти [177]. Чтобы исключить одновременный доступ процессов к памяти (чтение и запись в данном случае не различаются), использовано классическое семафорное решение. Имеется один общий ключ (моделируется фишкой в позиции *ключ*), который дает право на доступ. Пока работающий процесс не вернул его на место, ожидающий процесс не может перейти в рабочее состояние. В начальном состоянии оба процесса находятся в стадии ожидания доступа.

На рисунке 1.8 показан элемент памяти FIFO (очередь), состоящий из двух ячеек (позиции 1 и 2). Позиция 1 моделирует первую ячейку FIFO (куда поступают данные), позиция 2 — последнюю (откуда они забираются). В начальном состоянии системы обе ячейки свободны, на входе имеются три фишки “данных”.

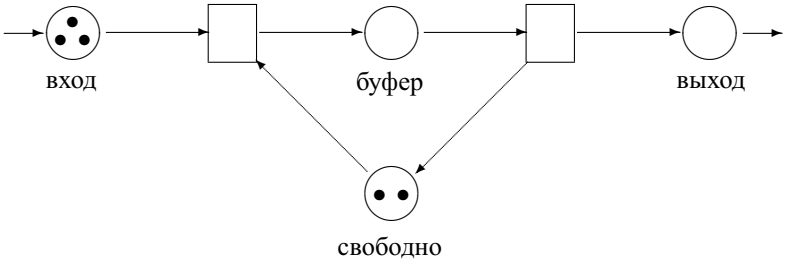


Рис. 1.6. Буфер объема 2

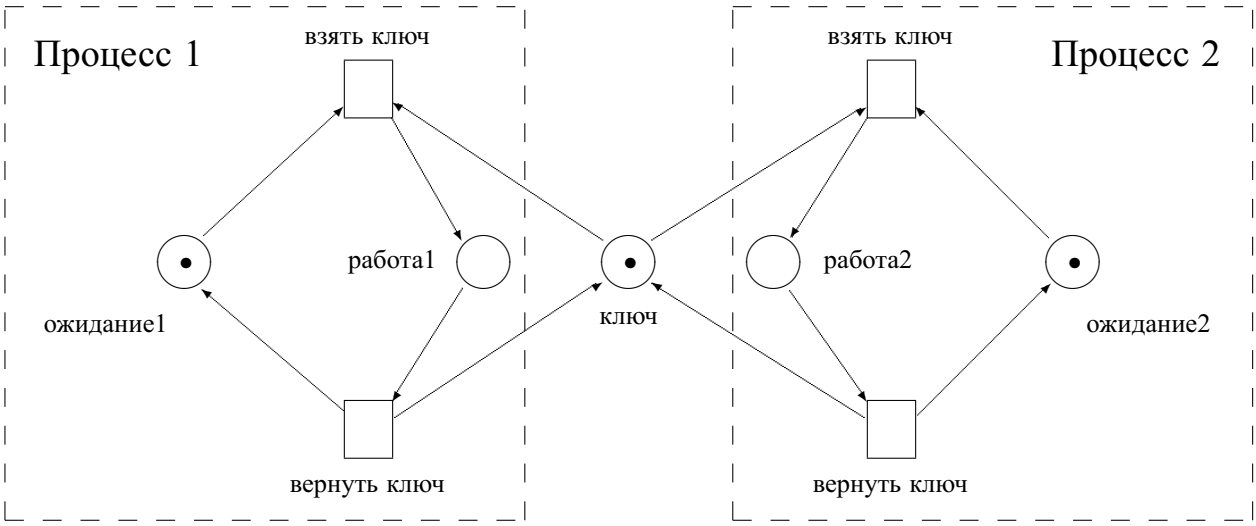


Рис. 1.7. Семафор (разделенный доступ к памяти)

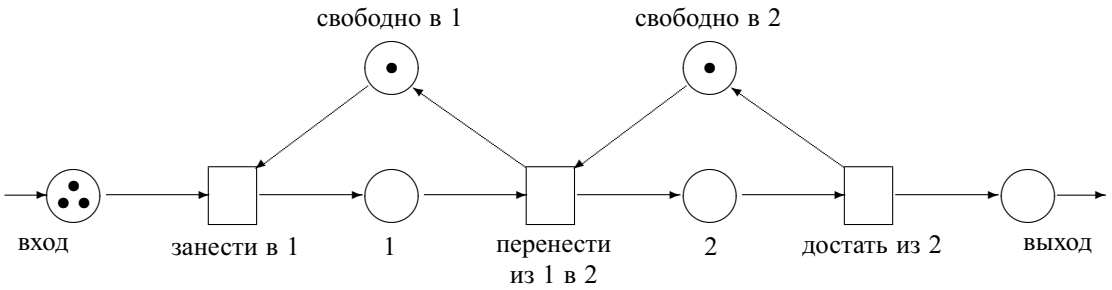


Рис. 1.8. Очередь (FIFO) из двух ячеек

Для анализа поведения систем, моделируемых сетями Петри, необходимо формально сопоставить реальные объекты (действия) элементам модели. Срабатывания переходов в сети Петри соответствуют различным наблюдаемым событиям в моделируемой системе. Чтобы идентифицировать их, переходы помечаются метками из Act.

Определение 1.27. Помеченной сетью Петри *называется набор* $N = (P, T, F, l)$, *где* (P, T, F) *— сеть Петри,* $l : T \to Act$ *— помечающая функция.*

Внешний наблюдатель видит не сам переход t, а метку срабатывания $l(t)$, которой он помечен. Если два перехода помечены одной и той же меткой, то их срабатывания считаются идентичными. Таким образом, внешний наблюдатель рассматривает систему как “черный ящих”, порождающий те или иные события. При этом внутренняя структура состояний для него недоступна.

Отношения бисимуляции для обыкновенных сетей Петри вводится следующим образом. Так как состоянием сети Петри является её разметка, бисимуляция определяется на множестве разметок сети, то есть на множестве всех мультимножеств над множеством позиций.

Определение 1.28. *Пусть* $N = (P, T, F, l)$ *— помеченная сеть Петри. Скажем, что отношение* $R \subseteq \mathcal{M}(P) \times \mathcal{M}(P)$ *обладает* свойством переноса*, если для любой пары разметок* $(M_1, M_2) \in R$ *и для любого перехода* $t \in T$*, такого что* $M_1 \xrightarrow{t} M_1'$*, найдется имитирующий переход* $u \in T$*, такой что* $l(t) = l(u)$*,* $M_2 \xrightarrow{u} M_2'$ *и* $(M_1', M_2') \in R$.

Свойство переноса может быть проиллюстрировано при помощи следующей диаграммы:

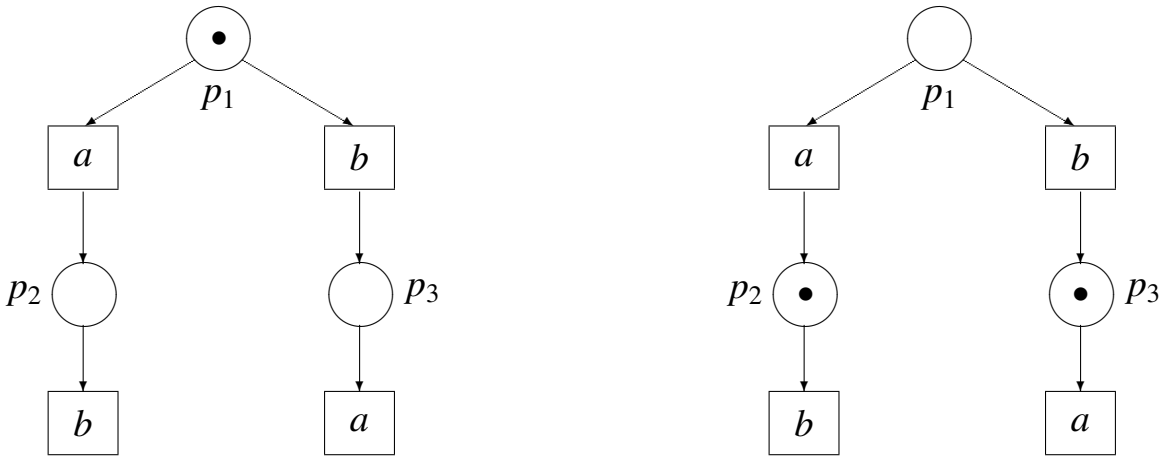


Рис. 1.9. Бисимулярные разметки в сети Петри

$$\begin{array}{ccc} M_1 & \sim & M_2 \\ \downarrow t & & \downarrow (\exists) u,\ l(u) = l(t) \\ M_1' & \sim & M_2' \end{array}$$

Определение 1.29. *Если отношения R и R^{-1} обладают свойством переноса, то R называется* бисимуляцией разметок.

На рисунке 1.9 приведен пример бисимулярных разметок в сети Петри. Выполняется $(1, 0, 0) \sim (0, 1, 1)$ (разметки записаны как вектора длины 3).

Бисимуляции разметок в сетях Петри по определению обладают всеми свойствами общего определения бисимуляции для LTS (так как сети Петри есть частный случай LTS).

Известно, что для некоторых важных подклассов сетей Петри бисимуляция разрешима:

1. Ограниченные сети Петри (так как они совпадают с классом конечных автоматов).

2. Сети Петри с одной неограниченной позицией.

3. Сети Петри с пометками один-к-одному (сети, в которых для каждой метки из *Act* существует не более одного помеченного ею перехода).

4. Бисимулярно-детерминированные сети Петри (в бисимулярно-детерминированных сетях разные переходы, имеющие одну и ту же метку, могут быть возбуждены одновременно, только если их срабатывания приводят к бисимулярным разметкам).

В 1994 году П.Жанкар доказал [140, 145], что в общем случае бисимуляция разметок в сетях Петри неразрешима. Более того, бисимуляция неразрешима даже для сетей с двумя неограниченными позициями. Это весьма существенное ограничение, не позволяющее использовать бисимуляцию разметок при изучении свойств достаточно широких классов систем с бесконечным числом состояний. Необходимо искать более сильные эквивалентности, лучше поддающиеся алгоритмическому анализу.

Примером таких отношений могут служить введенные Ф. Шнобеленом, С. Аутоном и Н. С. Сидоровой отношения бисимуляции позиций и корректного слияния позиций [53, 61, 62, 190]. В диссертации [6] нами было введено и исследовано промежуточное по силе между бисимуляцией разметок и слиянием позиций отношение подобия ресурсов.

1.4. Ресурсы в сетях Петри

В данном разделе представлено ключевое понятие ресурса сети Петри. Приводятся некоторые результаты из [6], касающиеся свойств отношений эквивалентности ресурсов, в частности, подобия ресурсов и бисимуляции ресурсов.

Два ресурса подобны, если, заменив в любой разметке один из них на другой, мы получим то же самое наблюдаемое поведение. Подобие ресурсов обладает естественной интерпретацией и позволяет выразить ряд важных свойств системы, в частности, эквивалентность двух различных ресурсов, избыточность

ресурса, эквивалентность двух различных действий при условии наличия дополнительного ресурса и т.п. Нахождение подобных ресурсов полезно для понимания характера моделируемого процесса и оптимизации ресурсных затрат.

Отношение подобия ресурсов сильнее отношения бисимуляции разметок, поэтому для него выполняется ряд конструктивных свойств. В общем случае подобие неразрешимо, однако обладает конечным базисом и может быть аппроксимировано.

1.4.1. Подобие ресурсов

Определение 1.30. *Пусть $N = (P, T, F)$ — обыкновенная сеть Петри.* Ресурсом *сети N называется мультимножество над множеством позиций P.*

Формально определение ресурса не отличается от определения разметки. Каждая разметка является ресурсом и каждый ресурс является разметкой. Мы различаем эти понятия из-за их различной интерпретации. Ресурсы являются частями разметок (в некотором смысле неделимыми), обеспечивающими то или иное поведение сети при любом её состоянии. Например, в сети на рисунке 1.5 две молекулы водорода и одна молекула кислорода составляют ресурс, достаточный для срабатывания перехода (т.е. для производства двух молекул воды).

В данном определении ресурс рассматривается не как “подмножество данной разметки”, а как “общая часть всех разметок, содержащих данное (мульти)множество фишек”. Суть различий наиболее ярко проявляется в определении подобных ресурсов:

Определение 1.31. *Пусть $N = (P, T, F, l)$ — помеченная сеть Петри. Ресурсы r и s называются* подобными *(обозначается $r \approx s$), если для любой разметки R, такой, что $r \subseteq R$, выполняется*

$$R \sim R - r + s.$$

Отношением подобия ресурсов *сети Петри N называется максимальное по вложению отношение на множестве ресурсов* $\mathcal{M}(P)$, *такое, что любые два связанных им ресурса подобны (обозначается* $\approx$*).*

Если ресурсы подобны, то в любой разметке мы можем заменить один из них на другой, и при этом дальнейшее наблюдаемое поведение системы не изменится (в смысле бисимулярности). Таким образом, подобие ресурсов отслеживает все ресурсы в сети Петри, обладающие идентичными свойствами при любом состоянии системы в целом.

Заметим, что для сети Петри, представляющей реакцию синтеза воды (рис. 1.5), одна молекула кислорода эквивалентна одной молекуле водорода с точки зрения бисимулярности разметок, так как и в том и в другом случае переход сети не сработает. Однако подобными ресурсами указанные две молекулы не являются.

Примером подобных ресурсов могут служить два набора монет — одна монета по десять копеек и две монеты по пять копеек, при условии кратности цен на интересующие нас товары десяти копейкам. В данном случае различный номинал монет соответствует различным позициям сети Петри, количество монет — разметке соответствующей позиции, покупка товара — срабатыванию перехода, а цена на товар — кратности исходящих из позиций дуг.

Рассмотрим модель подобной ситуации, изображенную на рисунке 1.10. Покупается товар стоимостью 20 копеек. Позиция “магазин” моделирует количество товара в магазине, “куплено” — количество уже купленного товара, “10к” и “5к” — количество неизрасходованных монет. Покупка возможна тремя способами — переходы t_1, t_2 и t_3. Во всех трех случаях тратится различное число различных монет.

В такой ситуации использование пятикопеечных монет избыточно, что и отражается в подобии ресурсов “10к”≈“5к”+“5к”.

Примеры подобных ресурсов приведены на рисунке 1.11.

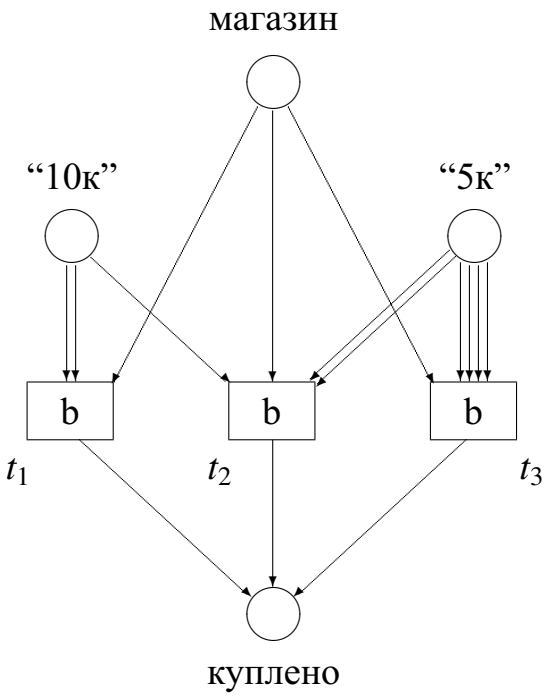
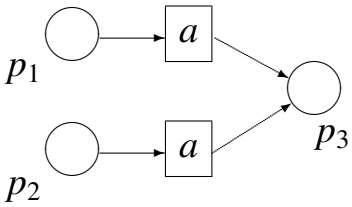


Рис. 1.10. Модель покупки товара с использованием двух видов монет

На рисунке а) изображена сеть Петри, содержащая два перехода, помеченных одинаковой меткой a и приводящих к одинаковой разметке p_3. Здесь подобны ресурсы p_1 и p_2, так как они приводят к полностью идентичному поведению сети — срабатыванию перехода с меткой a с помещением фишки в позицию p_3. Более того, подобны все ресурсы, содержащие одинаковое число фишек в позициях p_1 и p_2.

На рисунке б) изображена простейшая сеть, состоящая из одного перехода. В данном случае ресурс p_2 подобен пустому ресурсу, так как он никак не влияет на поведение сети (сколько бы фишек мы ни поместили в позицию p_2, переход сработать все равно не сможет).

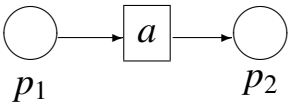
На рисунке в) изображен цикл, состоящий из одного перехода и одной позиции. Заметим, что множество разметок данной сети можно разделить на два непересекающихся подмножества — пустую разметку и все прочие. При пустой разметке переход сработать не может, при всех прочих — может сработать любое количество раз (более того, все прочие разметки бисимулярны между собой). Это свойство сети полностью отслеживается и подобием ресурсов. Вообще говоря, у этой сети максимальная бисимуляция разметок и подобие ресурсов совпадают.



$p_1 \approx p_2$

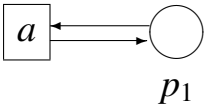
$kp_1 \approx kp_2$,
где $k > 0$

а) параллельные срабатывания



$p_2 \approx \emptyset$

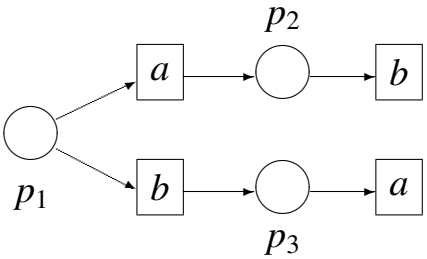
б) тупик



$p_1 \approx 2p_1$

$np_1 \approx mp_1$,
где $n, m > 0$

в) циклические срабатывания



$p_1 \approx p_2 + p_3$

г) ресурсы различной мощности

Рис. 1.11. Подобные ресурсы

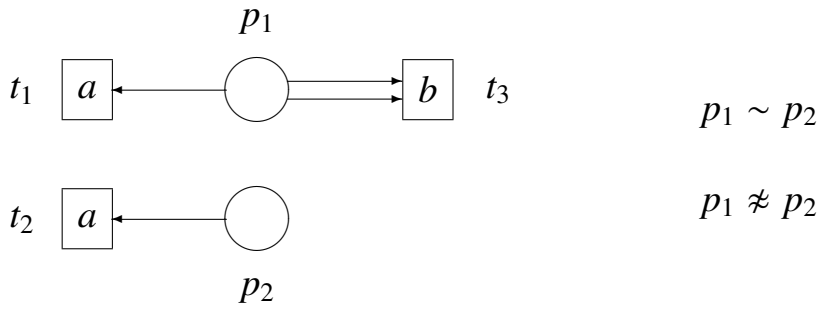


Рис. 1.12. Подобие ресурсов не совпадает с бисимуляцией разметок

На рисунке г) изображена более сложная сеть. Выполняется $p_1 \approx p_2 + p_3$, то есть замена одной фишки в позиции p_1 на две фишки — одну в позиции p_2 и еще одну в позиции p_2 никак не влияет на наблюдаемое поведение сети в целом.

1.4.2. Свойства подобия ресурсов

Утверждение 1.4. *[6] Пусть $N = (P, T, F, l)$ — помеченная сеть Петри, r, s — ресурсы сети N (которые также могут рассматриваться как разметки сети N). Тогда*

$$r \approx s \quad \Rightarrow \quad r \sim s.$$

Другими словами, $(\approx) \subseteq (\sim)$, то есть подобие ресурсов является сужением максимальной бисимуляции разметок.

Обратное утверждение ($r \sim s \quad \Rightarrow \quad r \approx s$) в общем случае неверно. В качестве контрпримера рассмотрим сеть, изображенную на рисунке 1.12.

В данной сети $p_1 \sim p_2$, так как при этих разметках может сработать только переход, помеченный a. Но $p_1 \not\approx p_2$, так как $2p_1 \not\sim 2p_2$ (при разметке $2p_1$ может сработать переход t_3, помеченный b).

Подобие ресурсов гораздо сильнее, чем бисимуляция разметок, так как оно не учитывает конкретной (начальной) разметки сети Петри, а выявляет закономерности, общие для ВСЕХ возможных разметок. Это снижает его выразительность по сравнению с бисимуляцией, однако добавляет полезное свойство аддитивной замкнутости, которое может быть использовано для построения конечного базиса.

Подобие ресурсов рефлексивно, симметрично, транзитивно и замкнуто относительно сложения:

Лемма 1.3. *[6] Пусть $N = (P, T, F, l)$ — помеченная сеть Петри, r, s, u, v — ресурсы сети N. Тогда*

1. $r \approx r$;
2. $r \approx s \;\Rightarrow\; s \approx r$;
3. $r \approx s \;\wedge\; s \approx u \;\Rightarrow\; r \approx u$;
4. $r \approx s \;\wedge\; u \approx v \;\Rightarrow\; r + u \approx s + v$.

Таким образом, подобие ресурсов обладает всеми свойствами AT-замыкания некоторого отношения (другими словами, отношения $\approx$ и $(\approx)^{AT}$ совпадают).

Например, из подобия ресурсов $r \approx s$ немедленно следует подобие ресурсов $2r \approx 2s,\ 2r+s \approx r+2s,\ 5r \approx 3r+2s$ и так далее, что может быть описано формулой

$$i_r r + i_s s \approx j_r r + j_s s, \text{ где } i_r, i_s, j_r, j_s \in Nat,\ i_r + i_s = j_r + j_s.$$

Из подобия ресурсов $r \approx 2r$ немедленно следует подобие ресурсов:

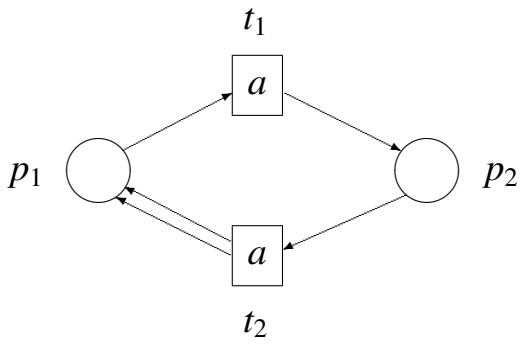
$$ir \approx jr, \text{ где } i, j \in Nat,\ i > 0,\ j > 0.$$

То есть любая пара подобных ресурсов может рассматриваться в качестве базиса для некоторого бесконечного отношения. С другой стороны, всё отношение подобия ресурсов также обладает конечным базисом:

Теорема 1.3. *[6] Подобие ресурсов в помеченных сетях Петри обладает конечным AT-базисом.*

Наличие конечного базиса — ключевое отличие подобия ресурсов от бисимуляции разметок. Поскольку подобие ресурсов кроме транзитивности еще и замкнуто относительно сложения, в любой сети оно имеет конечный базис.

В то же время в некоторых сетях бисимуляция разметок также может обладать конечным базисом:



$ip_1 + jp_2 \approx kp_1 + lp_2$, где
$i, j, k, l \in Nat,\ i + j > 0,\ k + l > 0.$

Рис. 1.13. Пример сети Петри, в которой подобие ресурсов и бисимуляция разметок совпадают

Пример 1.9. *На рисунке 1.13 изображена сеть Петри, представляющая собой цикл, состоящий из двух позиций и двух переходов, помеченных одним и тем же символом a.*

Поскольку один из переходов (p_1) при срабатывании сохраняет общее количество фишек в разметке сети, а другой (p_2) — увеличивает это количество на единицу, при любой непустой начальной разметке обе позиции — неограниченные, и переходы могут сработать сколь угодно много раз.

Таким образом, любые две непустые разметки — бисимулярны, то есть для данной сети мы можем построить следующую бисимуляцию разметок:

$$(\sim) = \{\emptyset, \emptyset\} \cup \{(ip_1 + jp_2, kp_1 + lp_2) \mid i, j, k, l \in Nat, i + j > 0, k + l > 0\}.$$

Это же отношение является подобием ресурсов данной сети:

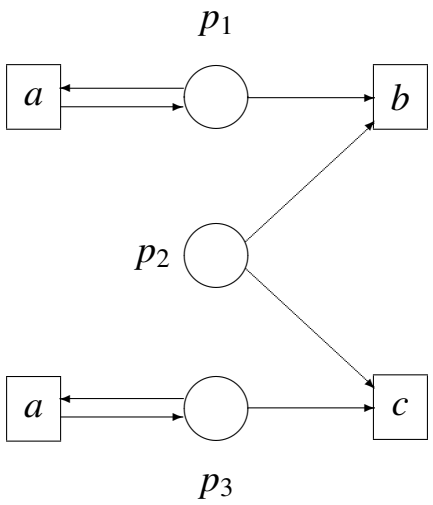
$$(\approx) = (\sim).$$

Подобие ресурсов задается следующим основным базисом:

$$(\approx)_s = \{(\emptyset, \emptyset),\ (p_1, p_1),\ (p_2, p_2),\ (p_1, p_2),\ (p_2, p_1),$$

$$(p_1, 2p_1),\ (2p_1, p_1),\ (p_2, 2p_2),\ (2p_2, p_2)\}.$$

Имеющее место в рассмотренном случае совпадение подобия ресурсов и бисимуляции разметок является достаточно нехарактерной ситуацией, поскольку, как правило, бисимуляция содержит гораздо больше пар, чем подобие ресурсов, и не замкнута относительно сложения.



$kp_1 \sim lp_3,\ k,l > 0$

$kp_1 \not\approx lp_3$, так как

$kp_1 + p_2 \not\sim lp_3 + p_2$

Рис. 1.14. Пример сети Петри, в которой существует АТ-замкнутое сужение бисимуляции разметок, не входящее в подобие ресурсов

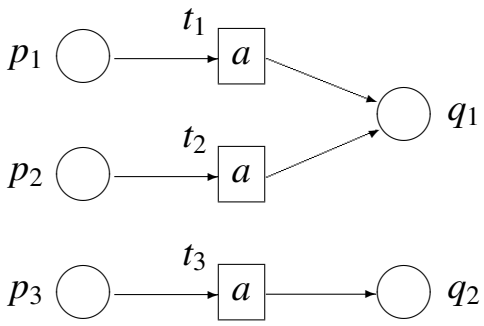
Заметим также, что подобие ресурсов не является наибольшим по вложению АТ-замкнутым сужением бисимуляции разметок:

Пример 1.10. *На рисунке 1.14 изображена сеть Петри, состоящая из трех позиций и четырех переходов. Бисимуляция разметок этой сети содержит пары вида $kp_1 \sim lp_3$, где $k,l > 0$. Действительно, при любой из разметок вида kp_1 или lp_3 могут сработать только переходы, помеченные "a" (сколь угодно много раз). Множество всех таких пар замкнуто относительно сложения и относительно транзитивности. С другой стороны, очевидно, что ни одна из них не принадлежит подобию ресурсов, так как $kp_1 + p_2 \not\sim lp_3 + p_2$.*

Бисимуляция разметок в каком-то смысле существенно более выразительна, чем подобие ресурсов — в ней могут существовать и другие обладающие конечными базисами подмножества. Однако подобие ресурсов всё же не настолько просто устроено, чтобы быть разрешимым:

Теорема 1.4. *[6] Проблема распознавания подобия ресурсов в сети Петри неразрешима.*

Таким образом, даже несмотря на существование конечного базиса, подобие ресурсов не может быть эффективно построено.



$$p_1 \approx p_2 \approx p_3$$

$$q_1 \approx q_2$$

$$B_1 = Id(P) \cup \{(p_1, p_3), (p_3, p_1)\}$$

$$B_2 = B_1 \cup \{(q_1, q_2), (q_2, q_1)\}$$

Рис. 1.15. Отношение B_1 — не бисимуляция ресурсов, отношение B_2 — бисимуляция ресурсов

1.4.3. Бисимуляция ресурсов

В работах Ф. Шнобелена и др. ([61, 62]) при определении вычислимых сохраняющих бисимулярность отношений на множестве позиций сети Петри был использован способ замыкания отношения относительно срабатываний переходов. Отношение на множестве разметок замкнуто относительно срабатываний переходов, если оно является бисимуляцией разметок.

В диссертации [6] аналогичный подход был применён для случая ресурсов.

Определение 1.32. *Пусть $N = (P, T, F, l)$ — помеченная сеть Петри. Симметричное 1-рефлексивное отношение $B \subseteq \mathcal{M}(P) \times \mathcal{M}(P)$ называется* бисимуляцией ресурсов *сети N, если B^{AT} есть бисимуляция разметок сети N.*

Теорема 1.5. *[6] Пусть $N = (P, T, F, l)$ — помеченная сеть Петри, $B \subseteq \mathcal{M}(P) \times \mathcal{M}(P)$ — бисимуляция ресурсов сети N и $(r, s) \in B$. Тогда $r \approx s$.*

Таким образом, всякая бисимуляция ресурсов есть сужение подобия ресурсов. Однако не всякое сужение подобия ресурсов есть бисимуляция ресурсов. Необходимо также, чтобы это отношение было замкнуто относительно срабатываний переходов, то есть было еще и бисимуляцией разметок.

Пример 1.11. *Рассмотрим сеть, изображенную на рисунке 1.15. Отношение B_1 является подмножеством подобия ресурсов, однако оно не замкнуто относительно срабатывания переходов.*

Рассмотрим пару разметок $(p_1, p_3) \in (B_1)^{AT}$. *Срабатывание* $p_1 \xrightarrow{t_1} q_1$ *может быть имитировано при разметке* p_3 *только срабатыванием перехода* $t_3 : \ p_3 \xrightarrow{t_3} q_2$. *Однако пара* (q_1, q_2) *не принадлежит отношению* $(B_1)^{AT}$. *Следовательно,* $(B_1)^{AT}$ *не является бисимуляцией разметок, и* B_1 *не является бисимуляцией ресурсов.*

Дополним отношение B_1 *парами* (q_1, q_2) *и* (q_2, q_1). *Полученное отношение* B_2 *является отношением бисимуляции ресурсов, так как* $(B_2)^{AT}$ *является биимуляцией разметок.*

Заметим, что отношение B_2 *всё ещё не содержит всех пар подобных ресурсов (например, оно не содержит пару* $p_1 \approx p_2$*).*

Итак, *бисимуляция ресурсов — это множество пар подобных ресурсов, являющееся к тому же бисимуляцией разметок*.

Следующая теорема описывает основные свойства бисимуляции ресурсов.

Теорема 1.6. *[6] Пусть* $N = (P, T, F, l)$ *— помеченная сеть Петри. Тогда*

1. *если* $B_1, B_2 \subseteq \mathcal{M}(P) \times \mathcal{M}(P)$ *— бисимуляции ресурсов, то* $B_1 \cup B_2$ *— бисимуляция ресурсов;*

2. *для сети N существует максимальная по вложению бисимуляция ресурсов (обозначается как* $\simeq$*);*

3. $(\simeq)$ *является отношением эквивалентности.*

Очевидно, что максимальная бисимуляция ресурсов $(\simeq)$ бесконечна и содержится в максимальной бисимуляции разметок. Однако в силу теоремы 1.1 даже максимальная бисимуляция ресурсов может быть представлена конечным числом пар, то есть конечной бисимуляцией ресурсов.

Взаимная вложенность максимальной бисимуляции разметок, подобия ресурсов и максимальной бисимуляции ресурсов изображена на рисунке 1.16.

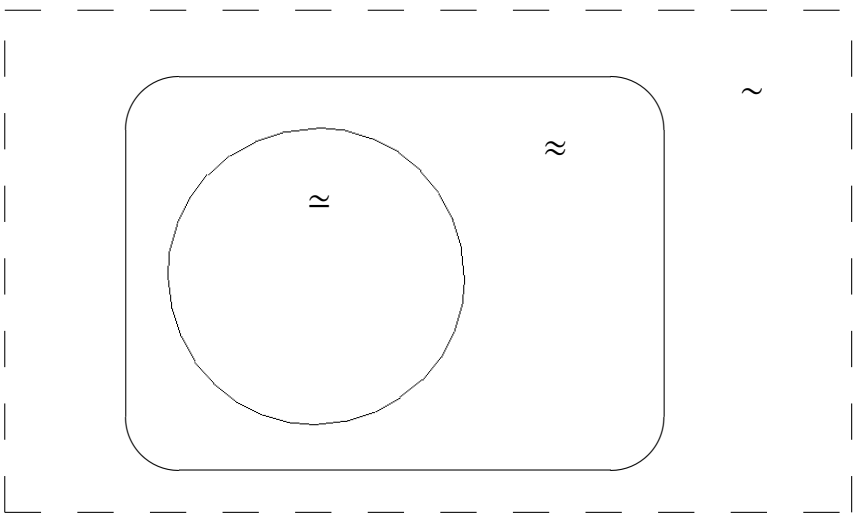


Рис. 1.16. Взаимная вложенность отношений на $\mathcal{M}(P)$

Сплошной линией обозначены отношения, обладающие конечным AT-базисом (в силу своей аддитивной и транзитивной замкнутости).

Как и любая бисимуляция разметок, AT-замыкание бисимуляции ресурсов обладает свойством переноса. Однако хорошая структурированность бисимуляции ресурсов (по сравнению с произвольной бисимуляцией разметок) позволяет использовать и более слабый (локальный) вариант свойства переноса — когда для перехода t рассматриваются только те разметки, при которых он может сработать.

Определение 1.33. *Отношение* $B \subseteq \mathcal{M}(P) \times \mathcal{M}(P)$ *обладает* слабым свойством переноса*, если для всех* $(r, s) \in B$, $t \in T$, *таких что* ${}^\bullet t \cap r \neq \emptyset$, *найдется имитирующий переход* $u \in T$, *такой что* $l(t) = l(u)$ *и, обозначив* $M_1 = {}^\bullet t \cup r$ *и* $M_2 = {}^\bullet t - r + s$, *мы имеем* $M_1 \xrightarrow{t} M_1'$ *и* $M_2 \xrightarrow{u} M_2'$, *где* $(M_1', M_2') \in B^{AT}$.

Слабое свойство переноса может быть представлено в виде следующей диаграммы:

$$
\begin{array}{ccc}
r & B & s \\
{}^\bullet t \cup r & & {}^\bullet t - r + s \\
\downarrow t & & \downarrow (\exists) u,\ l(u) = l(t) \\
M_1' & \sim_{B^{AT}} & M_2'
\end{array}
$$

Поясним данную диаграмму. Пусть имеются ресурсы r и s, связанные отношением B. Ресурс r при этом пересекается с предусловием перехода t. Обозначим результат срабатывания перехода t как M_1' :

$$
{}^\bullet t \cup r \xrightarrow{t} M_1'.
$$

Заменив в разметке ${}^\bullet t \cup r$ ресурс r на ресурс s, получим новую разметку ${}^\bullet t - r + s$. Слабое свойство переноса состоит в том, что обязательно найдется переход u с той же меткой $l(t)$, который может сработать от разметки ${}^\bullet t - r + s$:

$$
{}^\bullet t - r + s \xrightarrow{u} M_2',
$$

причём результат его срабатывания (разметка M_2') будет связан с разметкой M_1' отношением B^{AT}.

Теорема 1.7. *[6] Симметричное и 1-рефлексивное отношение $B \subseteq \mathcal{M}(P) \times \mathcal{M}(P)$ обладает слабым свойством переноса тогда и только тогда, когда B — бисимуляция ресурсов.*

Итак, для определения того, является ли данное конечное симметричное 1-рефлексивное отношение B бисимуляцией ресурсов, достаточно проверить выполнение слабого свойства переноса для конечного числа пар ресурсов.

В [6] приводится алгоритм, определяющий, является или нет данное конечное отношение на множестве ресурсов данной сети Петри бисимуляцией

ресурсов, и имеющий временную сложность $O(\max\{|P||B|^4, |T|^2|P||B_s|^3\})$, где B_s — простой базис отношения B.

Конечно, хотелось бы уметь не только проверять заданное отношение на бисимулярность, но и строить по данной сети Петри максимальную бисимуляцию ресурсов (отношение $(\simeq)$). В частности, проверка бисимулярности двух ресурсов сводится к проверке принадлежности данной пары отношению $(\simeq)$ (которое не известно заранее).

Одним из вариантов приближенного решения является построение аппроксимации максимальной бисимуляции ресурсов данной сети. Если рассматривать не все множество ресурсов сети (по определению бесконечное), а только его конечное подмножество (например, только ресурсы, чья мощность не превышает заданный параметр), то конечным становится и число пар в рассматриваемых отношениях, и мы можем использовать проверку слабого свойства переноса. В [6] приводится такой алгоритм аппроксимации.

До настоящего времени вопрос о разрешимости бисимуляции ресурсов остается открытым. Ниже приводится гипотеза, которая кажется наиболее вероятной (хотя и достаточно неожиданной).

Гипотеза 1.1. *Для любой помеченной сети Петри N выполняется* $(\simeq) = (\approx)$, *то есть максимальная бисимуляция ресурсов совпадает с подобием ресурсов.*

Таким образом, в соответствие с этой гипотезой на диаграмме, изображенной на рисунке 1.16, множества $(\approx)$ и $(\simeq)$ должны совпадать. Контрпримером являлась бы пара подобных ресурсов, не принадлежащих максимальной бисимуляции ресурсов.

Заметим также, что, если приведённая гипотеза верна, то, во-первых, бисимуляция ресурсов в сетях Петри неразрешима (так как подобие ресурсов неразрешимо по следствию 1.4), и, во-вторых, описанный в [6] способ приближения максимальной бисимуляции ресурсов (при помощи параметризованной аппрок-

симации) — единственно возможный. Другими словами, имея некоторое отношение $B \subseteq \mathcal{M}(P) \times \mathcal{M}(P)$, мы можем сказать, является ли это отношение бисимуляцией ресурсов, но не можем сказать, является ли оно *максимальной* бисимуляцией ресурсов.

1.4.4. Условное подобие ресурсов

Расмотрим еще один способ определения корректного с точки зрения бисимулярности отношения на $\mathcal{M}(P)$. При этом способе мы выделяем некоторую разметку (начальное условие) и рассматриваем все возможные пары ресурсов, которые одинаково влияют на поведение сети *при условии наличия данного начального ресурса*.

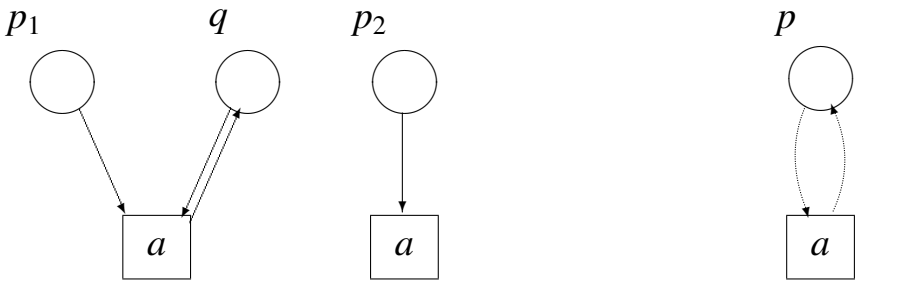
Определение 1.34. *Пусть* $r, s, b \in \mathcal{M}(P)$. *Ресурсы* r *и* s *называются* подобными при условии b *(обозначается* $r \approx_{|b} s$*), если для любого ресурса* $m \in \mathcal{M}(P)$, *такого, что* $b \subseteq m$, *выполняется* $m + r \sim m + s$.

Ресурсы r *и* s *называются* условно подобными *(обозначается* $r \approx_{|} s$*), если существует ресурс* $b' \in \mathcal{M}(P)$, *такой, что* $r \approx_{|b'} s$.

Условное подобие ресурсов обладает естественной интерпретацией. В некоторых ситуациях использовать именно условное подобие удобнее, чем обычное подобие ресурсов.

Рассмотрим сеть, изображенную на рисунке 1.17(а). Ресурсы p_1 и p_2 не подобны, так как при разметке p_1 ни один из переходов не может сработать, тогда как при разметке p_2 может сработать переход a. С другой стороны, они подобны *при условии* q, то есть при наличии в сети ресурса q ресурсы p_1 и p_2 полностью взаимозаменяемы.

Еще один пример приведен на рисунке 1.17(б). Очевидно, что в данной сети Петри любое количество фишек в позиции p может быть заменено любым другим ненулевым количеством фишек, но только *при условии* наличия в позиции p еще одной фишки.



а) $p_1 \approx_{|q} p_2$, $p_1 \not\approx p_2$ б) $p \approx_{|p} \emptyset$

Рис. 1.17. Примеры условно подобных ресурсов

Условное подобие ресурсов улавливает те же структуры во множестве состояний сети Петри, что и обычное подобие ресурсов. Однако, благодаря отдельному рассмотрению общей части (условия), с его помощью проще исследовать некоторые аспекты подобия ресурсов. В частности, с использованием условного подобия доказывается полулинейность множества пар подобных ресурсов.

Следующее утверждение устанавливает некоторые важные свойства условного подобия ресурсов:

Утверждение 1.5. *[6] Пусть* $r, s, b, b', m, m' \in \mathcal{M}(P)$. *Тогда*

1. $m + r \approx m + s \;\Leftrightarrow\; r \approx_{|m} s$.
2. $m \approx_{|} m' \;\wedge\; r \approx_{|} s \;\Rightarrow\; m + r \approx_{|} m' + s$.
3. $r \approx_{|b} s \;\wedge\; b \subseteq b' \;\Rightarrow\; r \approx_{|b'} s$.
4. $m + r \approx_{|b} m + s \;\Leftrightarrow\; r \approx_{|b+m} s$.
5. $m + r \approx_{|} m + s \;\Leftrightarrow\; r \approx_{|} s$.
6. $m \approx m' \;\wedge\; m + r \approx m' + s \;\Rightarrow\; r \approx_{|} s$.
7. $m \approx_{|b} m' \;\wedge\; m + r \approx_{|b'} m' + s \;\Rightarrow\; r \approx_{|} s$.
8. $m \approx_{|b} m' \;\wedge\; r \approx_{|b'} r' \;\Rightarrow\; m + r \approx_{|b \cup b'} m' + r'$.

Итак, условное подобие ресурсов замкнуто относительно сложения. Кроме того, оно инвариантно относительно расширения условия. Утверждения 1.5.4 и 1.5.5 показывают, что общая часть может быть удалена из подобных ресурсов. Утверждения 1.5.6 и 1.5.7 показывают, что разность как пар подобных, так и пар условно подобных ресурсов является парой условно подобных ресурсов. Кроме того, условное подобие ресурсов замкнуто относительно сложения ресурсов и объединения условий (утверждение 1.5.8).

Отметим, что, в отличие от подобия ресурсов, условное подобие замкнуто относительно вычитания. Это свойство может быть использовано при конструировании базиса отношения условного подобия ресурсов.

Следующее утверждение устанавливает связь между подобием ресурсов и условным подобием ресурсов:

Утверждение 1.6. *[6] Пусть* $r, s, m, m' \in \mathcal{M}(P)$, $m \approx m'$. *Тогда*

$$m + r \approx m' + s \;\Leftrightarrow\; r \approx_{|m} s.$$

Определение 1.35. *Пусть* $r, s, r', s', r'', s'' \in \mathcal{M}(P)$. *Пара условно подобных ресурсов* $r \approx_{|} s$ *называется* минимальной*, если ее нельзя представить как сумму двух других пар условно подобных ресурсов, то есть для любой непустой пары* $r' \approx_{|} s'$ *из* $r = r' + r''$ *и* $s = s' + s''$ *следует* $r = r'$ *и* $s = s'$.

Достаточно очевидным следствием утверждения 1.5.7 является:

Теорема 1.8. *[6] Любая пара условно подобных ресурсов может быть представлена в виде суммы минимальных пар условно подобных ресурсов.*

Поскольку пара условно подобных ресурсов может быть представлена как вектор длины $2|P|$ с целочисленными неотрицательными координатами, а минимальная пара условно подобных ресурсов минимальна также и относительно

обычного покоординатного сравнения таких векторов, для любой сети Петри множество минимальных пар условно подобных ресурсов конечно. Следовательно, множество всех пар условно подобных ресурсов является аддитивным замыканием конечного множества минимальных пар.

Определение 1.36. *Пара подобных ресурсов $r \approx s$ называется* минимальной*, если ее нельзя представить как сумму непустой пары подобных ресурсов и пары условно подобных ресурсов, то есть для любой непустой пары подобных ресурсов $r' \approx s'$ из $r = r' + r''$ и $s = s' + s''$ следует $r = r'$ и $s = s'$.*

Теорема 1.9. *[6] Любая пара подобных ресурсов может быть представлена в виде суммы одной минимальной пары подобных ресурсов и нескольких минимальных пар условно подобных ресурсов.*

Заметим, что по свойству покоординатного сравнения векторов с целочисленными неотрицательными компонентами, для любой пары условно подобных ресурсов $r \approx_{|} s$ множество ее минимальных условий (относительно покоординатного сравнения) конечно.

Обозначение 1.4. *Пусть $R \subseteq \mathcal{M}(P) \times \mathcal{M}(P)$ — некоторое подмножество множества пар условно подобных ресурсов ($r \approx_{|} s$ для всех $(r, s) \in R$). Пусть*

$$B = \{\ (u, v) \in \mathcal{M}(P) \times \mathcal{M}(P)\ \mid\ u \approx v\ \wedge\ \forall\ (r, s) \in R \quad u + r \approx v + s\}$$

— множество всех общих условий для R. Через $Cond(R)$ обозначим множество минимальных элементов B (относительно покоординатного сравнения, рассматривая элементы B как вектора длины $2|P|$).

Заметим, что из утверждения 1.6 следует, что для любой пары $(u, v) \in Cond(R)$ оба ресурса u и v являются условиями для любой пары $(r, s) \in R$. Кроме того, в силу свойства покоординатного сравнения векторов, для любого R множество $Cond(R)$ конечно.

Обозначение 1.5. *Пусть $u, v \in \mathcal{M}(P)$ и $u \approx v$. Через $S(u, v)$ обозначим множество всех потенциальных дополнений к паре (u, v) (корректных относительно подобия ресурсов):*

$$S(u, v) = \{(r, r') \in \mathcal{M}(P) \times \mathcal{M}(P) \mid u + r \approx v + r'\}.$$

Через $S_{\min}(u, v)$ обозначим множество всех минимальных относительно покоординатного сравнения элементов множества $S(u, v)$.

Утверждение 1.7. *[6] Пусть $u, v, u', v' \in \mathcal{M}(P)$ и $u \approx v$. Тогда*

1) $S(u, v)$ является отношением эквивалентности;

2) $S(u, v)$ замкнуто относительно сложения;

3) $u \approx v,\ u' \approx v',\ (u, v) \leq (u', v') \ \Rightarrow\ S(u, v) \subseteq S(u', v')$;

4) $S_{min}(u, v)$ конечно.

Из первой и второй части утверждения следует, что множество $S(u, v)$ обладает конечным АТ-базисом. В частности, согласно теореме 1.1, оно обладает основным базисом, составленным из минимальных относительно $\sqsubseteq$ элементов.

Обозначение 1.6. *Пусть N — помеченная сеть Петри. Через $A(N)$ обозначим множество всех множеств потенциальных дополнений в N:*

$$A(N) = \{H \mid \exists (u, v) \ :\ u \approx v \ \wedge\ H = S(u, v)\}.$$

Утверждение 1.8. *[6] Множество $A(N)$ конечно для любой сети N.*

Следующая теорема демонстрирует взаимосвязь между обычным подобием ресурсов и условным подобием ресурсов:

Теорема 1.10. *[6] Пусть N — помеченная сеть Петри, $(\approx)$ — множество всех пар подобных ресурсов в N, $(\approx_|)$ — множество всех пар условно подобных ресурсов в N. Тогда множество $(\approx)$ — полулинейно, причём существует конечное множество $R \subseteq (\approx_|)$, такое, что*

$$(\approx) = \bigcup_{Z \in 2^R} [\, Cond(Z) + lc(Z) \,],$$

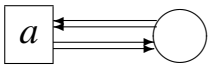


Рис. 1.18. Цикл со сдвоенными дугами

где 2^R — множество всех подмножеств множества R, $lc(Z)$ — множество всех линейных комбинаций над Z.

Подобие ресурсов представимо при помощи некоторого конечного набора пар условно подобных ресурсов (причем в виде полулинейного множества). Может возникнуть вопрос: нельзя ли использовать только *минимальные* условно подобные ресурсы в этом разложении? Действительно, это было бы гораздо удобнее. Однако, к сожалению, это невозможно. Только лишь минимальные условно подобные ресурсы не отражают всей структуры подобия ресурсов.

Рассмотрим в качестве примера сеть Петри, изображенную на рисунке 1.18. Минимальной парой условно подобных ресурсов для данной сети является пара $0 \approx_{|2} 1$. Одна фишка подобна любому числу фишек при условии наличия как минимум двух других фишек в единственной позиции сети. Однако существует и другая (не минимальная) условно подобная пара $1 \approx_{|1} 2$ с меньшим минимальным условием — только одной фишкой.

На рисунке 1.19 представлен другой пример, показывающий, что сумма минимальных условно подобных пар может иметь меньшее минимальное условие, чем слагаемые. Пары $m_1 \approx_{|b_1} m'_1$ и $m_2 \approx_{|b_2} m'_2$ являются минимальными парами условно подобных ресурсов, но пара $m_1 + m_2 \approx m'_1 + m'_2$ обладает пустым условием!

При аддитивном разложении подобных ресурсов неизбежно приходится учитывать не только минимальные условно подобные пары, но и некоторое количество других условно подобных пар (не обязательно минимальных). Выбор соответствующих пар зависит от раскладываемых ресурсов.

Итак, условное подобие сильно взаимосвязано с обычным подобием ресурсов. Это влияет на разрешимость:

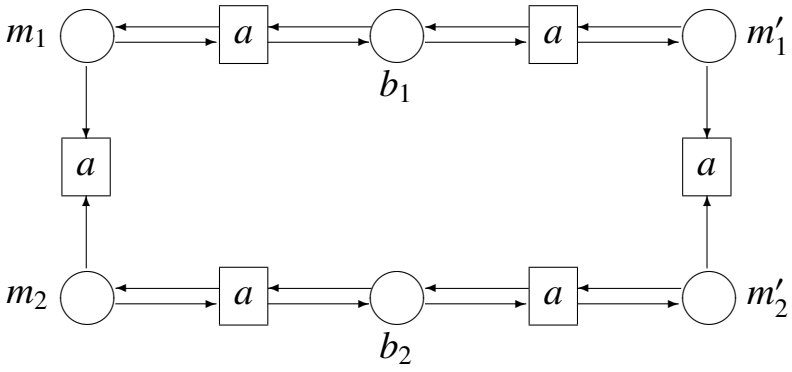


Рис. 1.19. Пример сети

Теорема 1.11. *[6] Проблема распознавания условного подобия ресурсов неразрешима для сетей Петри, то есть невозможно построить алгоритм, отвечающий на вопрос, являются ли данные ресурсы подобными при данном условии в данной сети Петри.*

Как и подобие ресурсов, условное подобие ресурсов в общем случае не может быть эффективно построено.

1.4.5. Редукция (оптимизация) модели

Подобие ресурсов может использоваться для редукции сети Петри, то есть сокращения ее размера при сохранении наблюдаемого поведения (в смысле бисимулярности разметок). В частности, такая редукция представляет собой достаточно мощное средство оптимизации сложных (в том числе распределенных) процессов и систем [61]. Спектр сходных задач весьма широк: от реинжиниринга бизнес-процессов до оптимизирующей компиляции параллельных программ.

Для определения эквивалентности поведения сетей различной структуры нам потребуется понятие бисимуляции разметок *между* двумя сетями Петри:

Определение 1.37. *Пусть* $N_1 = (P_1, T_1, F_1, l_1)$ *и* $N_2 = (P_2, T_2, F_2, l_2)$*— помеченные сети Петри. Отношение* $R \subseteq \mathcal{M}(P_1) \times \mathcal{M}(P_2)$ *обладает* свойством переноса*, если для любой пары разметок* $(M_1, M_2) \in R$ *и для любого перехода* $t \in T_1$*, такого, что в сети* N_1 *существует срабатывание* $M_1 \xrightarrow{t} M'_1$*, найдется имитирующий*

переход $u \in T_2$, *такой, что* $l_1(t) = l_2(u)$ *и в сети* N_2 *существует срабатывание* $M_2 \xrightarrow{u} M_2'$, *где* $(M_1', M_2') \in R$.

Если отношения R *и* R^{-1} *обладают свойством переноса, то* R *называется* бисимуляцией разметок между N_1 *и* N_2.

Маркированные сети Петри (N_1, M_1) *и* (N_2, M_2), *такие, что существует отношение бисимуляции* R, *для которого* $(M_1, M_2) \in R$, *называются* бисимулярными *(обозначается* $(N_1, M_1) \sim (N_2, M_2)$*).*

Известно, что для любых сетей Петри N_1 *и* N_2 *существует максимальная бисимуляция разметок между ними, обозначаемая как* $\sim_{[N_1,N_2]}$.

По определению подобные ресурсы при всех возможных разметках сети полностью взаимозаменяемы, то есть фишки одного можно заменить на фишки другого. Уже это позволяет в некотором смысле “редуцировать” сеть, заменяя в её начальной разметке больший (по количеству фишек) ресурс на меньший.

Однако наибольший интерес представляет изменение не начальной разметки сети, а её графовой структуры, то есть сокращение количества позиций, переходов и дуг. Информацию для такой редукции также можно получить из подобия ресурсов.

Заметим, что всякому подобию ресурсов (мультимножеств позиций) можно сопоставить “подобие” выходных дуг переходов. Другими словами, если срабатывание перехода t помещает в его выходные позиции некоторый ресурс r, у которого есть подобный ресурс s, то мы можем заменить у перехода t все выходные дуги, соответствующие r, на выходные дуги, соответствующие s, и наблюдаемое поведение полученной сети ничем не будет отличаться от поведения исходной (в смысле бисимулярности).

Утверждение 1.9. *[6]*(замена “подобных” выходных дуг) *Пусть* $N = (P, T, F, l)$ *— помеченная сеть Петри;* $r, s \in \mathcal{M}(P)$ *— ресурсы сети* N, *такие, что* $r \approx s$; $t \in T$ *— переход, такой, что* $r \subseteq {}^\bullet t$. *Пусть* $N' = (P, T, F', l)$ *— помеченная сеть Петри,*

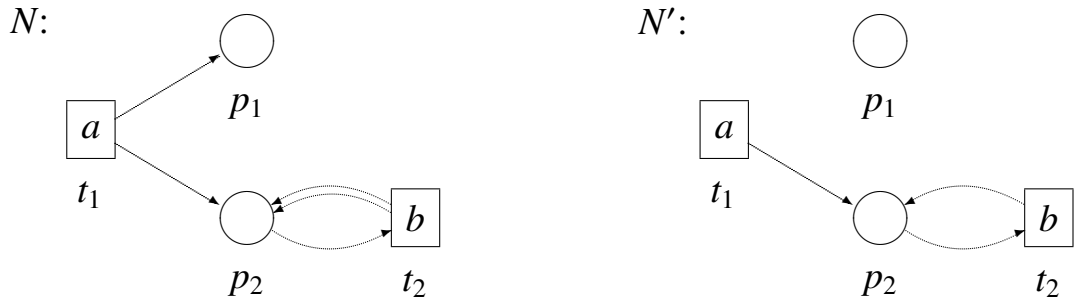


Рис. 1.20. Редукция заменой “подобных” выходных дуг. Для сети N выполняется $p_1 \approx \emptyset$, $2p_2 \approx p_2$. Соответствующие этим ресурсам дуги заменены у переходов t_1 и t_2.

такая, что $\forall p \in P\ F'(t,p) = F(t,p) - r(p) + s(p)$. *Тогда для любой* $M \in \mathcal{M}(P)$ *выполняется* $(N,M) \sim (N',M)$.

Таким образом, при одинаковых начальных разметках поведение сетей N и N' неразличимо (с точки зрения бисимулярности), и мы можем использовать для моделирования исходной системы сеть с меньшим числом дуг.

Замена подобных дуг во многих случаях позволяет добиваться существенного упрощения структуры сети.

Рассмотрим редукцию сети Петри N, изображенной на рисунке 1.20. Сеть N' получена из исходной сети N при помощи следующих замен “подобных” дуг:

1. $p_1 \approx \emptyset$. Дуга (t_1, p_1) заменена на $(t_1, \emptyset)$ (то есть удалена).

2. $2p_2 \approx p_2$. Две дуги (t_2, p_2) заменены на одну такую же.

Замена “подобных” выходных дуг может производиться даже у немаркированных сетей, поскольку одинаковые разметки исходной и редуцированной сетей бисимулярны.

Применить аналогичный прием для входных дуг переходов не удается без существенных модификаций.

Во-первых, заметим, что в случае входных дуг необходимо учитывать начальную разметку. Рассмотрим сеть, изображенную на рисунке 1.21. В данном случае дуги (p_1, t_1) и (p_2, t_1) заменены на дугу (p_3, t_1), в начальной разметке ре-

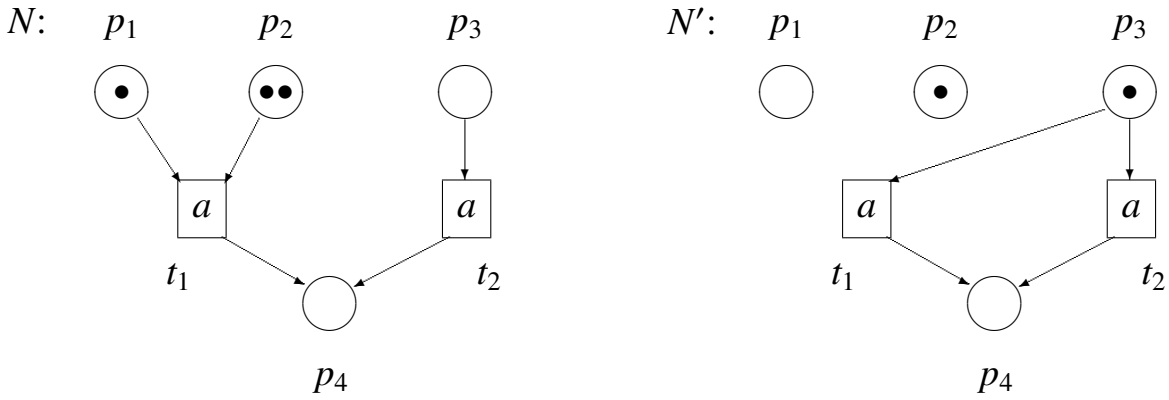


Рис. 1.21. Пример редукции заменой входных дуг и изменением начальной разметки. Для сети N выполняется $p_1 + p_2 \approx p_3$.

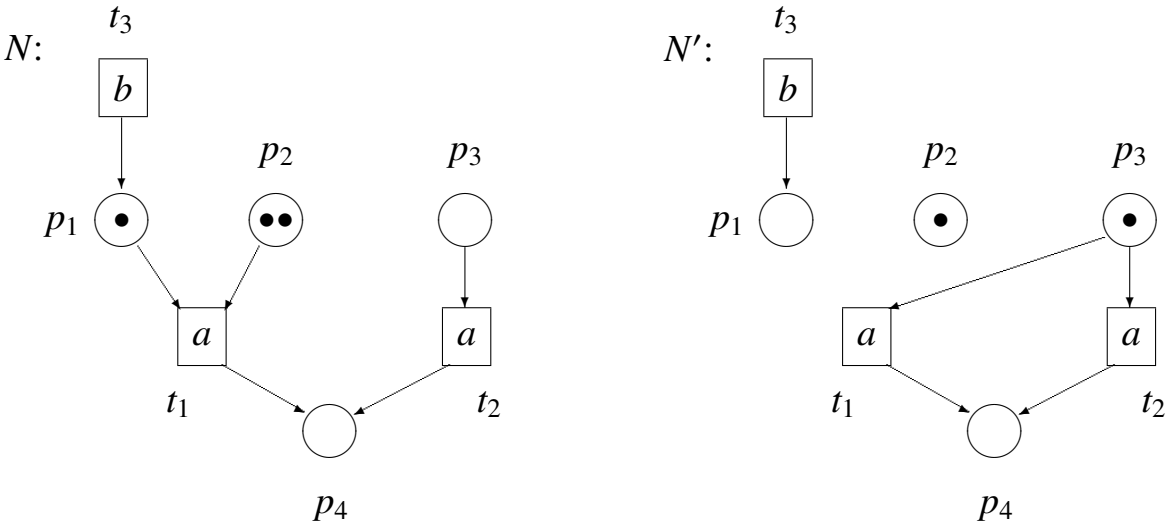


Рис. 1.22. Пример некорректной редукции. Для сети N выполняется $p_1 + p_2 \approx p_3$, однако $(N, p_1 + 2p_2) \not\sim (N', p_2 + p_3)$

сурс $p_1 + p_2$ заменен на p_3. Если бы мы не изменили начальную разметку, то в сети N' ни один переход не смог бы сработать.

Однако и замены начальной разметки не всегда достаточно для корректной редукции. Рассмотрим сети, изображенные на рисунке 1.22. Сеть N' получена из сети N тем же преобразованием, что и в предыдущем случае. Однако маркированные сети $(N, p_1 + 2p_2)$ и $(N', p_2 + p_3)$ не бисимулярны. Действительно, в сети $(N, p_1 + 2p_2)$ возможна последовательность срабатываний $t_3.t_1.t_1$, помеченная словом “baa”, тогда как в сети $(N', p_2 + p_3)$ переход, помеченный символом “a”, может сработать только один раз.

Следовательно, требуются дополнительные ограничения.

Рассмотрим следующее условие. Пусть заменяемый ресурс “неделим” с точки зрения начальной разметки сети и с точки зрения предусловий и постусловий ВСЕХ переходов. Другими словами, этот ресурс входит в начальную разметку “целое” число раз и всякий переход при срабатывании забирает из входных позиций и помещает в выходные позиции “целое” число таких ресурсов.

Это условие достаточно сильное, однако оно позволяет производить полную замену одного ресурса на другой (подобный):

Утверждение 1.10. *[6]*(замена подобных ресурсов) *Пусть* (N, M_0) *— помеченная маркированная сеть Петри, где* $N = (P, T, F, l)$ *— помеченная сеть Петри. Пусть* $r, s \in \mathcal{M}(P)$ *— ресурсы сети* N*, такие, что*

- $r \approx s$;
- $M_0 = nr + \overline{M}_0$, *где* $n \in Nat,\ \overline{M}_0 \cap r = \emptyset$;
- *для любого перехода* $t \in T$ *выполняется* ${}^\bullet t = i_t r + g_t$, *где* $i_t \in Nat,\ g_t \cap r = \emptyset$, *и* $t^\bullet = j_t r + h_t$, *где* $j_t \in Nat,\ h_t \cap r = \emptyset$.

Пусть $M'_0 = ns + \overline{M}_0$, $N' = (P, T, F', l)$ *— помеченная сеть Петри, такая, что* $\forall t \in T,\ \forall p \in P$

$$F'(p,t) = F(p,t) - i_t r(p) + i_t s(p) \quad и \quad F'(t,p) = F(t,p) - j_t r(p) + j_t s(p).$$

Тогда $(N, M_0) \sim (N', M'_0)$.

Очевидно, что после такой замены многие позиции могут остаться несвязанными ни с одним переходом, и их можно будет просто удалить из сети. Строго говоря,

Утверждение 1.11. *[6] Пусть выполнены все условия утверждения 1.10,* $p \in P$. *Тогда* $r(p) > 0 \ \wedge \ s(p) = 0 \quad \Rightarrow \quad \forall t \in T \quad F'(p,t) = 0 \ \wedge \ F'(t,p) = 0.$

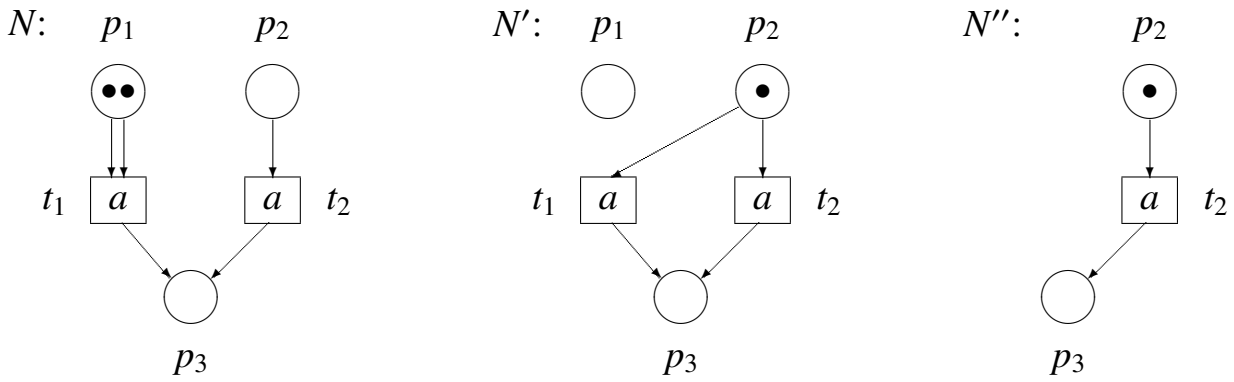


Рис. 1.23. Пример редукции заменой подобных ресурсов. Для сети N выполняется $2p_1 \approx p_2$. Произведена замена при $r = 2p_1,\ s = p_2$.

Другими словами, позиция p избыточна, если она принадлежит ресурсу r и не принадлежит ресурсу s.

Это не единственный “побочный” эффект от замены подобных ресурсов. При перемещении дуг по правилу, описанному в утверждении 1.10, в сети может возникнуть много одинаковых переходов. Очевидно, что при наличии в сети двух одинаковых переходов один из них можно удалить без последствий для наблюдаемого поведения.

На рисунке 1.23 приведен пример редукции заменой подобных ресурсов. Сеть N' получена из сети N заменой ресурса $2p_1$ на ресурс p_2. Сеть N'' получена из сети N' удалением избыточной позиции p_1 и перехода t_1, дублирующего переход t_2.

Итак, существуют два способа редукции обыкновенной сети Петри с использованием подобия ресурсов. Во-первых, можно редуцировать немаркированную сеть, заменяя “подобные” выходные дуги переходов. Во-вторых, можно редуцировать маркированную сеть с заданной начальной разметкой, полностью заменяя один ресурс на другой (при выполнении дополнительных условий, накладываемых на заменяемый ресурс). Очевидно, что оба этих способа можно использовать и одновременно, получая еще более сильную редукцию. Получаемая при этом сеть меньшего размера будет иметь то же поведение, что и исходная (с точки зрения бисимулярности).

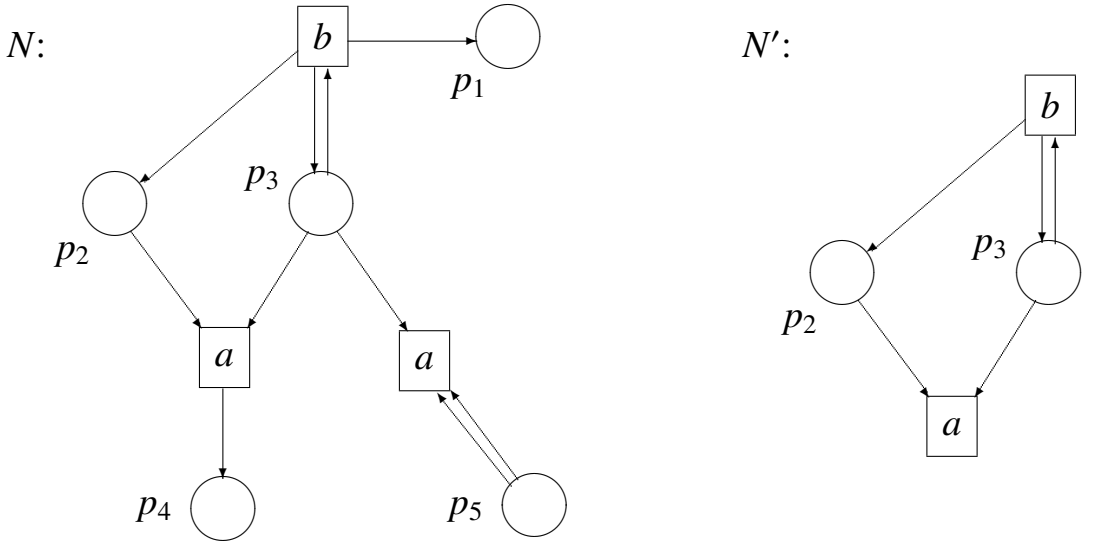


Рис. 1.24. Пример редукции сети Петри на основе подобия ресурсов. Для сети N выполняется $p_1 \approx p_4 \approx \emptyset,\ p_2 \approx 2p_5$.

Пример “комплексной” редукции сети Петри на основе подобия ресурсов приведен на рисунке 1.24. В данном случае объединены ресурсы p_2 и $2p_5$, удалены ресурсы (позиции) p_1 и p_4 (как тупиковые, то есть подобные пустому ресурсу $\emptyset$). Переходы с меткой a слиты в один переход.

В приведенных утверждениях использовано подобие ресурсов, однако очевидно, что на практике для редукции можно использовать и любые сужения подобия. Фактически, если известна только одна пара эквивалентных ресурсов, её уже можно использовать для редукции.

Глава 2

Некоторые методы поиска бисимуляционно-эквивалентных ресурсов

Данная глава посвящена исследованию поведенческих эквивалентностей ресурсов в сетях Петри. Рассматриваются возможности приближения неразрешимых в общем случае отношений бисимуляции разметок и подобия ресурсов. Предлагаются следующие подходы:

1. Построение множеств пар ресурсов, соответствующих некоторому конечному поведению сети (бисимуляции ограниченных разметок).
2. Приближение подобия ресурсов снизу (ограниченное подобие ресурсов).
3. Приближение подобия ресурсов сверху (расслоенное подобие ресурсов).
4. Выражение свойств эквивалентности не только пассивных ресурсов-фишек, но и активных агентов-переходов (подобие обобщённых ресурсов).
5. Использование отношений эквивалентности ресурсов для адаптивного управления процессами.

Каждый из рассматриваемых видов поведенческих эквивалентностей обладает специфическими конструктивными свойствами.

2.1. Бисимуляция в ограниченных сетях

В данном разделе рассматривается проблема поиска бисимулярных разметок в ограниченных сетях Петри. Для данного простейшего класса сетей Петри понятия ресурса и разметки неразделимы — число достижимых разметок конечно, поэтому каждая из них определяет тривиальный ресурс.

Вводятся и исследуются специальные виды бисимуляции для случая ограниченных сетей, в том числе расширение бисимуляции достижимых разметок — отношение, учитывающее кроме достижимых разметок еще и все разметки, бисимулярные достижимым (среди которых могут быть и неограниченные).

Доказывается разрешимость расширения бисимуляции разметок. Вводится понятие элементарного расширения бисимуляции — конечного подмножества полного расширения, в которое входят только минимальные (относительно вложения) пары разметок. Доказывается вычислимость элементарного расширения бисимуляции (предлагается алгоритм его построения).

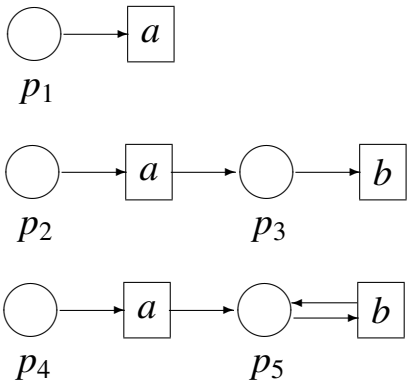
2.1.1. Расслоенная бисимуляция разметок

Напомним, что n-расслоенная бисимуляция разметок — это бисимуляция разметок в том случае, когда нам не важно, что будет происходить с сетью после n-го шага (см. определение в разделе 1.2.2). Все последующее поведение сети просто игнорируется.

На рисунке 2.1 приведен пример расслоенных бисимуляций разметок. При увеличении n количество пар в отношении уменьшается. Заметим, что на рисунке приведены не все пары разметок, а только те, которые содержат ровно одну фишку. На самом деле пар бисимулярных разметок существенно больше (например, $2p_3 \sim_2 p_5$). Пределом последовательности n-расслоенных бисимуляций разметок является достаточно простое отношение:

$$(\sim_\infty) = (\sim) = \{(M, M) \mid M \in \mathcal{M}(P)\} \cup$$
$$\{(M + kp_3 + lp_5, M + mp_3 + np_5) \mid M \in \mathcal{M}(P),\ k, m \geq 0,\ l, n > 0\}.$$

Напомним, что разметка M сети N называется *ограниченной*, если множество $\mathcal{R}(N, M)$ — конечно. Сеть (N, M) в таком случае называется ограниченной. Для случая ограниченных сетей известно следующее свойство, связывающее обычную и расслоённую бисимуляции разметок:



$n = 0: \quad p_1 \sim_0 p_2 \sim_0 p_3 \sim_0 p_4 \sim_0 p_5$

$n = 1: \quad p_1 \sim_1 p_2 \sim_1 p_4, \; p_3 \sim_1 p_5$

$n = 2: \quad p_2 \sim_2 p_4$

$n \geq 3: \quad \emptyset$

Рис. 2.1. Расслоенные бисимуляции разметок (перечислены только пары разметок, содержащих ровно одну фишку)

Лемма 2.1. *[144] Пусть* $M, M' \in \mathcal{M}(P)$, *где* $|\mathcal{R}(N, M)| = n < \infty$. *Тогда*

$$M \sim M' \quad \Longleftrightarrow \quad M \sim_n M' \;\wedge\; \mathcal{R}(N, M') \cap Inc(N, M) \neq \emptyset.$$

Здесь $Inc(N, M)$ обозначает множество несовместимых с M разметок сети N. Это полулинейное множество, которое может быть эффективно построено. Выполнение условия $\mathcal{R}(M') \cap Inc(N, M) \neq \emptyset$ может быть установлено при помощи алгоритма определения достижимости подразметки. Отсюда, в частности, следует разрешимость проблемы бисимулярности ограниченной и неограниченной разметок сети Петри [144].

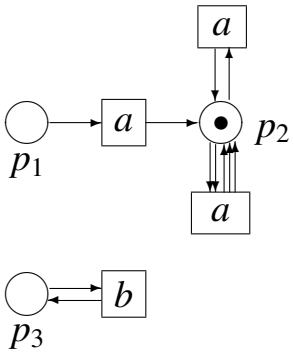
2.1.2. Специальные виды бисимуляции разметок

Введём три сужения отношения бисимуляции разметок, которые тем или иным образом учитывают свойства ограниченности и достижимости разметок ограниченной сети Петри.

Пусть $N = (P, T, F, l)$ — помеченная сеть Петри, M_0 — ограниченная разметка.

Определение 2.1. *Бисимуляцией достижимых разметок назовём отношение* $(\sim_{[r(M_0)]}) \subseteq \mathcal{M}(P) \times \mathcal{M}(P)$, *такое, что*

$$M_1 \sim_{[r(M_0)]} M_2 \quad \Leftrightarrow \quad M_1 \sim M_2 \;\wedge\; M_1 \in \mathcal{R}(N, M_0) \;\wedge\; M_2 \in \mathcal{R}(N, M_0).$$



$(\sim_{[r(M_0)]}) = \{(p_2, p_2)\}$

$(\sim_{[b]}) = \{(\emptyset, \emptyset), (p_1, p_1), (p_2, p_2), (p_1, p_2), (p_2, p_1)\} \cup \{(np_3, mp_3), (p_1 + np_3, p_2 + mp_3), (p_2 + np_3, p_1 + mp_3) \mid n, m > 0\}$

$(\sim_{[e(M_0)]}) = \{(np_1 + mp_2, kp_1 + lp_2) \mid n + m, k + l > 0\}$

$(\sim) = \{(\emptyset, \emptyset)\} \cup \{(np_1 + mp_2, kp_1 + lp_2), (sp_3, tp_3), (np_1 + mp_2 + sp_3, kp_1 + lp_2 + tp_3) \mid n + m, k + l, s, t > 0\}$

Рис. 2.2. Пример отношений $(\sim_{[r(M_0)]}), (\sim_{[b]}), (\sim_{[e(M_0)]})$ и $(\sim)$

Определение 2.2. *Бисимуляцией ограниченных разметок назовём отношение* $(\sim_{[b]}) \subseteq \mathcal{M}(P) \times \mathcal{M}(P)$, *такое, что*

$$M_1 \sim_{[b]} M_2 \quad \Leftrightarrow \quad M_1 \sim M_2 \ \wedge \ |\mathcal{R}(N, M_1)| < \infty \ \wedge \ |\mathcal{R}(N, M_2)| < \infty.$$

Определение 2.3. *Расширением бисимуляции (достижимых разметок) назовём отношение* $(\sim_{[e(M_0)]}) \subseteq \mathcal{M}(P) \times \mathcal{M}(P)$, *такое, что*

$$M_1 \sim_{[e(M_0)]} M_2 \quad \Leftrightarrow \quad \exists M_3 \in \mathcal{R}(N, M_0) \ : \ M_1 \sim M_2 \sim M_3.$$

Пример ограниченной сети Петри и отношений $(\sim_{[r(M_0)]}), (\sim_{[b]}), (\sim_{[e(M_0)]})$ и $(\sim)$ представлен на Рис. 2.2.

Утверждение 2.1. *Для любых сети* N *и ограниченной разметки* M_0 :

1. $|(\sim_{[r(M_0)]})| < \infty$;
2. $(\sim_{[r(M_0)]}) \subseteq (\sim_{[b]}) \subseteq (\sim)$;
3. $(\sim_{[r(M_0)]}) \subseteq (\sim_{[e(M_0)]}) \subseteq (\sim)$;
4. $(\sim_{[b]}), (\sim_{[r(M_0)]})$ *и* $(\sim_{[e(M_0)]})$ *— отношения бисимуляции.*

Доказательство: (1) Следует из конечности множества $\mathcal{R}(N, M_0)$.

(2-3) $(\sim_{[r(M_0)]}) \subseteq (\sim_{[b]})$ следует из того, что для любой разметки $M \in \mathcal{R}(N, M_0)$ выполняется $|\mathcal{R}(N, M)| < \infty$.

$(\sim_{[r(M_0)]}) \subseteq (\sim_{[e(M_0)]})$ следует из определений $(\sim_{[r(M_0)]})$ и $(\sim_{[e(M_0)]})$.

$(\sim_{[b]}) \subseteq (\sim)$ и $(\sim_{[e(M_0)]}) \subseteq (\sim)$ следуют из того, что $(\sim)$ — наибольшая бисимуляция разметок.

(4) Легко убедиться, что все эти отношения замкнуты относительно срабатывания переходов сети. □

Утверждение 2.2. *Найдутся сеть N и ограниченная разметка M_0, такие, что:*

1. $|(\sim_{[b]})| = |(\sim_{[e(M_0)]})| = \infty$;
2. $(\sim_{[r(M_0)]}) \subset (\sim_{[b]}) \subset (\sim)$;
3. $(\sim_{[r(M_0)]}) \subset (\sim_{[e(M_0)]}) \subset (\sim)$;
4. $(\sim_{[b]}) \not\subseteq (\sim_{[e(M_0)]}) \ \wedge \ (\sim_{[e(M_0)]}) \not\subseteq (\sim_{[b]})$.

Доказательство: См. пример сети на Рис. 2.2. □

Теорема 2.1. *Отношения $(\sim_{[b]}), (\sim_{[r(M_0)]})$ и $(\sim_{[e(M_0)]})$ разрешимы.*

(Существуют алгоритмы, отвечающие на вопросы "$M_1 \sim_{[b]} M_2$", "$M_1 \sim_{[r(M_0)]} M_2$" и "$M_1 \sim_{[e(M_0)]} M_2$" для произвольной сети N, произвольной ограниченной разметки M_0 и произвольных разметок M_1 и M_2.)

Доказательство: Поскольку отношениями $(\sim_{[b]})$ и $(\sim_{[r(M_0)]})$ могут быть связаны только ограниченные разметки, в случае неограниченности одной из разметок M_1 и M_2 ответ на вопрос отрицательный. Установить ограниченность разметок можно при помощи алгоритма построения полного покрывающего дерева [41]. Таким образом, для доказательства разрешимости $(\sim_{[b]})$ и $(\sim_{[r(M_0)]})$ достаточно доказать разрешимость бисимулярности ограниченных разметок. Заметим, что две ограниченные разметки порождают две системы с конечным числом состояний, а их бисимулярность можно установить простым перебором всех состояний.

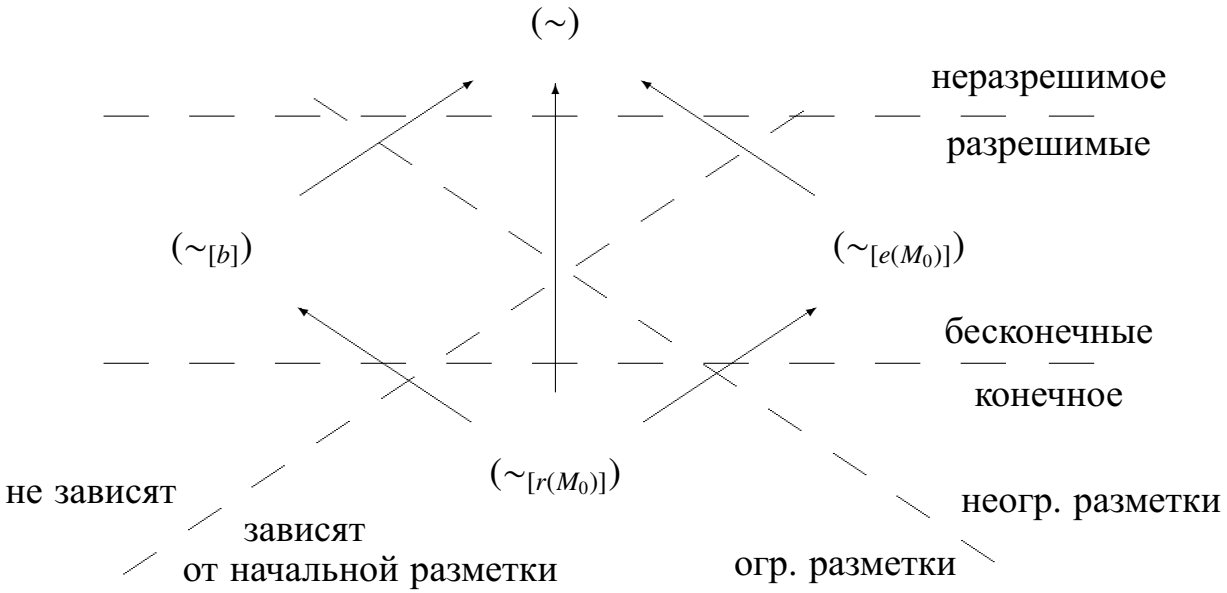


Рис. 2.3. Свойства отношений бисимуляции

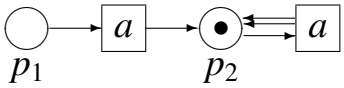
Отношением $(\sim_{[e(M_0)]})$ могут быть связаны неограниченные разметки. Однако из $(M_1, M_2) \in (\sim_{[e(M_0)]})$ следует существование достижимой разметки $M_3 \in \mathcal{R}(N, M_0)$, такой что $M_1 \sim M_2 \sim M_3$. Поскольку множество $\mathcal{R}(N, M_0)$ конечно и мы можем его эффективно перебрать, разрешимость $(\sim_{[e(M_0)]})$ сводится к разрешимости бисимулярности ограниченной и неограниченной разметок. Разрешимость этой проблемы следует из леммы 2.1. □

Свойства отношений $(\sim_{[b]}), (\sim_{[r(M_0)]})$ и $(\sim_{[e(M_0)]})$ можно представить в виде диаграммы, изображённой на Рис. 2.3.

2.1.3. Элементарное расширение бисимуляции ограниченных разметок

Из диаграммы видно, что наибольший практический интерес представляет отношение $(\sim_{[e(M_0)]})$. Во-первых, в нем учитываются неограниченные разметки. Во-вторых, не рассматриваются неактуальные разметки, то есть те состояния, которые в принципе не могут быть получены в моделируемой системе (например, состояния с фишками в позиции p_3 для сети на рисунке 1).

Однако множество $(\sim_{[e(M_0)]})$ может содержать бесконечно много пар разметок. Следовательно, для его практического использования необходимо уметь строить какие-то конечные представления или подмножества.



$$(\sim_{[e(M_0)]}) = \{\ (np_1 + mp_2, kp_1 + lp_2) \mid n + m, k + l > 0\ \}$$

$$(\sim_{[ee(M_0)]}) = \{\ (p_1, p_1), (p_2, p_2), (p_1, p_2), (p_2, p_1)\ \}$$

Рис. 2.4. Пример отношений $(\sim_{[e(M_0)]})$ и $(\sim_{[ee(M_0)]})$

Определение 2.4. *Элементарным расширением бисимуляции назовём отношение* $(\sim_{[ee(M_0)]}) \subseteq (\sim_{[e(M_0)]})$, *такое, что*

$$M_1 \sim_{[ee(M_0)]} M_2 \quad \Leftrightarrow \quad \nexists M_3 \in \mathcal{M}(P)\ :\ (M_1 \sim M_2 \sim M_3\ \wedge\ (M_3 \subset M_1 \vee M_3 \subset M_2)).$$

Другими словами, элементарное расширение — это подмножество (полного) расширения, где в каждом классе эквивалентности оставлены только попарно несравнимые разметки.

Утверждение 2.3. *Для любых сети* N *и ограниченной разметки* M_0 :

1. $|(\sim_{[ee(M_0)]})| < \infty$;
2. $(\sim_{[ee(M_0)]})$ — *отношение эквивалентности.*

Доказательство: (1) Следует из конечности $\mathcal{R}(N, M_0)$, а также из конечности любого множества попарно несравнимых векторов с целыми неотрицательными коэффициентами.

(2) Следует из определений. □

Утверждение 2.4. *Найдутся сеть* N *и ограниченная разметка* M_0, *такие, что* $(\sim_{[ee(M_0)]})$ *не является бисимуляцией разметок.*

Доказательство: См. Рис. 2.4. □

Пусть $M \in \mathcal{M}(P)$. Через $ActNames(M)$ обозначим множество меток переходов, готовых к срабатыванию:

$$ActNames(M) =_{def} \{a \in Act \mid \exists t \in T\ :\ (M \xrightarrow{t} M'\ \wedge\ l(t) = a)\}.$$

Аналогично для множества переходов $U \subseteq T$ через $ActNames(U)$ обозначим множество всех его меток:

$$ActNames(U) =_{def} \{a \in Act \mid \exists t \in U \ : \ l(t) = a\}.$$

Лемма 2.2. *Пусть* $M, M' \in \mathcal{M}(P)$. *Тогда*

$$ActNames(M) = ActNames(M') \ \Leftrightarrow \ M \sim_1 M'.$$

Доказательство: Следует из того, что $M \sim_0 M'$ для любых M и M'. □

Теорема 2.2. *Множество* $(\sim_{[ee(M_0)]})$ *вычислимо.*

В качестве доказательства приведем алгоритм построения.

Алгоритм 2.1. *(построения элементарного расширения бисимуляции)*
ввод : *Сеть Петри* $N = (P, T, F, l)$, *ограниченная разметка* $M_0 \in \mathcal{M}(P)$.
вывод : *Множество* $(\sim_{[ee(M_0)]})$.
шаг I : *Пусть* $(\sim_{[ee(M_0)]}) = \{(\emptyset, \emptyset)\}$.
шаг II : *Для каждой разметки* $M \in \mathcal{R}(N, M_0)$ *(далее* $n = |\mathcal{R}(M)|$*):*

1. *Положим* $B = \emptyset$, $C = \{M' \in \mathcal{M}(P) \mid \exists U \subseteq T \ : \ (ActNames(U) = ActNames(M) \wedge M' = {}^\bullet U)\}$.

2. *Пока в C есть элементы, будем выполнять следующие действия:*

2.1. *Удалим из C каждую разметку* M', *для которой найдется* $M'' \in B$, *такая что* $M'' \subseteq M'$.

2.2. *Удалим из C каждую разметку* M', *для которой выполняется* $\mathcal{R}(M') \cap Inc(N, M) \neq \emptyset$.

2.3. *Рассмотрим произвольную разметку* $M' \in C$. *Определим наибольшее* $m \leq n$, *такое что* $M \sim_m M'$. *(Из леммы 2 следует, что* $m > 0$.*)*

Если $m = n$, *то добавим* (M, M') *и* (M', M) *в* $(\sim_{[ee(M_0)]})$, *добавим* M' *в* B, *удалим* M' *из* C *и вернемся на шаг 2.*

Если $m < n$, *то возможны две ситуации:*

(а) для некоторого срабатывания $M \xrightarrow{\sigma} L$, *такого что* $|\sigma| = m$, *не найдется имитирующего срабатывания* $M' \xrightarrow{\sigma'} L'$, *такого что* $l(\sigma) = l(\sigma')$ *и* $L \sim_1 L'$;

(б) для некоторого срабатывания $M' \xrightarrow{\sigma'} L'$, *такого что* $|\sigma'| = m$, *не найдется имитирующего срабатывания* $M \xrightarrow{\sigma} L$, *такого что* $l(\sigma) = l(\sigma')$ *и* $L \sim_1 L'$.

Также возможна комбинация (а) и (б).

Заметим, что поскольку $M \sim_m M'$, *сами последовательности* σ *и* σ' *всегда существуют. Не выполниться может только требование* $L \sim_1 L'$. *Согласно лемме 2.2, условия (а) и (б) можно переписать:*

(а') ActNames$(L)\backslash$*ActNames*$(L') \neq \emptyset$; *(б') ActNames*$(L')\backslash$*ActNames*$(L) \neq \emptyset$.

Выполнение условия (б') означает, что от M *не достижима ни одна разметка, которая правильно 1-симулирует разметку* L', *достижимую от* M'. *В таком случае удаляем* M' *из* C *и возвращаемся на шаг 2.*

Рассмотрим ситуацию, когда выполнено только условие (а'):

Пусть $L'(\sigma) = \{L' \in \mathcal{M}(P) \mid \exists \sigma' \ : \ (M' \xrightarrow{\sigma'} L' \ \wedge \ l(\sigma) = l(\sigma'))\}$ *— множество "потенциально имитирующих разметок". Для каждой разметки* $L' \in L'(\sigma)$ *и для каждого* $U \subseteq T \ : \ ActNames(U) = ActNames(L) \setminus ActNames(L')$ *добавим в* C *разметку* $M' + (^\bullet U - L')$, *а затем вернемся на шаг 2.*

Алгоритм заканчивает свою работу за конечное число шагов, так как на шаге II рассматривается конечное множество $\mathcal{R}(N, M_0)$, а на шаге 1 в множество C помещается конечное число разметок. Кроме того, в ходе шага 2 при увеличении разметки (элемента множества C) строго увеличивается количество готовых к срабатыванию переходов. Очевидно, что это может привести либо к возникновению ситуации (б') (и прерыванию данной ветви алгоритма), либо к увеличению

значения m. Поскольку m ограничено сверху величиной n, этот процесс не может продолжаться бесконечно долго.

Из вычислимости множества $(\sim_{[ee(M_0)]})$ следует более слабое утверждение о разрешимости:

Следствие 2.1. *Отношение* $(\sim_{[ee(M_0)]})$ *разрешимо.*

2.2. Ограниченное подобие ресурсов

Неразрешимость фундаментального отношения подобия ресурсов вынуждает искать более грубые способы выявления бисимулярных структур на множестве ресурсов. Одним из вариантов могло бы стать ограничение сверху размера рассматриваемых мультимножеств. Однако, как будет далее показано, это также приводит к алгоритмической неразрешимости.

В данном разделе вводится понятие ограниченного подобия ресурсов в обыкновенных сетях Петри. Такое отношение не учитывает пары ресурсов, мощность хотя бы одного из которых превышает заданный параметр.

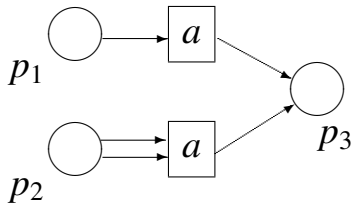
2.2.1. Определение и основные свойства

Определение 2.5. *Пусть* $n \in Nat$ *— целое неотрицательное число,* $N = (P, T, F, l)$ *— помеченная сеть Петри,* $(\approx)$ *— множество всех пар подобных ресурсов сети* N. *Через* $(\approx_{\overline{n}})$ *обозначим подмножество множества* $(\approx)$*, состоящее только из пар, получаемых аддитивно-транзитивным замыканием элементов* $(\approx)$*, мощности левой и правой частей которых не превышают* n :

$$(\approx_{\overline{n}}) =_{def} \{(r, s) \in (\approx) \mid (|r| \leq n \ \&\ |s| \leq n) \ \vee \ (r, s) \in (\approx_{\overline{n}})^{AT}\}.$$

Соответствующее отношение назовем n-ограниченным подобием ресурсов.

На рисунке 2.5 приведен пример 2-ограниченных подобных ресурсов. Две фишки в позиции p_2 эквивалентны одной фишке в позиции p_1. Очевидно, что



$p_1 \approx 2p_2$

$p_1 \not\approx_{\overline{1}} 2p_2$

$p_1 \approx_{\overline{2}} 2p_2$

Рис. 2.5. Подобные ресурсы n-ограниченно подобны не для всех n

эта эквивалентность не будет обнаружена, если рассматривать только ресурсы мощности не более единицы.

Рассмотрим основные свойства ограниченного подобия.

Утверждение 2.5. *Пусть* $N = (P, T, F, l)$ *— помеченная сеть Петри,* r, s, u, v *— ресурсы сети* N, $n \in Nat$. *Тогда*

1. $r \approx_{\overline{n}} r$ *для* $n > 0$;
2. $r \approx_{\overline{n}} s \;\Rightarrow\; s \approx_{\overline{n}} r$;
3. $r \approx_{\overline{n}} s \;\wedge\; s \approx_{\overline{n}} u \;\Rightarrow\; r \approx_{\overline{n}} u$;
4. $r \approx_{\overline{n}} s \;\wedge\; u \approx_{\overline{n}} v \;\Rightarrow\; r + u \approx_{\overline{n}} s + v$.

Доказательство: Из определений. □

Следствие 2.2. *n-ограниченное подобие ресурсов обладает конечным AT-базисом.*

Доказательство: Следует из аддитивной и транзитивной замкнутости отношения $\approx_{\overline{n}}$. □

Таким образом, ограниченное подобие ресурсов обладает теми же структурными свойствами, что и полное подобие ресурсов.

2.2.2. Ограниченное подобие как способ приближения полного подобия

Рассмотрим, как связаны между собой полное и ограниченное подобия.

Утверждение 2.6. *Для любого $n \in Nat$ выполняется*

$$r \approx_{\overline{n}} s \implies r \approx_{\overline{n+1}} s.$$

Другими словами, $(\approx_{\overline{n}}) \subseteq (\approx_{\overline{n+1}})$.

Доказательство: Из определений. □

Утверждение 2.7. *Для любой помеченной сети Петри N найдется число $n \in Nat$, такое что $(\approx) = (\approx_{\overline{n}})$.*

То есть подобие ресурсов — предел n-ограниченного подобия ресурсов.

Доказательство: В качестве n можно взять максимальную мощность компонентов конечного базиса отношения $(\approx)$ (по следствию 1.3 такой базис всегда существует). □

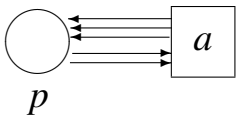
Итак, последовательность ограниченных подобий ресурсов стабилизируется начиная с некоторого n, причем пределом этой последовательности является полное подобие ресурсов:

$$(\approx_{\overline{0}}) \subseteq (\approx_{\overline{1}}) \subseteq \cdots \subseteq (\approx_{\overline{n-1}}) \subseteq (\approx_{\overline{n}}) = (\approx_{\overline{n+1}}) = \cdots = (\approx).$$

На рисунке 2.6 приведен пример такой ситуации. При увеличении параметра n ограниченное подобие “обрастает” новыми парами ресурсов: при $n = 1$ добавляется пара (p, p), при $n = 3$ — пары $(2p, 3p)$ и $(3p, 2p)$. Однако начиная с $n = 3$ множество эквивалентных пар стабилизируется, становясь равным полному подобию ресурсов $(\approx)$.

Использовать на практике ограниченное подобие в качестве приближения полного подобия проблематично, так как

Теорема 2.3. *n-ограниченное подобие ресурсов неразрешимо для любого $n > 0$.*



$$(\approx_{\overline{0}}) = \emptyset$$
$$(\approx_{\overline{1}}) = \{(p, p)\}^{AT}$$
$$(\approx_{\overline{2}}) = \{(p, p)\}^{AT}$$
$$(\approx_{\overline{3}}) = \{(p, p), (2p, 3p), (3p, 2p)\}^{AT}$$
$$(\approx_{\overline{4}}) = \{(p, p), (2p, 3p), (3p, 2p)\}^{AT}$$
$$\dots$$
$$(\approx) = \{(p, p), (2p, 3p), (3p, 2p)\}^{AT}$$

Рис. 2.6. “Стабилизация” ограниченного подобия при $n = 3$

Доказательство: Слияние позиций, исследованное в [53, 190], совпадает с 1−ограниченным подобием ресурсов. В [53] было доказано, что слияние позиций неразрешимо.

Для любого $n > 1$ из разрешимости n-ограниченного подобия ресурсов следовала бы и разрешимость 1−ограниченного подобия — достаточно было бы взять только пары ресурсов, компоненты которых содержат не более одной фишки. □

Следствие 2.3. *n-ограниченное подобие ресурсов невычислимо для любого $n > 0$.*

2.3. Расслоенное подобие ресурсов

Ограниченные подобия и бисимуляции ресурсов являются сужениями полного подобия ресурсов, то есть их можно рассматривать как его приближения “снизу”. В этом разделе мы рассмотрим способ приближения подобия ресурсов “сверху”, то есть при помощи расширения.

2.3.1. Определение расслоенного подобия ресурсов

Определение 2.6. *Пусть $n \in Nat$ — целое неотрицательное число, $N = (P, T, F, l)$ — помеченная сеть Петри, $(\sim_n)$ — n-расслоенная бисимуляция разметок сети N. Через $(\approx_n)$ обозначим множество пар ресурсов, задающее на $(\sim_n)$ отношение*

подобия:

$$(\approx_n) =_{def} \{\ (r, s) \in \mathcal{M}(P) \times \mathcal{M}(P)\ |$$

$$\forall M \in \mathcal{M}(P)\ (M + r, M + s) \in (\sim_n)\}.$$

Соответствующее отношение назовем n-расслоенным подобием ресурсов.

Для n-расслоенного подобия ресурсов верны аналоги утверждения 2.5 и следствия 2.2 (доказываются аналогично).

Ключевым отличием от уже рассмотренных ранее отношений на множестве ресурсов является тот факт, что при увеличении параметра n множество пар не увеличивается, а уменьшается:

Утверждение 2.8. *Для любого* $n \in Nat$, $n > 0$, *выполняется*

$$r \approx_{n+1} s \implies r \approx_n s.$$

Другими словами, $(\approx_{n+1}) \subseteq (\approx_n)$.

Доказательство: Из определений. □

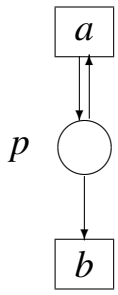
Кроме того, n-расслоенное подобие ресурсов не содержится в полном подобии, а содержит его в себе:

Утверждение 2.9. *Для любого* $n \in Nat$ *выполняется* $(\approx) \subseteq (\approx_n)$.

Доказательство: Следует из определений и свойства $(\sim) = (\sim_\infty) \subseteq (\sim_n)$. □

Предел у последовательности существует:

Утверждение 2.10. $(\approx) = (\approx_\infty)$.



Обозначим $I = \{(\emptyset, \emptyset), (p, p)\}$, тогда:

$$(\approx) = I^{AT}$$
$$(\approx_0) = (I \cup \{(p, \emptyset), (\emptyset, p)\})^{AT}$$
$$(\approx_1) = (I \cup \{(2p, p), (p, 2p)\})^{AT}$$
$$\cdots$$
$$(\approx_n) = (I \cup \{((n+1)p, np), (np, (n+1)p)\})^{AT}$$

Рис. 2.7. Пример сети, в которой для любого $n \in Nat$ выполняется $(\approx) \subset (\approx_n)$

Доказательство: Так как $(\sim) = (\sim_\infty)$. □

То есть подобие ресурсов — предел расслоенного подобия ресурсов (как и в случае ограниченного подобия). Однако, в отличие от ограниченного подобия, последовательность не всегда стабилизируется:

Замечание 2.1. *Существует сеть Петри $N = (P, T, F, l)$, такая что для любого $n \in Nat$ выполняется $(\approx) \subset (\approx_n)$. В качестве примера можно рассмотреть сеть на рисунке 2.7.*

2.3.2. Свойства расслоенного подобия ресурсов

Рассмотрим некоторые свойства расслоенного подобия ресурсов.

Утверждение 2.11. *$r \approx_0 s$ для любых r и s.*

Доказательство: Из определений. □

Пусть $M \in \mathcal{M}(P)$. Через $Act(M)$ обозначим множество меток переходов, готовых к срабатыванию:

$$Act(M) =_{def} \{a \in Act \mid \exists t \in T \ : \ (M \xrightarrow{t} M' \ \wedge \ l(t) = a)\}.$$

Утверждение 2.12.

$$r \approx_1 s \iff \forall m \in \mathcal{M}(P) \ : \ Act(r + m) = Act(s + m).$$

Доказательство: Из определений. □

Другими словами, для 1-расслоенного подобия двух ресурсов необходимо и достаточно того, чтобы множества активизируемых ими переходов обладали одинаковыми наборами меток. Следующее утверждение дает несколько более конструктивное толкование этого же свойства:

Утверждение 2.13. $r \approx_1 s \iff \forall t \in T$

1. $\exists t_1 \in T \ : \ l(t_1) = l(t) \ \& \ {}^\bullet t_1 \subseteq ({}^\bullet t \setminus r + s)$;

2. $\exists t_2 \in T \ : \ l(t_2) = l(t) \ \& \ {}^\bullet t_2 \subseteq ({}^\bullet t \setminus s + r)$.

Условия вида $({}^\bullet t \setminus r + s) \subseteq {}^\bullet t_1$ фактически означают возможность срабатывания перехода t_1.

Доказательство: ($\Longrightarrow$) Докажем первое утверждение правой части (второе аналогично).

Предположим противное: найдется t, такой, что для любого t_1 с той же меткой ${}^\bullet t_1 \not\subseteq ({}^\bullet t \setminus r + s)$.

Рассмотрим разметку $r \cup {}^\bullet t$. Из $r \approx_1 s$ следует

$$r \cup {}^\bullet t \sim_1 r \cup {}^\bullet t - r + s = ({}^\bullet t \setminus r + r) - r + s = {}^\bullet t \setminus r + s.$$

Из 1-бисимулярности $r \cup {}^\bullet t$ и ${}^\bullet t \setminus r + s$ следует существование имитирующего перехода u, такого, что ${}^\bullet u \subseteq ({}^\bullet t \setminus r + s)$ — противоречие.

($\Longleftarrow$)

Предположим противное: $r \not\approx 1s$, то есть найдется ресурс x, такой, что $r + x \not\sim_1 s + x$. Небисимулярность означает невозможность симулирования разметки $r + x$ разметкой $s + x$ или разметки $s + x$ разметкой $r + x$. Рассмотрим первую ситуацию (для второй аналогично):

Существует переход $r + x \xrightarrow{t} M'$, для которого не найдется перехода u с той же меткой, такого, что $s + x \xrightarrow{u} M''$ при $M' \sim_0 M''$.

Поскольку выполняется $M' \sim_0 M''$ для любых разметок, небисимулярность означает отсутствие перехода с меткой $l(t)$, активного при разметке $s + x$.

Имеем $r + x \xrightarrow{t}$, т.е. $r + x = {}^\bullet t + y$, следовательно, $x = {}^\bullet t \setminus r + z$. Подставляя получившееся выражение для x в $s + x$, получим

$$s + x = s + {}^\bullet t \setminus r + z.$$

Из первого утверждения правой части следует существование имитирующего перехода t_1, такого, что $l(t_1) = l(t)$ и при этом ${}^\bullet t_1 \subseteq ({}^\bullet t \setminus r + s)$ — противоречие. □

Определение 2.7. *Пусть $\sigma \in T^*$ — некоторая последовательность переходов сети N. Обозначим через ${}^\bullet\sigma$ минимальную относительно вложения разметку, при которой последовательность σ может сработать.*

Обозначим через $Mark(n)$ минимальную относительно вложения разметку, при которой могут сработать все последовательности длины n :

$$Mark(n) =_{def} \bigcup_{\forall \sigma \in T^* \,:\, |\sigma| = n} {}^\bullet\sigma.$$

Очевидно, что $Mark(n) \subseteq Mark(n + 1)$.

Утверждение 2.14. *Верны следующие свойства $Mark(n)$* :

1. $r \sim_n s \implies (r \cap Mark(n)) \sim_n (s \cap Mark(n))$;

2. $r \approx_n s \implies (r \cap Mark(n)) \approx_n (s \cap Mark(n))$.

Другими словами, если разметки (ресурсы) n-расслоенно бисимулярны (подобны) и при этом в каких-то позициях содержат фишек больше, чем $Mark(n)$, то лишние фишки можно убрать, и при этом бисимулярность (подобие) не нарушится.

Доказательство: Так как учитываются только последовательности переходов длины не более чем n, избыточные фишки не влияют ни на бисимуляцию, ни на подобие. □

Таким образом, для построения n-расслоенного подобия достаточно перебрать только ресурсы, не превышающие по размеру $Mark(n)$. Их число конечно, следовательно,

Теорема 2.4. *n-расслоенное подобие ресурсов вычислимо.*

В качестве доказательства приведем следующий алгоритм.

Алгоритм 2.2. *(построения n-расслоенного подобия ресурсов)*

ввод: *Помеченная сеть Петри* $N = (P, T, F, l)$*, параметр* $n \in Nat$.

вывод: *Отношение* $(\approx_n)$.

шаг 1: *Построим мультимножество* $Mark(n)$. *Для этого просмотрим все последовательности переходов длины* n.

шаг 2: *Построим* $Bis(n)$ *— множество пар n-расслоенно бисимулярных разметок, не превышающих* $Mark(n)$. *Для этого просмотрим все пары разметок, не превышающих* $Mark(n)$, *и все готовые сработать последовательности переходов длины не более* n.

шаг 3: *Положим* $X = \{(r, s) \mid r, s \subseteq Mark(n)\}$.

шаг 4: *Удалим из X все пары ресурсов, которые не являются подобными в смысле n-расслоенного подобия. Для проверки одной пары нужно рассмотреть каждую разметку, не превышающую* $Mark(n)$, *прибавить к ней* r *и* s *и выяснить n-расслоенную бисимулярность получившихся двух разметок (то есть перебрать все элементы множества* $Bis(n)$*).*

шаг 5: *Возвратим* $X \cup \{(r, s) \mid Mark(n) \subseteq r \ \& \ Mark(n) \subseteq s\}$ *— искомое множество* $(\approx_n)$. [1]

[1] Заметим, что бесконечное множество $\{(r, s) \mid Mark(n) \subseteq r \ \& \ Mark(n) \subseteq s\}$ однозначно задается конечным мультимножеством $Mark(n)$.

Представленный алгоритм вычисления ($\approx_n$) крайне неэффективен, однако он доказывает принципиальную возможность аппроксимации подобия ресурсов “сверху”.

2.4. Подобие обобщенных ресурсов

Во многих системах заменяемыми являются не только статические ресурсы (моделируемые фишками), но и динамические составляющие процесса — действия и события (моделируемые переходами). Например, в сетях потоков работ (workflow) под переходами зачастую понимаются сотрудники или устройства, выполняющие ту или иную индивидуальную работу [35].

В связи с этим возникают вопросы по поводу их взаимозаменяемости, сравнительной эффективности, избыточности и т.п. Проблема поиска эквивалентных переходов и поведений также важна для поддержки методологии адаптивного управления системой, согласно которой структура сети может изменяться непосредственно в ходе ее функционирования, например, в ответ на изменения внешних условий или же при возникновении внутрисистемных событий (болезнь сотрудника, отказ оборудования, внедрение новых элементов системы и т.п.).

В этом разделе определяется и исследуется отношение подобия на множестве так называемых обобщенных ресурсов. Обобщенный ресурс может содержать не только мультимножество фишек (материальная часть), но и мультимножество переходов (инструментальная часть). Два обобщенных ресурса подобны, если при любой разметке сети мы можем заменить фишки и срабатывания одного на фишки и срабатывания другого, и при этом поведение сети не изменится. Отношение подобия обобщенных ресурсов включает обычное подобие (“подобие материальных ресурсов”), а также позволяет отслеживать многие интересные свойства динамической составляющей системы, например, “сравнительную эффективность” и “эквивалентность при условии”.

2.4.1. Обобщенные ресурсы сети Петри

Определение 2.8. *Пусть* $N = (P, T, F, l)$ *— помеченная сеть Петри. Пара* (r, α), *где* $r \in \mathcal{M}(P), \alpha \in \mathcal{M}(T)$ *и* ${}^\bullet\alpha \subseteq r$, *называется* обобщенным ресурсом *сети* N.

Множество всех обобщенных ресурсов помеченной сети Петри N *обозначим как* $\Phi(N)$.

Другими словами, обобщенный ресурс можно рассматривать как мультимножество над множеством $P \cup T$ вершин графа сети Петри. Мы используем запись (r, α), поскольку из соображений синтаксиса более удобно явно разделять позиции и переходы.

Итак, обобщенный ресурс (r, α) включает в себя две составляющие — материальный ресурс r и инструментальный ресурс α. Первый показывает наличие в системе "вещественных", статичных элементов ("ресурсов" в обычном понимании этого слова), второй — динамическую составляющую процесса, то есть выполнение в данный момент тех или иных действий. Поскольку мы рассматриваем мультимножества, и материалы и инструменты могут обладать количеством.

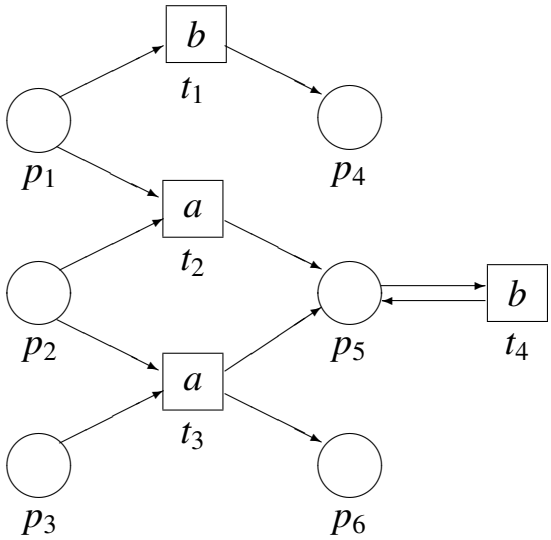
Условие "правильности" ${}^\bullet\alpha \subseteq r$ — естественное требование, которое гарантирует обеспеченность выполняемых действий необходимыми материальными ресурсами.

Обобщенные ресурсы также могут быть взаимозаменяемыми:

Определение 2.9. *Обобщенные ресурсы* (r, α) *и* (s, β) *называются* подобными *(обозначается* $(r, \alpha) \approx (s, \beta)$*), если*

1. $l(\alpha) = l(\beta)$;
2. *для любой разметки* $M \in \mathcal{M}(P)$ *и параллельного срабатывания* $M + r \xrightarrow{\alpha} M'$ *возможно параллельное срабатывание* $M + s \xrightarrow{\beta} M''$, *где* $M' \sim M''$.

Таким образом, если два обобщенных ресурса подобны, то мы можем без каких-либо последствий для наблюдаемого поведения процесса заменять один из



$$(p_1 + p_2 + p_6, t_2) \approx (p_2 + p_3, t_3)$$

$$(p_1, t_1) \approx (p_1 + p_4, t_1)$$

$$(p_5, t_4) \approx (2p_5, t_4)$$

$$(p_4, \emptyset) \approx (p_6, \emptyset)$$

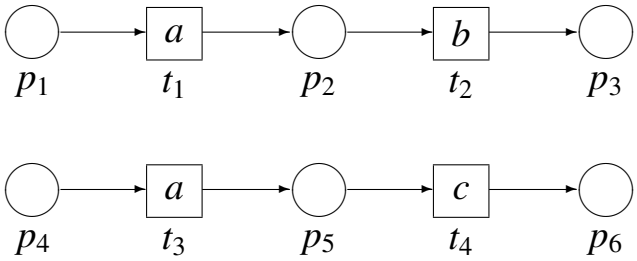
Рис. 2.8. Примеры подобных обобщенных ресурсов

них на другой. При этом замена материальной (статической) составляющей состоит в простом переносе соответствующих фишек, а замена инструментальной (динамической) составляющей означает “отмену” срабатывания первого набора переходов и запуск всех переходов из второго набора.

Несколько примеров подобных обобщенных ресурсов приведены на рисунке 2.8.

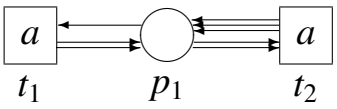
Подобие обобщенных ресурсов обладает естественной интерпретацией. Например, оно способно выразить возможность замены одного работника/подрядчика (выполняющего некоторую работу, описанную мультимножеством переходов в сети) на другого (другое мультимножество переходов), но только при условии одновременной замены мультимножества фишек в позициях сети (сырья на складе, денег на банковских счетах сотрудников) на другое мультимножество. При этом деньги и сырье — это материальный ресурс, а работник — инструментальный. Различные сотрудники потребляют и производят различные материальные ресурсы.

Заметим, что подобная “замена” не означает фактической замены переходов



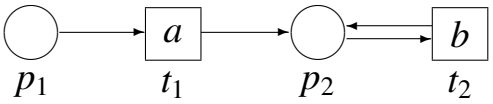
$$(p_1 + p_5, t_1) \approx (p_2 + p_4, t_3)$$
$$(p_3, \emptyset) \approx (p_6, \emptyset)$$

а) взаимозаменяемые ресурсы;



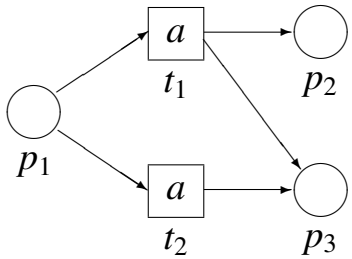
$$(p_1, t_1) \not\approx (p_1, t_2)$$
$$(2p_1, t_1) \approx (2p_1, t_2)$$

б) эквивалентные инструменты;



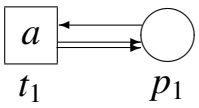
$$(p_1, \emptyset) \not\approx (p_1 + p_2, \emptyset)$$
$$(p_1, t_1) \approx (p_1 + p_2, t_1)$$

в) эквивалентные материалы;



$$(p_1, t_1) \approx (p_1 + p_2, t_2)$$

г) более эффективный инструмент;



$$(p_1, t_1) \approx (2p_1, t_1)$$
$$(p_1, \emptyset) \approx (2p_1, \emptyset)$$
$$(\emptyset, \emptyset) \not\approx (p_1, \emptyset)$$

д) избыточные материалы.

Рис. 2.9. Свойства системы, выявляемые при помощи подобия обобщенных ресурсов

в графе сети. Структура графа остается той же. Мы заменяем *срабатывания* переходов в самом процессе (поведении сети). Образно говоря, первый работник не увольняется, а лишь заменяется вторым для выполнения данной конкретной работы. В дальнейшем он опять может быть привлечен для ее выполнения.

При помощи подобия обобщенных ресурсов можно выразить ряд дополнительных эквивалентностных свойств. Некоторые из них показаны на рисунке 2.9.

$(r,\alpha) \approx (s,\beta)$	Обобщенные ресурсы (r,α) и (s,β) **взаимозаменяемы** в любом состоянии системы.
$(r,\alpha) \approx (r,\beta)$	Действия (инструменты) α и β **эквивалентны** при наличии материала r.
$(r,\alpha) \approx (s,\alpha)$	Материалы r и s **эквивалентны при условии** выполнения действия α.
$(r,\alpha) \approx (r+s,\beta)$	Действие (инструмент) α **эффективнее,** чем действие β.
$(r,\alpha) \approx (r+s,\alpha)$	Материал s **избыточен при условии** выполнения действия α.
$(r,\emptyset) \approx (r+s,\emptyset)$	Материал s **избыточен.**

2.4.2. Свойства подобия обобщенных ресурсов

Подобие обобщенных ресурсов в сетях Петри обладает рядом интересных свойств. Прежде всего, это отношение эквивалентности:

Утверждение 2.15. *Пусть* $(r,\alpha),(s,\beta),(u,\gamma) \in \Phi(N)$. *Тогда*

1. $(r,\alpha) \approx (r,\alpha)$;

2. $(r,\alpha) \approx (s,\beta) \;\Rightarrow\; (s,\beta) \approx (r,\alpha)$;

3. $(r,\alpha) \approx (s,\beta)\ \&\ (s,\beta) \approx (u,\gamma) \;\Rightarrow\; (r,\alpha) \approx (u,\gamma)$.

Доказательство: 1) Из определения.

2) Поскольку наибольшая бисимуляция разметок $\sim$ замкнута относительно симметричности.

3) Поскольку наибольшая бисимуляция разметок $\sim$ замкнута относительно транзитивности. □

Простейшие нетривиальные пары подобных обобщенных ресурсов можно получить, используя следующий факт:

Утверждение 2.16. *Пусть* $\alpha, \beta \in \mathcal{M}(T)$. *Тогда*

$$l(\alpha) = l(\beta) \quad \Rightarrow \quad ({}^\bullet\alpha + \beta^\bullet, \alpha) \approx ({}^\bullet\beta + \alpha^\bullet, \beta).$$

Доказательство: Из определения. □

Подобие обобщенных ресурсов замкнуто относительно "вычитания параллельного шага":

Утверждение 2.17. *Пусть* $(r, \alpha), (s, \beta) \in \Phi(N),\ \gamma, \delta \in \mathcal{M}(T)$. *Тогда*

$$(r, \alpha) \approx (s, \beta)\ \&\ l(\gamma) = l(\delta)\ \&\ \gamma \subseteq \alpha\ \&\ \delta \subseteq \beta \quad \Rightarrow$$
$$\Rightarrow \quad (r - {}^\bullet\gamma + \gamma^\bullet, \alpha - \gamma) \approx (s - {}^\bullet\delta + \delta^\bullet, \beta - \delta).$$

Доказательство: Предположим противное:

$$(r - {}^\bullet\gamma + \gamma^\bullet, \alpha - \gamma) \not\approx (s - {}^\bullet\delta + \delta^\bullet, \beta - \delta).$$

Во-первых, заметим, что $(r - {}^\bullet\gamma + \gamma^\bullet, \alpha - \gamma)$ и $(s - {}^\bullet\delta + \delta^\bullet, \beta - \delta)$ являются "правильными" обобщенными ресурсами, поскольку ${}^\bullet(\alpha - \gamma) \subseteq (r - {}^\bullet\gamma + \gamma^\bullet)$ и ${}^\bullet(\beta - \delta) \subseteq (s - {}^\bullet\delta + \delta^\bullet)$. Из $(r, \alpha) \approx (s, \beta)$ и $l(\gamma) = l(\delta)$ получим $l(\alpha - \gamma) = l(\beta - \delta)$.

Следовательно, мультимножества переходов $\alpha - \gamma$ и $\beta - \delta$ могут сработать параллельно при наличии в разметке соответствующих материальных ресурсов.

Таким образом, из нашего предположения следует, что для некоторой разметки $M \in \mathcal{M}(P)$ и параллельных шагов

$$M + (r - {}^\bullet\gamma + \gamma^\bullet) \stackrel{\alpha-\gamma}{\rightarrow} M' \tag{2.1}$$

и

$$M + (s - {}^\bullet\delta + \delta^\bullet) \stackrel{\beta-\delta}{\rightarrow} M'' \tag{2.2}$$

выполняется $M' \nsim M''$.

Рассмотрим α и r. Из условия $\gamma \subseteq \alpha$ имеем $\alpha = \gamma + \gamma'$. Из условий ${}^\bullet\alpha \subseteq r$ и $\gamma \subseteq \alpha$ имеем

$$r = {}^\bullet\gamma + {}^\bullet\gamma' + r'. \tag{2.3}$$

Применив (2.3), мы можем переписать шаг (2.1) в виде

$$M + (r - {}^\bullet\gamma + \gamma^\bullet) = M + \gamma^\bullet + {}^\bullet\gamma' + r' \xrightarrow{\gamma'} M + \gamma^\bullet + \gamma'^\bullet + r'.$$

Следовательно, $M' = M + \gamma^\bullet + \gamma'^\bullet + r'$.

Аналогично получим $\beta = \delta + \delta'$, $s = {}^\bullet\delta + {}^\bullet\delta' + s'$ для некоторого s' и $M'' = M + \delta^\bullet + \delta'^\bullet + s'$.

Из подобия обобщенных ресурсов $(r, \alpha) \approx (s, \beta)$ следует

$$M + r \xrightarrow{\alpha} G', \quad M + s \xrightarrow{\beta} G'', \quad G' \sim G''. \tag{2.4}$$

Применив (2.3), мы можем переписать шаг $M + r \xrightarrow{\alpha} G'$ в виде

$$M + r = M + {}^\bullet\gamma + {}^\bullet\gamma' + r' \xrightarrow{\gamma+\gamma'} G' = M + \gamma^\bullet + \gamma'^\bullet + r'.$$

Следовательно, $G' = M'$. Аналогично $G'' = M''$. Таким образом, мы получили противоречие между (2.4) и предположением $M' \nsim M''$. □

Заметим, что представленная операция не является полноценным вычитанием ресурсов, поскольку мы вычитаем только инструментальную часть ресурса. Материальная часть не вычитается, а трансформируется в соответствии со свойствами шага.

Подобие обобщенных ресурсов замкнуто относительно “прибавления параллельного шага”:

Утверждение 2.18. *Пусть* $(r, \alpha), (s, \beta) \in \Phi(N)$, $\gamma, \delta \in \mathcal{M}(T)$. *Тогда*

$$(r, \alpha) \approx (s, \beta) \;\&\; l(\gamma) = l(\delta) \;\&\; \gamma^\bullet \subseteq (r - {}^\bullet\alpha) \;\&\; \delta^\bullet \subseteq (s - {}^\bullet\beta) \quad \Rightarrow$$
$$\Rightarrow \quad (r - \gamma^\bullet + {}^\bullet\gamma, \alpha + \gamma) \approx (s - \delta^\bullet + {}^\bullet\delta, \beta + \delta).$$

Доказательство: Аналогично доказательству утверждения 2.17 о замкнутости подобия обобщенных ресурсов относительно “вычитания параллельного шага”. □

Это также не вполне сложение. Мы прибавляем только инструментальную компоненту, а материальная компонента трансформируется в зависимости от изменения инструментальной.

Однако мы можем складывать и обе компоненты одновременно. Подобие обобщенных ресурсов замкнуто относительно прибавления пар ресурсов:

Утверждение 2.19. *Пусть* $(r,\alpha),(s,\beta),(u,\gamma),(v,\delta) \in \Phi(N)$. *Тогда*

$$(r,\alpha) \approx (s,\beta)\ \&\ (u,\gamma) \approx (v,\delta)\ \Rightarrow\ (r+u,\alpha+\gamma) \approx (s+v,\beta+\delta).$$

Доказательство: Предположим противное:

$$(r+u,\alpha+\gamma) \not\approx (s+v,\beta+\delta).$$

Рассуждая так же, как в начале доказательства утверждения 2.17, получим, что для некоторой разметки $M \in \mathcal{M}(P)$ и параллельных шагов $M + r + u \xrightarrow{\alpha+\gamma} M'$ и $M + s + v \xrightarrow{\beta+\delta} M''$ выполняется $M' \nsim M''$.

Рассмотрим (r,α). Поскольку ${}^\bullet\alpha \subseteq r$, мы имеем

$$r \xrightarrow{\alpha} (r - {}^\bullet\alpha + \alpha^\bullet).$$

Обозначим мультимножество позиций $r - {}^\bullet\alpha + \alpha^\bullet$ символом r'. Аналогично, обозначим “результирующие материальные ресурсы” обобщенных ресурсов $(s,\beta),(u,\gamma)$ и (v,δ) символами s', u' и v' :

$$r \xrightarrow{\alpha} r', \qquad s \xrightarrow{\beta} s', \qquad u \xrightarrow{\gamma} u', \qquad v \xrightarrow{\delta} v'.$$

Очевидно, мы имеем $M' = M + r' + u'$ и $M'' = M + s' + v'$, и, следовательно, предположение может быть записано как:

$$M + r' + u' \not\sim M + s' + v'. \tag{2.5}$$

Рассмотрим $G = M + r'$. Из $(u, \gamma) \approx (v, \delta)$ имеем

$$G + u \xrightarrow{\gamma} G + u', \quad G + v \xrightarrow{\delta} G + v', \quad G + u' \sim G + v'. \tag{2.6}$$

Рассмотрим $H = M + v'$. Из $(r, \alpha) \approx (s, \beta)$ имеем

$$H + r \xrightarrow{\alpha} H + r', \quad H + s \xrightarrow{\beta} H + s', \quad H + r' \sim H + s'. \tag{2.7}$$

Бисимулярности (2.6) и (2.7) могут быть записаны как:

$$(M + r') + u' \sim (M + r') + v', \qquad (M + v') + r' \sim (M + v') + s'.$$

Поскольку наибольшая бисимуляция разметок $\sim$ замкнута относительно транзитивности, а сложение мультимножеств коммутативно и ассоциативно, имеем

$$M + r' + u' \sim M + s' + v',$$

что противоречит предположению (2.5). □

Замечание 2.2. *Подобие обобщенных ресурсов не замкнуто относительно вычитания пар ресурсов. В частности, это обусловлено тем, что вычитание может нарушить "правильность" ресурса.*

Следствие 2.4. *Подобие обобщенных ресурсов обладает конечным AT-базисом.*

Доказательство: Следует из аддитивной и транзитивной замкнутости отношения. □

Определим частичный порядок $\sqsubseteq$ на множестве $R \subseteq \Phi(N) \times \Phi(N)$ пар обобщенных ресурсов как "обобщение" случая $\mathcal{M}(P) \times \mathcal{M}(P)$:

$$\Big((r, \alpha), (s, \beta)\Big) \sqsubseteq \Big((u, \gamma), (v, \delta)\Big) \overset{def}{\Leftrightarrow} (r, s) \sqsubseteq (u, v) \;\&\; (\alpha, \beta) \sqsubseteq (\gamma, \delta).$$

Аналогично, через R_s обозначим множество всех минимальных относительно $\sqsubseteq$ элементов R, которое будем называть *основным базисом* отношения R.

Следствие 2.5. *Подобие обобщенных ресурсов обладает конечным основным базисом.*

2.4.3. Материальные и инструментальные ресурсы

Можно выделить два интересных специальных вида обобщенных ресурсов.

Определение 2.10. *Обобщенный ресурс вида* $(r, \emptyset)$ *называется* материальным ресурсом.

Обобщенный ресурс вида $({}^\bullet\alpha, \alpha)$ *называется* инструментальным ресурсом.

Рассматривая только пары подобных материальных ресурсов, мы получим отношение эквивалентности — *подобие материальных ресурсов.*

Легко видеть, что подобие материальных ресурсов совпадает с подобием ресурсов (с точностью до обозначений).

Утверждение 2.20. *Пусть* $r, s \in \mathcal{M}(P)$. *Тогда*

$$r \approx s \quad \Leftrightarrow \quad (r, \emptyset) \approx (s, \emptyset).$$

Доказательство: Из определений. □

Из неразрешимости подобия ресурсов (следствие 1.4) и утверждения 2.20 получаем

Следствие 2.6. *Подобие материальных ресурсов неразрешимо.*

Следствие 2.7. *Подобие обобщенных ресурсов неразрешимо.*

Рассматривая только пары подобных инструментальных ресурсов, мы получим *подобие инструментальных ресурсов.* Оказывается, оно менее выразительно, чем подобие материальных ресурсов, так как полностью определяется его специальным подмножеством:

Утверждение 2.21. *Пусть* $\alpha, \beta \in \mathcal{M}(T)$. *Тогда*

$$({}^\bullet\alpha, \alpha) \approx ({}^\bullet\beta, \beta) \quad \Leftrightarrow \quad (\alpha^\bullet, \emptyset) \approx (\beta^\bullet, \emptyset).$$

Доказательство:

($\Rightarrow$) Из утверждения 2.17.

($\Leftarrow$) Из утверждения 2.18. □

Это довольно естественно — мы можем заменять “инструменты” без дополнительных условий тогда и только тогда, когда они “полностью” подобны, то есть производят эквивалентные материальные ресурсы.

2.5. Адаптивное управление процессами на основе подобия ресурсов

Другая важная область применения отношений эквивалентности ресурсов — адаптивное управление.

На практике в ходе эксплуатации автоматизированных систем управления (в частности, систем управления потоками работ — Workflow Management Systems [35]) в процесс зачастую приходится вносить изменения непосредственно в ходе его выполнения. Это может происходить по причине доступности/недоступности тех или иных ресурсов, сбоев в отдельных модулях, в силу необходимости обрабатывать нестандартные ситуации/запросы и по многим другим причинам. Перенастройка системы вручную в таких ситуациях может быть сделана только специалистом, хорошо знающим все детали процесса и архитектуру системы, и является, таким образом, очень трудоемкой и затратной процедурой. К тому же внесение изменений в систему может приводить к появлению ошибок в ее функционировании.

В связи с этим необходимо уметь автоматически находить эквивалентные замены для отказавших элементов системы. Можно рассматривать различные критерии эквивалентности ресурсов, в частности, бисимуляцию ресурсов [67] или подобие ресурсов и отношения на его основе.

2.5.1. Управление “без потерь” на основе подобия обобщенных ресурсов

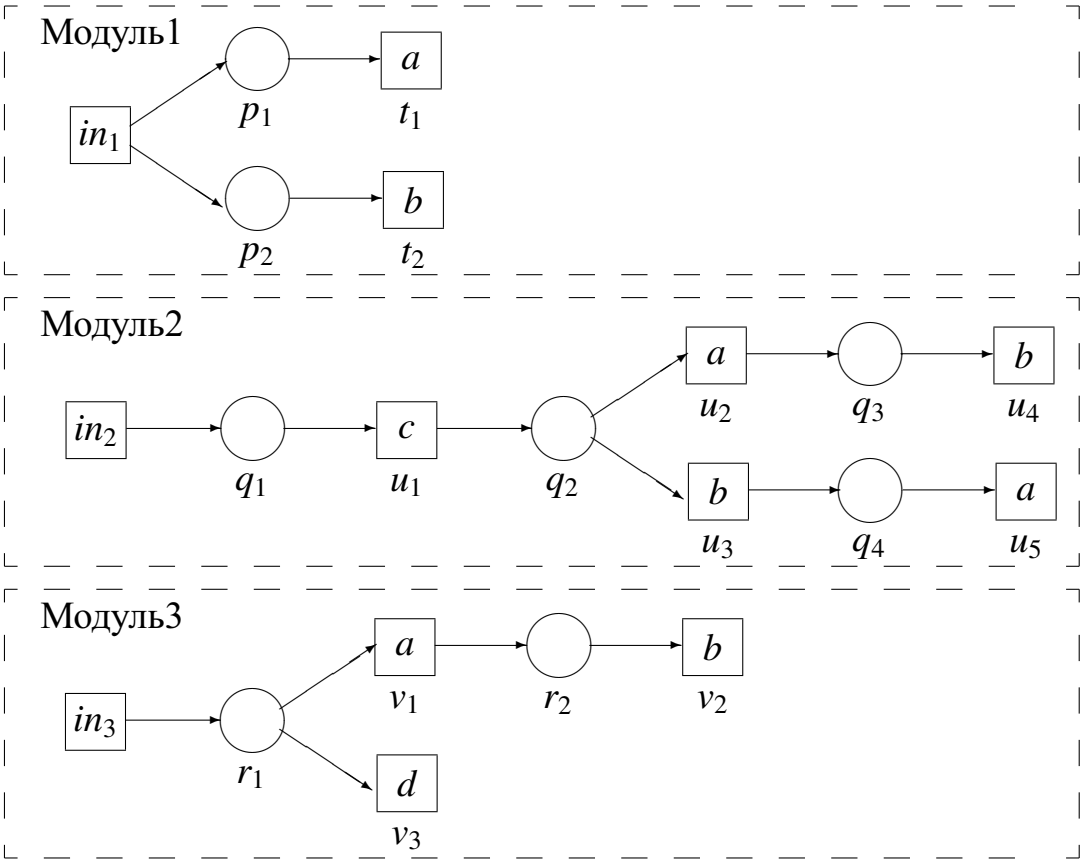
Рассмотрим небольшой пример, показывающий один из возможных способов практического использования подобия обобщенных ресурсов для адаптивного управления процессами.

На рисунке 2.10 изображена модель некоей довольно абстрактной системы обработки запросов. Это может быть организация, система web-сервисов или вычислительное устройство. Система состоит из трех отдельных модулей (подразделений, серверов или процессоров). Каждый из них умеет обрабатывать запросы только одного определенного типа. Алгоритмы обработки различных типов запросов в целом различны, однако некоторые из операций совпадают. А именно, в каждом модуле выполняются операции “a” и “b”. Поэтому мы хотим знать, каким образом эта похожесть модулей может быть использовано для *адаптивной обработки запросов* — обмена запросами (ресурсами) между модулями в целях уменьшения общей загрузки системы и поддержания ее работоспособности в случае отказа отдельных подсистем. При этом мы хотим сохранять наблюдаемое поведение системы.

Для решения таких задач интересные возможности предоставляет подобие обобщенных ресурсов. Например, в представленной модели системы обработки запросов можно выделить целый ряд нетривиальных подобных ресурсов.

Подобие материальных ресурсов (“обычное” подобие ресурсов) дает удобный набор правил для управления загрузкой подсистем. Например, используя правило $(q_2, \emptyset) \approx (p_1 + p_2, \emptyset)$, мы можем уменьшить загруженность подсистемы 2 (ценой увеличения загруженности подсистемы 1), удаляя фишку из q_2 и помещая фишки в p_1 и p_2. (Для простоты мы не вдаемся здесь в технические подробности процесса переноса запроса между подсистемами.)

Управление загрузкой обычно производится под воздействием каких-то внешних факторов и причин. В свою очередь, обработка исключительных ситуаций чаще всего имеет дело с внутренними проблемами системы. Рассмотрим



$$(p_1, \emptyset) \approx (q_4, \emptyset),\ (p_2, \emptyset) \approx (q_3, \emptyset) \approx (r_2, \emptyset)$$

$$(p_1 + p_2, \emptyset) \approx (q_2, \emptyset) \approx (q_3 + q_4, \emptyset)$$

$$(p_1, t_1) \approx (q_4, u_5),\ (p_2, t_2) \approx (q_3, u_4) \approx (r_2, v_2)$$

$$(p_1 + p_2, t_1) \approx (q_2, u_2) \approx (q_2 + q_3, u_5) \approx (r_1, v_1)$$

$$(p_1 + p_2, t_2) \approx (q_2, u_3) \approx (q_2 + q_3, u_4)$$

Рис. 2.10. Адаптивная обработка запросов

ситуацию, при которой в подсистеме 3 происходит сбой в момент выполнения задачи v_1 (или же эта задача просто выполняется слишком долго, в то время как другие — возможно, более мощные — подсистемы простаивают). В такой ситуации правило подобия $(r_1, v_1) \approx (p_1 + p_2, t_1) \approx (q_2, u_2)$ позволяет нам перенести всю входную информацию из подсистемы 3 в другую подсистему (1 или 2), а затем (незаметно для пользователя) рестартовать задачу “a”. Такая манипуляция не является традиционным откатом транзакции (rollback), поскольку мы запоминаем не только состояние системы (разметку), но и множество выбранных действий (переходов).

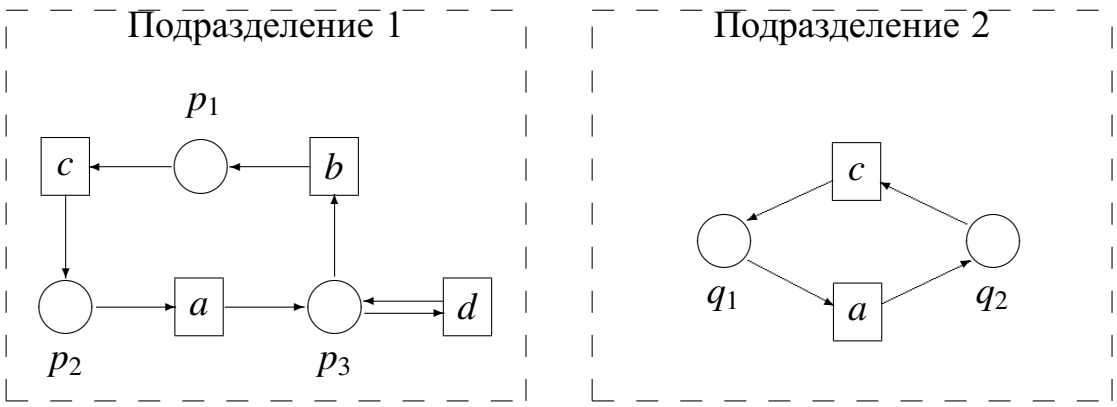
Заметим, что изложенный выше подход позволяет изменять структуру системы без каких-либо потерь в её поведении (относительно бисимуляции).

2.5.2. Управление в условиях ограниченного времени на основе расслоенного подобия

Ещё один вид адаптивного управления — управление “с потерями”. Дается гарантия на сохранение поведения только до определенного времени (шага). В дальнейшем системе позволяется изменить свое поведение по сравнению с эталонным. Такое управление может потребоваться при обработке каких-либо кризисных ситуаций, когда нужно быстро и правильно выполнить срочные (тактические) действия, и при этом не так важно, будет ли в будущем соблюдена долговременная стратегия (или есть возможность через несколько шагов опять изменить структуру процесса).

Рассмотрим пример на рисунке 2.11. Здесь представлены два циклических бизнес-процесса (потока работ) некоей абстрактной фирмы. За каждый процесс отвечает конкретное подразделение, которое имеет жесткую структуру и поэтому может функционировать только по раз и навсегда определенному алгоритму.

Очевидно, что подразделения фирмы не являются взаимозаменяемыми. То есть в случае проблем в одном из процессов (на любой его стадии) мы не мо-



$$p_1 \approx_1 q_2,\ p_2 \approx_1 q_1$$
$$p_1 \approx_2 q_2$$

Рис. 2.11. Тактическое адаптивное управление

жем “увести” его в другое подразделение “навсегда”. В этом ключевое отличие от примера из предыдущего параграфа — там некоторые состояния различных модулей были полностью эквивалентными. Однако *на время* подразделения все-таки могут обмениваться друг с другом работой. Действительно, из рисунка видно, что на один “такт” мы можем перенести задание из состояния p_2 первого подразделения в состояние q_1 второго (и наоборот).

Для решения задач тактического адаптивного управления бизнес-процессами удобнее всего использовать расслоенное подобие ресурсов. Это отношение позволяет адекватно учесть границы временного интервала, кроме того, оно разрешимо (то есть, в отличие от обычного подобия, здесь не требуется аппроксимация).

Вычислив расслоенные подобия для сети Петри на рисунке 2.11, мы можем сделать вывод о допустимости следующих переносов заданий:

- На один такт: между p_1 и q_2, между p_2 и q_1.
- На два такта: между p_1 и q_2.

Необходимо заметить, что в данном случае под квантами “времени” понимаются срабатывания переходов сети, а не реальные временные интервалы.

Глава 3

Некоторые методы анализа сетей с одномерным ресурсом

Данная глава посвящена исследованию систем, содержащих один потенциально неограниченный ресурс, моделируемый целочисленным неотрицательным счетчиком. Примерами могут служить “заявки” или “исполнители” в схемах потоков работ (workflow), “пакеты” или “задержки” в моделях сетевых протоколов, “финансы” в схемах бизнес-процессов и т.д.

Односчетчиковые сети, известные также как одномерные системы векторного сложения с состояниями (1-dim Vector Addition Systems with States — VASS), эквивалентны сетям Петри с одной неограниченной позицией, а также магазинным автоматам с односимвольным стековым алфавитом. Это достаточно известная модель вычислений, которую часто рассматривают как одну из самых слабых среди систем с бесконечным множеством состояний. Ограничение на число счетчиков делает данный формализм менее выразительными, чем обыкновенные сети Петри, однако существенно облегчает анализ. Проблемы достижимости, верификации формул различных темпоральных логик, подобия, бисимуляции и других поведенческих эквивалентностей для данного класса моделей рассматривались, среди прочих, в работах [58, 131, 132, 143, 156, 157, 191].

Максимальное допустимое число неограниченных счетчиков — важный параметр, который порождает несколько содержательных иерархий классов в теории сетей Петри. Он позволяет установить ряд интересных границ сложности и разрешимости. Например, односчетчиковые сети являются наибольшим классом счетчиковых сетей с разрешимой бисимулярностью [142], двухсчетчиковые сети — наибольший класс с полулинейной достижимостью [138]. В более выразительном случае (автоматы с проверкой на ноль) односчетчиковые системы — наибольший класс с разрешимой эквивалентностью языков [199]. В [109] было

доказано, что любой недетерминированный конечный автомат над унарным алфавитом может быть представлен в стандартном виде, называемом нормальной формой Хробака.

Заметим, что конструктивные части всех упомянутых результатов были получены путем обнаружения какой-либо периодичности соответствующего бесконечного множества состояний.

В данной работе мы используем теоретико-числовой метод, основанный на числах Фробениуса, для изучения периодичности достижимых значений счетчика односчетчиковой сети. Этот подход позволяет получить ряд ключевых свойств самого класса систем, а также набор новых символьных инструментов для построения алгоритмов анализа моделей.

3.1. Одномерные полулинейные множества

Дж. Хопкрофт и Ж. Ж. Пансио [138] доказали (в терминах систем векторного сложения), что сети Петри с двумя неограниченными позициями всегда обладают полулинейным множеством достижимости, и, кроме того, привели пример сети с тремя неограниченными позициями, множество достижимости которой неполулинейно.

В данном разделе мы будем исследовать ещё более простой класс — сети Петри с одной неограниченной позицией (односчетчиковые сети). Таким образом, объектом исследования будут простейшие из полулинейных множеств — одномерные.

3.1.1. Полулинейные множества натуральных чисел

Для удобства введём новое обозначение одномерных линейных множеств (линейных множеств натуральных чисел). Пусть $m \subseteq Nat$ линейно, тогда для

некоторого $l \in \mathbf{Z}_+$ выполняется

$$m = Lin\{v, \{w_1, \ldots, w_l\}\} =_{\text{def}} \{v + n_1 w_1 + \ldots + n_l w_l \mid n_1, \ldots, n_l \in Nat\},$$

где $v, w_1, \ldots, w_l \in Nat$ фиксированы.

Множество $m \subseteq Nat$ назовём *ограниченно неполным линейным множеством,* если $m = m' \setminus m''$, где m' — линейное множество, а m'' — конечное множество. Если $m' = Lin\{v, \{w_1, \ldots, w_l\}\}$ и $w \in Nat$ — наибольший элемент m'', то мы обозначаем m как $DLin\{v, w + 1, \{w_1, \ldots, w_l\}\}$.

Отметим, что выражение $DLin\{v, w, E\}$ не является точным описанием множества m — это приближение сверху. Здесь нет точной информации о том, какие именно элементы удалены из исходного линейного множества. Указана лишь граница w между “испорченной” головой множества и бесконечным линейным хвостом.

Множество $Lin\{v, \{w\}\}$ представляет собой *арифметическую прогрессию* с разностью w.

3.1.2. Числа Фробениуса

Рассмотрим решение задачи Фробениуса о размене монет, называемое также числами Фробениуса. Требуется найти число, являющейся крупнейшей денежной суммой, не набираемой монетами указанных номиналов. Например, крупнейшая сумма, которая не может быть получена, используя только монеты в 3 и 5 единиц, составляет 7.

Задачу для двух переменных (двух номиналов монет) решил Сильвестр в [195]:

Факт 3.1. *Для любых взаимно простых натуральных a и b и натурального c, такого что $c \geq (a-1)(b-1)$, диофантово уравнение $ax + by = c$ имеет натуральное решение; при этом уравнение $ax + by = c - 1$ не имеет натурального решения.*

Обобщение задачи Фробениуса для произвольного числа переменных (номиналов монет) до сих пор не имеет точного решения. Насколько нам известно, наилучшим приближением сверху является квадрат наибольшего номинала [100, 117] (см. также исчерпывающий обзор [184]):

Факт 3.2. *Для любых попарно взаимно простых натуральных $a_1, \ldots, a_k$ и натурального c, такого что $c \geq max\{a_1, \ldots, a_k\}^2$, диофантово уравнение $a_1x_1 + \cdots + a_kx_k = c$ имеет решение в натуральных числах.*

3.1.3. Единственность периода бесконечной части

Использование чисел Фробениуса позволяет нам доказать важное свойство одномерных линейных множеств:

Лемма 3.1. *Пусть $m = Lin\{v, \{w_1, w_2\}\}$ — линейное множество с двумя периодами. Тогда m может быть представлено как ограниченно неполное множество с одним периодом.*

Обозначив $p = НОД(w_1, w_2)$ и $b = v + p(\frac{w_1}{p} - 1)(\frac{w_2}{p} - 1)$, получим

$$m = DLin\{v, b, \{p\}\}.$$

Будем говорить, что множество m распадается на "неполную" (в некотором смысле хаотичную) "голову" $m_0 \subseteq \{b - kp \mid k \in \{1, 2, \ldots, (\frac{w_1}{p} - 1)(\frac{w_2}{p} - 1)\}\}$ и простой бесконечный периодический "хвост" $m_\infty = \{b + kp \mid k \in Nat\}$.

Доказательство: Легко заметить, что для любого $z \in m$ найдется $l \in Nat$, такое, что $z = v + lp$.

Нам нужно доказать, что для любого $k \in Nat$ и $z = b + kp$ выполняется $z \in m$, то есть $z - v$ может быть представлено как линейная комбинация w_1 и w_2. Рассмотрим z:

$$z = v + p(\frac{w_1}{p} - 1)(\frac{w_2}{p} - 1) + kp = v + p((\frac{w_1}{p} - 1)(\frac{w_2}{p} - 1) + k)$$

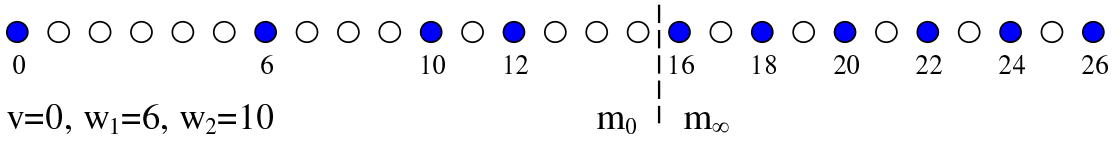


Рис. 3.1. Линейное множество с двумя периодами

Обозначим $z' = (\frac{w_1}{p} - 1)(\frac{w_2}{p} - 1) + k$.

Очевидно, что НОД$(\frac{w_1}{p}, \frac{w_2}{p}) = 1$ и $z' \geq (\frac{w_1}{p} - 1)(\frac{w_2}{p} - 1)$. Из теоремы о числах Фробениуса получим, что $z' = (\frac{w_1}{p})x_1 + (\frac{w_2}{p})x_2$ для некоторых неотрицательных целых x_1 и x_2. Следовательно,

$$z = v + pz' = v + p((\frac{w_1}{p})x_1 + (\frac{w_2}{p})x_2) = v + w_1x_1 + w_2x_2. \qquad \square$$

Итак, любое двухпериодическое линейное множество натуральных чисел является подмножеством некоего однопериодического множества, причем мы можем найти точную верхнюю границу “неполной” части. Пример приведен на Рис. 3.1 — здесь два различных периода (6 и 10) превращаются в один (2), начиная с числа 16 (начало “хвоста”).

Замечание 3.1. *Структура множества m_0 является в некотором смысле “хаотической”. Однако и здесь есть определённая симметрия. Во-первых, заметим, что $(\frac{w_1}{p} - 1)(\frac{w_2}{p} - 1)$ всегда чётно. Это наблюдение приводит нас к следующему интересному факту из теории чисел ([37], олимпиадная задача 3.22):*

Факт 3.3. Пусть $\alpha_1, \alpha_2 \in Nat$ — взаимно простые. Обозначим $c = \alpha_1\alpha_2 - \alpha_1 - \alpha_2$. Мы будем называть число $n \in Nat$ “хорошим”, если существуют неотрицательные целые x_1 и x_2, такие, что $n = \alpha_1x_1 + \alpha_2x_2$, и “плохим” в противном случае. Тогда для любого n, такого, что $n \leq c$, если n — “хорошее”, то $(c - n)$ — “плохое”, и наоборот.

Заметим, что поскольку 0 всегда “хорошее”, с всегда “плохое”. Структура множества m_0 в действительности такая же (при $\alpha_i = \frac{w_i}{p}$ и всех числах, умноженных на p и затем увеличенных на v). Здесь “плохое” число с эквивалентно

$b - p$ *(заметим, что* $b - p = (v + p(\frac{w_1}{p} - 1)(\frac{w_2}{p} - 1)) - p = v + p((\frac{w_1}{p})(\frac{w_2}{p}) - \frac{w_1}{p} - \frac{w_2}{p})$*), то есть* $b - p \notin m_0$*. Это означает, что наша оценка для* b *точна.*

Задача нахождения эффективного аналитического представления для всех элементов m_0 *остаётся открытой (более того, вряд ли такое представление вообще существует). Однако упомянутая выше симметрия по крайней мере позволяет ускорить (в два раза) процесс построения множества:*

$$b - kp \in m_0 \iff v + (k-1)p \notin m_0.$$

Достаточно вычислить только "половину" от m_0 *(то есть* $k \leq \frac{(\frac{w_1}{p}-1)(\frac{w_2}{p}-1)}{2}$*), оставшаяся "половина" может быть получена при помощи симметрии.*

Лемма 3.1 может быть обобщена:

Лемма 3.2. *Пусть* $Lin\{v, \{w_1, \ldots, w_s\}\}$ *— линейное множество с* s *периодами. Тогда* m *может быть представлено как ограниченно неполное множество с одним периодом.*

Обозначив $p = НОД(w_1, \ldots, w_s)$, $c = max\{w_1, \ldots, w_s\}^2$ *и* $b = v + \frac{c}{p}$*, получим*

$$m = DLin\{v, b, \{p\}\}.$$

Доказательство: Применим тот же метод, что и в Лемме 3.1, учитывая известную оценку обобщенных чисел Фробениуса (Факт 3.2). Также отметим, что любое $z \in m$ может быть представлено как $z = v + ip$ для некоторого i. □

Рассмотрим полулинейное множество над *Nat*. Как оказалось, оно тоже обладает единственным "периодом", однако в данном случае это уже не интервал, а вектор.

Лемма 3.3. *Для любых ограниченно неполных линейных множеств с одним периодом* $m' = DLin\{v', b', \{p'\}\}$ *и* $m'' = DLin\{v'', b'', \{p''\}\}$ *полулинейное множество* $m = m' \cup m''$ *распадается на конечное множество и конечное семейство линейных множеств с одинаковым периодом. Обозначив* $p = LCM(p', p'')$

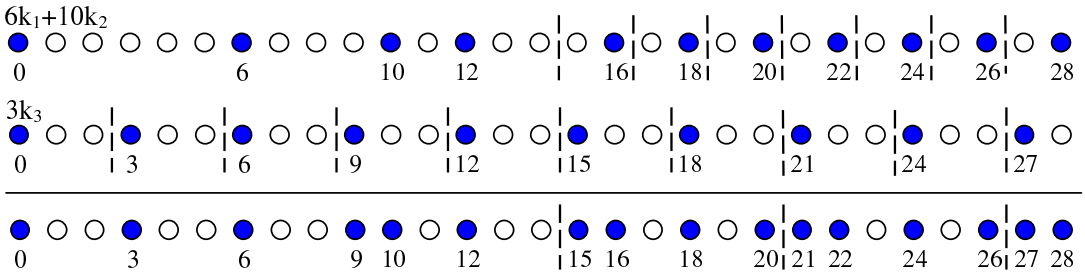


Рис. 3.2. Полулинейное множество, полученное объединением двух линейных

и $b = max\{b', b''\}$, *получим, что существует* характеристическое множество $\Psi \subseteq \{b, b+1, b+2, \ldots, b+(p-1)\}$, *такое, что*

$$m = m_0 \cup m_\infty, \quad \textit{где} \quad m_0 \subseteq \bigcup_{k=1}^{[\frac{b}{p}]} (\Psi \triangleleft kp), \; m_\infty = \bigcup_{k=0}^{\infty} (\Psi \triangleright kp).$$

Рассмотрим двоичный вектор v длины p, такой, что $v[i] = 0$ для $b + i \notin \Psi$ и $v[i] = 1$ для $b + i \in \Psi$. Говоря неформально, мы можем использовать этот *вектор периода* в качестве повторяющегося шаблона при "пометке" всех входящих в m чисел, расположенных "справа" от b. По построению множество m_0 конечно и $m_0 \cap m_\infty = \varnothing$.

Доказательство: Заметим, что множество всех $x \in m' \cup m''$, таких, что $x > max\{b', b''\}$, является периодическим с периодом НОК(p', p''). □

Пример приведен на Рис. 3.2 — здесь два линейных множества начинают "перекрываться" от числа 15 (начало "хвоста"), образуя бесконечную повторяющуюся последовательность "строк" 110101 (в данном случае характеристическое множество $\Psi = \{15, 16, 18, 20\}$).

Дальнейшее обобщение Леммы 3.3 на произвольное число линейных множеств с произвольным числом периодов:

Теорема 3.1. *Любое полулинейное множество* $m \subseteq Nat$ *распадается на конечное множество и конечное семейство линейных множеств с одинаковым периодом: для некоторых* $p, b \in Nat$, *существует* характеристическое множество $\Psi \subseteq \{b, b+$

$1, b+2, \ldots, b+(p-1)\}$, *такое, что*

$$m = m_0 \cup m_\infty, \quad \textit{где} \quad m_0 \subseteq \bigcup_{k=1}^{[\frac{b}{p}]}(\Psi \triangleleft kp), \; m_\infty = \bigcup_{k=0}^{\infty}(\Psi \triangleright kp). \tag{3.1}$$

Доказательство: Доказательство очевидно. □

Замечание 3.2. *Пусть все линейные подмножества находятся в одноперио-дической форме. Тогда наименьшее b не превышает наибольшего базового элемента всех линейных подмножеств m, а наименьшее p равно наименьшему общему кратному всех их периодов. В частности, если все эти периоды попарно взаимно просты, то наименьшее b в точности равно наибольшему базовому элементу всех линейных подмножеств.*

Теорема 3.2. *Пусть $m \subseteq Nat$ — полулинейное множество, представленное в форме (3.1), $x, y \in Nat$. Пусть $\{A^{(i)}\}$ — последовательность полулинейных множеств, такая что*

$$A^{(0)} = m, \; A^{(i+1)} = (A^{(i)} \triangleleft x) \triangleright y.$$

Тогда существует $j \leq max\{[\frac{b}{|x-y|}], НОК(p, |x-y|)\} + 1$, такое, что

$$\bigcup_{i=1}^{\infty} A^{(i)} = \bigcup_{i=1}^{j} A^{(i)}.$$

Доказательство: **($x \geq y$)** Легко заметить, что, начиная с некоторого $j \leq [\frac{b}{|x-y|}] + 1$ “неполная” голова (префикс) множества A_j становится пустой и множество A_j превращается в чисто периодическое с наименьшим возможным базовым элементом. С другой стороны, не более чем за НОК$(p, |x-y|) + 1$ шагов мы исчерпаем все возможные сдвиги базового элемента.

($x < y$) Заметим, что “неполная” голова является подмножеством множества с теми же периодом и базовым элементом, что и у бесконечного периодического хвоста. Следовательно, сдвиги её элементов вправо (к бóльшим значениям) не “нарушат” вектор периода. Как и в предыдущем случае, не более чем за

НОК$(p, |x-y|)+1$ шагов мы исчерпаем все возможные сдвиги базового элемента. □

Теорема раскрывает важное свойство одномерных полулинейных множеств: конечно определенная аддитивная последовательность стабилизируется за конечное число шагов. Это свойство стабилизации было доказано как лемма Дж. Хопкрофтом и Ж. Ж. Пансио в [138], но только для сложения (сдвига вправо) и без каких-либо оценок требуемого количества шагов.

3.1.4. Одно периодический базис

Рассмотрим двоичный вектор v длины p, такой что $v[i] = 0$ для $b + i \notin \Psi$ и $v[i] = 1$ для $b + i \in \Psi$. Теорема 3.1 утверждает, что этот вектор является "битовой маской" для периодического "закрашивания" натурального ряда справа от числа b. Таким образом, мы можем использовать в качестве конечного символьного представления произвольного полулинейного одномерного множества m его *одно периодический базис* (m_0, b, p, v), состоящий из

- конечного *базового множества* m_0,
- *базового элемента* b,
- *длины периода* p,
- *вектора периода* v.

Данное представление проще формул арифметики Пресбургера [101, 111], однако может быть использовано только в одномерном случае.

Пример 3.1. $m = \{2 + 3k \mid k \in Nat\} \cup \{6k_1 + 9k_2 \mid k_1, k_2 \in Nat\}$.

$$m = \{0,\quad 2,\quad 5, 6,\quad 8, 9,\quad 11, 12,\quad 14, 15,\quad \ldots\}.$$

Одно периодический базис: $Z = (\{0, 2\}, 4, 3, (0, 1, 1))$.

Определение 3.1. *Базис* $Z = (m_0, b, p, v)$ *полулинейного множества* $m \subseteq Nat$ *называется* минимальным, *если для любого базиса* $Z' = (m'_0, b', p', v')$ *множества* m *выполняется* $p < p'$ *или* $(p = p'$ *и* $b \leq b')$.

Утверждение 3.1. *Для любого одномерного полулинейного множества* $m \subseteq Nat$ *минимальный базис существует и единственен.*

Доказательство: (**существование**) Предположим противное: минимальный базис отсутствует. Тогда из Теоремы 3.1 следует, что существуют по крайней мере два базиса с несравнимыми базовыми элементами и периодами:

$$Z = (m_0, b, p, v),\ Z' = (m'_0, b', p', v'),$$

$$\neg\Big((p < p' \vee (p = p' \wedge b \leq b')) \vee (p > p' \vee (p = p' \wedge b \geq b'))\Big). \tag{3.2}$$

Утверждение 3.2 может быть преобразовано в:

$$\neg(p < p') \wedge \neg(p = p' \wedge b \leq b') \wedge \neg(p > p') \wedge \neg(p = p' \wedge b \geq b').$$

Тогда $p = p'$ и, следовательно, $\neg(b \leq b') \wedge \neg(b \geq b')$, то есть $(b > b') \wedge (b < b')$ — противоречие.

(**единственность**) Предположим противное: имеются два минимальных базиса — Z и Z'. По определению минимального базиса они оба имеют один и тот же базовый элемент b и один и тот же период p:

$$Z = (m_0, b, p, v),\ Z' = (m'_0, b, p, v').$$

Рассмотрим $m_0 \neq m'_0$. Без потери общности мы можем предположить, что существует $x \in m_0$, такой, что $x \notin m'_0$. Но из $b > max(m'_0)$ следует, что x не может быть порождён Z' — противоречие.

Рассмотрим $v \neq v'$. Без потери общности мы можем предположить, что $v_i = 1$, $v'_i = 0$ для некоторого $0 \leq i < p$. Но из этого следует, что число $b + i$ может быть порождено базисом Z и не может быть порождено базисом Z' — противоречие. □

Минимальный базис множества m обозначим как $Base(m)$; множество, определяемое базисом Z, обозначим как $Set(Z)$.

Таким образом, мы выяснили, что в классе полулинейных базисов есть нормальная форма. Существует и простая процедура нормализации:

Утверждение 3.2. *Произвольный базис* (m_0, b, p, v) *полулинейного множества* $m \subseteq Nat$ *может быть преобразован в минимальный базис* $Base(m)$ *за полиномиальное время относительно* $b * p$.

Доказательство: Схема алгоритма:

1. Разобьём вектор v на самые короткие возможные повторяющиеся векторы v' (их длину обозначим p').

2. Уменьшим ("передвинем влево") базовый элемент настолько, насколько это возможно (сначала передвигая максимально влево с шагом p'; затем с шагом 1, соответственно "побитово сдвигая" вектор v').

Очевидно, что все приведённые вычисления могут быть выполнены за полиномиальное время относительно $b * p$. □

Обозначим процедуру минимизации полулинейного базиса (m_0, b, p, v) как $Mmz(m_0, b, p, v)$.

Для двоичных векторов $v, v' \in \{0,1\}^p$ через $NOT(v)$, $AND(v, v')$ и $OR(v, v')$ обозначим покомпонентное умножение, сложение и отрицание:

$$AND(v, v')[i] =_{\text{def}} min\{v[i], v'[i]\},\ OR(v, v')[i] =_{\text{def}} max\{v[i], v'[i]\},$$

$$NOT(v)[i] =_{\text{def}} (1 - v[i]).$$

Через v^k обозначим конкатенацию k векторов v:

$$v^k =_{\text{def}} \underbrace{v.v.v.\ldots.v}_{k}.$$

Теоретико-множественные операции и отношения могут эффективно вычисляться не над множествами, а непосредственно над их однопериодическими базисами:

Теорема 3.3. *Пусть* $m, m' \subseteq Nat$ *— полулинейные,* $Base(m) = (m_0, b, p, v)$, $Base(m') = (m'_0, b', p', v')$, $y \in Nat$. *Обозначим* $K = max\{b, b'\}$ *и* $L = НОК(p, p')$. *Пусть* $K = b + ip = b' + jp'$ *для некоторых* $i, j \in Nat$. *Тогда:*

1. $Base(Nat) = (\varnothing, 0, 1, (1))$;
2. $Base(m \cup m') = Mmz(\{x \in m \cup m' \mid x < K\}, K, L, OR(v^{\frac{L}{p}}, (v')^{\frac{L}{p'}}))$;
3. $Base(m \cap m') = Mmz(\{x \in m \cap m' \mid x < K\}, K, L, AND(v^{\frac{L}{p}}, (v')^{\frac{L}{p'}}))$;
4. $Base(m \setminus m') = Mmz(\{x \in m \setminus m' \mid x < K\}, K, L, AND(v^{\frac{L}{p}}, NOT((v')^{\frac{L}{p'}})))$;
5. $m \subseteq m' \iff AND(v^{\frac{L}{p}}, (v')^{\frac{L}{p'}}) = v^{\frac{L}{p}} \ \wedge \ \forall x \in m \ (x < K \Rightarrow x \in m')$;
6. $Base(m \triangleright y) = Mmz(\{x + y \mid x \in m_0\}, b + y, p, v)$;
7. $Base(m \triangleleft y) = Mmz(\{x - y \mid x \in m, x < B, x \geq y\}, B, p, v)$, *где* $B = min_{k \in Nat}\{b + kp - y \mid b + kp - y \geq 0\}$.

Доказательство: Доказательства всех этих утверждений очевидны. Мы используем базовые свойства соответствующих операций и периодическую структуру полулинейных множеств над *Nat*. □

Заметим, что ограничение $K = b + ip = b' + jp'$ носит технический характер — оно позволяет записать формулы в более краткой форме. Это ограничение не является существенным — мы легко можем трансформировать *любую* пару полулинейных базисов таким образом, чтобы она удовлетворяла данному условию (“сдвинув” вправо базовый элемент одного из базисов).

Все приведенные операции эффективны, то есть выполняются за полиномиальное время относительно размеров входных базисов. В некоторых случаях

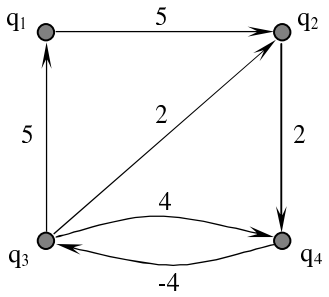


Рис. 3.3. Диаграмма переходов односчетчиковой сети.

мы использовали операцию проверки принадлежности на правой стороне определения (например, $x \in m \cup m'$ во втором утверждении). Однако проверяемое множество всегда ограничено сверху каким-либо значением (соответственно, $x < max\{b, b'\}$), то есть принадлежность разрешима.

3.2. Односчетчиковые контуры

В данном разделе рассматриваются циклические управляющие структуры односчетчиковой сети. Показано, что основным структурным свойством, определяющим границу между “хаотической” и чисто периодической частями пространства состояний, является наибольший общий делитель эффектов всех циклов сильно связных компонент диаграммы переходов сети.

3.2.1. Односчетчиковые сети

(Непомеченной) односчетчиковой сетью называется пара $N = (Q, T)$, где Q — конечное множество управляющих состояний, $T \subset Q \times Q \times \mathbf{Z}$ — конечное множество переходов.

Односчетчиковую сеть удобно изображать при помощи её *диаграммы переходов* — взвешенного ориентированного графа с вершинами из Q и дугами из T (Рис. 3.3).

Состояние сети описывается парой $q|c$, где $q \in Q$ — текущее управляющее состояние, $c \in Nat$ — текущее значение счетчика.

Переход $t = (q, q', z)$ *активен* в состоянии $q|c$, если $c + z \geq 0$. Активный переход может *сработать*, переводя сеть в состояние $q'|c+z$ (обозначается $q|c \xrightarrow{t} q'|c + z$).

Для перехода $t = (q, q', z)$ величина z также называется *эффектом* t (обозначается Eff(t)). *Ресурс* перехода определяется как:

$$\mathrm{Supp}(t) =_{\mathrm{def}} \begin{cases} 0, & z \geq 0; \\ |z|, & z < 0. \end{cases}$$

Последовательность переходов $\sigma = t_1 \ldots . t_n \in T^*$ может сработать в состоянии $s \in Q \times Nat$, если существуют состояния $s_1, \ldots, s_n \in Q \times Nat$, такие что $s \xrightarrow{t_1} s_1 \xrightarrow{t_2} \ldots \xrightarrow{t_n} s_n$ (обозначается $s \xrightarrow{\sigma} s_n$).

Для последовательности переходов $\sigma = t_1.t_2 \ldots t_{n-1}.t_n$ определим её предусловие и постусловие: ${}^\bullet t\sigma = {}^\bullet t(t_2 \ldots t_{n-1}.t_n) + |\mathrm{Eff}(t_1)|$ при $\mathrm{Eff}(t_1) < 0$ и ${}^\bullet t\sigma = {}^\bullet t(t_2 \ldots t_{n-1}.t_n) \ominus \mathrm{Eff}(t_1)$ при $\mathrm{Eff}(t_1) \geq 0$; $t^\bullet\sigma = t^\bullet(t_1.t_2 \ldots t_{n-1}) \ominus |\mathrm{Eff}(t_n)|$ при $\mathrm{Eff}(t_n) < 0$ и $t^\bullet\sigma = t^\bullet(t_1.t_2 \ldots t_{n-1}) + \mathrm{Eff}(t_n)$ при $\mathrm{Eff}(t_n) \geq 0$ (где $\ominus$ — усечённое вычитание до нуля).

Состояние $s' \in Q \times Nat$ *достижимо* от состояния $s \in Q \times Nat$ (обозначается $s \xrightarrow{*} s'$), если существует последовательность переходов $\sigma \in T^*$, такая что $s \xrightarrow{\sigma} s'$.

Размеченной односчетчиковой сетью называется пара (N, s_0), где $N = (Q, T)$ — односчетчиковая сеть, $s_0 \in Q \times Nat$ – начальное состояние (начальная разметка).

Для размеченной односчетчиковой сети (N, s_0) её *множество достижимости* $\mathcal{R}(N, s_0)$ определяется как:

$$\mathcal{R}(N, s_0) =_{\mathrm{def}} \{s \in Q \times Nat \mid s_0 \xrightarrow{*} s\}.$$

Размеченная сеть (N, s_0) называется неограниченной, если $\mathcal{R}(N, s_0)$ бесконечно.

Для выделенного управляющего состояния $q \in Q$ его множество достижимых значений счетчика $\mathcal{R}(N, s_0)[q]$ определяется следующим образом:

$$\mathcal{R}(N, s_0)[q] =_{\mathrm{def}} \{c \in Nat \mid s_0 \xrightarrow{*} q|c\}.$$

Управляющее состояние $q \in Q$ достижимо, если $\mathcal{R}(N, s_0)[q] \neq \emptyset$.

Граф достижимости $\mathcal{RG}(N, s_0)$ размеченной односчетчиковой сети (N, s_0) представляет собой орграф с вершинами из $\mathcal{R}(N, s_0)$, в котором дуга с пометкой t связывает вершины s и s' тогда и только тогда, когда в сети возможен переход $s \xrightarrow{t} s'$.

Диаграмма переходов односчетчиковой сети представляет обой взвешенный ориентированный граф. Напомним несколько основных понятий теории графов.

Маршрут в графе — это чередующаяся последовательность вершин и рёбер, начинающаяся и заканчивающаяся вершинами, в которой вершины, идущие до и после дуги, являются её началом и концом соответственно.

Ориентированный граф называется *сильно связным*, если для любых двух вершин p и q существует маршрут из p в q.

Мы рассматриваем только непустые маршруты, содержащие по крайней мере одну дугу.

Маршрут *замкнут*, если его первая и последняя вершины совпадают.

Цепью (ориентированной цепью) называется маршрут без повторяющихся дуг (вершины могут повторяться).

Простая цепь — это цепь, в которой вершины не повторяются.

Циклом называется замкнутая цепь.

Простой цикл — это цикл, в котором вершины не повторяются (кроме первой и последней).

Эффект и ресурс маршрута определяются индуктивно. Пусть t — переход, σ – маршрут, такой, что конец перехода t является началом первого перехода из σ. Обозначив как $t\sigma$ маршрут, полученный сцеплением t и σ, мы получим:

$$\mathrm{Eff}(t\sigma) =_{\mathrm{def}} \mathrm{Eff}(t) + \mathrm{Eff}(\sigma); \quad \mathrm{Supp}(t\sigma) =_{\mathrm{def}} \mathrm{Supp}(t) + (\mathrm{Supp}(\sigma) \ominus \mathrm{Eff}(t)).$$

Здесь $\ominus$ обозначает усеченное вычитание: для $x, y \in Nat$ положим $x \ominus y =_{\mathrm{def}} \max\{0, x - y\}$.

Положительным (*отрицательным*) называется маршрут с положительным

(соответственно, отрицательным) эффектом. Очевидно, что эффект цикла не зависит от выбора начального/конечного узла.

Узел q называется *(положительным) генератором*, если существует положительный маршрут от q до q (образующий положительный цикл) с нулевым ресурсом.

Лемма 3.4. *Любой положительный цикл содержит хотя бы один генератор.*

Доказательство: Индукция по длине цикла. Заметим также, что без ограничения общности мы можем рассматривать только циклы четной длины с чередующимися положительными и отрицательными дугами. □

Узел q называется *отрицательным генератором*, если существует отрицательный маршрут θ от q до q (образующий отрицательный цикл), такой, что $\mathrm{Supp}(\theta) = -\mathrm{Eff}(\theta)$.

Лемма 3.5. *Любой отрицательный цикл содержит хотя бы один генератор.*

Доказательство: Аналогично предыдущей лемме. □

3.2.2. Односчетчиковые контуры

Определение 3.2. *Односчетчиковая сеть называется сильно связной, если её диаграмма переходов представляет собой сильно связный орграф.*

Определение 3.3. *Пусть $N = (Q, T)$ — сильно связная сеть. Назовём сеть N:*

- положительным контуром*, если она содержит по крайней мере один положительный цикл;*
- отрицательным контуром*, если она содержит по крайней мере один отрицательный цикл и ни одного положительного;*

- нейтральным контуром *в прочих случаях (т.е. когда все циклы имеют нулевой эффект).*

Все дальнейшие определения и утверждения данного раздела касаются сильно связных сетей (контуров).

Определение 3.4. *Пусть $q \in Q$ — управляющее состояние. Определим два важных поддерживающих ресурса для q:*

- *$RSupp(q)$ — наименьшее значение счетчика, такое что в $(N, q|RSupp(q))$ все управляющие состояния достижимы;*
- *$USupp(q)$ — наименьшее значение счетчика, такое что $(N, q|USupp(q))$ не ограничена.*

Утверждение 3.3. *1) Если N — положительный контур, то $USupp(q)$ равно наименьшему значению среди ресурсов всех маршрутов от q до генераторов; в остальных случаях (для отрицательных и нейтральных контуров) $USupp(q)$ не определено.*
2) Если N — положительный контур, то $RSupp(q) = USupp(q)$.

Доказательство: 1) Случай отрицательного и нейтрального контуров очевиден.

Легко заметить, что в случае положительного контура неограниченность сети влечёт неограниченность всех управляющих состояний (так как сеть является сильно связной). С другой стороны, неограниченность генератора влечёт неограниченность сети (опять же, из-за её сильной связности). Генератор ограничен тогда и только тогда, когда он не достижим, то есть достижимость хотя бы одного генератора эквивалентна неограниченности сети.

2) Очевидно, что $RSupp(q) \leq USupp(q)$. Предположим противное: $RSupp(q) < USupp(q)$. Согласно определению положительного контура, найдётся хотя бы

один положительный цикл θ. Он содержит по крайней мере один генератор q'. Произвольная цепь σ от q до q' требует ресурса, не меньшего, чем $\mathrm{RSupp}(q)$ (по определению), следовательно, согласно предыдущему свойству мы можем использовать $\mathrm{RSupp}(q)$ в качестве $\mathrm{USupp}(q)$ — противоречие. □

Отметим, что:

- оба ресурса могут быть эффективно вычислены (например, при помощи алгоритма Тарьяна [197]).
- в частности, из 1) следует, что сильно связная сеть структурно ограничена (ограничена при любой начальной разметке) тогда и только тогда, когда она не является положительным контуром.

Определение 3.5. *Назовём наибольший общий делитель эффектов всех циклов* периодом *контура и обозначим его как* $\Delta(N)$.

Период определен только для положительных и отрицательных контуров. Контур с периодом Δ мы также будем называть Δ-контуром.

Утверждение 3.4. *В* Δ*-контуре:*

1. *эффект любого замкнутого маршрута кратен* Δ;
2. *для любой пары управляющих состояний* (q, q') *найдётся неотрицательное целое число* $Off(q, q')$ *(назовём его* смещением*), такое что* $Off(q, q') < \Delta$ *и для любой простой цепи* σ *от* q *до* q' *выполняется* $Eff(\sigma) = Off(q, q') + k\Delta$ *для некоторого* $k \in \mathbf{Z}$;
3. *для любого маршрута* σ *от* q *до* q' *выполняется* $Eff(\sigma) = Off(q, q') + k\Delta$ *для некоторого* $k \in \mathbf{Z}$.

Доказательство: 1) Очевидно, так как любой замкнутый маршрут представляет собой комбинацию циклов (простой цикл и конечное число циклов, прикрепленных к его вершинам).

2) Пусть σ_1 и σ_2 — две различные простые цепи от q до q'. Пусть также

$$\mathrm{Eff}(\sigma_1) = \delta_1 + k_1\Delta \quad \text{и} \quad \mathrm{Eff}(\sigma_2) = \delta_2 + k_2\Delta,$$

где $\delta_1, \delta_2 \in Nat,\ \delta_1 < \Delta,\ \delta_2 < \Delta$ и $k_1, k_2 \in \mathbf{Z}$. Достаточно доказать, что $\delta_1 = \delta_2$ (и, следовательно, равно $\mathrm{Off}(q_1, q_2)$).

Сеть является сильно связной, следовательно, существует простая цепь σ' от q' до q. Пусть $\mathrm{Eff}(\sigma') = e + m\Delta$, где $e \in Nat,\ e < \Delta$ и $m \in \mathbf{Z}$.

Маршруты $\sigma_1\sigma'$ и $\sigma_2\sigma'$ замкнуты, следовательно,

$$\mathrm{Eff}(\sigma_1\sigma') = n_1\Delta \quad \text{и} \quad \mathrm{Eff}(\sigma_2\sigma') = n_2\Delta \quad \text{для некоторых} \quad n_1, n_2 \in \mathbf{Z}.$$

Таким образом, мы имеем

$$\delta_1 + k_1\Delta + e + m\Delta = n_1\Delta; \quad \delta_2 + k_2\Delta + e + m\Delta = n_2\Delta.$$

Равенства могут быть переписаны:

$$\delta_1 + e = (n_1 - k_1 - m)\Delta; \quad \delta_2 + e = (n_2 - k_2 - m)\Delta.$$

Натуральные числа δ_1, δ_2 и e меньше, чем Δ. Следовательно, $\delta_1 = \delta_2$.

3) Заметим, что любой маршрут от q до q' представляет собой простую цепь от q до q', к узлам которой прицеплено конечное число замкнутых маршрутов. □

Это утверждение, в частности, показывает, что положительные и отрицательные контуры обладают удобным графическим представлением. Диаграмма переходов контура может быть изображена в виде кольца, состоящего из Δ секторов. Например, контур на Рис. 3.4 содержит 8 секторов (здесь мы видим сильно связную сеть с Рис. 3.3, изображенную в контурном виде). Отсчет секторов начинается с верхнего правого сектора (с нулевым номером) по часовой стрелке.

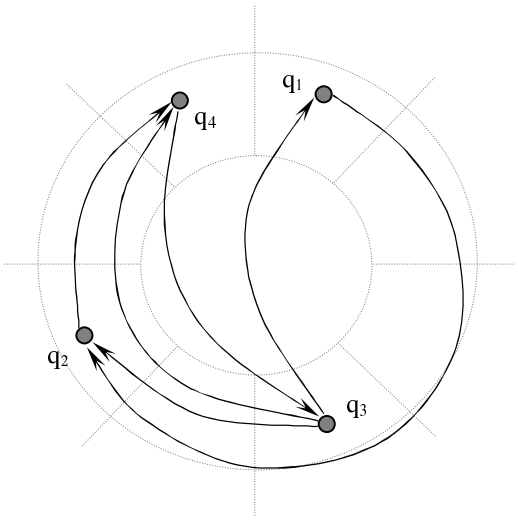


Рис. 3.4. Графическое представление Δ-контура при Δ = 8.

Сектор с номером n соответствует смещению n относительно нулевого. Если начальное управляющее состояние q поместить в сектор 0, то все остальные управляющие состояния однозначным образом распределятся по секторам в соответствие со своими смещениями относительно q (например, если $\mathrm{Off}(q, q') = n$, то q' попадает в n-ый сектор). Положительные переходы будут ориентированы по часовой стрелке, отрицательное — против часовой стрелки. Эффект перехода естественным образом определяется его длиной (количеством пересекаемых границ секторов).

Положительные контуры

Обозначим через C_+ и C_- конечные множества всех положительных и отрицательных циклов положительного контура. Рассмотрим множество замкнутых маршрутов C_*, построенное из элементов C_- добавлением наименьшего необходимого числа совместимых (инцидентных) положительных циклов таким образом, чтобы получались замкнутые положительные маршруты:

$$C_* =_{\mathrm{def}} \{\sigma^k\theta \mid \theta \in C_-,\ \sigma \in C_+,\ k \in Nat,\ \text{ такие что } \mathrm{Eff}(\sigma^k\theta) > 0 \ \wedge\ \mathrm{Eff}(\sigma^{k-1}\theta) \le 0\}.$$

Поскольку сеть сильно связная, подходящий положительный цикл всегда найдётся. Заметим также, что наибольший эффект среди всех замкнутых маршрутов

из C_* не превышает наибольший эффект среди всех положительных циклов (в противном случае значение $\mathrm{Eff}(\sigma^{k-1}\theta)$ было бы положительным).

Обозначим $C =_{\mathrm{def}} C_+ \cup C_*$. Это множество содержит все положительные циклы и некоторое количество замкнутых положительных маршрутов. Заметим, что, поскольку каждый положительный цикл содержит генератор, каждый элемент C по построению также содержит генератор.

Обозначим множество всех эффектов элементов множества C как E.

Лемма 3.6. *Для положительного* Δ*-контура выполняется НОД*$\{E\} = \Delta$.

Доказательство: Обозначим $g = \mathrm{НОД}\{E\}$. Предположим противное: $g \neq \Delta$.

Рассмотрим случай $g < \Delta$. Он невозможен, так как согласно Утв. 3.4(1) эффект любого замкнутого маршрута кратен Δ.

Рассмотрим случай $g > \Delta$. Поскольку C содержит все положительные циклы, должен существовать отрицательный цикл θ, эффект которого не делится на g. С другой стороны, C содержит замкнутый положительный маршрут $\phi = \sigma^k\theta$, где σ — положительный цикл, $k \in Nat$. $\mathrm{Eff}(\phi) = \mathrm{Eff}(\sigma^k\theta) = k\mathrm{Eff}(\sigma) + \mathrm{Eff}(\theta)$. Цикл σ — положительный, то есть он принадлежит C и его эффект делится на g. Следовательно, $\mathrm{Eff}(\phi) = kmg + \mathrm{Eff}(\theta)$ для некоторого $m \in Nat$. Замкнутый маршрут ϕ также принадлежит C, то есть $\mathrm{Eff}(\phi) = ng$ для некоторого $n \in Nat$. Мы получили, что $kmg + \mathrm{Eff}(\theta) = ng$, то есть $\mathrm{Eff}(\theta)$ делится на g — противоречие. □

Пусть q — управляющее состояние, $U(q)$ – множество всех положительных замкнутых цепей (циклов) от q до q, содержащих генераторы, $F(q)$ – множество эффектов этих цепей. Обозначим $W(q) =_{\mathrm{def}} \mathrm{НОК}\{F(q)\}$ – наименьшее значение эффекта, которое может быть получено “посещениями” любого из генераторов от управляющего состояния q. Другими словами, $W(q)$ — это наименьшее число, которое мы можем получить, итерируя любой цикл из множества $U(q)$. Очевидно, что $W(q)$ кратно Δ.

Теорема 3.4. *Пусть* $N = (Q, T)$ *— положительный* Δ*-контур,* $e \in Nat$ *– наиболь-*

ший эффект среди всех циклов в N, $q \in Q$ *– управляющее состояние. Тогда*

$$\mathcal{R}(N, q|u)[q] = DLin\{USupp(q), USupp(q) + W(q) + e^2, \{\Delta\}\}.$$

Доказательство: Другими словами, теорема утверждает, что, обозначив $u = \mathrm{USupp}(q)$ и $w = W(q)$, множество $\mathcal{R}(N, q|u)[q]$ является объединением двух непересекающихся множеств: конечного множества R_0, элементы которого не превосходят $u + w + e^2$, и арифметической прогрессии R_∞ с базой $u + w + e^2$ и разностью Δ :

$$\mathcal{R}(N, q|u)[q] = R_0 \bigsqcup R_\infty, \quad \text{где}$$

$$R_0 \subseteq \{x \in Lin\{u, \{\Delta\}\} \mid x < u + w + e^2\}, \quad R_\infty = Lin\{u + w + e^2, \{\Delta\}\}.$$

1) Рассмотрим достижимые значения счетчика, не превосходящие $u + w + e^2$. Достаточно показать, что “дистанция” от u до любого из них кратна Δ. Это следует из Утв. 3.4(3) и того очевидного факта, что $\mathrm{Off}(q, q) = 0$. Таким образом, структура R_0 доказана.

2) Рассмотрим достижимые значения счетчика, равные и превосходящие величину $u + w + e^2$.

Напомним, что $u = \mathrm{USupp}(q)$. Из Утв. 3.3 (2) мы получим $u = \mathrm{RSupp}(q)$. Следовательно, все генераторы уже достижимы.

Построим множества C и E способом, описанным в Лемме 3.6. Также напомним, что $\max\{E\} = e$ по построению C.

Множество эффектов всех возможных замкнутых маршрутов, начинающихся в q, содержит в себе линейное множество $X = Lin\{w, E\}$. Заметим, что слагаемое w добавлено с единственной целью — чтобы гарантировать, что всевозможные линейные комбинации эффектов из E добавляются к одному и тому же базовому значению — ведь все соответствующие генераторы “посещаются” от q маршрутами (повторяющимися циклами) с одним и тем же эффектом w.

Из Факта 2 следует, что множество элементов X, превосходящих $w + e^2$, представляет собой арифметическую прогрессию с разностью НОД$\{E\}$. Из Лем-

мы 3.6 следует, что НОД$\{E\} = \Delta$. Дополнительно, из Утв. 3.4(3) и Off$(q, q) = 0$ мы получим, что от управляющего состояния q нет замкнутых маршрутов с эффектами больше e^2, не вошедших в X. Действительно, Δ — наименьшее расстояние между достижимыми значениями счетчика для q (как и для любого другого управляющего состояния). Таким образом, структура R_∞ также доказана. □

Замечание 3.3. *Мы полагаем, что полученные оценки для "нижней границы регулярности" ($u+w+e^2$) могут быть улучшены (уменьшены). Например, можно попытаться уменьшить w, используя "ресурсы" из u. Однако это наверняка приведёт к ещё более тяжеловесным формулам.*

Отрицательные контуры

Теорема 3.5. *Пусть $N = (Q, T)$ — отрицательный Δ-контур, $e \in \mathbf{Z}_-$ – наименьший эффект цикла в N, $f \in \mathbf{Z}_-$ – наименьший эффект перехода в N, $q \in Q$ – управляющее состояние. Обозначим $\overline{\mathcal{R}}(N, q) =_{def} \{c \in Nat \mid 0 \in \mathcal{R}(N, q|c)[q]\}$. Тогда*

$$\overline{\mathcal{R}}(N, q) = DLin\{0, |f| + |e|^2, \{\Delta\}\}.$$

Доказательство: Заметим, что ресурс $|f|$ достаточен для запуска любого цикла в сети. □ Множество $\overline{\mathcal{R}}(N, q)$ полностью описывает все возможные уменьшения счетчика, которые может произвести отрицательный контур.

Соединение контуров

Определение 3.6. *Пусть $X \subseteq Nat$, $Y \subseteq Nat$. Через $X + Y$ и $X \ominus Y$ обозначим множества, полученные, соответственно, прибавлением/вычитанием элементов Y к элементам/из элементов X (сдвиги из Y применены к числам из X):*

$$X + Y =_{def} \{x + y \in Nat \mid x \in X, y \in Y\}.$$

$$X \ominus Y =_{def} \{x \ominus y \in Nat \mid x \in X, y \in Y\}.$$

Утверждение 3.5. *Пусть* $X = DLin\{a, b, \{c\}\}$ *и* $Y = \{d\}$, *где* $a, b, c, d \in Nat$. *Тогда:*

$$X + Y = DLin\{a + d, b + d, \{c\}\};\ X \ominus Y = DLin\{a \ominus d, b \ominus d, \{c\}\}.$$

Доказательство: Очевидно. □

Утверждение 3.6. *Если* $X = DLin\{a, b, \{c\}\}$ *и* $Y = DLin\{d, e, \{f\}\}$, *где* $a, b, c, d, e, f \in Nat$, *то, обозначив* $g = НОД(c, f)$, *мы имеем*

$$X + Y = DLin\{a + d, b + e + g(c/g - 1)(f/g - 1), \{g\}\}.$$

Доказательство: Легко убедиться в том, что неполная ("стираемая") часть $X+Y$ действительно имеет такую структуру (поскольку если слагаемые делятся на g, то и сумма тоже делится на g).

Рассмотрим бесконечную часть. Заметим, что все элементы $X+Y$, превышающие $b+e$, могут быть получены только прибавлением линейных комбинаций из c и f. Из Леммы 3.1 следует, что множество всех линейных комбинаций, превышающих $g(c/g - 1)(f/g - 1)$, является арифметической прогрессией с разностью g. □

Утверждение 3.7. *Если* $X = DLin\{a, b, \{c\}\}$ *и* $Y = DLin\{d, e, \{f\}\}$, *где* $a, b, c, d, e, f \in Nat$, *то, обозначив* $g = НОД(c, f)$, *мы имеем*

$$X \ominus Y = DLin\{a \ominus d, b \ominus e + g(c/g - 1)(f/g - 1), \{g\}\}.$$

Доказательство: Очевидно. □

Утверждения 3.6 и 3.7 показывают, что, если подать "на вход" контуру ограниченно неполное линейное множество, то на выходе также получится ограниченно неполное линейное множество. Более того, граничное значение, отделяющее хаотичную неполную "голову" от простого бесконечного "хвоста", растёт при этом достаточно медленно. Также заметим, что если в положительном контуре имеются отрицательные циклы, то их эффект может только уменьшить значение этого граничного значения.

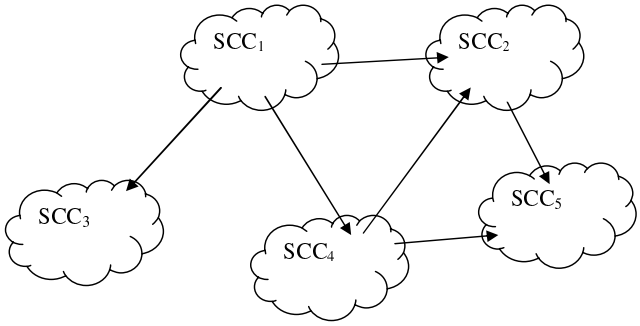


Рис. 3.5. Результат конденсации сильно связных компонент.

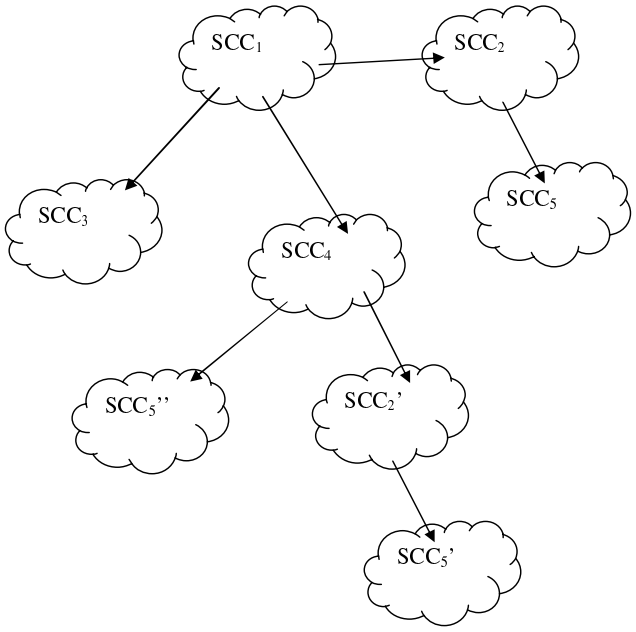


Рис. 3.6. Конденсированное дерево сильно связных компонент.

3.2.3. Дерево односчетчиковых контуров

Любая односчетчиковая сеть N содержит конечное число сильно связных компонент. Как известно из теории графов, если все вершины внутри каждой сильно связной компоненты орграфа объединить в единую “супервершину”, то результирующий “сжатый” орграф $\hat{N}$, сстоящий из таких “супервершин” (граф-конденсация) будет ациклическим (Рис. 3.5).

Путем копирования “супервершин” ациклический орграф $\hat{N}$ очевидным образом может быть трансформирован в дерево $\tilde{N}$ (Рис. 3.6).

Таким образом, любая односчетчиковая сеть может быть преобразована в дерево контуров. Каждому управляющему состоянию такой сети соответствует либо конечное, либо ограниченно неполное линейное множество достижимых значений счетчика. Периоды и граничные значения таких неполных линейных множеств являются структурными свойствами соответствующих контуров и не зависят от начального состояния сети. Грубо говоря, для односчетчиковых сетей проблема достижимости является *структурной* проблемой. Бесконечная часть множества достижимости оказывается слишком легко предсказуемой.

3.3. Алгоритмы анализа

Одноперiодические базисы представляют собой новый инструмент символьных вычислений, который может применяться для верификации систем с бесконечным числом состояний наряду с формулами арифметики Пресбургера [101, 124, 125], периодическими множествами [95], аффинными ограничениями и др. [98, 126]

3.3.1. Глобальная достижимость

(Локальная) задача достижимости состоит в выяснении того, достижимо ли данное состояние от начального. Эта проблема разрешима для сетей Петри ([168], см. также новый алгоритм [159]), хоть и весьма трудоёмка — все известные алгоритмы экспоненциальны по памяти [120, 121].

(Глобальная) задача достижимости состоит в построении эффективного конечного представления всего множества достижимости (под эффективностью понимается возможность “быстрого” решения локальной проблемы на основе данного представления).

Мы покажем, что для односчетчиковых сетей такое представление (*одноперiодическая свёртка*) существует, и его можно построить при помощи одноперiодических базисов. Это конечное дерево, корнем которого является начальное

состояние, а узлами-наследниками — пары вида $(q|Z)$, где q — управляющее состояние, Z — базис полулинейного множества возможных значений счетчика.

Алгоритм 3.1. *(глобальная достижимость)*

1. Корень: $(q_0|Base(c_0))$*, пометим узел как "новый".*

2. Пока есть "новые" узлы, для такого узла $(q|Z)$ *выполним:*

2.1. Если $\forall t = (q, q', z) \in T$ *выполняется* $z < 0$ *и* $(Set(Z) \triangleleft |z|) = \varnothing$*, пометим узел как "тупик".*

2.2. Если $Set(Z) \subseteq Reach(q)$*, где* $Reach(q)$ *— объединение всех уже вычисленных значений счетчика для q, то пометим узел как "старый".*

2.3. Если на пути от корня найдется $(q|Z')$ *с* $Set(Z') \subset Set(Z)$*, то срабатывание последовательности* $(q|Z') \xrightarrow{U} (q|Z)$ *— цикл, увеличивающий счетчик. Заменим* $(q|Z)$ *на* $(q|Z^{(\infty)})$*, где*

$$Z^{(0)} = Z,\ Z^{(i+1)} = Base\Big(Set(Z^{(i)}) \cup \big((Set(Z^{(i)}) \triangleleft {}^{\bullet}tU) \triangleright t^{\bullet}U\big)\Big).$$

2.4. Иначе пометим $(q|Z)$ *как "старый" и для любого активного перехода* $t = (q, q', z)$ *добавим "новый" узел* $(q'|Z')$*, где*

$$Z' = Base(Set(Z) \triangleleft |z|) \text{ при } z < 0 \text{ и } Z' = Base(Set(Z) \triangleright z) \text{ при } z \geq 0.$$

Доказательство: Ветвления в графе достижимости конечны, все теоретико-множественные операции вычислимы (Теорема 3.3). Из Теоремы 3.2 также следует, что базис $Z^{(\infty)}$ вычислим за конечное число шагов.

Остаётся доказать, что алгоритм не может произвести бесконечную последовательность расширяющихся множеств. Заметим, что даже двухсчетчиковые сети являются *плоскими системами* [160, 161]. Это означает, что в них отношение достижимости исчерпывается простыми (невложенными) циклами. Следовательно, и в нашем случае достаточно рассмотреть только простые циклы. Их

число конечно, то есть конечно число возможных наборов периодов, при помощи которых могут быть порождены полулинейные множества. Следовательно, порождаемые алгоритмом последовательности расширяющихся множеств рано или поздно стабилизируются.

Для подтверждения эффективности получаемого символьного представления отношения достижимости достаточно заметить, что проверка принадлежности для одноперiодических базисов может быть выполнена за полиномиальное время (Теорема 3.3) □

Символьный алгоритм для глобальной проблемы достижимости в двухсчетчиковых сетях был предложен в [138]. Он работает похожим образом, конструируя дерево, узлы которого помечены символьными описаниями множеств состояний. Однако там “метки” описывают не полулинейные, а линейные множества, поэтому дерево получается экстремально высоким и широким.

Наш алгоритм использует память гораздо более эффективно, поскольку полулинейное представление существенно компактнее линейного. Однако он может быть применен лишь в одномерном случае. Пример работы алгоритма приведен на Рис. 3.7.

3.3.2. Глобальная верификация

Помеченной односчетчиковой сетью называется набор $N = (Q, T, l)$, где (Q, T) — односчетчиковая сеть, $l : T \to \Sigma$ — помечающая функция, Σ — конечный алфавит имен (меток) действий.

Темпоральная логика EF ([169], называемая также UB^- в [118]) обладает следующим синтаксисом:

$$\varphi ::= \mathbf{true} \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid E\langle a\rangle\varphi \mid EF\varphi.$$

Пусть $LTS = (S, s_0, \to, l)$ — система помеченных переходов (в качестве которой мы рассматриваем диаграмму состояний некоей помеченной односчетчиковой сети), S — множество состояний (бесконечное множество достижимых

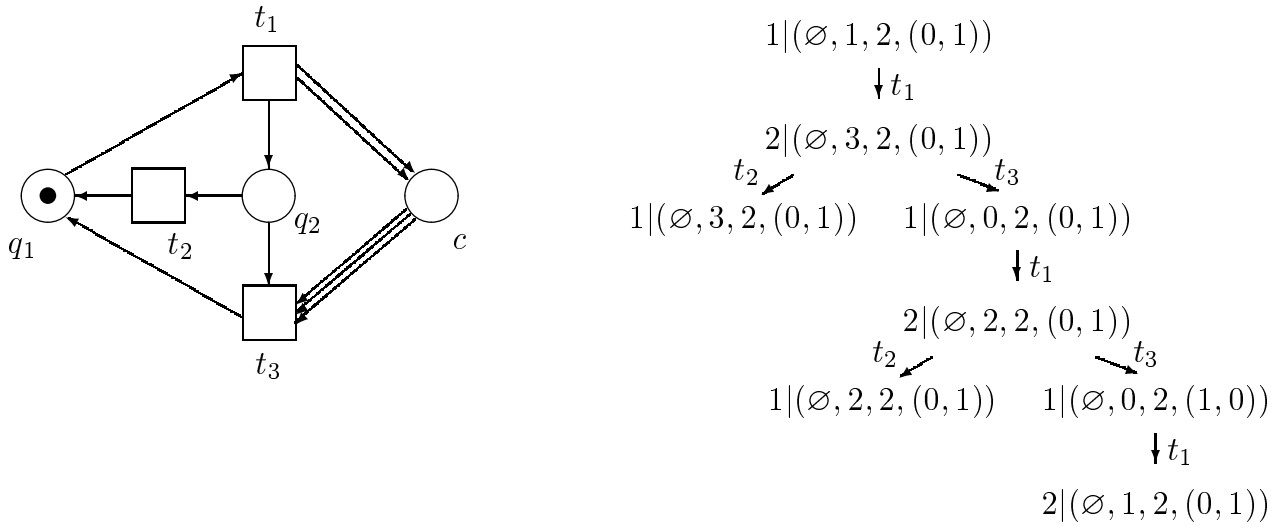


Рис. 3.7. Пример построения символьного дерева достижимости

состояний сети), $s_0 \in S$ — начальное состояние, $\rightarrow \subseteq S \times S$ — отношение переходов, $l : (\rightarrow) \rightarrow \Sigma$ — помечающая функция (соответствующая меткам переходов сети).

Отношение выполнимости $\models$ для состояния s системы LTS, темпоральной формулы φ и метки $a \in \Sigma$ определяется индуктивно:

- $s \models \mathbf{true}$;
- $s \models \neg\varphi \;\Leftrightarrow\; s \not\models \varphi$;
- $s \models \varphi_1 \wedge \varphi_2 \;\Leftrightarrow\; s \models \varphi_1 \wedge s \models \varphi_2$;
- $s \models E\langle a\rangle\varphi \;\Leftrightarrow\; \exists s' \in S \quad s \xrightarrow{a} s' \wedge s' \models \varphi$;
- $s \models EF\varphi \;\Leftrightarrow$ для некоторого пути $(s_1, s_2, \ldots)$, где $s = s_1,\ \exists i \geq 1\ s_i \models \varphi$.

Стандартным образом определяются двойственные темпоральные операторы: $A\langle a\rangle\varphi = \neg E\langle a\rangle\neg\varphi$, $AG\varphi = \neg EF\neg\varphi$.

Логика EF формализует свойства достижимости; она является расширением логики Хеннесси-Милнера и сужением логики CTL.

Формула $E\langle a\rangle\varphi$ означает “может сработать переход с меткой a, переводящий систему в какое-то новое состояние, в котором выполняется формула φ”. Двойственная к ней формула $A\langle a\rangle\varphi$ означает “срабатывание любого перехода с меткой a переводит систему в состояние, в котором выполняется формула φ”. Таким образом, оператор $E\langle\rangle$ формализует понятие “существования”, оператор $A\langle\rangle$ — понятие “неизбежности”.

Формула $EF\varphi$ означает “может сработать последовательность переходов, переводящая систему в какое-то новое состояние, в котором выполняется формула φ”. Двойственная к ней формула $AG\varphi$ означает “срабатывание любой последовательности переходов переводит систему в состояние, в котором выполняется формула φ”.

Логика EF позволяет формализовать всевозможные свойства достижимости, например:

- $AG\ EF\ A\langle a\rangle$**true** — в системе всегда существует вероятность наступления события “a” (точнее, из любого достижимого состояния достижимо состояние, при котором может выполниться только переход “a”). Это свойство может служить признаком правильной завершаемости процесса (если “a” — действие, возможное только в финальном состоянии).

- $EF\ AG\ E\langle a\rangle$**true** — система может сработать таким образом, что в ней возникнет единственный постоянно активный переход “a”. Например, этот переход может сигнализировать о возникновении переполнения памяти.

- $AG(E\langle a\rangle$**true** $\Rightarrow E\langle b\rangle$**true**$)$ — событие “b” возможно только в таких состояниях системы, в которых возможно событие “a”. Например, подтверждение транзакции возможно только тогда, когда возможен и её откат.

- $A\langle a\rangle\ EF\ E\langle b\rangle$**true** — если возможно “a”, то после его выполнения остается хотя бы одна возможность когда-то в будущем выполнить “b”. Например, в качестве “a” может выступать событие принятия рискованного решения, в

качестве “b” — событие получения крупных дивидендов (другими словами, “риск не является бессмысленным”).

- $AG\ A\langle a\rangle\ AG\ E\langle b\rangle\mathbf{true}$ — если возможно “a”, то после его выполнения рано или поздно неизбежно наступает “b”. Например, это может соответствовать правильности написания параллельной программы (если “a” — событие распараллеливания, “b” — событие синхронизации, то есть соединения распараллеленных ветвей вычисления).

- $AG((EF\ E\langle a\rangle\mathbf{true}) \vee (E\langle b\rangle\mathbf{true}))$ — всегда возможно либо возникновение в будущем “a”, либо прямо сейчас — “b”. Например, “a” — это правильное завершение процесса, “b” — предусмотренный аварийный выход.

Темпоральная логика EF не является универсальной (в частности, не позволяет формализовать существование бесконечной последовательности срабатываний одного и того же перехода), однако для большинства задач проверки свойств достижимости достаточно выразительна.

Задача (локальной) верификации состоит в определении того, выполняется ли данная формула темпоральной логики в данном состоянии.

Задача глобальной символьной верификации состоит в построении конечного эффективного представления множества всех состояний, в которых выполняется данная формула темпоральной логики.

Задача локальной верификации для односчетчиковых сетей (и для более общего случая односчетчиковых процессов) достаточно хорошо изучена ([132, 143]). В частности, известно [132], что задача верификации формул логики EF в односчетчиковых сетях является P^{NP}–полной.

Для положительного решения более общей задачи глобальной символьной верификации требуется наличие конечного символьного представления для множества выполнимости произвольной формулы. Покажем, что одноперiодические базисы позволяют построить такое представление.

В качестве инструмента используем символьное дерево достижимости (однопериодическую свертку пространства состояний). Обозначим как $Res(q, \varphi)$ множество значений счетчика, при которых формула φ выполняется в управляющем состоянии q. Значение $Res(q, \varphi)$ может быть вычислено индуктивно по структуре формулы:

Алгоритм 3.2. *(глобальная верификация)*

- $Res(q, \mathbf{true}) = Nat$
- $Res(q, \neg\varphi) = Nat \setminus Res(q, \varphi)$
- $Res(q, \varphi_1 \wedge \varphi_2) = Res(q, \varphi_1) \cap Res(q, \varphi_2)$
- $Res(q, E\langle a \rangle \varphi) = \bigcup_{(q|Z) \xrightarrow{t} (q'|Z') \,:\, l(t)=a} \Big(((Res(q', \varphi) \cap Set(Z')) \triangleleft t^{\bullet} t) \triangleright {}^{\bullet} tt \Big)$
- $Res(q, EF\varphi) = \bigcup_{(q|Z) \xrightarrow{U} (q'|Z')} \Big(((Res(q', \varphi) \cap Set(Z')) \triangleleft t^{\bullet} U) \triangleright {}^{\bullet} tU \Big)$

В определении алгоритма запись "$(q|Z) \xrightarrow{t} (q'|Z')$" означает "для всех Z, t, q', Z' таких, что в символьном дереве достижимости есть дуга t, соединяющая узлы $(q|Z)$ и $(q'|Z')$"; запись "$(q|Z) \xrightarrow{U} (q'|Z')$" означает "для всех Z, U, q', Z' таких, что в символьном дереве достижимости есть цепочка дуг U, соединяющая узлы $(q|Z)$ и $(q'|Z')$".

Очевидно, что предложенный алгоритм имеет полиномиальную трудоемкость по времени относительно размера формулы и размера символьного дерева достижимости.

3.3.3. Аппроксимация глобальной бисимуляции

Рассмотрим, каким образом однопериодические базисы могут быть применены для приближенного решения проблемы построения конечного символьного представления отношения бисимуляции состояний ($\sim$) односчетчиковой сети. Эта проблема является более сложной, чем хорошо изученная разрешимая проблема проверки бисимулярности двух данных состояний сети, поскольку требует

рассмотрения всего (в общем случае бесконечного) множества классов эквивалентности.

Насколько нам известно, вычислимость наибольшей бисимуляции остается открытой проблемой. Мы представляем специфический символьный метод аппроксимации сверху отношения наибольшей бисимуляции, основанный на использовании одноперiодической арифметики.

Локальная проблема бисимулярности состоит в проверке того, выполняется ли $M_1 \sim M_2$ для данных $M_1, M_2 \in Q{\times}Nat$.

Для пары состояний односчетчиковой сети это может быть проделано при помощи процедуры, строящей так называемой дерево расширений [142].

Глобальная проблема бисимулярности состоит в построении эффективного представления множества

$$(\sim) = \{(M_1, M_2) \mid M_1, M_2 \in Q{\times}Nat,\ M_1 \sim M_2\}.$$

Здесь под “эффективностью” также понимается возможность эффективной проверки бисимулярности разметок по получившемуся представлению $(\sim)$.

Напомним, что n-расслоенная бисимуляция разметок — это бисимуляция разметок в том случае, когда нам не важно, что будет происходить с сетью после n-го шага (см. определение в разделе 1.2.2).

Известно [145], что для любого $i \in Nat$ отношение $\sim_i$ является эквивалентностью, более того, $\sim_i \subseteq \sim_{i-1}$. Также известно, что $M_1 \sim M_2 \Leftrightarrow M_1 \sim_i M_2$ для любого $i \in Nat$. Кроме того, если $\sim_i = \sim_{i-1}$, то $\sim_i = \sim \infty = \sim$. Таким образом, построение расслоённых бисимуляций может рассматриваться в качестве аппроксимации полной бисимуляции разметок.

Расслоенная i-бисимуляция (множество i-бисимулярных пар состояний) может быть естественным образом представлена конечным множеством непересекающихся классов эквивалентности на множестве достижимых состояний:

$$Cl(\sim_i) = \{E_i^1, E_i^2, \ldots, E_i^{m_i}\},$$

где $E_i^j \subseteq Q{\times}Nat$, $E_i^1 \cup \ldots \cup E_i^{m_i} = Q{\times}Nat$, $E_i^j \cap E_i^k = \varnothing$ для $j \neq k$, и для любых $M \in E_i^j, M' \in E_i^k$ мы имеем $M \sim_i M'$ при $j = k$ и $M \not\sim_i M'$ в противном случае.

Напомним, что множество $Q{\times}Nat$ полулинейно у любой одно- и двухсчетчиковой сети [138]. Для произвольного управляющего состояния множество значений счетчика, i-бисимулярных данному значению, также полулинейно:

Утверждение 3.8. *Для любых $i, j \in Nat$, таких что $1 \leq j \leq m_i$, и для любого $q \in Q$ множество чисел $\{c \in Nat \mid (q, c) \in E_i^j\}$ полулинейно.*

Доказательство: Поскольку отношение $(\sim_i)$ индуктивно построено из тривиально полулинейного отношения $(\sim_0)$ посредством конечного числа трансформаций (переходов). □

Итак, мы рассматриваем только полулинейные подмножества $Q{\times}Nat$. Для удобства обозначим полулинейное множество состояний, соответствующее данному управляющему состоянию $q \in Q$, как пару $(q|A)$, где q — управляющее состояние, A — одномерный базис полулинейного множества соответствующих состоянию q значений счетчика. Например, обозначив $n = |Q|$, имеем

$$E_i^j = \left\{(q_1 \middle| E_i^j(q_1)), (q_2 \middle| E_i^j(q_2)), \ldots, (q_n \middle| E_i^j(q_n))\right\}.$$

Здесь $E_i^j(q_k)$ обозначает одномерный базис полулинейного множества значений счетчика для всех состояний из E_i^j с управляющим состоянием q_k.

Для полулинейного множества состояний $S \subseteq Q{\times}Nat$ и перехода $t \in T$ определим "обратное расширение" множества состояний:

$$Back(S, t) = S \cup \left\{\left(start(t) \middle| S\left(fin(t)\right) \triangleleft \delta^+(t) \triangleright \delta^-(t)\right)\right\},$$

где $start(t)$ и $fin(t)$ — начальное и финальное управляющие состояния перехода t, $\delta^+(t)$ и $\delta^-(t)$ — положительный и отрицательный эффекты перехода. Неформально, $Back(S, t)$ вычисляется из S добавлением всех состояний, обратно достижи-

мых от S посредством t. Это обозначение позволяет нам определить процедуру однопериодического построения ($\sim_{i+1}$) следующим образом:

Теорема 3.6. *Для любого* $E_i^j \in Cl(\sim_i)$ *найдутся* $E \subseteq Cl(\sim_{i-1})$, $U \subseteq T$, *такие что*

$$E_i^j = \bigcup_{S \in E, t \in U} Back(S, t).$$

Доказательство: Доказательство чисто техническое и основано на определениях однопериодической арифметики. □

Поскольку множество $Cl(\sim_{i-1})$ конечно, множество E также конечно, и, следовательно, расслоенное свойство переноса может быть эффективно проверено для любого кандидата E и набора переходов U. Следовательно, мы получили символьный алгоритм вычисления $Cl(\sim_n)$:

Алгоритм 3.3. *(аппроксимация бисимуляции)*
Input: Односчетчиковая сеть $N = (Q, T, l)$, *состояние* M_0, *параметр* $n \in Nat$.
Output: Однопериодическое представление $Cl(\sim_n)$ *расслоенной бисимуляции* $\sim_n$.

1. *Вычислим однопериодический базис* $Q \times Nat$ *(по Алгоритму 3.1).*

2. *Вычислим* $Cl(\sim_0)$ *как однопериодическое представление* $(Q \times Nat) \times (Q \times Nat)$, *содержащее единственный класс эквивалентности* $Id(Q \times Nat)$.

3. *Для i от 1 до n выполним:*

   Раасмотрим все разбиения-кандидаты множества $Cl(\sim_{i-1})$, *полученные посредством всевозможных обратных расширений (их число конечно в силу конечности T), и проверим для них выполнение расслоенного свойства переноса. Выберем наименьшего (относительно вложения) кандидата со свойством переноса в качестве* $Cl(\sim_i)$.

Доказательство алгоритма основано на конечности $Cl(\sim_i)$ и T и на вычислимости однопериодических операций. Очевидно, что для его работы требуется

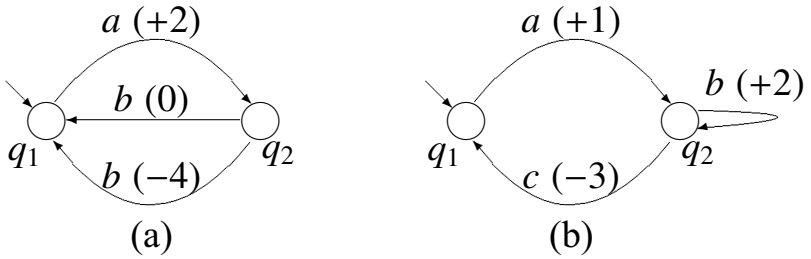


Рис. 3.8. Сети с конечным (a) и бесконечным (b) множествами классов эквивалентности.

экспоненциальный объем памяти (точнее, каждый из шагов 1 и 3 требует экспоненциальной памяти).

Алгоритм может завершиться досрочно, если $Cl(\sim_i) = Cl(\sim_{i-1})$ для некоторого $i \leq n$, и, следовательно, $\sim_i = \sim$ (Пример 3.2). Однако в общем случае последовательность расслоенных бисимуляций может оказаться бесконечной, так что наш алгоритм не является полуразрешающей процедурой для глобальной бисимуляции, которая в принципе может содержать бесконечное число классов эквивалентности (Пример 3.3).

Пример 3.2. *Рассмотрим сеть на Рис. 3.8 (a). Здесь*

$Q \times Nat = \left\{(q_1 \middle| (\varnothing, 0, 2, (1, 0))), (q_2 \middle| (\varnothing, 0, 2, (1, 0)))\right\}$.

Применив алгоритм, получим:

$Cl(\sim_0) = \left\{E_0^1\right\}$, *где* $E_0^1 = \{(q_1 \middle| (\varnothing, 0, 2, (1, 0))), (q_2 \middle| (\varnothing, 0, 2, (1, 0)))\}$;

$Cl(\sim_1) = \left\{E_1^1, E_1^2\right\}$, *где*

$E_1^1 = \{(q_1 \middle| (\varnothing, 0, 2, (1, 0)))\}, E_1^2 = \{(q_2 \middle| (\varnothing, 0, 2, (1, 0)))\}$;

$Cl(\sim_2) = Cl(\sim_1)$, *и, следовательно,* $Cl(\sim) = Cl(\sim_1)$.

Пример 3.3. *Рассмотрим сеть на Рис. 3.8 (b). Здесь*

$Q \times Nat = \left\{(q_1 \middle| (\varnothing, 0, 2, (1, 0))), (q_2 \middle| (\varnothing, 0, 2, (0, 1)))\right\}$.

Применив алгоритм, получим:

$Cl(\sim_0) = \left\{E_0^1\right\}$, *где* $E_0^1 = \{(q_1 \middle| (\varnothing, 0, 2, (1, 0))), (q_2 \middle| (\varnothing, 0, 2, (0, 1)))\}$;

$Cl(\sim_1) = \left\{E_1^1, E_1^2, E_1^3\right\}$, *где*

$E_1^1 = \{(q_1 \big| (\varnothing, 0, 2, (1, 0)))\}$, $E_1^2 = \{(q_2 \big| \{1\})\}$, $E_1^3 = \{(q_2 \big| (\varnothing, 2, 2, (0, 1)))\}$;

$Cl(\sim_2) = \left\{E_2^1, E_2^2, E_2^3, E_2^4, E_2^5\right\}$, *где*

$E_2^1 = \{(q_1 \big| \{0\})\}$, $E_2^2 = \{(q_1 \big| (\varnothing, 2, 2, (1, 0)))\}$,

$E_2^3 = \{(q_2 \big| \{1\})\}$, $E_2^4 = \{(q_2 \big| \{3\})\}$, $E_2^5 = \{(q_2 \big| (\varnothing, 4, 2, (0, 1)))\}$;

$Cl(\sim_n) = \left\{E_n^1, \ldots, E_n^{n-1}, E_n^n, E_n^{n+1}, \ldots, E_n^{2n+1}, E_n^{2n+2}\right\}$, *где*

$E_n^1 = \{(q_1 \big| \{0\})\}$, $E_n^{n-1} = \{(q_1 \big| \{2n-4\})\}$, $E_n^n = \{(q_1 \big| (\varnothing, 2n-2, 2, (1, 0)))\}$, $E_n^{n+1} = \{(q_2 \big| \{1\})\}$,

$E_n^{2n+1} = \{(q_2 \big| \{2n-1\})\}$, $E_n^{2n+2} = \{(q_2 \big| (\varnothing, 2n, 2, (0, 1)))\}$.

Интуитивно очевидно, что $Cl(\sim_\infty) = Cl(\sim) = Id(Q \times Nat)$.

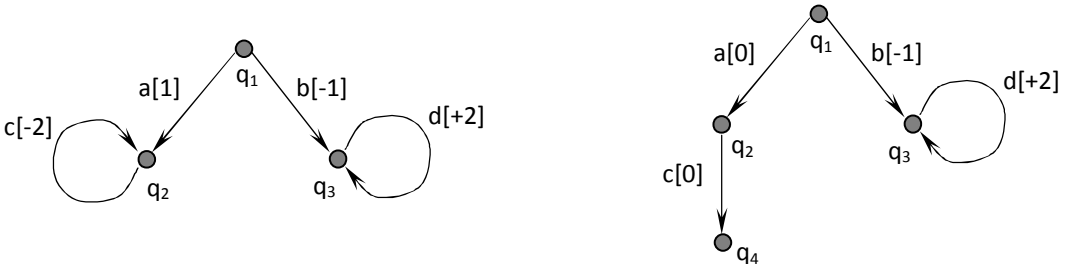
Теорема 3.7. *Последовательность расслоенных бисимуляций односчетчиковой сети N стабилизируется тогда и только тогда, когда N бисимулярна некоторому конечному автомату.*

Доказательство: ($\Rightarrow$) Поскольку максимальное отношение бисимуляции по построению содержит конечное число классов эквивалентности.

($\Leftarrow$) Предположим противное: последовательность различных расслоенных бисимуляций бесконечна. Тогда для любого $i \in Nat$ выполняется $\sim_i \subset \sim_{i-1}$. То есть по крайней мере один из классов эквивалентности отношения $\sim_{i-1}$ не является классом эквивалентности отношения $\sim_i$. Составлявшие данный класс элементы не могут попасть в другие унаследованные классы эквивалентности, следовательно, они образуют несколько (более одного) новых классов. Таким образом, число классов эквивалентности монотонно растёт с ростом i.

С другой стороны, в силу бисимулярности сети и автомата, а также в силу конечности числа состояний автомата, отношение эквивалентности $\sim$ содержит конечное число классов эквивалентности — противоречие. □

Таким образом, алгоритм 1 также может быть использован для проверки конечности наблюдаемого поведения односчетчиковой системы.



Начальное состояние – $q_1|1$.

Рис. 3.9. Неправильно (слева) и правильно (справа) организованные односчетчиковые сети, порождающие один и тот же язык.

3.4. Правильная организованность

Сравним две помеченные сети, изображенные на Рис. 3.9. Они порождают один и тот же язык $\{ac, bd, bdd, \ldots\}$. Однако сразу видно, что правая сеть “проще” — ветвь “ac” не использует счетчик. Цикл “c” в левой сети на самом деле избыточен, поскольку переход может сработать только один раз. Это цикл можно рассматривать как “ошибку проектирования” – мы использовали неограниченное хранилище (счетчик) для хранения априорно ограниченной величины (количества срабатываний перехода “c”). Это подобно тому, как если бы мы использовали целочисленную переменную для хранения булевских значений. В каком-то другом контексте поведение этих двух сетей может стать весьма различным (например, при начальном состоянии $q_1|3$ левая сеть будет дополнительно генерировать “acc”).

Неформально говоря, сеть является *правильной* (правильно организованной), если она использует свой счетчик только при порождении бесконечной части множества достижимости. Как мы покажем, неправильность является устранимой проблемой — для любой односчетчиковой сети существует эквивалентная ей правильная, которую можно эффективно построить.

Пусть $N = (Q, T)$ — односчетчиковая сеть.

Определение 3.7. *Размеченная сеть* $(N, q_0|c_0)$ *называется* правильной, *если для*

любого $m \in Nat$ *выполняется или* $\mathcal{R}(N, q_0|c_0 + m) = \{(q|c + n) : (q|c) \in \mathcal{R}(N, q_0|c_0)\}$, *или* $\mathcal{R}(N, q_0|c_0) = \{(q|c + n) : (q|c) \in \mathcal{R}(N, q_0|c_0 + m)\}$ *для некоторого* $n \in Nat$.

Другими словами, множество $\mathcal{R}(N, q_0|c_0 + m)$ может быть получено из множества $\mathcal{R}(N, q_0|c_0)$ посредством сдвига. Очевидно, что

Утверждение 3.9. *Если* $(N, q_0|c_0)$ — *правильная, то* $(N, q_0|c_0 + m)$ — *правильная для любого* $m \in Nat$.

Неформально, сеть является правильной, если начального значения счетчика достаточно для запуска всех возможных циклов и ветвей в графе достижимости (как мы показали в предыдущем разделе, этот граф бесконечен в общем случае, однако имеет достаточно простую структуру). Сеть не является правильной, если начальное значение счетчика используется также как часть конечной "управляющей подсистемы" сети. Действительно, если какой-нибудь из циклов или какая-нибудь из ветвей $\mathcal{R}(N, q_0|c_0 + m)$ недоступна из $\mathcal{R}(N, q_0|c_0)$, то она "отключена" срабатыванием какого-то элемента сети (отрицательного цикла, как будет видно далее), который использует счетчик в "конечном" стиле — в качестве невозобновляемого ресурса.

Рассмотрим некоторые дополнительные определения:

Определение 3.8. *Управляющее состояние* $q \in Q$ *называется* правильным, *если существует* $C \in Nat$, *такое что сеть* $(N, q|C)$ *правильна. Наименьшее из удовлетворяющих этому условию значений* C *называется* правильным ресурсом *для управляющего состояния* q.

Односчетчиковая сеть называется структурно правильной, *если все её управляющие состояния правильны.*

Односчетчиковая сеть называется структурно неправильной, *если ни одно из её управляющих состояний не является правильным.*

Односчетчиковые контуры обладают рядом ценных свойств с точки зрения структурной правильности:

Утверждение 3.10. *Пусть $N = (Q, T)$ — положительный Δ-контур, $q_0 \in Q$ — управляющее состояние. Тогда, обозначив $z = RSupp(q_0)$, для любого $m \in Nat$ имеем*

$$\mathcal{R}(N, q_0|z{+}m) = \begin{cases} \{(q|c + m) : (q|c) \in \mathcal{R}(N, q_0|z)\}, & \text{если в диаграмме переходов } N \\ & \text{нет отрицательныхных циклов;} \\ \{(q|c + (m \bmod \Delta)) : (q|c) \in \mathcal{R}(N, q_0|z)\}, & \text{в противном случае.} \end{cases}$$

Доказательство: Случай контура без отрицательных циклов тривиален.

Из Утв. 3.3(2) получим, что $\mathrm{RSupp}(q_0) = \mathrm{USupp}(q_0)$ и, следовательно, этого ресурса достаточно для порождения любого неограниченного поведения контура. Таким образм, все возможные вычитания (по модулю Δ), производимые отрицательными циклами, уже учтены в $\mathcal{R}(N, q_0|\mathrm{RSupp}(q_0))$. □

Таким образом, мы получаем непосредственное следствие:

Следствие 3.1. *Каждое управляющее состояние q положительного контура правильно с ресурсом $RSupp(q)$.*

Другие тривиальные свойства правильной организованности:

Утверждение 3.11. *1. Неотрицательные контуры структурно правильны.*

2. Отрицательные контуры структурно неправильны.

Доказательство: (1) Следует из определения структурной правильности и Следствия 3.1.

(2) Предположим противное: управляющее состояние $q \in Q$ правильно с ресурсом $c \in Nat$. Очевидно, что множество достижимости отрицательного контура конечно при любом начальном состоянии. Следовательно, существует $m \in Nat$, такое что $|\mathcal{R}(N, q|c)| = m$. Более того, из определения правильности получим $|\mathcal{R}(N, q|c + k)| = m$ для любого $k \in Nat$.

Пусть s — ресурс некоторого отрицательного цикла, начинающегося от состояния q (поскольку N сильно связна, такой цикл найдётся всегда). Очевидно, что $(q|c) \in \mathcal{R}(N, q|c + s)$, и, более того, $(q|c + i * s) \in \mathcal{R}(N, q|c + m * s)$ для любого $i \in Nat$, такого что $0 \leq i \leq m$. Но из этого следует, что $|\mathcal{R}(N, q|c + m * s)| > m$ — противоречие. □

Рассмотрим произвольные односчетчиковые сети, представленные в форме деревьев контуров. Контуры, непосредственно достижимые от корня (без прохода через другие положительные или отрицательные контуры), назовём *верхними*.

Утверждение 3.12. *Размеченная односчетчиковая сеть* $(N, q_0|c_0)$ *правильна тогда и только тогда, когда выполнены два условия:*

1. *в дереве контуров все верхние контуры неотрицательны;*
2. *значение* c_0 *не меньше максимального правильного ресурса входных узлов всех верхних контуров.*

Доказательство: ($\Leftarrow$) – очевидно.

($\Rightarrow$) Предположим противное.

Во-первых, предположим, что в дереве один из верхних контуров отрицателен. В таком случае мы имеем ситуацию, подобную изображенной на Рис. 3.9. Увеличивая начальное значение счетчика, мы будем производить все более длинные последовательности вычитаний и, следовательно, постоянно увеличивающееся в размерах множество достижимых значений счетчика для управляющих состояний соответствующего отрицательного контура — противоречие с определением правильности.

Далее, предположим, что не выполняется второе условие. Это означает, что в дереве имеется неотрицательный верхний контур, для полноценного срабатывания которого не хватает имеющегося начального ресурса. Опять же, при увеличении начального значения счетчика мы получим увеличение множеств достижимости — противоречие. □

Правильные сети не содержат избыточных отрицательных контуров (которые хорошо видны в соответствующем дереве). Очевидно, что даже если такой контур присутствует, для любого начального значения счетчика он произведёт только конечное число достижимых состояний (с управляющими состояниями из данного контура), и, следовательно, может быть безболезненно заменен на эквивалентный конечный автомат, не использующий счетчик (см. на Рис. 3.9).

Следствие 3.2. *Для любой односчетчиковой сети существует эквивалентная (обладающая тем же множеством достижимости) правильная односчетчиковая сеть.*

Правильные сети могут рассматриваться как “корректно сконструированные”. Более того, правильные деревья контуров могут использоваться в качестве “структурно правильного” способа конструирования односчетчиковых сетей.

3.5. Потоки работ с ресурсом

В последние годы всё более активно развивается ещё одна важная область применения сетей Петри — моделирование потоков работ (workflow). Потоки работ используются для формализации управления всевозможными технологическими процессами, бизнес-процессами, web-сервисами, распределенными вычислениями и т.д. В потоках работ возможны циклы, также возможно распараллеливание и синхронизация. Однако есть и ограничения, в частности, существуют одно начальное и одно конечное состояния, запрещены тупики.

Для моделирования процессов потоков работ используется специальный подкласс сетей Петри — так называемые WF-сети [202, 203].

Одной из основных проблем при разработке workflow является обеспечение их правильной организованности — отсутствия тупиков, избыточных переходов, необоснованного использования ресурсов и т.п. Применяется формальный критерий *бездефектности* — говорят, что исполнение процесса завершается коррект-

но, если от любой разметки, достижимой от начальной (одна фишка в начальной позиции), достижима финальная (одна фишка в финальной позиции), и при этом в сети по завершению работы не остаётся лишних управляющих фишек.

Бездефектность WF-сетей разрешима [202, 204]. Более того, известно несколько разрешимых модификаций классического понятия бездефектности, например, k-бездефектность [207], структурная бездефектность [198], бездефектность вложенных моделей [205] и структурированных сетей [110].

При разработке workflow-процессов большое внимание уделяется управлению ресурсами. Ресурсы в данном случае понимаются в обобщённом смысле — это могут быть исполнители (люди или устройства), сырьё, финансы и т.д. Для того, чтобы лучше учесть ресурсную составляющую систем, базовый формализм WF-сетей многими авторами был расширен до различных вариантов “сетей с ресурсами”, что повлекло необходимость соответствующих модификаций понятия бездефектности, и, как правило, усложнило проверку этого свойства.

В [64, 65] был исследован специальный класс сетей с разделяемыми ресурсами (WFR-сетей), для которого также была установлена разрешимость бездефектности. В [192, 206] был введён более общий класс — так называемые ресурсно-ограниченные сети (Resource-Constrained Workflow Nets, RCWF-сети). В обоих случаях авторы накладывают на ресурсы два ограничения. Во-первых, количество доступных ресурсов на момент окончания работы процесса должно совпадать с начальным количеством. Во-вторых, при любой достижимой разметке количество доступных ресурсов не может превышать начальное количество.

В [206] было доказано, что для RCWF-сетей с одномерным ресурсом бездефектность разрешима за полиномиальное время. В [192] доказано, что бездефектность разрешима и в общем случае (сведением к задаче о домашней разметке).

В перечисленных статьях рассматриваются ресурсы, которые находятся в системе постоянно и в фиксированном количестве. Они не уничтожаются и не создаются, а всего лишь блокируются и освобождаются при срабатываниях пере-

ходов сети. Таким образом, множества состояний RCWF- и WFR-сетей конечны.

В данном разделе мы определяем и исследуем более общий класс сетей с произвольными трансформациями ресурсов, которые требуются, например, в случае открытых и/или адаптивных workflow-систем. Определяется формализм, названный ресурсными WF-сетями (RWF-сетями), в котором не накладывается никаких ограничений на работу с ресурсными позициями. Даже бездефектные RWF-сети могут обладать бесконечными множествами достижимых состояний, поэтому известные методы анализа бездефектности для них неприменимы.

Для ресурсов произвольной размерности вопрос разрешимости бездефектности остаётся открытым, однако для одномерного ресурса мы представляем алгоритмы проверки бездефектности и определения минимального необходимого ресурса. Одномерный ресурс — интересное сужение RWF-сетей, имеющее достаточно много практических приложений (например, моделирование дискретного времени, объёма памяти, доступных финансов и т.п.)

3.5.1. Сети потоков работ с ресурсами

Приведём определение специального подкласса сетей Петри, который используется при моделировании потоков работ (workflow) — это так называемые WF-сети [35].

Определение 3.9. *Пусть* $N = (P, T, F)$ *— обыкновенная сеть Петри. Сеть* N *называется* WF-сетью *(сетью потока работ), если*

1. *в множестве* P *имеются две специальные позиции* i *и* o*, такие, что* ${}^\bullet i = o^\bullet = \emptyset$;
2. *любой элемент множества* $P \cup T$ *лежит на пути из* i *в* o.

Позиция i называется *начальной*, а позиция o — *финальной* позицией сети N. Начальная разметка сети потока работ состоит из одной фишки в позиции i.

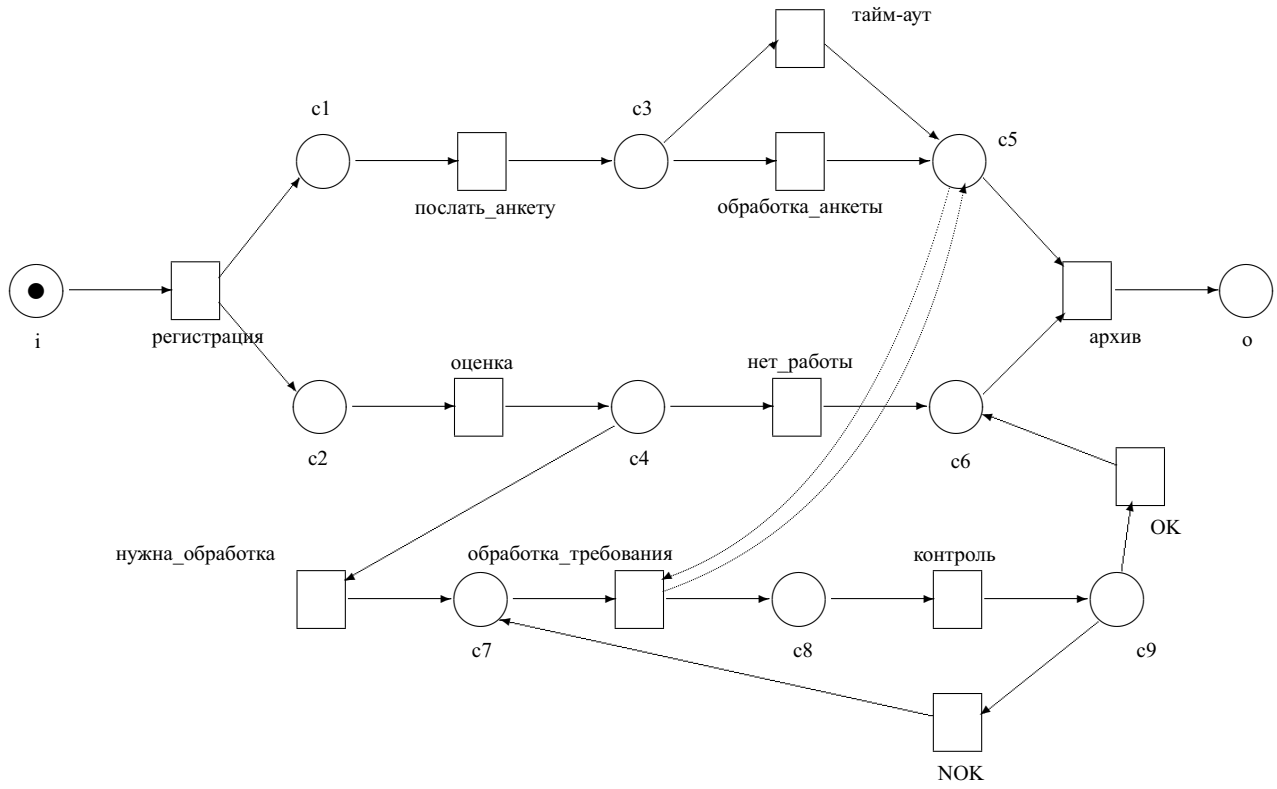


Рис. 3.10. Модель потока работ (workflow) бизнес-процесса обработки страховых требований

Правильное завершение процесса, моделируемого сетью, гарантируется выполнением следующего свойства:

Определение 3.10. *WF-сеть N называется* бездефектной*, если для любой достижимой разметки $M \in \mathcal{R}(N, i)$ выполняется*

1. $o \in \mathcal{R}(N, M)$;

2. $o + M' \in \mathcal{R}(N, M) \;\Rightarrow\; M' = \emptyset$.

Другими словами, из любого достижимого состояния бездефектной сети достижимо финальное состояние, при этом в финальном состоянии не может остаться никаких “лишних” фишек. Свойство бездефектности может быть эффективно проверено [202].

Легко заметить, что бездефектные сети потоков работ (WF-сети) являются подклассом ограниченных сетей Петри.

На рисунке 3.10 изображена сеть Петри, моделирующая процесс обработки страховых требований (этот пример взят из книги [35]). Прежде всего, тре-

бование регистрируется (задача *регистрация*), затем параллельно выполняются две задачи: клиенту посылается анкета (задача *послать_анкету*) и производится оценка требования (задача *оценка*). Если анкета возвращается в течение двух недель, то выполняется задача *обработка_анкеты*. Если в течение двух недель анкета не возвращена, то результат анкетирования игнорируется (задача *тайм-аут*). На основании результата оценки требование либо рассматривается, либо нет. Реальная работа с требованием (задача *обработка_требования*) откладывается до того момента, когда будет выполнено условие c5, т.е. либо анкета обработана, либо истекло время ее ожидания. Обработка требования контролируется с помощью задачи *контроль*. В конце концов, выполняется задача *архив*.

В сетях с ресурсами мы дополнительно вводим конечное множество ресурсных позиций.

Определение 3.11. Сетью потоков работ с ресурсами *(RWF-сетью) называется набор* $N = (P_c, P_r, T, F_c, F_r)$*, где*

- $N_c = (P_c, T, F_c)$ *— обыкновенная WF-сеть (называемая* управляющей подсетью *сети N, при этом элементы множеств* P_c *и* F_c *называются управляющими позициями и дугами соответственно);*
- P_r *— конечное множество ресурсных позиций,* $P_c \cap P_r = \emptyset$*;*
- $F_r : (P_r \times T) \cup (T \times P_r) \to Nat$ *— конечное множество ресурсных дуг.*

Заметим, что из определения следует, в частности, что $\forall t \in T \ \exists p \in P_c : F_c(p,t) + F_c(t,p) > 0$, то есть каждый переход инцидентен какой-нибудь управляющей позиции — этим гарантируется невозможность “неконтролируемых” модификаций ресурсов.

Разметка RWF-сети распадается на две составляющие — управляющую и ресурсную часть. Мультимножество $c + r$, в котором $c \in \mathcal{M}(P_c)$ и $r \in \mathcal{M}(P_r)$, мы будем обозначать как $(c|r)$.

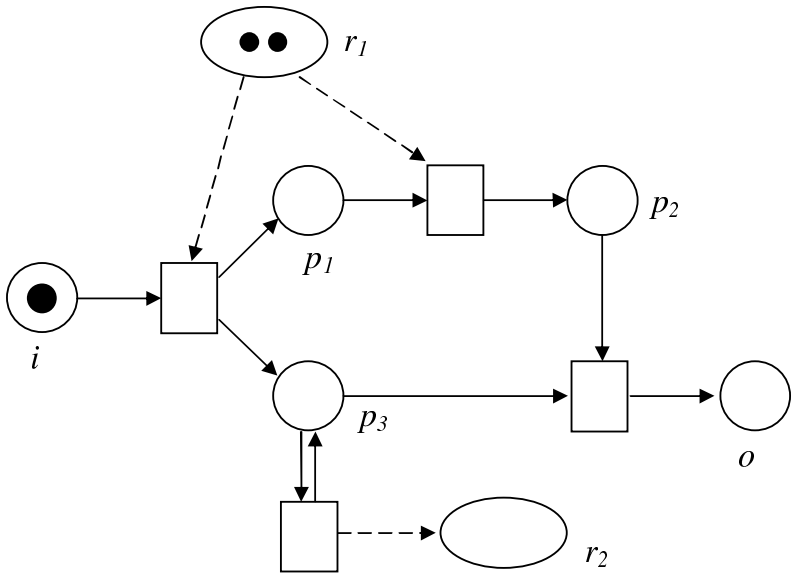


Рис. 3.11. WF-сеть с ресурсами

Определение 3.12. *Для сети N* ресурсом *называется мультимножество над P_r.* Управляемым ресурсом *называется мультимножество над $P_c \cup P_r$.*

На Рис. 3.11 приведен пример RWF-сети с ресурсными позициями r_1 и r_2. Графически ресурсные позиции обозначаются овалами, ресурсные дуги — пунктирными стрелками.

3.5.2. Бездефектность

Введём определение бездефектности для сетей с ресурсами.

Определение 3.13. *Размеченная RWF-сеть* $(N, c|r)$ *называется* бездефектной, *если* $\forall s \in \mathcal{M}(P_r),\ \forall M \in \mathcal{R}(N, c|r + s)$ *выполняется:*

1. $\exists s' \in \mathcal{M}(P_r)\ :\ o|s' \in \mathcal{R}(N, M)$;
2. $c'|r' \in \mathcal{R}(N, M)\ \Rightarrow\ c' = o\ \vee\ c' \cap o = \emptyset$.

Таким образом, во-первых, процесс может правильно завершиться при любом развитии событий, и, дополнительно, увеличение начального ресурса не нарушает свойства правильной завершаемости.

Заметим, что наше определение существенно отличается от определения бездефектных RCWF-сетей [206]. Мы не запрещаем создание и уничтожение ресурсов — они могут производиться и непосредственно в ходе исполнения процесса. Из этого, в частности, следует возможная неограниченность бездефектных RWF-сетей.

Утверждение 3.13. *Если размеченная RWF-сеть* $(N, i|r)$ *бездефектна, то размеченная WF-сеть* (N_c, i) *также бездефектна.*

Доказательство: Предположим противное. Пусть $(N, i|r)$ — бездефектная RWF-сеть, но при этом WF-сеть (N_c, i) не бездефектна. Тогда найдётся разметка $c \in \mathcal{R}(N_c, i)$, такая, что либо финальная разметка o недостижима от c, либо $o \in c$ и $c \neq \{o\}$.

Поскольку в управляющей подсети разметка c достижима от начальной разметки i после некоторой конечной последовательности срабатываний переходов, мы всегда можем выбрать начальный ресурс s настолько большим, чтобы та же последовательность могла сработать в сети с ресурсами от разметки $(N, i|r + s)$, достигнув того же управляющего состояния c.

Если в управляющей подсети финальное состояние не было достижимо от c, то добавление ресурсных позиций никак не могло сделать его достижимым в сети с ресурсами, то есть для $(N, i|r + s)$ — что противоречит бездефектности $(N, i|r)$. Если же $o \in c$ и $c \neq \{o\}$, то мы также получаем противоречие с бездефектностью $(N, i|r)$, поскольку управляющее состояние c достижимо в $(N, i|r + s)$. □

Обратное неверно: существуют не бездефектные RWF-сети, управляющие подсети которых бездефектны. Пример такой сети приведён на Рис. 3.12.

Пусть N — RWF-сеть. Через $C(N)$ обозначим *множество всех достижимых в* N_c *управляющих разметок*, т. е. $C(N) = \mathcal{R}(N_c, i)$.

Утверждение 3.14. *Если размеченная RWF-сеть* $(N, i|r)$ *бездефектна, то*

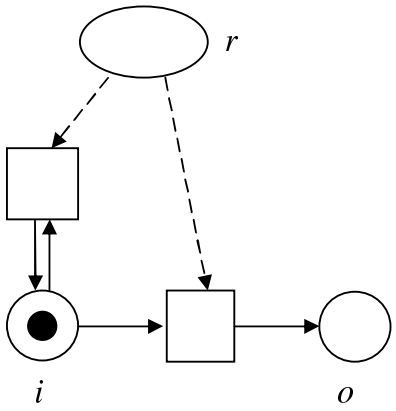


Рис. 3.12. RWF-сеть с бездефектной управляющей подсетью, для которой не существует бездефектных ресурсов

1. *для любой достижимой управляющей разметки* $c \in C(N)$ *найдётся ресурс* r'*, такой, что* $(N, c|r')$ *бездефектна;*

2. *для любых двух управляющих разметок* $c_1, c_2 \in C(N)$ *выполняется* $c_1 \not\subset c_2$ *и* $c_2 \not\subset c_1$.

Доказательство: (1) Как и в случае доказательства Утв. 3.13, мы можем взять достаточно большой начальный ресурс $r + s$.

(2) Предположим противное. Пусть для некоторых $c_1, c_2 \in C(N)$ выполняется $c_2 = c_1 + c'$, где $c' \neq \emptyset$. Из Утв. 3.14(1) следует, что найдутся ресурсы r_1 и r_2, такие, что RWF-сети $(N, c_1|r_1)$ и $(N, c_2|r_2)$ бездефектны. Тогда сети $(N, c_1|r_1 + r_2)$ и $(N, c_2|r_1 + r_2)$ также бездефектны. Следовательно, финальная разметка $o|r'$ достижима от разметки $c_1|r_1 + r_2$, то есть разметка $o + c'|r'$ достижима от бóльшей разметки $c_2|r_1 + r_2$ — противоречие со свойством бездефектности RWF-сети $(N, c_2|r_1 + r_2)$. □

Далее мы будем называть RWF-сеть N бездефектной (без указания конкретного ресурса), если существует такой ресурс r, при котором размеченная RWF-сеть $(N, i|r)$ бездефектна.

Из второй части Утв. 3.14 и известной леммы Диксона [114] получим, что

Следствие 3.3. *Для любой бездефектной RWF-сети N множество $C(N)$ достижимых управляющих состояний конечно.*

Замечание 3.4. *Поскольку управляющая подсеть бездефектной RWF-сети N ограничена, множество $C(N)$ может быть эффективно построено (например, при помощи покрывающего дерева [41]).*

Определение 3.14. *Пусть N — RWF-сеть, $c \in C(N)$. Определим:*

1. $res(c) =_{def} \{r \in \mathcal{M}(P_r) \mid (N, c|r)$ *бездефектна*$\}$ — *множество всех бездефектных ресурсов для c;*
2. $mres(c) =_{def} \{r \in res(c) \mid \nexists r' \in res(c) : r' \subset r\}$ — *множество всех минимальных бездефектных ресурсов для c.*

Из леммы Диксона немедленно получаем:

Утверждение 3.15. *Для любой бездефектной RWF-сети N и для любой её управляющей разметки $c \in C(N)$ множество $mres(c)$ конечно.*

Вопросы о вычислимости $mres(c)$ и разрешимости бездефектности для RWF-сетей пока остаются открытыми. В следующем подразделе мы дадим положительные ответы на оба этих вопроса для случая одномерного ресурса.

3.5.3. Бездефектность в сетях с одномерным ресурсом

Пусть $N = (P_c, P_r, T, F_c, F_r)$ — RWF-сеть, в которой $P_r = \{p_r\}$ (то есть ресурсная позиция всего одна). Такие сети мы будем называть RWF-сетями с одномерным ресурсом или просто одномерными RWF-сетями. Пример подобной сети приведён на Рис. 3.13.

Поскольку нас интересуют вопросы бездефектности ресурсов, в дальнейшем мы предполагаем, что в рассматриваемых сетях управляющая подсеть уже бездефектна (в противном случае в силу Утв. 3.13 сама сеть также автоматически была бы не бездефектной), и, следовательно, ограничена (Следствие 3.3).

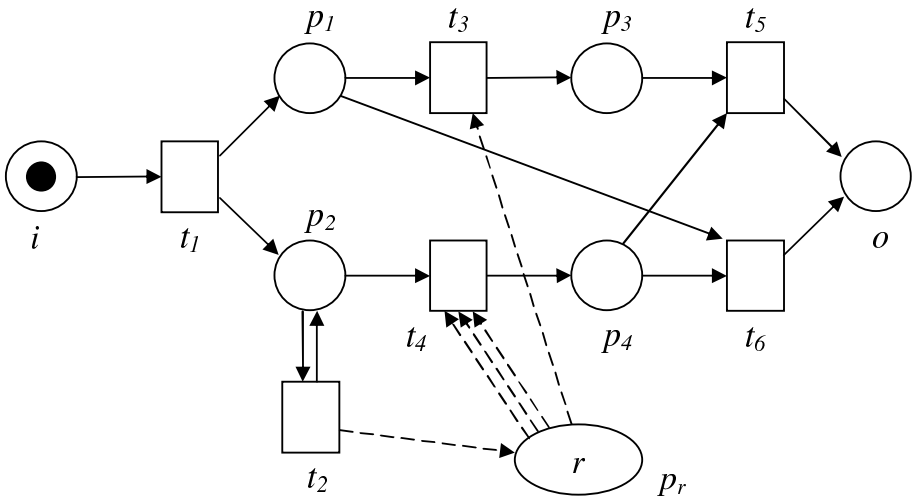


Рис. 3.13. Одномерная RWF-сеть N_1

Управляющий автомат

Ограниченная управляющая подсеть RWF-сети может быть представлена в виде конечного автомата (системы переходов). Состояниями этого автомата являются все управляющие состояния исходной сети (элементы $C(N)$), переходами — её переходы. Если мы теперь пометим переходы арифметическими действиями, которые они совершают над ресурсной позицией (добавлением и удаление фишек), то получим в точности односчётчиковую сеть с двумя выделенными состояниями — начальным i (без входящих дуг) и финальным o (без исходящих дуг). Кроме того, из определения WF-сети следует, что в этой сети все состояния лежат на путях из i в o.

Напомним, что эффект перехода t обозначается как Eff(t), а состояние односчётчиковой сети — как $(c|r)$, где $c \in C(N)$ — управляющее состояние, $r \in Nat$ — текущее значение счётчика (количество ресурса).

Рассмотрим пример на Рис. 3.14. Здесь показан управляющий автомат одномерной RWF-сети, изображённой на Рис. 3.13. Состояния обозначены восьмиугольниками, помеченными соответствующими управляющими разметками сети. Переходы помечены соответствующими именами и ресурсными эффектами. Например, $t_2[+1]$ означает, что перед нами переход t_2 исходной сети, который

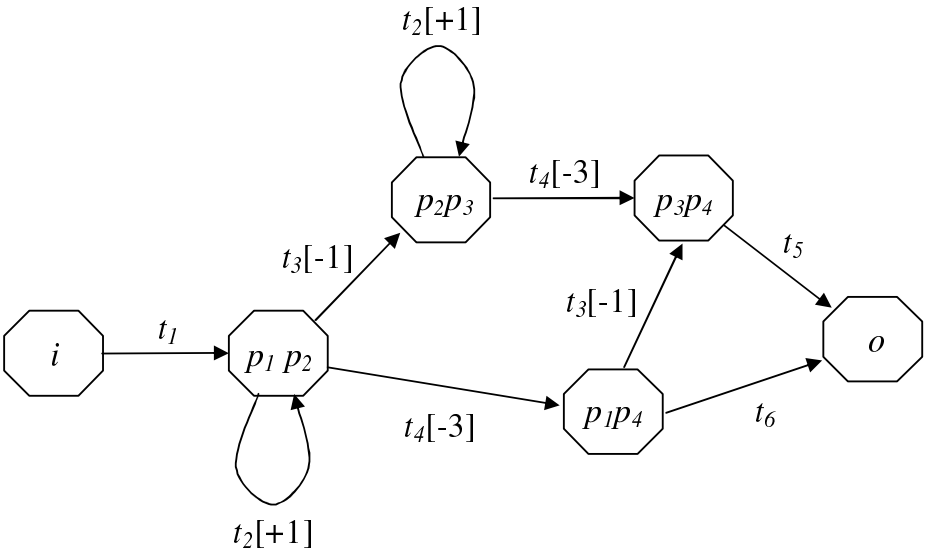


Рис. 3.14. Управляющий автомат для N_1

увеличивает значение счётчика на единицу.

Разрешимость бездефектности для размеченных сетей

Пусть $(N, i|r_0)$ — одномерная RWF-сеть с заданной начальной разметкой. Допуская вольность обозначений, будем тем же символом N обозначать её управляющий автомат, представленный в виде односчётчиковой сети.

Существует два типа нежелательного поведения сети, приводящего к нарушению бездефектности — тупик (deadlock) и динамический тупик (livelock), называемый также динамической блокировкой. Для краткости далее мы везде будем использовать термины “тупик” и “блокировка”.

Определение 3.15. *Состояние сети* $(c|r) \in C(N)\times Nat$ *называется* тупиком*, если* $c \neq o$ *и не существует перехода* $t \in T$*, такого, что* $(c|r) \xrightarrow{t} (c'|r')$.

Конечное подмножество $L \subset C(N)\times Nat$ *множества состояний сети называется* блокировкой*, если*

1. $|L| > 1$;
2. *для любых* $(c|r), (c'|r') \in L$ *существует конечная последовательность переходов* $\sigma \in T^*$*, такая, что* $(c|r) \xrightarrow{\sigma} (c'|r')$;

3. *для любых* $(c|r) \in L$ *и* $t \in T$, *таких, что* $(c|r) \xrightarrow{t} (c''|r'')$, *выполняется* $(c''|r'') \in L$.

Блокирующим состоянием *называется состояние, принадлежащее какой-либо блокировке.*

Заметим, что по определению $(o|r) \notin L$ для любого r.

Утверждение 3.16. *Если* $(c|r)$ *— тупик, то для любого* $t \in T$, *такого, что* $c \xrightarrow{t} c'$, *выполняется* $Supp(t) > r$.

Доказательство: Очевидно. □

Заметим, что из Утв. 3.16 следует, что $\mathrm{Eff}(t) < 0$, то есть мы можем его переформулировать:

Следствие 3.4. *Если* $(c|r)$ *— тупик, то:*

1. $\forall t \in T$, *такого, что* $c \xrightarrow{t} c'$ *для некоторого* c', *выполняется* $Eff(t) < 0$;
2. $r < min\{|Eff(t)| \ : \ c \xrightarrow{t} c' \ \text{для некоторого} \ c'\}$.

Таким образом, тупики могут возникать (1) только в управляющих состояниях, у которых все исходящие переходы — отрицательные; (2) только для конечного числа различных значений счётчика — когда ресурса недостаточно ни для одного из исходящих переходов.

Утверждение 3.17. *Множество тупиков конечно.*

Доказательство: Множество “потенциальных тупиков” — управляющих состояний с исходящими отрицательными переходами — конечно. Для данного “потенциального тупика” множество “опасных” значений счётчика (натуральных чисел, меньших, чем наименьший из ресурсов, требуемых для срабатывания исходящего перехода) также конечно. □

Следовательно, все тупики могут быть обнаружены перебором управляющих состояний.

Теперь рассмотрим блокировки.

Утверждение 3.18. *Если $L \subset C(N) \times Nat$ — блокировка, то найдутся состояние $(c|r) \in L$ и отрицательный переход $t \in T$, такие, что $c \xrightarrow{t} c'$ для некоторого c', и при этом $Eff(t) < -r$.*

Доказательство: Очевидно, поскольку управляющая подсеть RWF-сети N бездефектна. □

Утверждение 3.19. *Множество блокировок конечно.*

Доказательство: Во-первых, заметим, что если $(c|r), (c|r + x) \in L$, то L не является тупиком. Действительно, в этом случае последовательность переходов $(c|r) \xrightarrow{\sigma} (c|r + x)$ является положительным циклом, который порождает бесконечное множество состояний — а это противоречит конечности числа состояний, входящих в блокировку. Следовательно, каждое управляющее состояние может встретиться в блокировке не более одного раза.

Предположим противное: существует бесконечно много блокировок. Тогда среди них бесконечно много блокировок с одним и тем же набором управляющих состояний. Эти блокировки отличаются друг от друга лишь значением счётчика. Следовательно, в этом множестве блокировок существуют блокировки со сколь угодно большим значением счётчика, и мы можем выбрать такую, из которой будет достижимо управляющее состояние o — а это противоречит определению блокировки. □

Следовательно, все блокировки могут быть найдены перебором конечных подмножеств множества $C(N)$, замкнутых относительно срабатывания переходов (сильно связных компонент) и удовлетворяющих свойству, сформулированному в Утв. 3.18.

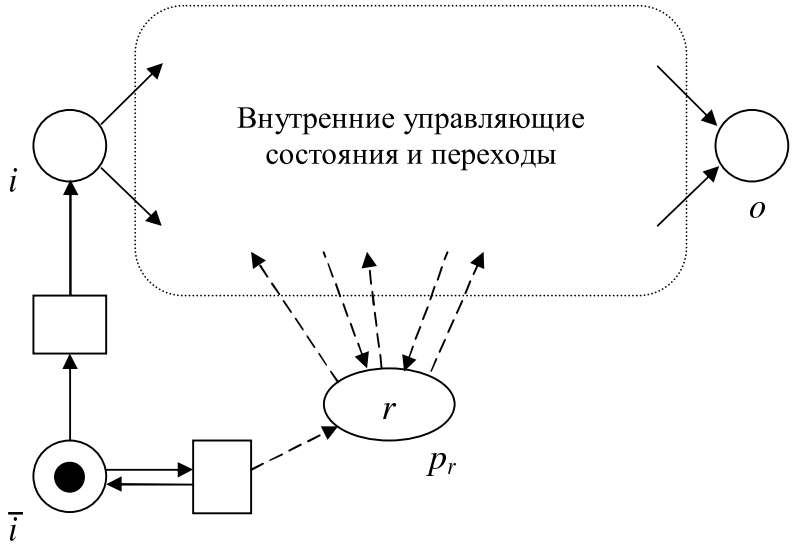


Рис. 3.15. Модифицированная RWF-сеть $\overline{N}$

Теорема 3.8. *Бездефектность разрешима для размеченных одномерных RWF-сетей.*

Доказательство: Следующее доказательство использует приём, подобный использованному в [198] для доказательства разрешимости структурной бездефектности WF-сетей.

Для данной одномерной RWF-сети N построим модифицированную RWF-сеть $\overline{N}$, добавив новую начальную позицию $\overline{i}$ и два новых перехода так, как показано на Рис. 3.15. Очевидно, что исходная сеть $(N, i|r)$ бездефектна тогда и только тогда, когда ни один из тупиков и ни одна из блокировок не достижимы в сети $(\overline{N}, \overline{i}|r)$.

Поскольку множества тупиков и блокировок конечны и вычислимы, проблема бездефектности размеченной одномерной RWF-сети сводится к проблеме достижимости в односчётчиковой сети Петри, которая является разрешимой (в частности, мы можем использовать для её решения алгоритм построения однопериодической свёртки пространства состояний). □

Разрешимость бездефектности для неразмеченных сетей

Определение 3.16. *(Неразмеченная) RWF-сеть N называется* бездефектной*, если существует ресурс r, такой, что размеченная сеть $(N, i|r)$ бездефектна.*

Теорема 3.8 предоставляет только полуразрешающую процедуру для проблемы бездефектности сети. С её помощью можно проверить бездефектность при данной начальной разметке, но если ответ будет отрицательным, то мы не будем знать, существует ли бóльшая начальная разметка, при которой сеть станет бездефектной.

Следствие 3.4 даёт нам необходимое условие для тупика, достижимого от *некоторой* начальной разметки. Далее мы докажем более сильную теорему, предоставляющую необходимое и достаточное условие существования нарушающего бездефектность тупика (то есть тупика, достижимого от бесконечного числа различных начальных разметок).

Теорема 3.9. *Неразмеченная одномерная RWF-сеть не бездефектна из-за тупиков тогда и только тогда, когда в ней существуют тупиковое состояние $(c|r)$, отрицательный генератор q и простой маршрут $q \xrightarrow{\sigma} c$, такие, что*

$$Eff(\sigma) \ominus Supp(\sigma) \leq r.$$

Доказательство: ($\Leftarrow$) Достаточно показать, что для любого (достаточно большого) начального ресурса r_0 найдётся ещё бóльший начальный ресурс $r_0 + x$, такой, что от $(i|r_0 + x)$ будет достижим тупик.

Рассмотрим произвольный достаточно большой начальный ресурс r_0, такой, что выполнено

$$(i|r_0) \xrightarrow{\tau} (q|s)$$

для некоторого маршрута τ и ресурса s (всегда можно подобрать такой ресурс, поскольку управляющая подсеть бездефектна, и, следовательно, любое управляющее состояние достижимо при достаточно большом начальном количестве

ресурса). Пусть $\theta = qc_1 \dots c_j q$ — простой отрицательный цикл с генератором q, то есть $\mathrm{Supp}(\theta) = -\mathrm{Eff}(\theta)$. Обозначим $z = s \bmod \mathrm{Supp}(\theta)$ и рассмотрим бóльший начальный ресурс $r_0 + z + \mathrm{Supp}(\sigma)$.

Мы имеем:

$$(i|r_0 + z + \mathrm{Supp}(\sigma))$$

$$\downarrow \tau$$

$$(q|s + z + \mathrm{Supp}(\sigma))$$

$$\downarrow \theta^{((s+z)/\mathrm{Supp}(\theta))}$$

$$(q|\mathrm{Supp}(\sigma))$$

$$\downarrow \sigma$$

$$(c|\mathrm{Eff}(\sigma) \ominus \mathrm{Supp}(\sigma))$$

и, следовательно, тупик.

($\Rightarrow$) Предположим противное: сеть не бездефектна из-за наличия тупиков, однако ни для одного тупикового состояния не выполнено свойство существования связанного отрицательного генератора.

Количество тупиковых состояний конечно, следовательно, некоторое тупиковое состояние $(c|r)$ достижимо от бесконечного числа различных начальных состояний (различных начальных значений счётчика).

Каждая последовательность переходов $\sigma = t_1.t_2.\dots.t_n$ от $(i|r_0)$ до $(c|r)$ соответствует маршруту σ в графе управляющего автомата. Поскольку существует бесконечно много порождающих данный тупик начальных состояний, множество соответствующих маршрутов также бесконечно. Каждый из этих маршрутов может быть разбит на последовательность чередующихся простых циклов и ациклических маршрутов:

$$\sigma = \tau_1(\theta_1)^{k_1}\tau_2(\theta_2)^{k_2} \dots \tau_{n-1}(\theta_{n-1})^{k_{n-1}}\tau_n.$$

Заметим, что такое разбиение не всегда единственно: например, *ababa* может быть представлено и как $(ab)^2a$, и как $a(ba)^2$. Для определённости мы всегда рассматриваем “декомпозицию справа налево”, то есть в данном случае выбираем вариант $a(ba)^2$.

Покажем, что среди всех этих маршрутов найдётся хотя бы один с отрицательным последним циклом θ_{n-1}. Действительно, если последний цикл положителен (или нейтрален) с эффектом x, то мы можем перейти к рассмотрению увеличенного начального ресурса $r_0 + x * k_{n-1}$ и укороченного маршрута

$$\sigma' = \tau_1(\theta_1)^{k_1}\tau_2(\theta_2)^{k_2}\dots\tau_{n-1}\tau_n,$$

имеющего то же самое завершение — тупик. Далее, новый маршрут σ' также может быть разбит на простые циклы и маршруты, затем его последний положительный цикл (если он опять окажется положительным) также может быть удален засчёт увеличения начального ресурса, и т.д. В конце этого процесса мы получим либо маршрут с отрицательным “последним циклом”, либо полностью ациклический маршрут (простой маршрут из i в c). В графе существует конечное число ациклических маршрутов, но у нас в наличии бесконечно много различных приводящих к тупику начальных разметок, так что второй вариант не может повторяться бесконечно много раз —- и, следовательно, мы неизбежно построим маршрут с отрицательным циклом.

Рассмотрим такой “тупиковый” маршрут σ'' с окончанием $\theta^k\tau$, где θ — отрицательный цикл, а τ — ациклический маршрут. Пусть $\theta = c_1c_2\dots c_i\dots c_mc_1$, где c_i — отрицательный генератор (по Лемме 3.5 такое c_i всегда существует). Маршрут $((c_i\dots c_mc_1)\tau)$ является простым маршрутом (напомним, что мы раскладывали последовательность “справа налево” и, следовательно, $\theta\tau$ не содержит циклов, отличных от θ). Поскольку финальным состоянием всего маршрута σ'' является $(c|r)$, для любого окончания ϕ маршрута σ'' выполняется

$$\mathrm{Eff}(\phi) \ominus \mathrm{Supp}(\phi) \le r.$$

Это выполняется и для $((c_i \dots c_n c_1)\tau)$. Но это простой маршрут, ведущий от отрицательного генератора к тупиковому управляющему состоянию — что и требовалось найти. □

Аналогичный Теореме 3.9 результат можно доказать и для блокировок:

Теорема 3.10. *Неразмеченная одномерная RWF-сеть не бездефектна из-за блокировок тогда и только тогда, когда в ней существуют блокирующее состояние* $(c|r)$, *отрицательный генератор* q *и простой маршрут* $q \xrightarrow{\sigma} c$, *такие, что*

$$Eff(\sigma) \ominus Supp(\sigma) \leq r.$$

Доказательство: Аналогично доказательству Теоремы 3.9. □

Следствие 3.5. *Бездефектность разрешима для неразмеченных одномерных RWF-сетей.*

Доказательство: Все простые (отрицательные) циклы могут быть найдены при помощи алгоритма Тарьяна, тупики и блокировки — при помощи перебора управляющих состояний и проверки свойств, указанных в Утв. 3.16 и Утв. 3.18 соответственно. Множество простых маршрутов конечно и может быть легко построено. □

Вычислимость минимального бездефектного ресурса

Рассмотрим “наивный” (и, возможно, не самый эффективный) алгоритм поиска минимального ресурса r, при котором сеть $(N, i|r)$ бездефектна. Этот алгоритм применим только к бездефектным сетям (бездефектность может быть проверена предварительно при помощи алгоритма, описанного в доказательстве Следствия. 3.5).

Алгоритм 3.4. *(вычисление минимального бездефектного ресурса)*
Input: Бездефектная одномерная RWF-сеть N.

Output: Наименьшее r, *такое, что сеть* $(N, i|r)$ *бездефектна.*

Шаг 0: Пусть $r := 0$.

Шаг 1: Проверим, бездефектна ли $(N, i|r)$: *будем искать тупики и блокировки, достижимые в размеченной служебной сети* $(\overline{N}, \bar{i}|r)$*(см. Теорему 3.8). Если "да", то возвращаем* r, *иначе* $r := r + 1$ *и возвращаемся на Шаг 1.*

Поскольку сеть бездефектна, количество итераций алгоритма конечно.

Глава 4

Модели с активными и обобщёнными ресурсами

Обыкновенные сети Петри представляют собой низкоуровневый формализм с очень простым набором основных элементов: позиция, переход, дуга и фишка. Отсутствуют сколько-нибудь удобные инструменты для высокоуровневых конструкций, таких как модуль и иерархия. Также обыкновенные сети Петри не вполне удобны для моделирования мультиагентных систем с динамической структурой, поскольку в них невозможно изменять в ходе функционирования сети структуру множества переходов. Моделировать возникновение новых агентов и исчезновение старых приходится изменением разметки вершин-позиций, тем самым разрешая или запрещая срабатывания зависящих от них переходов.

Существует ряд формализмов, построенных на основе обыкновенных сетей Петри, в которых тем или иным способом более явно выделяется понятие агента, а также вносятся конструкции модульности и иерархичности. В частности, в *сетях Петри высокого уровня* (например, в алгебраических сетях [127, 187, 193], раскрашенных сетях [113, 147, 148], объектных сетях [122, 123, 200, 201] и многих других [46, 90, 91, 116, 151, 155, 167, 178]) вводятся охранные функции на переходах и выражения на дугах, позволяющие усложнить условие срабатывания перехода и производимое им действие. Во вложенных сетях Петри [43, 44, 163] усложняется структура ресурса: он сам может быть сетью Петри. В рекурсивных сетях допускается даже неограниченная вложенность сетей-слоёв [45, 164]. В большинстве случаев получающиеся формализмы сопадают по выразительной мощности с обыкновенными сетями Петри, одно из редких исключений — вложенные сети.

Отдельным и гораздо менее развитым направлением исследований являются модификации базового языка сетей Петри с целью изменения самого способа взаимодействия между активной и пассивной составляющими модели. При этом

предлагается, в частности, более сбалансированное использование двудольности графа сети и прочих двойственностей в определении (позиция/переход, производство/потребление, агент/ресурс, …). Отметим, что ещё в [179] К.-А. Петри указывал, что

> “In general I would like to say that the exploitation of dualities is a main source of deep insight in any theory of dynamic systems as it is in mathematics; net theory abounds in dualities and this is not by chance.”

В [158] К. Лаутенбах предложил концепцию двойственных сетей позиций/переходов. В этом формализме переходы так же, как и позиции, помечаются специальными маркерами, названными “t-фишками”. Предназначение t-фишек в блокировании срабатывания соответствующих переходов. Переход, содержащий t-фишку, не может быть активен ни при какой разметке позиций. Позиция в сети может быть активной и срабатывает по “зеркальным” правилам. Срабатывания позиций изменяют разметку переходов (мультимножество t-фишек), дуги при этом берутся те же, что и для переходов, но перевернутые в обратную сторону. Таким образом, получившаяся сеть может быть легко дуализирована (“вывернута наизнанку”). К. Лаутенбах в своей работе предлагал использовать двойственные P/T-сети для моделирования процесса распространения системных сбоев.

В работах М. Колера и Х. Рольке [152–154] 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 „склеены“ 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 вложенной и иерархической структурой переходов).

4.1. Сети активных ресурсов

В супер-двойственных сетях сохранена двудольность графа сети — переход может быть связан дугой только с позицией, позиция — только с переходом. Однако здесь это разделение выглядит несколько искусственно — ведь в силу симметричности определений позиции и переходы в таких сетях обладают похожими свойствами. Возникает вопрос — что произойдет с формализмом сетей Петри при полном “слиянии” понятий позиции и перехода, то есть при отказе от требования двудольности графа?

4.1.1. Базовые определения и свойства

Определение 4.1. Сетью активных ресурсов (AP-сетью) *назовем набор* $AR = (V, I, O)$*, где*

- V *— конечное множество вершин (ресурсов);*
- $I : V \times V \to Nat$ *— множество потребляющих дуг;*
- $O : V \times V \to Nat$ *— множество производящих дуг.*

Графически вершины сети изображаются кружками, потребляющие дуги — пунктирными стрелками, производящие дуги — непрерывными стрелками (Рис. 4.1).

Естественно вводятся следующие термины:

Пусть $i = (v_1, v_2)$ — потребляющая дуга. Тогда дуга i называется *входной* для вершины v_2 и *потребляющей* для вершины v_1. Ресурс в вершине v_1 — *потребляемый* по дуге i, ресурс 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 *производящей* для вершины v_2. Ресурс в вершине v_1 — *производящий* по дуге o, ресурс в вершине v_2 — *производимый* по дуге o.

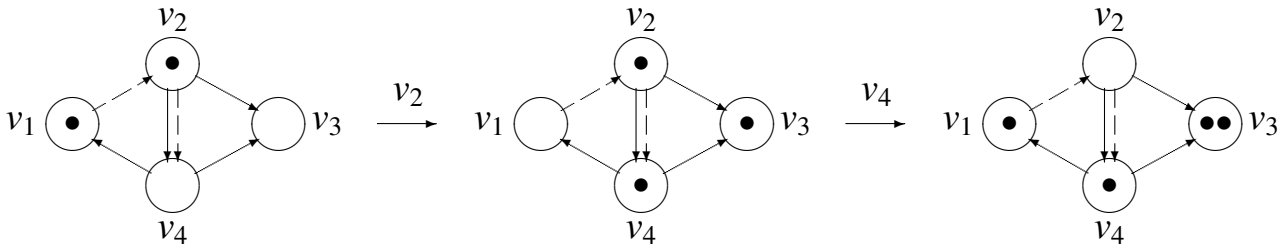


Рис. 4.1. Сеть активных ресурсов

Не бывает выходных потребляющих и входных производящих дуг. Один и тот же ресурс может быть одновременно производящим, потребляющим, производимым и потребляемым (по разным инцидентным дугам).

Определение 4.2. Размеченной сетью активных ресурсов *(сетью с начальной разметкой) назовем пару* (AR, M_0), *где* $AR = (V, I, O)$ *— сеть активных ресурсов,* $M_0 : V \to Nat$ *— начальная разметка.*

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

Определение 4.3. *Ресурс* $v \in V$ активен *при разметке* M*, если*

- $M(v) > 0$ *(узел* v *непустой);*
- $\forall w \in V \;\; M(w) \geq I(w, v)$ *(в потребляемых узлах содержится достаточное количество фишек).*

Активный при разметке M *ресурс* v *может* сработать*, порождая при этом новую разметку* M'*, где* $\forall w \in V \;\; M'(w) =_{def} M(w) - I(w, v) + O(v, w)$.

Таким образом, в срабатывании ресурса участвуют его входные и выходные дуги, в трансформации ресурса (изменении разметки соответствующей вершины) — потребляющие и производящие его дуги.

Структура АР-сетей отличается от структуры сетей Петри. Сеть активных ресурсов — это два ориентированных псевдографа на общем множестве вершин,

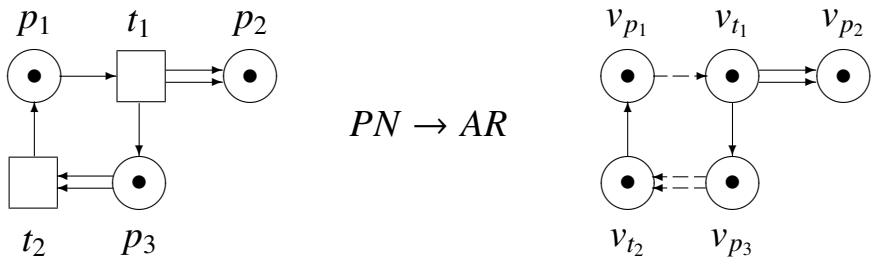


Рис. 4.2. Симулирование сети Петри АР-сетью

тогда как сеть Петри — это двудольный ориентированный псевдограф. При этом они определяют один и тот же класс систем:

Теорема 4.1. *Класс сетей активных ресурсов совпадает с классом обыкновенных сетей Петри.*

Доказательство: 1) Докажем, что для любой сети Петри существует эквивалентная АР-сеть.

Действительно, мы можем переделать сеть Петри в АР-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. Пример такой трансформации приведен на Рис. 4.2.

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

Каждой вершине v в АР-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.

Пример такой трансформации приведен на Рис. 4.3. □

Поскольку процедуры трансляции линейны по времени (это чисто синтаксические преобразования), все основные методы анализа сетей Петри (полное покрывающее дерево, инварианты, матричные методы, поведенческие эквивалентности и т.п. [41, 189]) могут быть легко обобщены [38] на АР-сети — про-

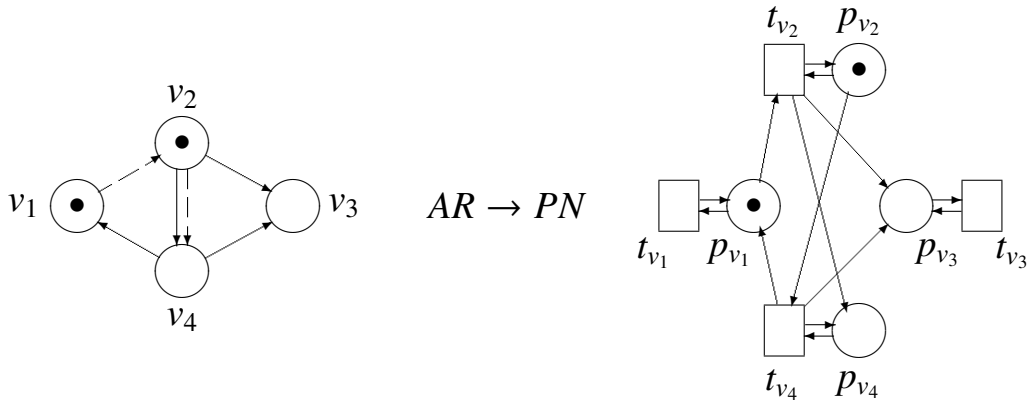


Рис. 4.3. Симулирование АР-сети сетью Петри

Рис. 4.4. Самоуничтожающее и самовоспроизводящее срабатывания

межуточная трансляция не требуется.

4.1.2. Возможности моделирования

Сети активных ресурсов представляют собой 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 АР-сетях появляется возможность моделирования динамического изменения набора агентов, вплоть до самоуничтожения и самовоспроизводства агента (пример приведен на Рис. 4.4).

Формализация ряда структурных семантических свойств системы (то есть свойств, не зависящих от её начального состояния) приведена в таблице 4.1.

Таблица 4.1. Формализация некоторых структурных свойств вершины v.

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

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 4.2.

На Рис. 4.5 приводятся примеры моделирования AP-сетями различных способов перемещения ресурсов (что также можно рассматривать как смену состояний каких-то неэлементарных компонентов системы, представленных в виде наборов вершин).

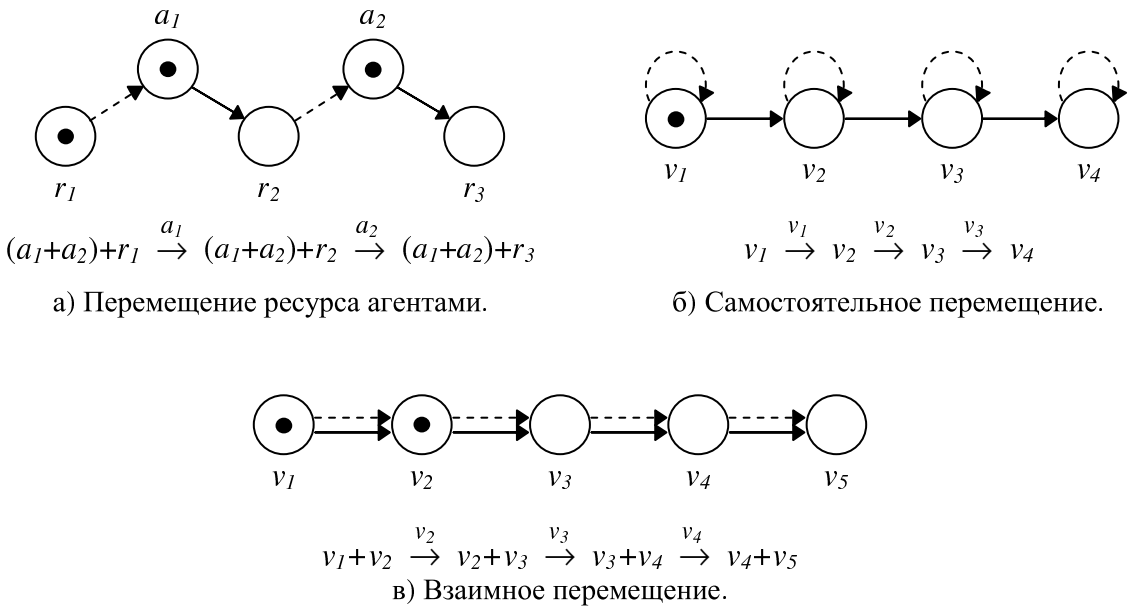
Первый способ соответствует тому, как ресурсы (фишки) переносятся в обыкновенных сетях Петри. В модель вводятся специальные фиксированные агенты („переходы“), которые занимаются исключительно переносом фишек и не могут сами ни возникать, ни пропадать.

Таблица 4.2. 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).

Следующие два примера используют специфические возможности языка AP-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 там для этого придется вводить дополнительные позиции и переходы, серьезно усложняющие модель.



а) Перемещение ресурса агентами.

б) Самостоятельное перемещение.

в) Взаимное перемещение.

Рис. 4.5. Моделирование перемещения (смены состояния) объекта

4.2. Модульные АР-сети

В настоящее время существует довольно много модификаций сетей Петри, в которых тем или иным способов добавляется модульность. В частности, существуют достаточно эффективные и применимые на практике высокоуровневые формализмы [92, 148, 201]. Многие авторы используют алгебраический подход к композиции и декомпозиции [93, 94], или же применяют какие-либо эффективные алгебраические методы модульной верификации [150]. Некоторые модели даже допускают рекурсию [45, 55, 134].

Как правило, в композиционных сетях Петри модули связываются между собой посредством синхронизированных переходов [43, 108, 150] или общих интерфейсных позиций [149]. Это — естественное следствие классического синтаксиса сетей Петри, в котором структура сети явно поделена на два класса элементов — позиции и переходы.

Определение АР-сети двойственно определению сети Петри — множество дуг явным образом (на уровне синтаксиса) разбито на два множества — входных и выходных дуг. При этом переходы и позиции объединены в единое множество

узлов. Таким образом, появляется возможность определить модульные трансформации не по отдельности для переходов и позиций (как это делается для сетей Петри, например в [93, 94, 108, 150], а в общем синтаксисе.

4.2.1. Модули в AP-сетях

В данном разделе исследуются композиционные свойства AP-сетей. Модуль определяется тривиально — как подсеть, заданная некоторым подмножеством узлов исходной сети. Интерфейсом модуля (набором его *связей*) является множество дуг, связывающих узлы модуля с узлами внешней подсети. Таким образом, модуль может иметь четыре типа связей: вход, выход, производство и потребление. Первые два из них представляют действия самого модуля, два других представляют действия соседей. Синтаксически модуль с инцидентными связями можно рассматривать как узел с инцидентными дугами — это обобщение является вполне естественным и позволяет сохранить однородность графа сети.

Определение 4.4. *Пусть $N = (V, I, O)$ — AP-сеть.* Модуль μ *сети N определяется некоторым подмножеством узлов $V_\mu \subseteq V$. Для модуля μ сети N обозначим:*

- $I_\mu = \{(v, v') \in I \mid v, v' \in V_\mu\}$ – *внутренние входные дуги;*
- $O_\mu = \{(v, v') \in O \mid v, v' \in V_\mu\}$ – *внутренние выходные дуги;*
- $N_\mu = (V_\mu, I_\mu, O_\mu)$ – *сеть модуля μ;*
- $A^i_\mu = \{(v, v') \in I \mid v \in (V \setminus V_\mu), v' \in V_\mu\}$ – *входные связи;*
- $A^o_\mu = \{(v, v') \in O \mid v \in V_\mu, v' \in (V \setminus V_\mu)\}$ – *выходные связи;*
- $R^i_\mu = \{(v, v') \in I \mid v \in V_\mu, v' \in (V \setminus V_\mu)\}$ – *потребляющие связи;*
- $R^o_\mu = \{(v, v') \in O \mid v \in (V \setminus V_\mu), v' \in V_\mu\}$ – *производящие связи.*

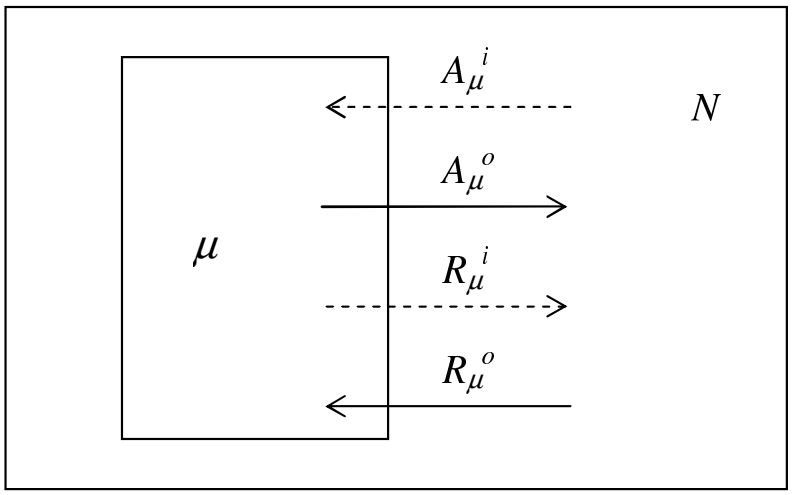


Рис. 4.6. Четыре типа связей в модульных АР-сетях

Неформально, A-связи описывают наблюдаемое поведение модуля, R-связи описывают его роль в качестве ресурса (Рис. 4.6).

Для размеченной сети (N, M_0) и модуля μ размеченная сеть модуля $(N_\mu, (M_0)_\mu)$ определяется очевидным образом: $(M_0)_\mu =_{\text{def}} M_0[V_\mu]$.

Определим также *дополнение* $\overline{\mu}$ модуля μ как модуль, заданный подмножеством узлов $V \setminus V_\mu$. Дополнение модуля может рассматриваться как *системная подсеть* сети.

На Рис. 4.7 приведена хорошо известная модель обедающих философов. Данная задача получила статус классической и в разных вариантах упоминается во множестве учебников. Это объясняется тем, что в ней при кажущейся простоте имеются все элементы распределенных систем: выбор, конфликт, распараллеливание и синхронизация. Философы сидят за столом с общим блюдом спагетти посередине (dish). Между каждыми двумя философами на салфетке лежит вилка (fork). Есть спагетти можно только одновременно двумя вилками, поэтому соседи потенциально конкурируют. Каждый философ может находиться в одном из двух состояний: он либо ждёт (waiting), либо ест (eating). Переходы между состояниями — take (взял обе вилки) и put (вернул на место). В состоянии eating он также может выполнить действие eat (взять спагетти).

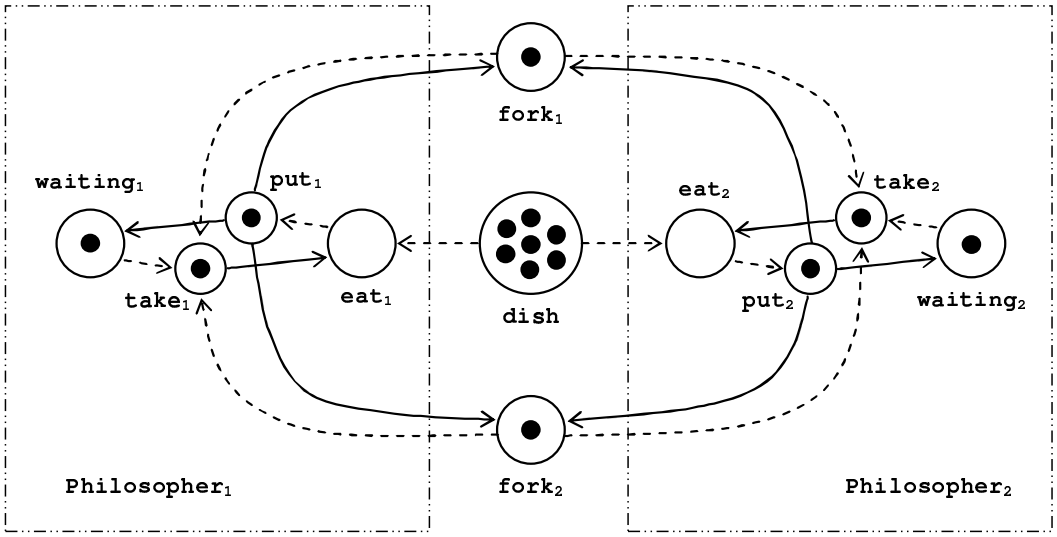


Рис. 4.7. Два обедающих философа

Для простоты мы рассматриваем только двух участников. Определен модуль, представляющий первого философа. Заметим, что он имеет только входные и выходные связи (элементы множеств A^i_μ и A^o_μ), то есть этот модуль может рассматриваться в качестве чистого агента.

Модуль в АР-сети внешне очень похож на отдельный узел: он тоже может производить и потреблять ресурсы других модулей, а его ресурсы, в свою очередь, могут быть потреблены или произведены другими модулями. Более того, взаимосвязи между модулями естественным образом обозначены теми же конструктивными элементами, что и на “низком” уровне узлов: входными и выходными дугами (связями). Таким образом, получающийся иерархический синтаксис весьма универсален и компактен.

Особенно интересны модули, обладающие не всеми четырьмя типами связей. Мы будем называть модуль μ A-модулем (соответственно, R-модулем), если он имеет только A-связи (соответственно, R-связи). Например, $\texttt{philosopher}_1$ является A-модулем. Модули с более ограниченным интерфейсом будут обозначаться с использованием соответствующего верхнего индекса: например, $\mathrm{A}^i\mathrm{R}^o$-модуль имеет только входные и производящие связи. Произвольная АР-сеть может рассматриваться как композиция модулей различных типов (Рис. 4.8).

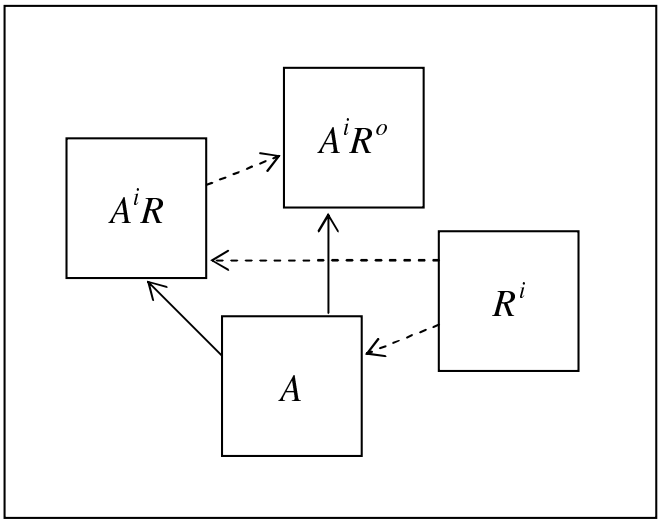


Рис. 4.8. Связи определяют роль модуля в системе.

4.2.2. Наследственные свойства

Рассмотрим понятие наследственного свойства из теории графов [57, 97]. Свойство *Prop* называется наследственным, если из наличия *Prop* у графа G следует наличие *Prop* у произвольного подграфа H графа G.

Примерами наследственных свойств являются полнота, планарность, двудольность, ацикличность. Не является наследственной связность.

Здесь мы рассмотрим два основных семантических свойства сетей Петри — ограниченность и живость. Напомним их определения в терминах сетей активных ресурсов.

Узел v *ограничен* в размеченной сети (N, M_0), если существует натуральное $n \in Nat$, такое что для любой достижимой разметки $M \in \mathcal{R}(N, M_0)$ выполняется $M(v) \leq n$. Размеченная сеть *ограничена,* если все её узлы ограничены.

Узел v *жив* в (N, M_0), если для любой $M \in \mathcal{R}(N, M_0)$ найдётся $M' \in \mathcal{R}(N, M)$, такая что v активен в M'. Размеченная сеть *жива,* если живы все узлы, обладающие инцидентными входными или выходными дугами.[1]

Очевидно, что ограниченность и живость не являются наследственными в

[1] Узлы–ресурсы, имеющие только потребляющие или производящие дуги, являются “структурно мёртвыми”, поэтому их мы не учитываем.

общем случае. Тем не менее, классификация модулей позволяет найти некоторые случаи конструктивного наследования (направленного как вниз — от сети к подсетям, так и вверх — от подсети к сети в целом):

Теорема 4.2. *Пусть* (N, M_0) *— размеченная сеть,* μ *— модуль сети* N*. Тогда:*

1. *если* (N, M_0) *ограничена, а* μ *—* A^oR*-модуль, то* $(N_\mu, (M_0)_\mu)$ *ограничена;*
2. *если* (N, M_0) *жива, а* μ *—* A^oR^i*-модуль, то* $(N_\mu, (M_0)_\mu)$ *жива;*
3. *если* $(N_\mu, (M_0)_\mu)$ *жива, а* μ *—* A^o*-модуль, то* $(N_{\overline{\mu}}, (M_0)_{\overline{\mu}})$ *неограничена;*
4. *если* $(N_\mu, (M_0)_\mu)$ *ограничена, а* μ *—* R^i*-модуль, то* $(N_{\overline{\mu}}, (M_0)_{\overline{\mu}})$ *не жива.*

Доказательство: (1) Очевидно, удаление выходных дуг не может сделать ограниченную сеть неограниченной. Следовательно, сеть (N', M_0), где N' получена из N удалением всех A^o-связей модуля μ, всё так же ограничена.

Теперь модуль μ является R-модулем, и, следовательно, не зависит от статической разметки системной подсети $\overline{\mu}$ – только от её активных действий (срабатываний узлов). Системная подсеть с точки зрения R-модуля — множество чистых агентов (переходов сети Петри) с “непредсказуемым” поведением. Но удаление агентов (переходов) не может сделать сеть неограниченной — в противном случае было бы достаточно рассмотреть последовательности срабатываний только “прочих” (оставшихся) переходов в исходной сети для доказательства её неограниченности.

(2) Срабатывания A^oR^i-модуля не зависят от разметки системной подсети $\overline{\mu}$. С другой стороны, срабатывания $\overline{\mu}$ могут только уменьшить разметку μ. Следовательно, не-живость узла v в μ повлекла бы за собой не-живость этого же узла в сети в целом.

(3)-(4) Очевидно. □

Среди всех 11 возможных типов модулей чистые A- и R-модули (агенты и ресурсы) являются наиболее важными. Оказывается, любой интерфейс между

любыми двумя модулями может быть преобразован в эквивалентный относительно достижимости A/R-интерфейс, при котором один из модулей является агентом, другой — ресурсом. Рассмотрим простую процедуру преобразования входных и выходных связей в производящие и потребляющие связи:

Лемма 4.1. *Пусть* (N, M_0) *— размеченная AP-сеть,* $v \in V$ *— узел, такой что* $I(\bullet, v) \neq \varnothing$ *или* $O(v, \bullet) \neq \varnothing$ *(*v активен: он может потреблять и/или производить фишки*).*

Пусть N' *— сеть, построенная из* N *удалением всех дуг, задействованных в* $I(\bullet, v)$ *и* $O(v, \bullet)$ *(*v становится пассивным*), и добавлением нового узла* v_t *с* $I(v_t, \bullet) = O(\bullet, v_t) = \varnothing$ *(*v_t не может быть произведен или потреблен*),* $I(\bullet, v_t) = I(\bullet, v) \cup \{(v, v_t)\}$*,* $O(v_t, \bullet) = O(v, \bullet) \cup \{(v_t, v)\}$ *(*v_t симулирует в N' срабатывание v в N*).*

Пусть M'_0 *— разметка сети* N'*, такая что* $M'_0[V] = M_0$*,* $M'_0(v_t) = 1$*. Тогда*

$$\mathcal{R}(N, M_0) = \mathcal{R}(N', M'_0) \cap (V \times V) \text{ и } \forall M' \in \mathcal{R}(N', M'_0) \;\; M'(v_t) = 1.$$

Доказательство: Изображение подобной трансформации приведено на Рис. 4.9. Рассмотрены все возможные связи узла. В действительности мы всего лишь разделили активные и пассивные свойства узла v. Новый узел v_t является *переходом*: он ведёт себя в точности так же, как переход обыкновенной сети Петри. Аналогично, узел v в N' по терминологии сетей Петри является *позицией.* □

Реструктуризация, описанная в Лемме 4.1, расширяет множество узлов сети на один узел-переход, *всегда* помеченный одной фишкой. Так что мы можем не учитывать его при рассмотрении множества достижимости новой сети.

Следствие 4.1. *Произвольный модуль AP-сети может быть преобразован в R-модуль без изменения множества достижимости сети.*

Таким образом, интерфейс любого модуля может быть упрощен до R-интерфейса или A-интерфейса. Разумеется, при этом мы меняем структуру сети (добавляется v_t). Однако эта модификация локальна и не затрагивает “внутреннюю”

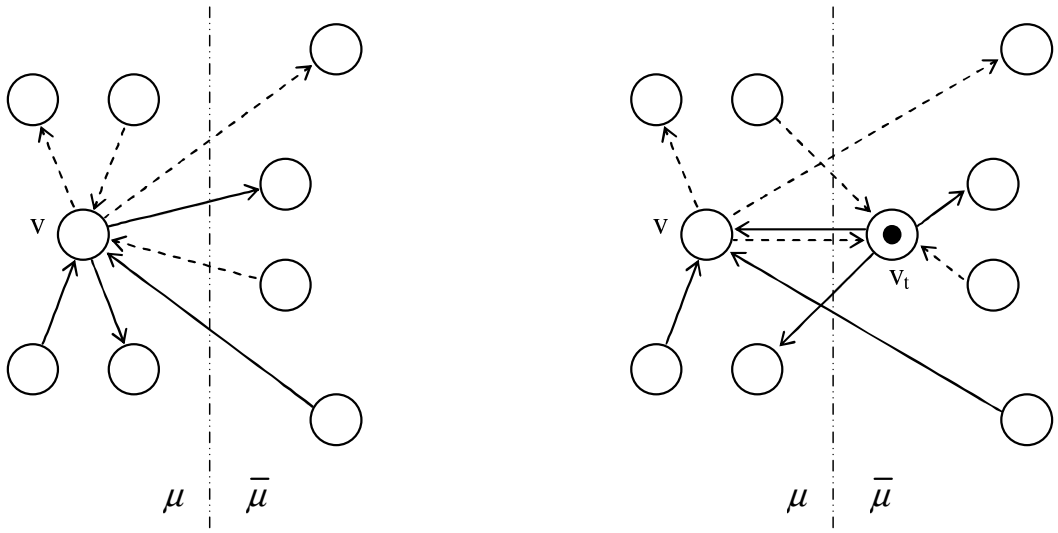


Рис. 4.9. Трансформация узла в пару (позиция, переход).

часть модуля. На практике A- и R-модули могут рассматриваться в качестве “активных” и “пассивных” частей системы (“управление” и “данные”). Лемма 4.1 утверждает, что такое разделение относительно: мы легко можем преобразовать агент в ресурс и наоборот. Такая двойственность в модульных AP-сетях проявляется очень четко.

Мы будем называть такую трансформацию *R-нормализацией* модуля μ и обозначать $\mu^{\langle R\rangle}$.

Следствие 4.2. *Пусть* (N, M_0) *— размеченная AP-сеть,* μ *— модуль сети* N*. Тогда*

1. (N, M_0) *ограничена* $\Rightarrow (N_{\mu^{\langle R\rangle}}, (M_0)_{\mu^{\langle R\rangle}})$ *ограничена;*

2. $(N_{\mu^{\langle R\rangle}}, (M_0)_{\mu^{\langle R\rangle}})$ *неограничена* $\Rightarrow (N, M_0)$ *неограничена.*

Доказательство: Следует из Теоремы 4.2 (утверждения (1) и (3)) и определения нормализации. Заметим, что нормализованная сеть обладает тем же отношением достижимости, что и исходная, и, следовательно, сохраняет ограниченность. □

Определение 4.5. Плоской модуляризацией Ω *сети* N *называется разбиение* V *на непересекающиеся модули* $\{\mu_1, \ldots, \mu_n\}$.

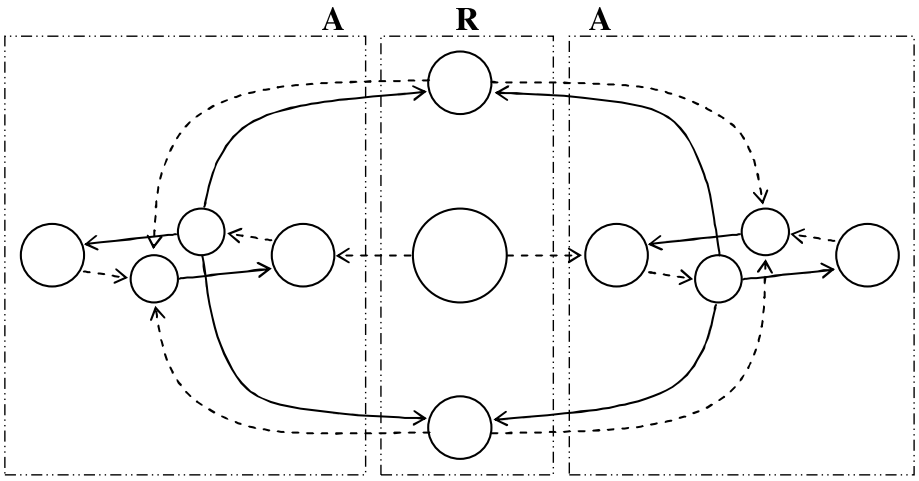


Рис. 4.10. Плоская A/R-модуляризация

Плоская модуляризация называется плоской A/R-модуляризацией*, если каждый модуль является либо A-модулем, либо R-модулем.*

Следствие 4.3. *Пусть* $\Omega = \{\mu_1, \ldots, \mu_n\}$ *— плоская модуляризация сети N. Пусть G — граф с вершинами из* Ω*, в котором вершины* μ_i *и* μ_j *соединены ребром тогда и только тогда, когда существует связь между модулями* μ_i *и* μ_j *в N.*

Сеть N может быть преобразована в эквивалентную (относительно достижимости) сеть N'*, для которой* Ω *является A/R-модуляризацией, тогда и только тогда, когда хроматическое число графа G равно 2.*

Доказательство: Модули одного цвета преобразуются в A-модули, другого — в R-модули. □

Следствие 4.3 утверждает, что любое двухцветное разбиение сети может быть преобразовано в разбиение на агенты и ресурсы. Следовательно, мы можем достаточно просто находить управляющие структуры и структуры данных, соответствующие конкретному разбиению: управляющие модули, обращающиеся к общим данным, модули данных, имеющие общее управление, цепочки связанных модулей и т.п. Более того, подсети управления и хранения данных дуализируемы (Следствие 4.1).

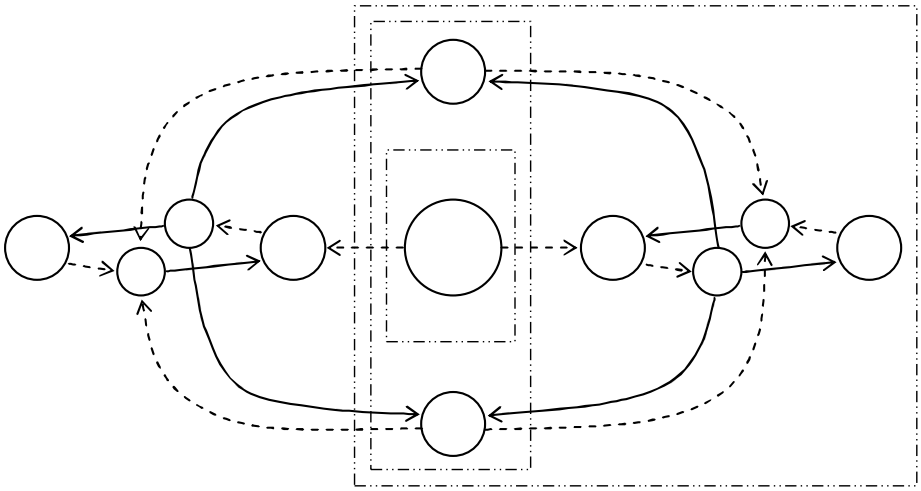


Рис. 4.11. Вложенная R-модуляризация

Определение 4.6. Вложенной модуляризацией Ω *сети N называется разбиение V на модуль* μ *и системную часть* $\overline{\mu}$*, где* μ*, возможно, тоже модуляризован.*

Вложенная модуляризация называется вложенной α-модуляризацией*, если каждый модуль является* α*–модулем (*$\alpha \in \{A/R, A, R\}$*).*

Следствие 4.4. *Для любой вложенной модуляризации* Ω *сети N эта сеть может быть преобразована в эквивалентную (относительно достижимости) сеть N′, в которой* Ω *является вложенной* α*-модуляризацией N′ (*$\alpha \in \{A/R,A,R\}$*).*

Доказательство: Очевидно. Трансформацию (где она необходима) начинаем с самого внутреннего модуля. □

Вложенная модуляризация позволяет нам конструировать иерархическую структуру с любым типом межуровневых взаимодействий. Возможна также комбинация плоской и вложенной модуляризации.

A/R-модуляризации интересны в силу того, что позволяют выстроить естественную иерархию на множестве модулей. Подобная иерархия может иметь много приложений в расширенных формализмах. Например, активные модули могут быть ответственны за межмодульную передачу данных. Другой пример — безопасность: R-модуль способен скрывать точные моменты срабатывания сво-

их переходов от агента (A-модуля), поскольку агент наблюдает уже измененное состояние ресурса.

Как уже отмечалось выше, для обыкновенных сетей Петри существует достаточно много методов модульного и композиционального анализа [43, 86, 93, 94, 108, 149, 150]. Основное отличие нашего подхода состоит в том, что в качестве межмодульного интерфейса рассматривается не множество каких-то общих элементов (разделяемых ресурсов-позиций [108, 150] или синхронизирующих переходов [43, 108]), а набор связей между узлами двух соединяемых модулей (т.е. набор дуг, пересекающих “периметр”). Тем самым упрощается математическое определение самого модуля — не требуется никаких дополнительных обозначений (меток синхронизации [43], разделяемых позиций [108], интерфейсных переходов [150] и т.п.) и ограничений на исходную структуру сети. Отметим также, что предложенный способ декомпозиции в силу своей простоты допускает, в частности, очевидные многоуровневые и рекурсивные обобщения.

4.2.3. Эквивалентность модулей

В этом разделе мы рассматриваем эквивалентные модули. Ключевая проблема — проверка того, может ли один модуль быть заменен другим без нарушения поведения системы в целом. Мы исследуем эквивалентность отношений достижимости двух получившихся сетей — фундаментальную поведенческую эквивалентность, которая сильнее эквивалентности трасс и бисимулярности.

Определение 4.7. *Рассмотрим AP-сети* N_1 *и* N_2 *и их модули* μ_1 *и* μ_2 *соответственно, такие что* $(N_1)_{\overline{\mu}_1} = (N_2)_{\overline{\mu}_2} = N_{sys}$ *для некоторой AP-сети* $N_{sys} = (V_{sys}, I_{sys}, O_{sys})$ *(общей системной сети). Рассмотрим разметки* M_1, M_2 *и* M_{sys} *сетей* μ_1, μ_2 *и* N_{sys} *соответственно.*

Размеченные модули (μ_1, M_1) *и* (μ_2, M_2) *назовём* эквивалентными относительно системной достижимости *для размеченной системной сети* (N_{sys}, M_{sys})

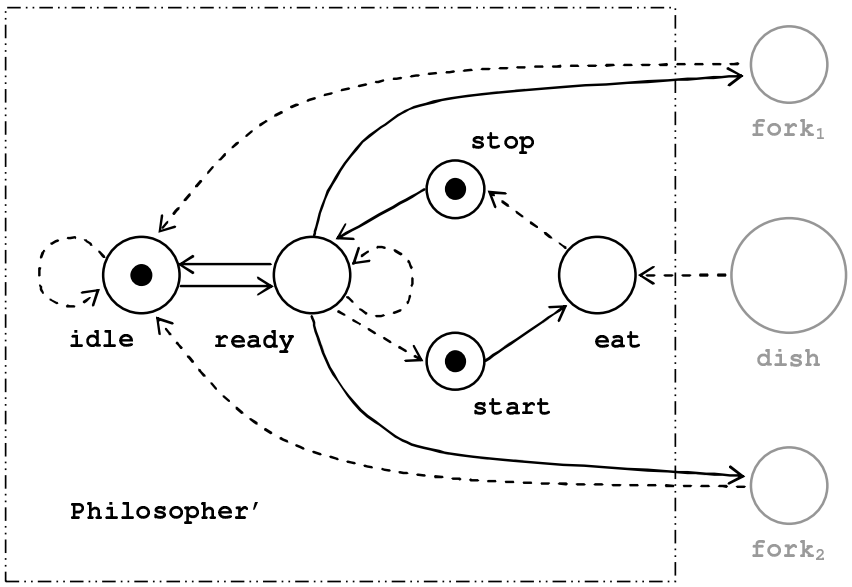


Рис. 4.12. СД-эквивалентный философ

(СД эквивалентными), если

$$\mathcal{R}(N_1, M_1 + M_{sys})[V_{sys}] = \mathcal{R}(N_2, M_2 + M_{sys})[V_{sys}].$$

Говоря неформально, два СД-эквивалентных модуля одинаково воздействуют на системную часть сети. Они взаимозаменяемы без “порчи” системной достижимости.

Пример приведен на Рис. 4.12. Модуль `Philosopher'` СД-эквивалентен модулю $\texttt{Philosopher}_1$, изображенному на Рис. 4.7. Но при этом модули существенно различны: у `Philosopher'` имеется новое состояние `ready`, в котором у него есть обе вилки, но при этом он не ест.

Теорема 4.3. *СД-эквивалентность неразрешима для АР-сетей.*

Доказательство: Следует из неразрешимости эквивалентности множеств достижимости в сетях Петри. Действительно, мы можем поместить все “узлы-агенты” (“переходы”) сравниваемых сетей в соответствующие модули (а все “узлы-ресурсы”, то есть “позиции”, — в системные сети) и попытаться проверить их СД-эквивалентность. □

Заметим, что в доказательстве мы использовали активные модули (А-модули). Следовательно, СД-эквивалентность неразрешима уже даже для А-модулей. Интересно было бы исследовать эту эквивалентности и для других специфических типов модулей, в частности, для R-модулей. Мы полагаем, что для них она также неразрешима.

Рассмотрим более ограниченный случай. Пусть оба сравниваемых модуля имеют одинаковый интерфейс, то есть одно и то же множество связей и связанных узлов системной сети.

$\forall v \in V_{sys}$

$$\sum_{v_1 \in V_{\mu_1}} A^o_{\mu_1}(v_1, v) = \sum_{v_2 \in V_{\mu_2}} A^o_{\mu_2}(v_2, v); \quad \sum_{v_1 \in V_{\mu_1}} R^i_{\mu_1}(v_1, v) = \sum_{v_2 \in V_{\mu_2}} R^i_{\mu_2}(v_2, v);$$
$$\sum_{v_1 \in V_{\mu_1}} A^i_{\mu_1}(v, v_1) = \sum_{v_2 \in V_{\mu_2}} A^i_{\mu_2}(v, v_2); \quad \sum_{v_1 \in V_{\mu_1}} R^o_{\mu_1}(v, v_1) = \sum_{v_2 \in V_{\mu_2}} R^o_{\mu_2}(v, v_2).$$

Будем говорить, что модуль *совместим* с системной сетью (и наоборот), если сеть содержит все узлы, требующиеся для внешних связей модуля.

Определение 4.8. *Размеченные модули* (μ_1, M_1) *и* (μ_2, M_2)*, имеющие одинаковый интерфейс, называются* универсально эквивалентными относительно системной достижимости *(УСД-эквивалентными), если они СД-эквивалентны для любой совместимой размеченной системной сети.*

УСД-эквивалентность двух модулей означает, что они производят одинаковые множества разметок на пассивных интерфейсных узлах системы (по активным агентским связям) и подчиняются одинаковым наборам ограничений и команд, поступающих с активных интерфейсных узлов системы (по пассивным ресурсным связям).

УСД-эквивалентность является сужением СД-эквивалентности. Однако мы полагаем, что она также неразрешима.

Рассмотрим одну из простейших нетривиальных замен модуля – пусть один из модулей (обозначенный μ) будет произвольной АР-сетью (с некоторыми огра-

ничениями), а другой (обозначенный ν) - простейшей АР-сетью, то есть отдельным узлом без дуг.

Теорема 4.4. *Пусть* (μ, M) *— размеченный модуль, такой что*

1. *Все активные интерфейсные узлы модуля* μ *имеют одинаковые наборы (мультимножества) активных связей:*

$$\exists A^i, A^o \in \mathcal{M}(V_{sys}) \quad \forall v \in V_\mu$$
$$(A^i_\mu(\bullet, v) = A^o_\mu(v, \bullet) = \varnothing) \ \vee \ (A^i_\mu(\bullet, v) = A^i \ \wedge \ A^o_\mu(v, \bullet) = A^o).$$

2. *Все активные интерфейсные узлы системной сети оперируют целыми количествами одного и того же мультимножества внутренних узлов:*

$$\exists R^{io} \in \mathcal{M}(V_\mu) \quad \forall v \in V_{sys} \quad \exists k^i(v), k^o(v) \in Nat$$
$$(R^i_\mu(\bullet, v) = k^i(v) \times R^{io} \ \wedge \ R^o_\mu(v, \bullet) = k^o(v) \times R^{io}).$$

3. *Все активные внутренние узлы модуля, воздействующие на узлы из* R^{io}, *выполняют одну и ту же операцию над целым количеством мультимножеств* R^{io}*:*

$$\exists k^c, k^p \in Nat \quad \forall v \in V_\mu \quad (A^i_\mu(\bullet, v) \cap R^{io} \neq \varnothing \ \vee \ A^o_\mu(v, \bullet) \cap R^{io} \neq \varnothing) \quad \Rightarrow$$
$$(\exists X, Y \in \mathcal{M}(V_\mu) \quad (X \cap R^{io} = Y \cap R^{io} = \varnothing) \wedge$$
$$(A^i_\mu(\bullet, v) = k^c \times R^{io} + X) \ \wedge \ (A^o_\mu(v, \bullet) = k^p \times R^{io} + Y)).$$

4. *Разметка* M *может быть представлена как* $M = M' + m \times R^{io}$, *где* $m \in Nat$ *и* $M' \cap R^{io} = \varnothing$.

5. *Размеченная внутренняя сеть* $(N_\mu, \overline{M})$ *жива, где* $\overline{M}$ *обозначает разметку, полученную из* M *обнулением разметок всех пассивных интерфейсных узлов модуля* μ :

$$\forall v \in V_\mu \quad \overline{M}(v) =_{def} \begin{cases} 0 & \text{if } (A^i_\mu(\bullet, v) \cup A^o_\mu(v, \bullet)) \neq \varnothing; \\ M(v) & \text{в противном случае.} \end{cases}$$

Тогда модуль (μ, M) *УСД-эквивалентен модулю-узлу* (ν, M_ν), *где*

- $V_\nu = \{w\}$;
- $I_\nu(w, w) = k^c;\ O_\nu(w, w) = k^p$;
- $A^i_\nu(\bullet, w) = A^i;\ A^o_\nu(w, \bullet) = A^o$;
- $\forall v \in V_{sys} \quad R^i_\nu(w, v) = k^i(v),\ R^o_\nu(v, w) = k^o(v)$;
- $M_\nu(w) = m$.

Доказательство: Доказательство чисто техническое. Оно основано на том, что живость “недоразмеченного” модуля влечёт живость этого же модуля при любой большей начальной разметке (которая может быть получена вводом данных из системы). Система не зависит от реального поведения внутренней сети (заметьте, что эффект от всех интерфейсных срабатываний одинаков). Поэтому достаточно знать, что хотя бы какое-нибудь из срабатываний рано или поздно станет возможным. □

Пример корректной замены приведен на Рис. 4.13. Здесь модуль `Procedure` (живой и неограниченный) моделирует некие вычисления, которые могут быть выполнены одним из трех компьютеров, изначально размещенных в узле `idle`. Сервис `loader` запускает вычисление, загружая в один из компьютеров входные данные. Компьютер выполняет вычисления (возможно, бесконечно долго) и время от времени производит результаты (в `output`). Он также может быть выгружен (остановлен) другим сервисом `unloader`. Внешнее поведение модуля Procedure таково, что он может быть заменен единичным узлом.

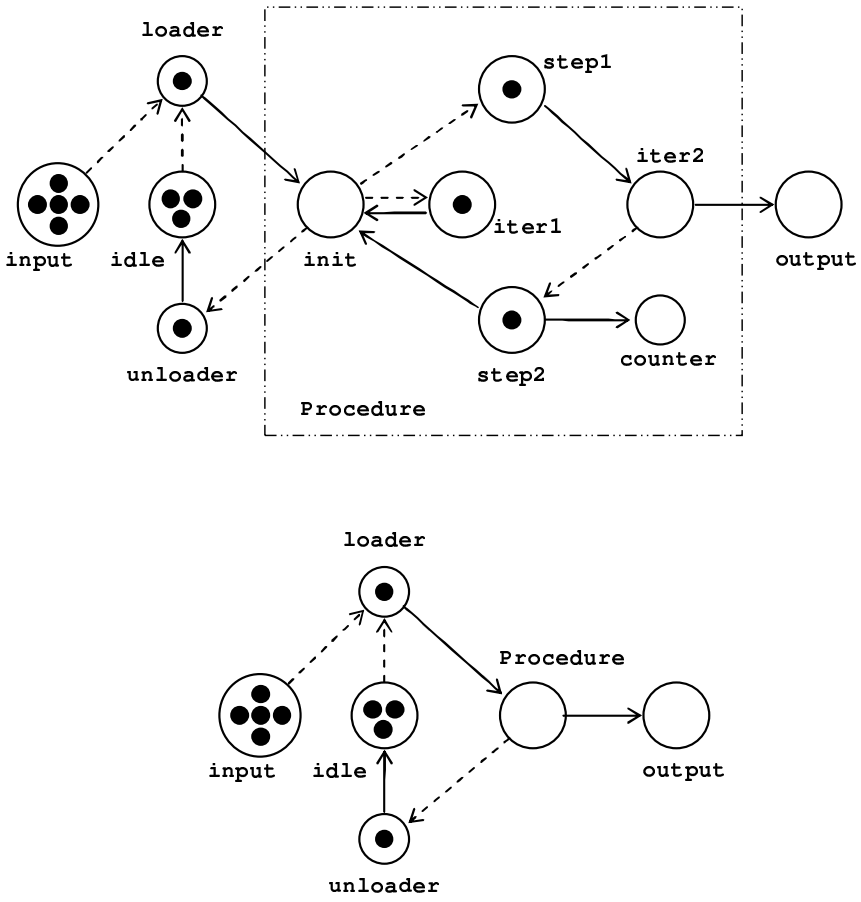


Рис. 4.13. Замена модуля УСД-эквивалентным узлом

4.3. Модифицированные АР-сети

4.3.1. Расширения класса

В данном разделе определяются и исследуются расширения класса сетей активных ресурсов, получающиеся при введении в модель нелокальных проверок памяти (ингибиторные дуги) и нелокальных действий над памятью (обнуляющие дуги).

Сети с ингибиторными дугами

Одним из самых известных расширений класса сетей Петри являются сети Петри с ингибиторными дугами. Это сети, в которых кроме обычных дуг могут присутствовать так называемые ингибиторные, обозначаемые стрелками с наконечниками в виде черных точек (Рис. 4.14).

Ингибиторная дуга всегда ведет от позиции к переходу. Переход может сработать только тогда, когда во всех позициях, которые связаны с ним ингибиторными дугами, отсутствуют фишки. Таким образом, при помощи этих моделей можно осуществлять проверку позиции на пустоту. Известно, что сети Петри с ингибиторными дугами эквивалентны машинам Тьюринга. Таким образом, для них неразрешимы все основные алгоритмические проблемы: останова, ограниченности, достижимости, живости и др.

Рассмотрим, каким образом можно вводить нелокальные проверки памяти в сетях активных ресурсов. Вначале нам потребуется ряд вспомогательных утверждений.

Теорема 4.5. *Для любой сети Петри с ингибиторными дугами существует эквивалентная ей сеть, в которой каждой позиции инцидентно не более одной ингибиторной дуги.*

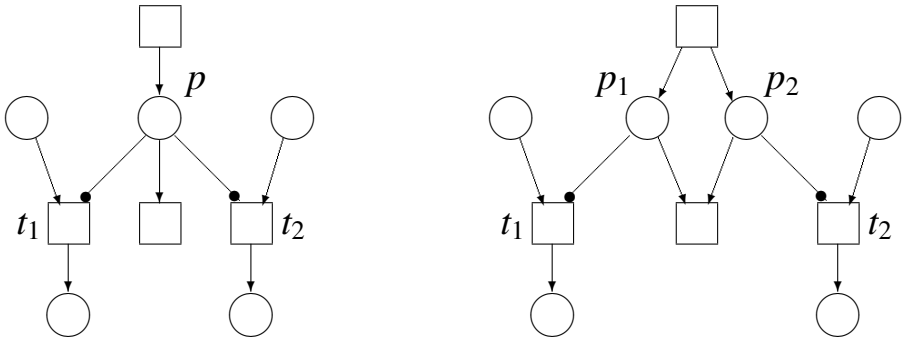


Рис. 4.14. Построение сети, в которой каждой позиции инцидентно не более одной ингибиторной дуги

Доказательство: В качестве доказательства рассмотрим следующее преобразование.

Пусть в исходной сети позиции p инцидентны две ингибиторные дуги — (p, t_1) и (p, t_2). Преобразуем исходную сеть, заменив в ней позицию p на две позиции — p_1 и p_2 с такими же наборами обыкновенных дуг, что и у исходной позиции p. Добавим ингибиторные дуги (p_1, t_1) и (p_2, t_2).

(Пример преобразования изображен на Рис. 4.14. Здесь и далее для иллюстрации преобразований мы рассматриваем в качестве исходной сети (слева) схему, включающую все возможные связи моделируемого синтаксического элемента с другими компонентами системы. В данном случае рассматривается позиция и две инцидентные ей ингибиторные дуги, а также полный набор всевозможных различных связанных с ними элементов. Так, с позицией связаны два перехода – входной и выходной. С переходами, в которые ведут ингибиторные дуги, могут быть связаны входные и выходные позиции.)

В качестве результирующей сети (справа) изображена сеть после преобразований. Переходы и позиции без подписей соответствуют таким же по расположению переходам и позициям исходной сети.

Начальную разметку получим из исходной разметки M, поместив в p_1 и p_2 по $M(p)$ фишек.

Очевидно, что из-за полного совпадения начальной разметки и наборов

инцидентных обычных дуг разметки p_1 и p_2 всегда будут совпадать. □

Теорема 4.6. *Для любой сети Петри с ингибиторными дугами существует эквивалентная ей сеть, в которой каждому переходу инцидентно не более одной ингибиторной дуги.*

Доказательство: В качестве доказательства рассмотрим следующее преобразование.

Во-первых, добавим в сеть "выключатель" — новую позицию *key* с дугами $(key(s), s)$ и $(s, key(s))$ для каждого перехода s. Таким образом, любой переход сети может сработать только тогда, когда в позиции *key* есть фишка. В начальной разметке в позицию-выключатель поместим одну фишку.

Пусть в исходной сети переходу t инцидентны две ингибиторные дуги — (t, p_1) и (t, p_2). Преобразуем исходную сеть, заменив в ней переход t на три перехода — t_1, t_2 и t_3. У перехода t_1 тот же набор входных дуг, что и у t, и нет выходных; у перехода t_2 — тот же набор выходных дуг, что и у t, и нет входных; у перехода t_3 нет входных дуг, а множество выходных дуг соответствует множеству входных дуг исходного перехода t.

Добавим новую позицию p (с нулевой начальной разметкой) и три новые дуги — (t_1, p), (p, t_2) и (p, t_3). Исходные ингибиторные дуги (t, p_1) и (t, p_2) преобразуем в ингибиторные дуги (t_1, p_1) и (t_2, p_2).

Далее, выключатель *key* соединим забирающей фишку дугой (key, t_1) с первым переходом t_1, возвращающей фишку дугой (t_2, key) со вторым переходом t_2 и возвращающей фишку дугой (t_3, key) с третьим переходом t_3. .

Пример преобразования изображен на Рис. 4.15.

Срабатывание перехода t_1 соответствует началу (попытке) срабатывания перехода t в исходной сети и может произойти только при отсутствии фишек в p_1. По построению после срабатывания перехода t_1 может последовать только одно из двух срабатываний: t_2 или t_3 (все остальные переходы "выключены"). При

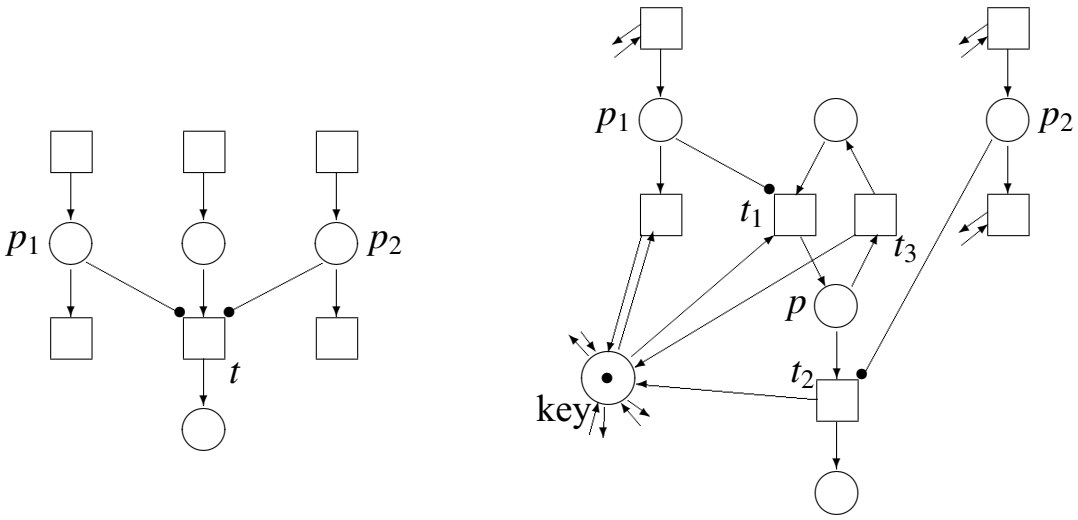


Рис. 4.15. Построение сети, в которой каждому переходу инцидентно не более одной ингибиторной дуги

этом t_2 может сработать только в случае отсутствия фишек в p_2. Срабатывания t_2 и t_3 “включают” остальные переходы. При этом срабатывание t_2 соответствует успешному завершению срабатывания t в исходной сети, а t_3 — отмене срабатывания.

Очевидно, что множество достижимости построенной сети совпадает с множеством достижимости исходной сети (без учета разметки новых позиций, которые могут содержать не более одной фишки). □

Теорема 4.7. *Класс сетей Петри с ингибиторными дугами, в которых каждой вершине инцидентно не более одной ингибиторной дуги, совпадает с классом машин Тьюринга.*

Доказательство: Заметим, что преобразования, описанные в доказательстве двух предыдущих теорем, не увеличивают число ингибиторных дуг, инцидентных переходу, и число ингибиторных дуг, инцидентных позиции, соответственно. Поэтому их последовательное применение (в любом порядке) приведет к сети, в которой каждой вершине инцидентно не более одной ингибиторной дуги. □

Определим полный аналог ингибиторной дуги для случая сетей активных ресурсов:

Определение 4.9. АР-сетью с ингибиторными дугами *назовем набор* $AR = (V, I, O, Inh)$, *где*

- V — *конечное множество вершин (ресурсов);*
- $I : V \times V \to Nat$ — *множество потребляющих дуг;*
- $O : V \times V \to Nat$ — *множество производящих дуг.*
- $Inh : V \times V \to \{0, 1\}$ — *множество ингибиторных дуг.*

В сетях с ингибиторными дугами изменяется определение активности ресурса:

Определение 4.10. *Ресурс* $v \in V$ активен *при разметке* M, *если*

- $M(v) > 0$;
- $\forall w \in V \;\; M(w) \geq I(w, v)$;
- $\forall w \in V \;\; M(w) * Inh(w, v) = 0$ *(пусто во всех потребляемых узлах, связанных с данным узлом посредством ингибиторных дуг).*

Определение срабатывания остается прежним. Таким образом, единственное отличие этих сетей от обыкновенных сетей активных ресурсов — возможность проверки разметки потребляемого узла на пустоту.

Графически ингибиторная дуга в АР-сетях изображается стрелкой с наконечником в виде черной точки (Рис. 4.16, правая часть).

Теорема 4.8. *Класс сетей активных ресурсов с ингибиторными дугами совпадает с классом машин Тьюринга.*

Доказательство: Достаточно показать, что класс сетей активных ресурсов с ингибиторными дугами совпадает с классом сетей Петри с ингибиторными дугами.

Для построения по данной сети Петри с ингибиторными дугами моделирующей ее АР-сети с ингибиторными дугами применим тот же алгоритм, что и в доказательстве первой части Теоремы 4.1. При этом обычные дуги от позиций к переходам будем заменять на потребляющие дуги, обычные дуги от переходов к позициям — на производящие дуги, ингибиторные дуги — на ингибиторные дуги.

Для построения по данной АР-сети с ингибиторными дугами моделирующей ее сети Петри с ингибиторными дугами применим тот же алгоритм, что и в доказательстве второй части Теоремы 4.1. При этом потребляющие и производящие дуги будем заменять на обычные дуги, ингибиторные дуги — на ингибиторные дуги. □

Используя ту же схему и утверждение Теоремы 4.7, легко доказать, что:

Теорема 4.9. *Класс АР-сетей с ингибиторными дугами, в которых каждой вершине инцидентно не более одной ингибиторной дуги, совпадает с классом машин Тьюринга.*

Далее для краткости АР-сети с ингибиторными дугами, в которых каждой вершине инцидентно не более одной ингибиторной дуги, будем называть ИД-сетями.

Сети с “каналами”

В сетях Петри с ингибиторными дугами (и в АР-сетях с ингибиторными дугами) проверка на пустоту осуществляется при помощи дуги, то есть отношения на множестве узлов системы. Сети активных ресурсов благодаря своей двойственной структуре предоставляют и другие достаточно простые синтаксические возможности для проверки нелокальных свойств системы, обладающие

при этом естественной интерпретацией. Например, мы можем “ограничить” область видимости операции проверки на пустоту самим срабатывающим узлом.

Рассмотрим сети активных ресурсов, в которых в качестве агентов могут выступать пустые вершины. Такие вершины мы будем называть “каналами”. Вершина-канал может сработать только в том случае, когда в ней отсутствуют фишки (“канал свободен”). Графически канал изображается в виде двойного кружка (Рис. 4.16).

Определение 4.11. АР-сетью с каналами *назовем набор* $AR = (V, I, O)$*, где*

- $V = V_{simple} \cup V_{channel}$ *— конечное множество вершин, где* V_{simple} *— вершины–ресурсы,* $V_{channel}$ *— вершины–каналы, причем* $V_{simple} \cap V_{channel} = \varnothing$*;*
- $I : V \times V \to Nat$ *— множество потребляющих дуг;*
- $O : V \times V \to Nat$ *— множество производящих дуг.*

Активность вершины–ресурса и срабатывание вершины–ресурса определяются так же, как и в случае обыкновенных АР-сетей. Для канала вместо активности определяется готовность к передаче ресурсов:

Определение 4.12. *Канал* $c \in V_{channel}$ готов к передаче *при разметке* M*, если*

- $M(c) = 0$ *(свободен сам канал* c*);*
- $\forall w \in V \;\; M(w) \geq I(w, c)$ *(в потребляемых узлах содержится достаточное количество фишек).*

Готовый к передаче при разметке M *канал* c *может* сработать*, порождая при этом разметку* M'*, где* $\forall w \in V \;\; M'(w) =_{def} M(w) - I(w, c) + O(c, w)$.

Как видно из определения, канал обладает общими свойствами агентов в АР-сетях: он может изменять количество фишек в сети (набор ресурсов). Также здесь есть и проверка на пустоту, хотя и ограниченная: мы можем проверять на

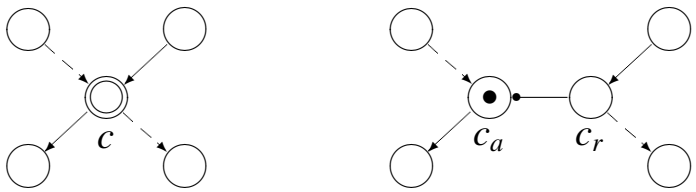


Рис. 4.16. Построение ИД-сети, моделирующей данную АР-сеть с каналами

пустоту не произвольный узел, как в сетях с ингибиторными дугами, а только сам канал. Однако этого оказывается достаточно для получения универсальной вычислительной системы:

Теорема 4.10. *Класс сетей активных ресурсов с каналами совпадает с классом машин Тьюринга.*

Доказательство: Достаточно показать, что класс АР-сетей с каналами совпадает с классом ИД-сетей.

Для построения по данной АР-сети с каналами моделирующей ее ИД-сети преобразуем каждый канал c в две вершины — c_a и c_r. Вершина c_a будет соответствовать “активной” составляющей исходной вершины, ей будут инцидентны входные и выходные для c дуги. Вершина c_r будет соответствовать “пассивной” составляющей исходной вершины, ей будут инцидентны производящие и потребляющие c дуги. Также добавим новую ингибиторную дугу (c_r, c_a). Начальная разметка $M(c) = k$ преобразуется в разметку $M(c_a) = 1$, $M(c_r) = k$.

Пример преобразования изображен на Рис. 4.16.

Для построения по данной ИД-сети моделирующей ее АР-сети с каналами применим следующий алгоритм.

Во-первых, заметим, что в силу того, что в ИД-сетях каждой вершине может быть инцидентно не более одной ингибиторной дуги, нам достаточно предложить способ преобразования ингибиторной дуги, связывающей два узла с наборами обычных производящих и потребляющих дуг (Рис. 4.17).

Рассмотрим изображенное на Рис. 4.17 преобразование. Исходной паре уз-

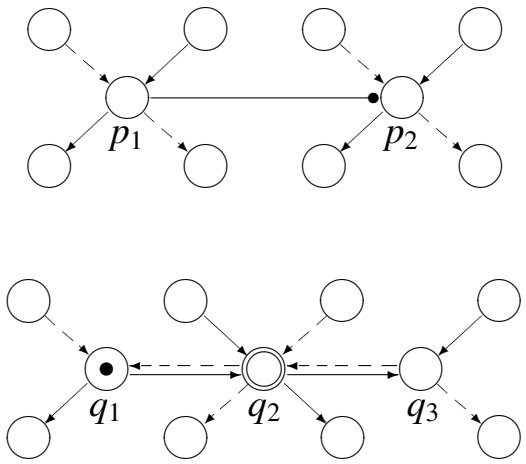


Рис. 4.17. Построение АР-сети с каналами, моделирующей данную ИД-сеть

лов p_1 и p_2 сопоставлена тройка новых узлов — q_1, q_2 и q_3. Узел q_1 соответствует активной составляющей (агенту) узла p_1, узел q_3 — пассивной составляющей (ресурсу) узла p_2. Узел q_2 объединяет в себе свойства ресурса p_1 и агента p_2. При этом добавлены двунаправленные связи: пара дуг $\{(q_2, q_1), (q_1, q_2)\}$ гарантирует, что агент q_1 сработает только при наличии ресурса q_2, а пара $\{(q_3, q_2), (q_2, q_3)\}$ гарантирует, что агент q_2 сработает только при наличии ресурса q_3. Тем самым мы гарантируем правильность функционирования полученной сети в смысле обычных потребляющих и производящих дуг. Для моделирования ингибиторной дуги узел q_2 сделан в виде канала. □

Из приведенного построения можно сделать интересный (хотя и достаточно очевидный) вывод: ингибиторная дуга “задействует” пассивную составляющую того узла, в котором она начинается, и активную составляющую того узла, в котором она заканчивается. При этом непосредственно от самой дуги можно избавиться, совместив соответствующий ресурс и агент в форме единого узла — канала.

Сети с одновременным срабатыванием

Рассмотрим сети активных ресурсов, в которых срабатывание вершины может зависеть не только от наличия в ней фишек, но и от их количества. Подобные

вершины мы будем называть вершинами (узлами) с одновременным срабатыванием и обозначать при помощи прямоугольника в кружке (Рис. 4.18, левая часть). Все фишки в такой вершине могут сработать только одновременно, и только при наличии достаточного для их одновременного срабатывания числа входных ресурсов.

Определение 4.13. АР-сетью с одновременным срабатыванием *назовем набор* $AR = (V, I, O)$, *где*

- $V = V_{simple} \cup V_{simult}$ — *конечное множество вершин, где* V_{simple} — *обычные вершины–ресурсы,* V_{simult} — *вершины с одновременным срабатыванием фишек, причем* $V_{simple} \cap V_{simult} = \varnothing$;
- $I : V \times V \to Nat$ — *множество потребляющих дуг;*
- $O : V \times V \to Nat$ — *множество производящих дуг.*

Активность вершины–ресурса и срабатывание вершины–ресурса определяются так же, как и в случае обыкновенных АР-сетей. Для вершины с одновременным срабатыванием в определениях учитывается полная разметка самой срабатывающей вершины:

Определение 4.14. *Вершина с одновременным срабатыванием* $v \in V_{simult}$ активна *при разметке* M, *если* $\forall w \in V \quad M(w) \geq M(v) * I(w, v)$.

Активная при разметке M *вершина* $v \in V_{simult}$ *может* сработать, *порождая при этом новую разметку* M', *где* $\forall w \in V \quad M'(w) =_{def} M(w) - M(v) * I(w, v) + M(v) * O(v, w)$.

Подобное усложнение правила срабатывания также приводит к увеличению выразительности сразу до универсальных моделей вычислений:

Теорема 4.11. *Класс сетей активных ресурсов с одновременным срабатыванием совпадает с классом машин Тьюринга.*

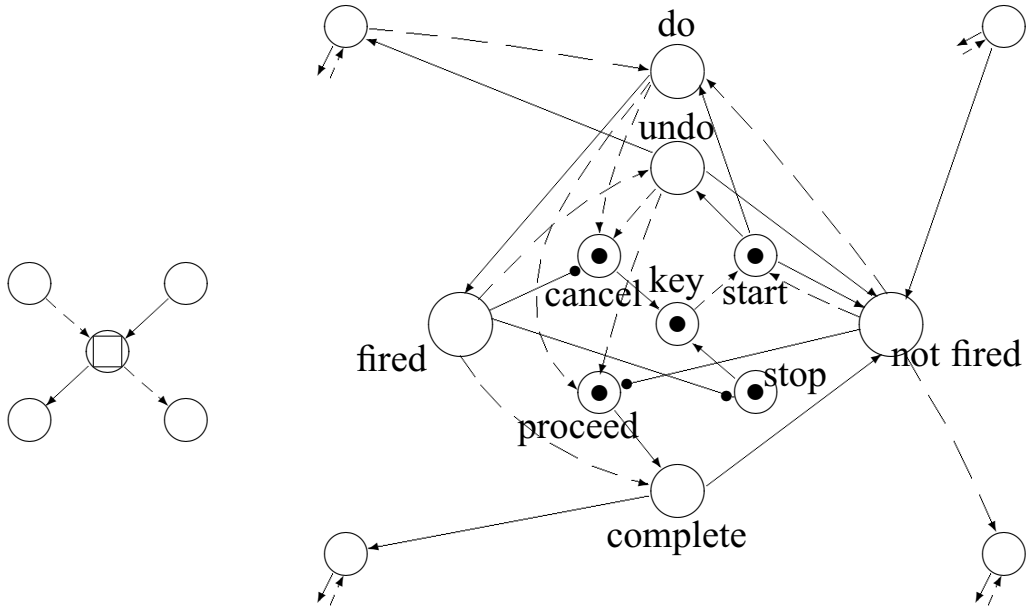


Рис. 4.18. Построение ИД-сети, моделирующей данную АР-сеть с одновременным срабатыванием

Доказательство: Достаточно показать, что класс АР-сетей с одновременным срабатыванием совпадает с классом ИД-сетей.

Докажем, что для любой сети с одновременным срабатыванием существует эквивалентная ей ИД-сеть.

Пример преобразования изображен на Рис. 4.18.

В ходе данного преобразования в сеть опять добавляется узел-выключатель “key”. Исходный узел с одновременным срабатыванием заменяется на набор узлов. Ресурсной составляющей исходного узла соответствует новый узел “not fired”, именно туда помещается его начальная разметка. Запуск имитации одновременного срабатывания осуществляет агент “start”, который выключает все внешние узлы сети и активизирует агентов “do” и “undo”. Действие агента “do” соответствует срабатыванию одной фишки исходной вершины (при наличии нужного числа ресурсов в потребляемых внешних узлах), действие “undo” — отмене одного срабатывания. Выйти из набора действий “do” и “undo” можно только двумя способами. Во-первых, в случае отстутствия фишек в “not fired” срабатывание агента “proceed” переводит систему в режим завершения

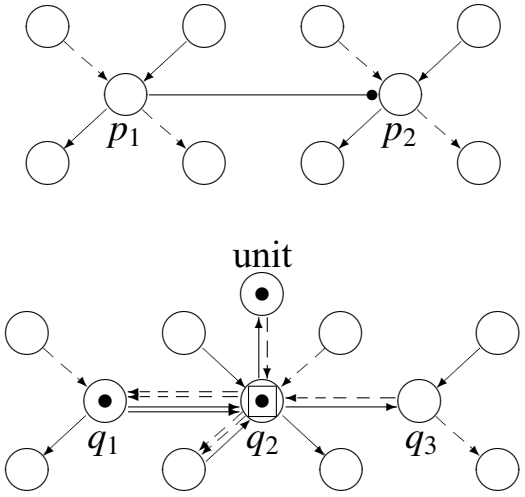


Рис. 4.19. Построение АР-сети с одновременным срабатыванием, моделирующей данную ИД-сеть

срабатывания: выключаются агенты “do” и “undo”, при этом включается агент “complete”, которому остается только произвести нужное количество фишек для внешних узлов, а затем передать управление агенту “stop”. В противном случае (при отсутствии фишек в “fired”), управление передается агенту “cancel”, который переводит систему в исходное состояние (“одновременное срабатывание не произошло”).

Таким образом, построенная сеть правильно моделирует одновременное срабатывание: выйти из состояния “key=∅” можно только двумя способами — без изменения разметки (“cancel”) и с многократным (соответствующим исходному числу фишек в “not fired”) применением действий “do”, а затем “proceed”, что как раз соответствует полному потреблению и производству ресурсов исходным одновременным срабатыванием.

Заметим, что доказанное выше утверждение следует из тезиса Черча-Тьюринга. Гораздо менее очевидна возможность обратного моделирования ИД-сети при помощи одновременного срабатывания. Пример такого моделирования приведен на Рис. 4.19.

Здесь применен подход, похожий на способ моделирования ИД-сети при помощи сети с каналами. Точно так же мы объединяем в одном узле ресурс

начала ингибиторной дуги и агент ее завершения. Но здесь пустота моделируется не отсутствием фишек, а наличием ровно одной фишки в вершине q_2 (при любом большем количестве фишек узел q_2 не сработает, так как в служебном узле “unit” всегда находится ровно одна фишка). □

Итак, в случае сетей активных ресурсов для получения универсальных (в смысле машин Тьюринга) моделей можно использовать три различных способа расширения синтаксиса:

1. ингибиторные дуги (как и в сетях Петри);
2. срабатывания пустых узлов–каналов;
3. срабатывания узлов с учетом их полной разметки, т.е. одновременные срабатывания фишек.

Сети с обнуляющими дугами

Рассмотрим расширение класса сетей Петри, занимающее строго промежуточное (по выразительности) положение между обыкновенными сетями Петри и машинами Тьюринга — сети Петри с обнуляющими дугами. Известно, что для сетей Петри с обнуляющими дугами разрешима проблема останова, но неразрешима проблема достижимости [115].

Обнуляющая дуга связывает переход с позицией. В отличие от ингибиторной дуги, она никак не влияет на активность перехода. При срабатывании перехода разметка всех позиций, связанных с ним обнуляющими дугами, обнуляется (независимо от того, сколько в них было фишек). Графически обнуляющие дуги изображаются при помощи перечеркнутых линий (Рис. 4.20).

Так же, как и в случае ингибиторных дуг, сети с обнуляющими дугами допускают некоторое упрощение структуры без потери выразительности:

Теорема 4.12. *Для любой сети Петри с обнуляющими дугами существует эквивалентная ей сеть, в которой каждой позиции инцидентно не более одной*

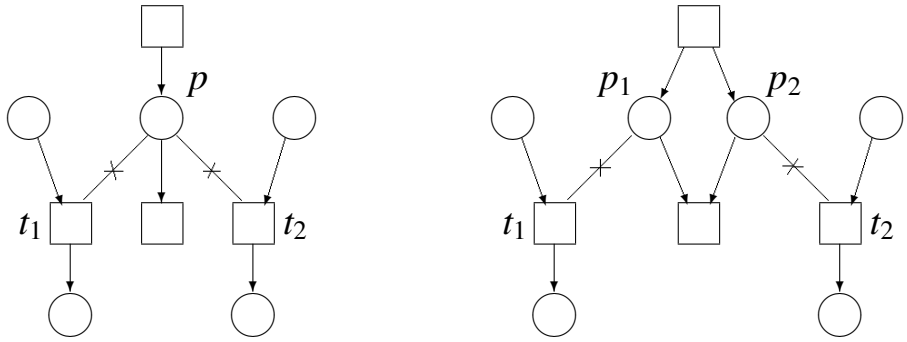


Рис. 4.20. Построение сети, в которой каждой позиции инцидентно не более одной обнуляющей дуги

обнуляющей дуги.

Доказательство: В качестве доказательства рассмотрим преобразование на Рис. 4.20, аналогичное преобразованию в доказательстве Теоремы 4.5. Единственное отличие состоит в том, что ингибиторные дуги заменены на обнуляющие.

□

Теорема 4.13. *Для любой сети Петри с обнуляющими дугами существует эквивалентная ей сеть, в которой каждому переходу инцидентно не более одной обнуляющей дуги.*

Доказательство: Рассмотрим преобразование, представляющее собой упрощенную версию преобразования из доказательства Теоремы 4.6. В отличие от ингибиторной дуги обнуляющая дуга не ограничивает возможностей срабатывания инцидентного перехода, поэтому здесь мы можем не моделировать откат после первого обнуления. Пример преобразования приведен на Рис. 4.21.

□

Теорема 4.14. *Класс сетей Петри с обнуляющими дугами, в которых каждой вершине инцидентно не более одной обнуляющей дуги, совпадает с классом сетей Петри с обнуляющими дугами.*

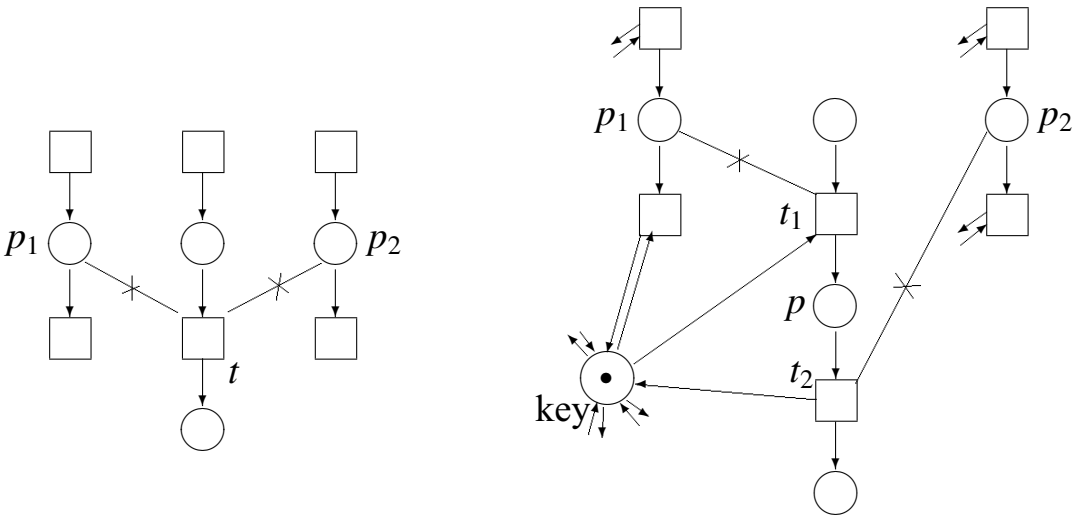


Рис. 4.21. Построение сети, в которой каждому переходу инцидентно не более одной обнуляющей дуги

Доказательство: Заметим, что преобразования, описанные в доказательстве предыдущих теорем, не увеличивают число обнуляющих дуг, инцидентных переходу, и число обнуляющих дуг, инцидентных позиции, соответственно. Поэтому их последовательное применение (в любом порядке) приведет к сети, в которой каждой вершине инцидентно не более одной обнуляющей дуги. □

Рассмотрим непосредственное обобщение понятия обнуляющей дуги на случай сетей активных ресурсов.

Определение 4.15. АР-сетью с обнуляющими дугами *назовем набор* $AR = (V, I, O, Reset)$*, где*

- V *— конечное множество вершин (ресурсов);*
- $I : V \times V \to Nat$ *— множество потребляющих дуг;*
- $O : V \times V \to Nat$ *— множество производящих дуг.*
- $Reset : V \times V \to \{0, 1\}$ *— множество обнуляющих дуг.*

В сетях с обнуляющими дугами не изменяется определение активности ресурса, но изменяется определение срабатывания:

Определение 4.16. *Активный при разметке M ресурс v может* сработать, *порождая при этом новую разметку M', где* $\forall w \in V \quad M'(w) =_{def} M(w) * (1 - Reset(w, v)) + O(v, w)$.

Таким образом, единственное отличие таких сетей от обыкновенных сетей активных ресурсов — возможность обнуления разметки одного узла при срабатывании другого.

Графически обнуляющая дуга в АР-сетях изображается перечеркнутой стрелкой (здесь, в отличие от сетей Петри, необходимо явно задавать направление дуги).

Теорема 4.15. *Класс сетей активных ресурсов с обнуляющими дугами совпадает с классом сетей Петри с обнуляющими дугами.*

Доказательство: Для построения по данной сети Петри с обнуляющими дугами моделирующей ее АР-сети с обнуляющими дугами применим тот же алгоритм, что и в доказательстве первой части Теоремы 4.1. При этом обычные дуги от позиций к переходам будем заменять на потребляющие дуги, обычные дуги от переходов к позициям — на производящие дуги, обнуляющие дуги — на обнуляющие дуги.

Для построения по данной АР-сети с обнуляющими дугами моделирующей ее сети Петри с обнуляющими дугами применим тот же алгоритм, что и в доказательстве второй части Теоремы 4.1. При этом потребляющие и производящие дуги будем заменять на обычные дуги, обнуляющие дуги — на обнуляющие дуги. □

Используя ту же схему и утверждение Теоремы 4.14, легко доказать, что:

Теорема 4.16. *Класс АР-сетей с обнуляющими дугами, в которых каждой вершине инцидентно не более одной обнуляющей дуги, совпадает с классом сетей Петри с обнуляющими дугами.*

Сети с обнуляющим срабатыванием

Рассмотрим сети активных ресурсов, в которых обнуление производится непосредственно в срабатывающей вершине. Такие вершины мы будем называть вершинами с “обнуляющим срабатыванием”. Графически они будут изображаться в виде частично перечеркнутого кружка (Рис. 4.22).

Определение 4.17. АР-сетью с обнуляющим срабатыванием *назовем набор* $AR = (V, I, O)$*, где*

- $V = V_{simple} \cup V_{reset}$ *— конечное множество вершин, где* V_{simple} *— обычные вершины–ресурсы,* V_{reset} *— вершины с обнуляющим срабатыванием, причем* $V_{simple} \cap V_{reset} = \varnothing$*;*
- $I : V \times V \to Nat$ *— множество потребляющих дуг;*
- $O : V \times V \to Nat$ *— множество производящих дуг.*

Активность вершины–ресурса и срабатывание вершины–ресурса определяются так же, как и в случае обыкновенных АР-сетей. Изменяются определения для вершины с обнуляющим срабатыванием:

Определение 4.18. *Вершина с обнуляющим срабатыванием* $v \in V_{reset}$ активна *при разметке* M*, если* $\forall w \in V \;\; M(w) \geq I(w, v)$.

Активная при разметке M *вершина с обнуляющим срабатыванием* v *может* сработать*, порождая при этом новую разметку* M'*, где* $\forall w \in V \setminus \{v\}\; M'(w) =_{def} M(w) - I(w, c) + O(c, w);\;\; M'(v) = O(v, v)$.

В отличие от АР-сетей с обнуляющими дугами здесь мы не можем обнулять произвольный узел. Обнулиться может только тот узел, который срабатывает. Однако, как оказалось, это ограничение также является чисто синтаксическим и не сужает выразительных возможностей формализма:

Теорема 4.17. *Класс АР-сетей с обнуляющим срабатыванием совпадает с классом сетей Петри с обнуляющими дугами.*

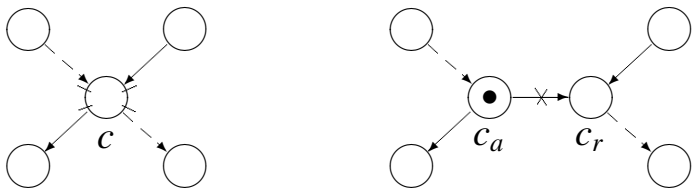


Рис. 4.22. Построение АР-сети с обнуляющими дугами, моделирующей данную АР-сеть с обнуляющим срабатыванием

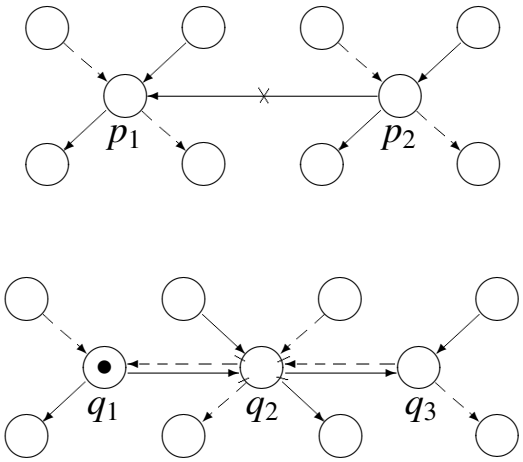


Рис. 4.23. Построение АР-сети с обнуляющим срабатыванием, моделирующей данную АР-сеть с обнуляющими дугами

Доказательство: Достаточно показать, что класс АР-сетей с обнуляющим срабатыванием совпадает с классом АР-сетей с обнуляющими дугами, в которых каждой вершине инцидентно не более одной обнуляющей дуги.

Схемы соответствующих преобразований моделей изображены на Рис. 4.22 и 4.23.

Интересно, что, несмотря на переход в другой класс систем (от машин Тьюринга к менее мощным сетям с обнуляющими дугами), общая схема моделирования мало изменилась. □

Итак, в случае сетей активных ресурсов для получения систем из класса сетей с обнуляющими дугами можно использовать как стандартный способ (непосредственно сами обнуляющие дуги), так и специфическую для АР-сетей синтаксическую конструкцию — вершину (агент, ресурс) с возможностью “са-

моуничтожения”.

4.3.2. Синтаксические расширения

Итак, мы увидели, что существует опасность при расширении синтаксиса получить расширение класса моделируемых систем, и в результате прийти к неразрешимости тех или иных алгоритмических свойств. В данном разделе рассматриваются новые синтаксические конструкции, которые не меняют алгоритмические свойства формализма, но при этом упрощают моделирование, поскольку лучше отражают специфические особенности работы более сложных агентов и ресурсов.

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.

Определение 4.19. Сетью активных ресурсов с расширенными срабатываниями *(PC-сетью) назовём 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 назовём тройку (E, M, M_e), где $E = (V, V_e, I, O)$ — PC-сеть, $M \in \mathcal{M}(V \cup V_e)$ — обычная разметка (ожидающие ресурсы), $M_e \in \mathcal{M}(V_e)$ — расширенная разметка (работающие расширенные ресурсы).

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 AP-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

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

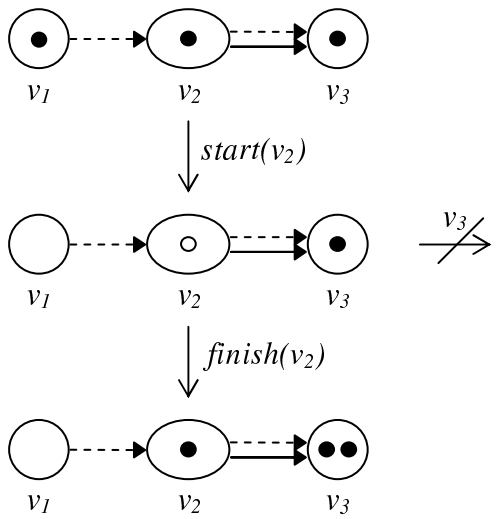


Рис. 4.24. Работающий агент

- $M'(v) = M(v) - I(v, v) - 1$;
- $M'_e(v) = M_e(v) + 1$.

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. 4.24.

Введение расширенного срабатывания не меняет выразительности формализма:

Теорема 4.18. *Класс сетей активных ресурсов с расширенными срабатываниями эквивалентен классу обыкновенных сетей Петри.*

Доказательство: В силу Теоремы 4.1 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. □

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. 4.25 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 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 4.3). Также достаточно компактно формализуются свойства взаимных блокировок (табли-

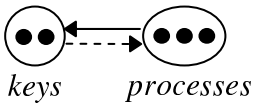


Рис. 4.25. Модель разделяемого доступа (mutex) с тремя процессами и двумя ключами

ца 4.4).

Таблица 4.3. Формализация свойств одновременной работы агентов в PC-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.

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.

Определение 4.20. Сетью активных ресурсов с ненадежными срабатываниями *(НС-сетью) назовём набор* $U = (V, I, I_u, O, O_u)$*, где*

- V *— конечное множество вершин (ресурсов);*

Таблица 4.4. 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.

- $I \in \mathcal{M}(V \times V)$ — *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

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

Активная при разметке M вершина v может *сработать*, порождая любую разметку M' из конечного множества

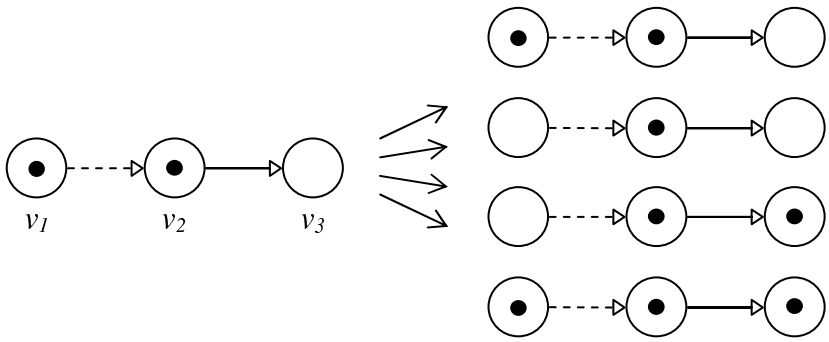


Рис. 4.26. Ненадежное срабатывание (в смысле потребления и в смысле производства ресурсов)

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

Пример ненадежного срабатывания приведен на рис. 4.26. Здесь агент может сработать в четырех различных режимах.

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

Теорема 4.19. *Класс сетей активных ресурсов с ненадежными срабатываниями эквивалентен классу обыкновенных сетей Петри.*

Доказательство: В силу Теоремы 4.1 нам достаточно доказать, что НС-сети эквивалентны АР-сетям.

Поскольку НС-сети являются синтаксическим расширением АР-сетей, нам достаточно доказать, что для любой НС-сети существует эквивалентная ей АР-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.

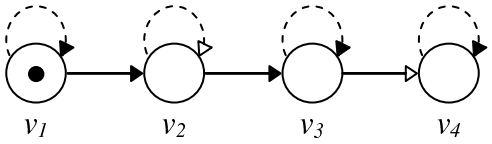


Рис. 4.27. Модель канала с шумами

В качестве начальной разметки поместим в каждую вершину из $Repl(v)$ ровно $M_0(v)$ фишек (где M_0 — начальная разметка исходной НС-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 НС-сети. □

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. 4.27) Здесь второе звено цепи ненадежно в том смысле, что может „забыть“ о том, что пакет уже передан, и передать его ещё 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. 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.

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 соответствующим 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 АР-сетей). То есть непосредственно в саму схему процесса распределенного приложения мы можем заложить принцип динамического изменения набора исполняющих модулей (агентов). Например, при

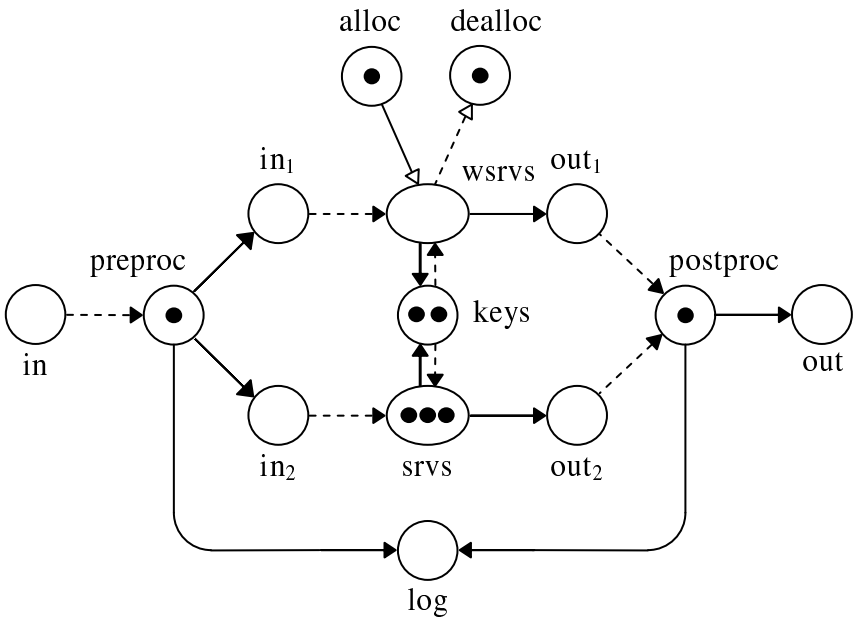


Рис. 4.28. Управляющий процесс распределенного приложения

помощи данной технологии можно организовывать приложения в виде динамических композиций веб-сервисов.

Рассмотрим простой пример подобной системы, приведенный на рис. 4.28 (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).

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.

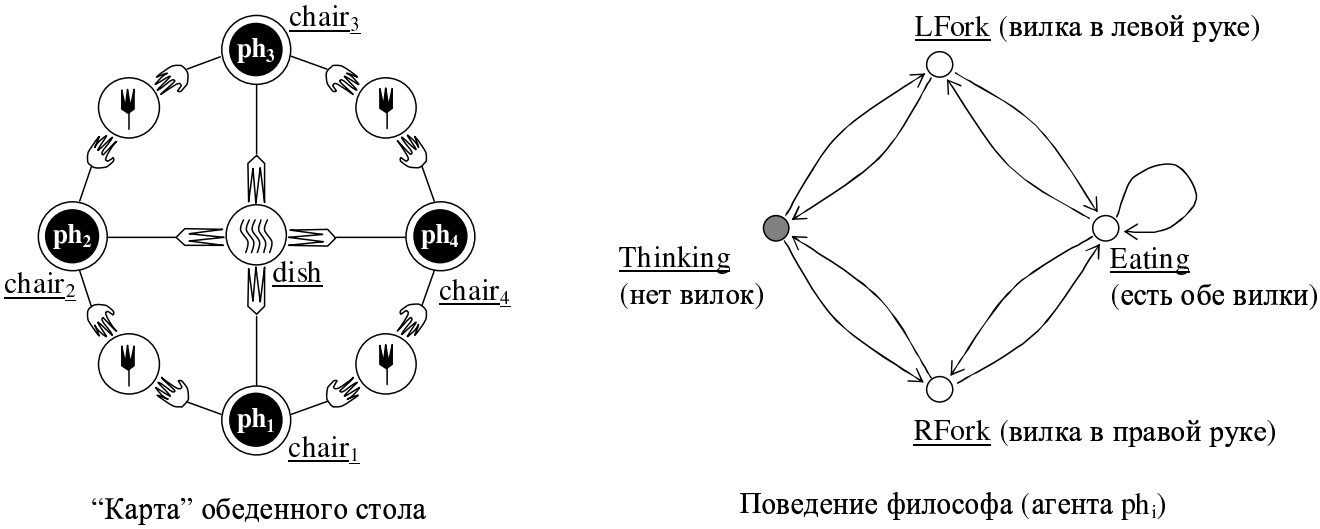
Простейший анализ семантических свойств модели средствами AP-сетей позволяет выяснить следующую полезную информацию о системе:

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.

4.4. Автоматы, управляемые ресурсами

4.4.1. Мотивация

Вернёмся к проблеме Обедающих Философов. В более сложной постановке философы могут брать/класть левую и правую вилки по отдельности. Таким образом, каждый из них может находиться в одном из четырех состояний: Thinking (нет вилок), Eating (есть обе вилки), RFork (только правая) и LFork (только левая).



"Карта" обеденного стола

Поведение философа (агента ph_i)

Рис. 4.29. Обедающие философы — неформальная схема

На Рис. 4.29 изображена неформальная постановка проблемы в самом, как мы посчитали, понятном и естественном графическом виде. Левая часть рисунка описывает "пространственную" ситуацию — стулья, вилки, возможность дотянуться руками до вилок, а вилками до тарелки. Это своего рода карта с понятными условными обозначениями. Справа показано поведение отдельного философа — проще всего это оказалось сделать при помощи даграммы переходов конечного автомата.

Классический способ моделирования проблемы сетью Петри изображен на Рис. 4.30. Можно сказать, что здесь мы видим "развёртку" предыдущей модели — поведение философов "встроено" в системную сеть в виде отдельных повторяющихся фрагментов. Очевидно, что эта модель уже не является агентно-ориентированной.

Обратим внимание, что на Рис. 4.29 ни философы, ни стол, вокруг которого они сидят, не требуют для своего моделирования всех возможностей сетей Петри. Они гораздо примитивнее: философ — просто конечный автомат, а схема стола является своего рода диаграммой коммуникаций. На этой схеме у нас нет переходов — только взаимосвязи ("порты"), изображенные при помощи дуг разного вида. Причем взаимосвязи явно стоит разделить на два типа: двусторонние ("рука" может брать и возвращать вилку) и односторонние (спагетти могут

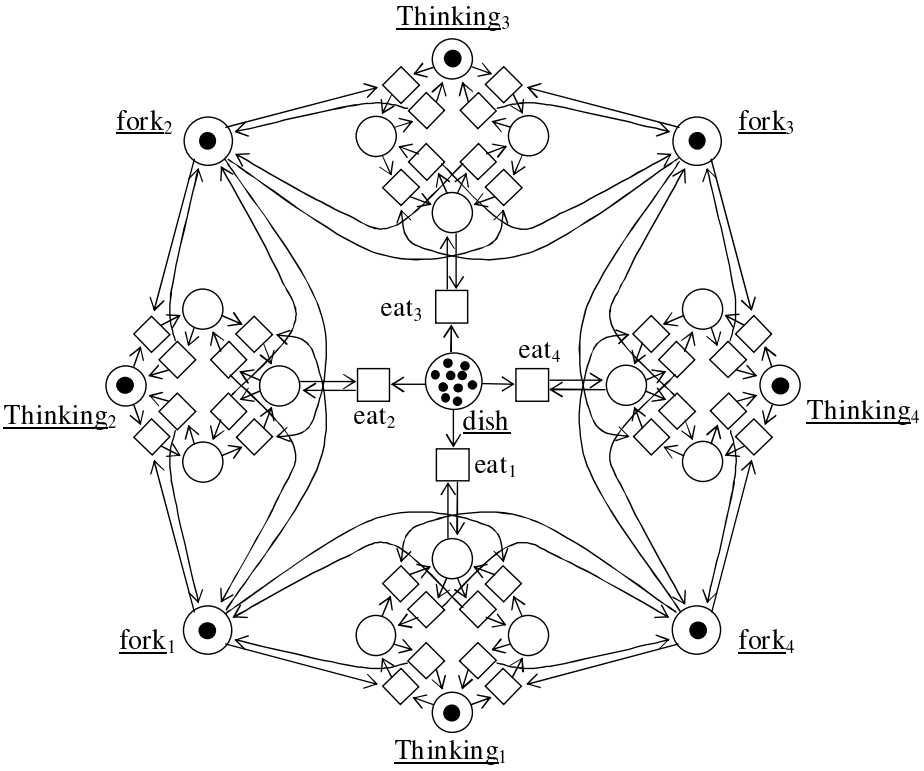
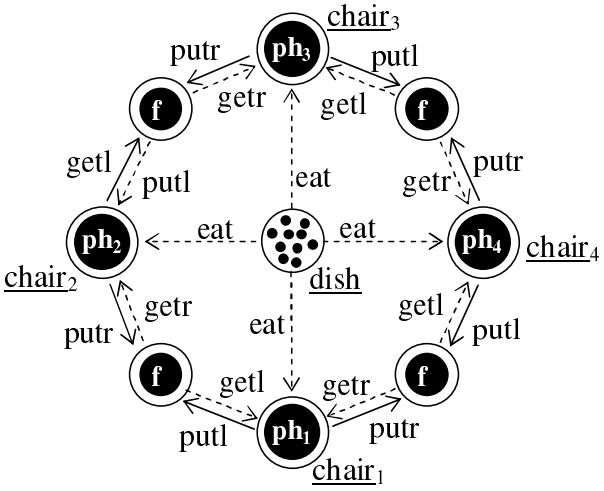


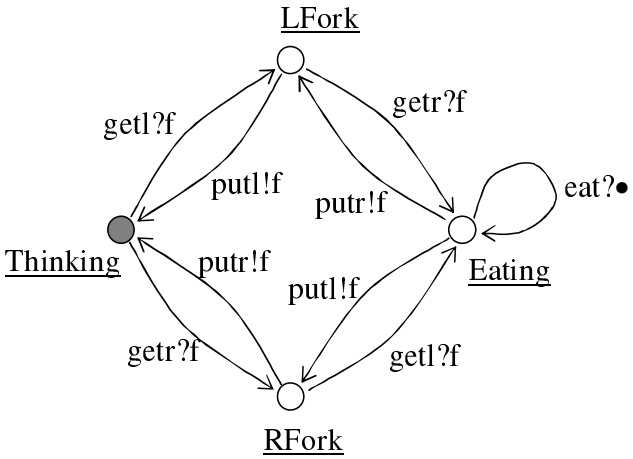
Рис. 4.30. Обедающие философы — представление в виде сети Петри

быть перемещены только из блюда к философу). Кроме того, каждый порт имеет активную сторону (это всегда философ) и пассивную сторону (вилки и тарелки).

Очевидно, что АР-сети могут быть весьма удобны для моделирования “внешнего” уровня подобных систем. В этом разделе мы расширяем язык АР-сетей засчёт использования подхода “сеть-внутри-сети” (net-within-net). В новом формализме, названном Р-сетями (сетями управляемых ресурсами автоматов), любая модель имеет два уровня. Системым (внешним) уровнем является обыкновенная плоская АР-сеть. Но фишками в этой сети являются не обычные “активные ресурсы”, а полноценные агенты, обладающие собственным поведением (конечные автоматы). Во время своей работы агенты потребляют и производят локальные ресурсы, являясь одновременно потенциальными ресурсами для своих соседей. При этом системную сеть можно рассматривать как “карту” связей (портов), описывающих всевозможные отношения, в которых могут находиться агенты. Агент использует эти порты для взаимодействия с системными ресурсами, находящимися в соседних узлах. В частности, он может естественным образом



Обеденный стол (системная сеть)



Философ (объектный автомат ph_i)

Рис. 4.31. Обедающие философы — представление в виде P-сети

производить / потреблять / копировать / перемещать других агентов, или даже самого себя.

Отметим, что, в отличие от ссылочной семантики для объектных сетей, рассматриваемой в [91, 155, 178, 201], в случае P-сетей агенты не “размазаны” по системной сети, а находятся в конкретных узлах (как во вложенных сетях Петри [47, 166]).

На Рис. 4.31 приведена модель обедающих философов в синтаксисе P-сетей — сетей автоматов, управляемых ресурсами.

Модель построена непосредственно на основе неформального описания проблемы. На системном уровне мы определяем типы отношений между узлами при помощи свойств дуг: пунктирная дуга обозначает потребление ресурса агентом (фишкой в конце дуги), сплошная — производство ресурса агентом (фишкой в начале дуги). Дуги называются *портами*, каждый порт помечен именем порта. Например, пунктирные дуги с меткой “getl” (“взять слева”) используются в качестве входных портов для философов и описывают возможности взять что-то левой рукой. Напротив, “putr” описывает возможность положить что-то с правой стороны. Порты “eat” моделируют возможность что-то съесть.

Агент (философ) моделируется конечным автоматом. Переходы в нём мы помечаем специальными ресурсными выражениями, которые определяют, какие

системные ресурсы должны производиться и потребляться агентом в момент срабатываний этих переходов. Каждое ресурсное выражение состоит из трех частей: имя порта, тип порта (“?” для входного и “!” для выходного) и ресурс. Например, `putr!f` означает, что вилка (обозначаемая константой `f`) должна быть положена справа, `eat?•` — что простая фишка (обозначающая спагетти) должна быть “употреблена” через порт “eat”.

В формализм P-сетей заложены три ключевые особенности, отличающие его от обыкновенных и выскоуровневых сетей Петри:

- “ортогональный” синтаксис системного уровня: сети активных ресурсов вместо сетей Петри;
- более простой объектный уровень: конечные автоматы вместо сетей Петри;
- новый способ межуровневой синхронизации: коммуникационные порты вместо синхронизированных переходов.

4.4.2. Сети автоматов, управляемых ресурсами (P-сети)

Мы определяем P-сеть как AP-систему, в которой в роли фишек выступают специальные расширенные конечные автоматы — так называемые автоматы, управляемые ресурсами (P-автоматы). Таким образом, мы расширяем понятие фишки (ресурса) путем добавления ей собственного поведения. В ходе своей работы автоматная фишка потребляет и производит фишки-ресурсы (в т.ч. автоматные), находящиеся в узлах, соседних тому, в котором она сама находится. Дуги в AP-сети мы помечаем именами портов, указывая тем самым узлы с доступными ресурсами.

Пусть Ω — конечное множество *типов*, *Const* — конечное множество типизированных индивидуальных объектов, которые мы называем константами. Тип константы $c \in Const$ обозначается как $Type(c)$. Для $a \in \Omega$ через $Const(a)$ мы обозначаем множество всех констант типа a.

Далее, пусть П — конечное множество типизированных (элементами Ω) *портов* (имён портов). Тип порта $\pi \in \Pi$ обозначим $Type(\pi)$.

Лежащая в основе Р-сети сеть активных ресурсов с дугами, помеченными именами портов, называется *системной сетью*.

Определение 4.21. Системная сеть — *это набор* $SN = (V, I, O, \pi)$*, где* (V, I, O) — *АР-сеть,* $\pi : (I \cup O) \to \Pi$ — *функция, помечающая дуги именами портов.*

Определение 4.22. Разметкой M *системной сети* SN *называется функция*

$$M : V \to \mathcal{M}(Const),$$

сопоставляющая каждому узлу сети мультимножество находящихся в нём объектов.

Размеченной системной сетью *называется пара* (SN, M_0), *где* $SN = (V, I, O, \pi)$ — *системная сеть, а* M_0 — *её* начальная разметка.

Обозначим через Var множество типизированных переменных с типами из Ω. Пусть $a \in \Omega$. Определим язык $L(a)$ *ресурсных выражений* типа a следующим образом.

Пусть $\pi \in \Pi$. *Входной терм* есть терм вида $\pi?e$, где e — переменная или константа типа $Type(\pi)$. Аналогично, *выходной терм* есть терм вида $\pi!e$ с теми же условиями на π и e. *Ресурсное выражение* есть терм вида $\alpha_1; \alpha_2; \ldots; \alpha_k$, где α_j $(j = 1, \ldots, k)$ представляет собой входной или выходной терм (типы подвыражений $\alpha_1, \alpha_2, \ldots, \alpha_k$ могут различаться).

Определение 4.23. Управляемым ресурсами автоматом типа $a \in \Omega$ *(ресурсным автоматом, Р-автоматом) называется набор* $A = (S_A, T_A, l_A)$*, где* S_A — *конечное множество состояний,* $T_A \subseteq S_A \times S_A$ — *отношение перехода и* $l_A : T_A \to L(a)$ — *функция пометки переходов.*

Таким образом, Р-автомат — это всего лишь конечный автомат с помеченными переходами и без выделенного начального состояния. В Р-сетях Р-автоматы

будут играть роль сетевых фишек. В вырожденном случае, когда P-автомат содержит одно состояние и не содержит переходов, такие фишки представляют собой обычные цветные фишки, которые мы будем называть *элементарными ресурсами*.

Определение 4.24. *Пусть* $\Omega = \{a_1, \ldots, a_k\}$ *— конечное множество типов,* $\mathcal{A} = (A_1, \ldots, A_k)$ *— конечное множество P-автоматов, такое, что*

1. *для типа* $a_i \in \Omega$ *множество* $Const(a_i)$ *определяется как множество всех состояний P-автомата* A_i*;* $Const =_{def} \bigcup_{a \in \Omega} Const(a)$*;*
2. *каждый* A_i *есть P-автомат типа* a_i *над множеством типов* Ω *и множествами* Ω*-типизированных переменных* Var*, имён портов* Π *и констант* $Const$*.*

Сеть управляемых ресурсами автоматов *(P-сеть)* $RN = (\Omega, SN, (A_1, \ldots, A_k))$ *состоит из описанного выше конечного множества P-автоматов* $(A_1, \ldots, A_k)$ *и системной сети* SN *над множеством типов* Ω *и множествами* Ω*-типизированных имён портов* Π *и констант* $Const$*.*

Разметкой (состоянием) P-сети называется разметка лежащей в её основе системной сети.

Допуская некоторую вольность в обозначениях, условимся не различать автомат A и его имя/тип a. Далее мы будем писать $(a|s)$ для обозначения константы, соответствующей состоянию s автомата A типа a. В зависимости от контекста такая константа может называться *агентом*, *ресурсом* или просто *объектом*.

Определим интерливинговую семантику для P-сетей. Срабатыванием P-сети является последовательность срабатываний переходов её агентов. Таким образом, только агенты могут менять состояние системы.

Для срабатывания перехода t в агенте a требуются ресурсы, перечисленные во входных подтермах помечающего t ресурсного выражения. Входной терм $p?e$

описывает ресурс e, который должен быть получен через порт p системной сети (то есть этот ресурс должен находиться в том узле системной сети, из которого ведёт дуга порта p). Аналогично, выходные термы определяют ресурсы, производимые агентом (и узлы-получатели для этих ресурсов).

Режимы срабатывания перехода определяются возможными означиваниями переменных.

Определение 4.25. Означиванием (переменных) *называется функция*

$$\mathbf{bind} : Var \to Const,$$

такая, что для любой переменной $\varphi \in Var$ *выполняется* $Type(\mathbf{bind}(\varphi)) = Type(\varphi)$.

Пусть M — разметка Р-сети $RN = (\Omega, SN, (A_1, \ldots, A_k))$, $a|s$ — Р-автомат в состоянии s, находящийся в узле v сети RN при разметке M, t – переход в $a|s$. Пусть b — означивание переменных. Переход $t = (s, s')$ с меткой $l_a(t)$ *активен* при разметке M, если существует взаимно однозначное соответствие между входными подтермами в $l_a(t)$ и объектами (фишками) в M, такое, что для каждого подтерма $p?e$ существует объект $e[b]$ в узле v', соединенном с узлом v входной дугой, помеченной именем порта p.

Описанное выше соответствие определяет ’подразметку’ $\breve{M}$ системной разметки, описываемую входными подтермами выражения $l_a(t)$. Она включает все объекты, потребляемые срабатыванием перехода t при означивании b. Заметим, что для данных t и b разметка $\breve{M}$ определяется недетерминированно (может быть несколько вариантов разметок, удовлетворяющих условиям). Через $\mathbf{in}(t[b])$ мы обозначим множество всех таких разметок.

Аналогично, определим множество $\mathbf{out}(t[b])$ разметок, сопоставляющих узлам объекты, производимые переходом t при означивании b в соответствие с выходными подтермами $l_a(t)$.

Определение 4.26. *Пусть* $(a|s)$ *— агент, находящийся в узле* $v \in V$ *при разметке* M, $t \in T_a$ *— переход, такой, что* $t = (s, s')$ *для некоторого* s'.

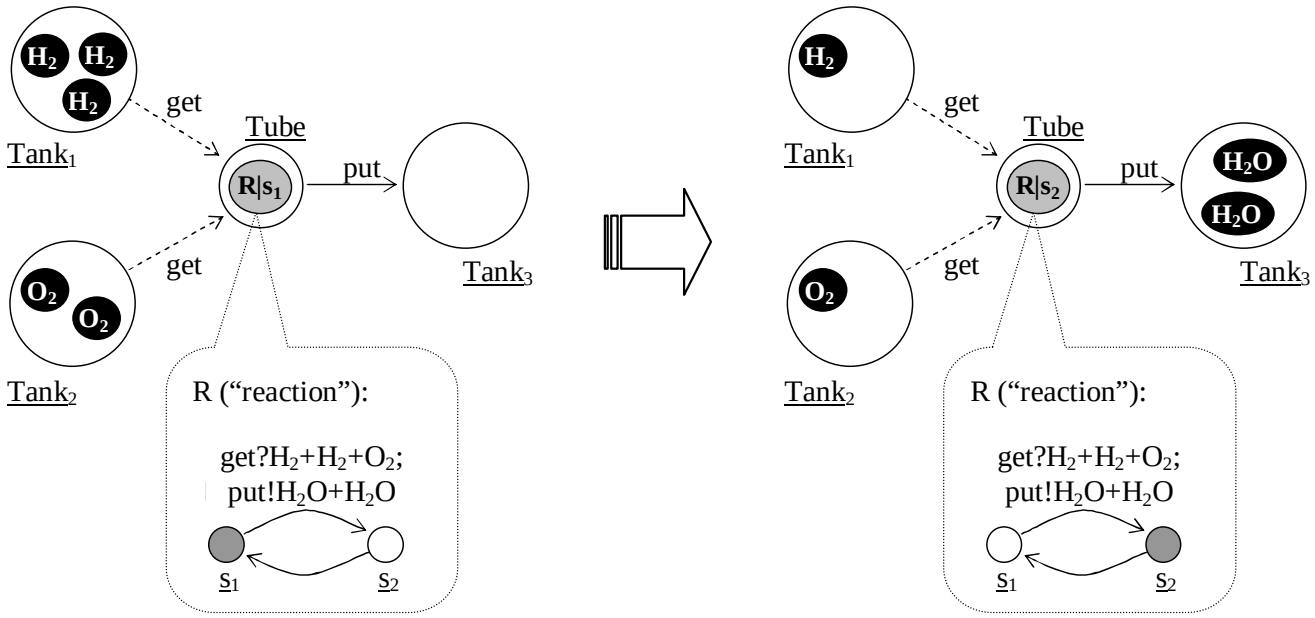


Рис. 4.32. Пример P-сети: химическая реакция

Переход t активен *с означиванием переменных b, если существует разметка* $\check{M} \in \mathbf{in}(t[b])$*, такая, что* $\forall u \in V : \check{M}(u) \subseteq M(u)$.

Активный означенный переход может (недетерминированно) сработать*, преобразуя M в новую разметку M′, такую, что* $\forall u \in V : M'(u) = M(u) - \check{M}(u) + \hat{M}$*, где* $\check{M} \in \mathbf{in}(t[b])$ *и* $\hat{M} \in \mathbf{out}(t[b])$.

Сети управляемых ресурсами автоматов позволяют моделировать различные аспекты ресурсно-зависимых систем. Во-первых, рассмотрим простейший классический пример — химическую реакцию (Рис. 4.32). Единственный активный агент данной системы – автомат *R* – определяет сам алгоритм реакции (“две молекуля водорода и одна молекула кислорода превращаются в две молекулы воды”). Системная сеть описывает физическую среду: три резервуара и трубу, в которой происходит реакция. Порты *get* и *put* связывают эти два уровня (или два аспекта) модели. С системной точки зрения порты — это физические “отверстия”, с точки зрения логики агента — это имена аргументов (параметров).

Системный уровень P-сети представляет собой помеченную сеть активных ресурсов, то есть помеченный ориентированный граф с двумя типами дуг — входными и выходными портами. Однако семантически эта сеть всё же отличается от

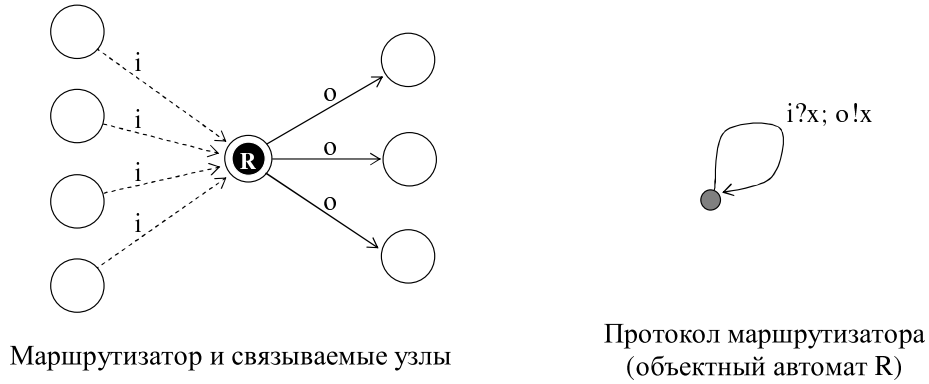


Рис. 4.33. Пример сети: недетерминированный маршрутизатор

AP-сетей. В AP-сетях инцидентные дуги однозначно определяют все ресурсы, которые *должны* быть потреблены и произведены агентом, расположенным в узле. В P-сетях они призваны определить узлы, которые *могут* быть использованы в качестве источников/получателей потребляемых/производимых агентом ресурсов. Так, например, в P-сети на Рис. 4.33 маршрутизатор может получить ресурс (пакет) по любому из четырех входных портов, и затем отправить по любому из трёх выходных. Подобный недетерминизм специфичен именно для P-сетей, посколку основан и на выборе означивания переменных (недетерминированном), и на выборе распределения задействуемых ресурсов по узлам сети (также недетерминированном). Разумеется, первый способ возможен и в сетях Петри высокого уровня — там также применяются означивание переменных и выражения на дугах. Однако второй способ требует большей гибкости в “пространственном” распределении ресурсов, которая возможна только в P-сетях.

Несмотря на новые возможности моделирования, P-сети обладают теми же возможностями анализа, что и сети Петри:

Теорема 4.20. *Класс P-сетей по выразительной мощности эквивалентен классу обыкновенных сетей Петри.*

Доказательство: Поскольку сети активных ресурсов (AP-сети) эквивалентны сетям Петри, нам достаточно доказать, что P-сети эквивалентны AP-сетям.

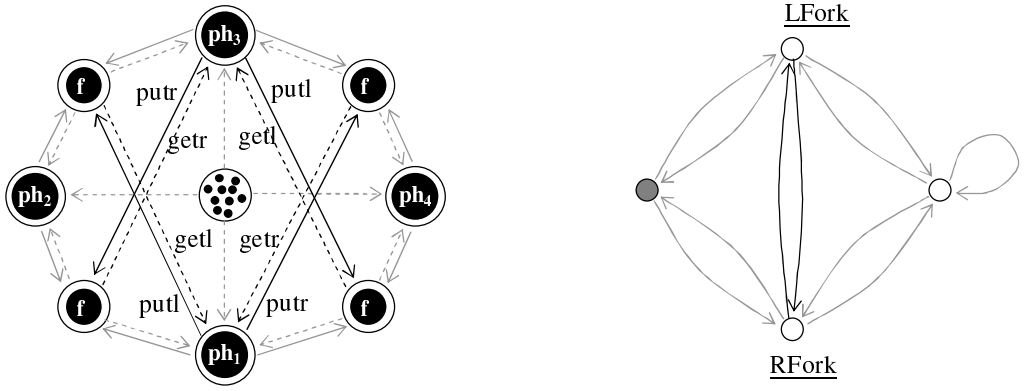
(⊇) Опишем схему трансформации AP-сети в эквивалентную P-сеть.

Во-первых, каждой дуге сопоставим уникальное имя порта. Далее, для каждого отдельного узла АР-сети, заменим каждую находящуюся в нём фишку на автомат с одним состоянием и одним переходам (простую петлю), переход в котором помечен ресурсным выражением, в котором упомянуты все инцидентные дуги (по их именам портов; входные — во входных подтермах, выходные — в выходных). Очевидно, что срабатывание этого перехода эквивалентно срабатыванию узла в исходной АР-сети.

(⊆) Опишем схему трансформации Р-сети в эквивалентную АР-сеть.

Во-первых, заметим, что в произвольной сети число различных типов Р-автоматов конечно. Кроме того, каждый Р-автомат содержит конечное число состояний. Следовательно, мы можем “закодировать” разметку системного узла мультимножеством над этим множеством пар (тип автомата, состояние). Заменим узел v данной системной Р-сети на множество узлов-“позиций” АР-сети, соответствующих парам (тип объектной сети a, состояние s автомата A). Разметка узла (a, s) будет соответствовать числу агентов типа a, находящихся в состоянии s в узле v исходной Р-сети.

Далее, заметим, что для любого перехода Р-автомата A, находящегося в узле v системной сети, существует конечное число различных означиваний переменных и распределений входных и выходных ресурсов по смежным узлам. Добавим в АР-сеть по одному новому узлу-“переходу” (с одной обычной фишкой внутри) для каждой такой комбинации, и свяжем его входными или выходными дугами с узлами-“позициями”, содержащими входные или выходные ресурсы соответствующего Р-автомата при соответствующем означивании переменных и пространственном распределении ресурсов. Срабатывание подобных узлов-“переходов” будет полностью эквивалентно срабатыванию переходов в агентах исходной Р-сети (с учётом означиваний). □



Прямоугольный стол (длиннорукие философы?)

Забывчивый философ

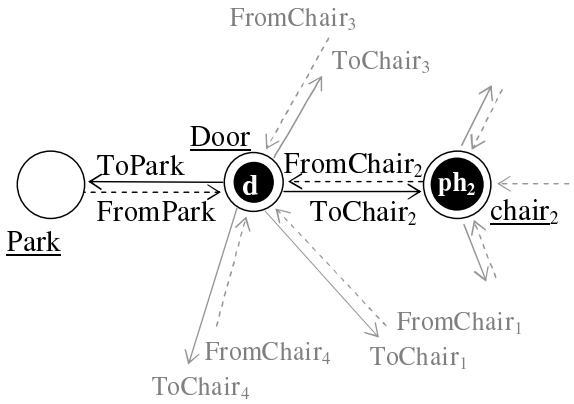
Рис. 4.34. Обедающие философы — простые модификации

4.4.3. Модельный пример: Обедающие Философы

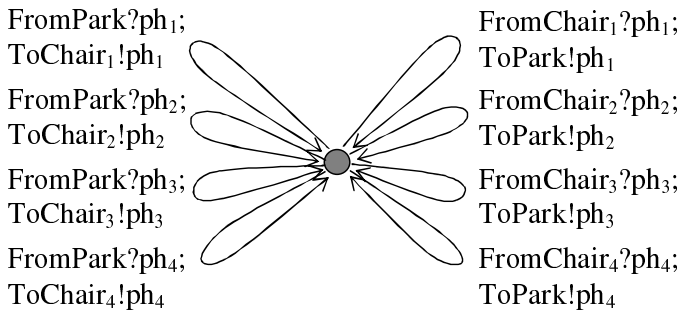
Разделение системного и объектного уровней позволяет нам достаточно легко моделировать иерархические структуры. Мы можем менять верхний и нижний слои независимо друг от друга, получая новые интересные модели (Рис. 4.34). Здесь в системной сети (слева) мы добавили новые порты для философов ph_1 and ph_3, которые позволяют им дотягиваться до *всех* вилок на столе. В P-автомате (справа) мы добавили два новых перехода, которые позволяют философу перекладывать вилки из одной руки в другую. Каждая из этих модификаций может привести к ситуации, когда на одной салфетке будут лежать несколько вилок.

Заметим, впрочем, что в обоих случаях не был изменен сам механизм синхронизации уровней — порты. В качестве более сложного примера рассмотрим модель “обедающих и прогуливающихся философов”. Теперь философы могут вставать из-за стола и уходить на прогулку в парк. На Рис. 4.35 представлена модель дворецкого, который провожает философа от стола в парк и обратно.

Также рассмотрим “компактный” вариант дворецкого — Рис. 4.36. Здесь мы использовали переменную *x* типа *philosopher* для упрощения ресурсных выражений. Модель действительно получилась более компактной, однако при этом в ней возник серьёзный сбой – дворецкий ослеп – он не способен разглядеть в руках философа вилку, которую тот по забывчивости мог не вернуть на стол (ведь дворецкий не различает константы типа *philosopher*). Так что теперь философы

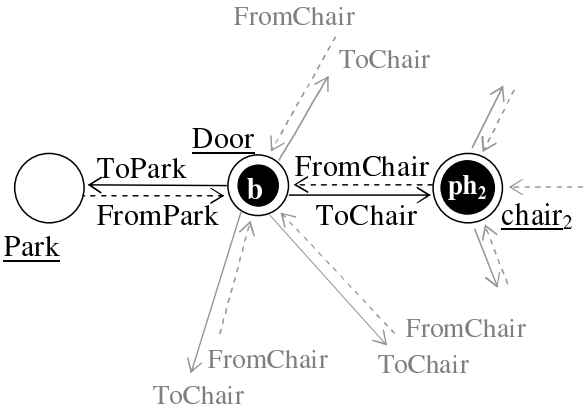


Парк, дверь и соответствующие порты (системная сеть)

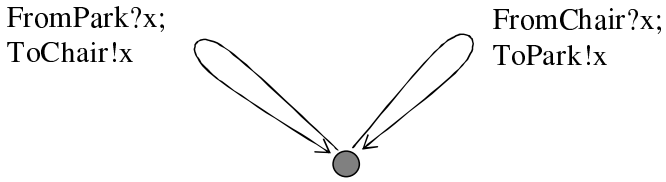


Дворецкий (объектный автомат d)

Рис. 4.35. Обедающие философы — дворецкий



Парк, дверь и соответствующие порты (системная сеть)



Слепой дворецкий (объектный автомат d)

Рис. 4.36. Обедающие философы — “слепой” дворецкий

могут гулять по парку с вилками в руках.

Более того, дворецкий не способен различать стулья (все его выходные порты имеют одно и то же имя), поэтому вполне может подвести нескольких философов к одному и тому же (уже занятому) стулу.

В ситуации с дворецким мы моделируем механизм пассивной транспортировки объектов. Философы — это всего лишь ресурсы для дворецкого, они не принимают участия в принятии решений. Но мы можем рассмотреть и ситуацию, в которой философы перемещаются самостоятельно (Рис. 4.37). Здесь использован более сложный агент — “бродячий” философ. Он способен сам перемещаться из одного места в другое, а также занимать любой незанятый стул (для пометки незанятых стульев используется специальный маркер *free*).

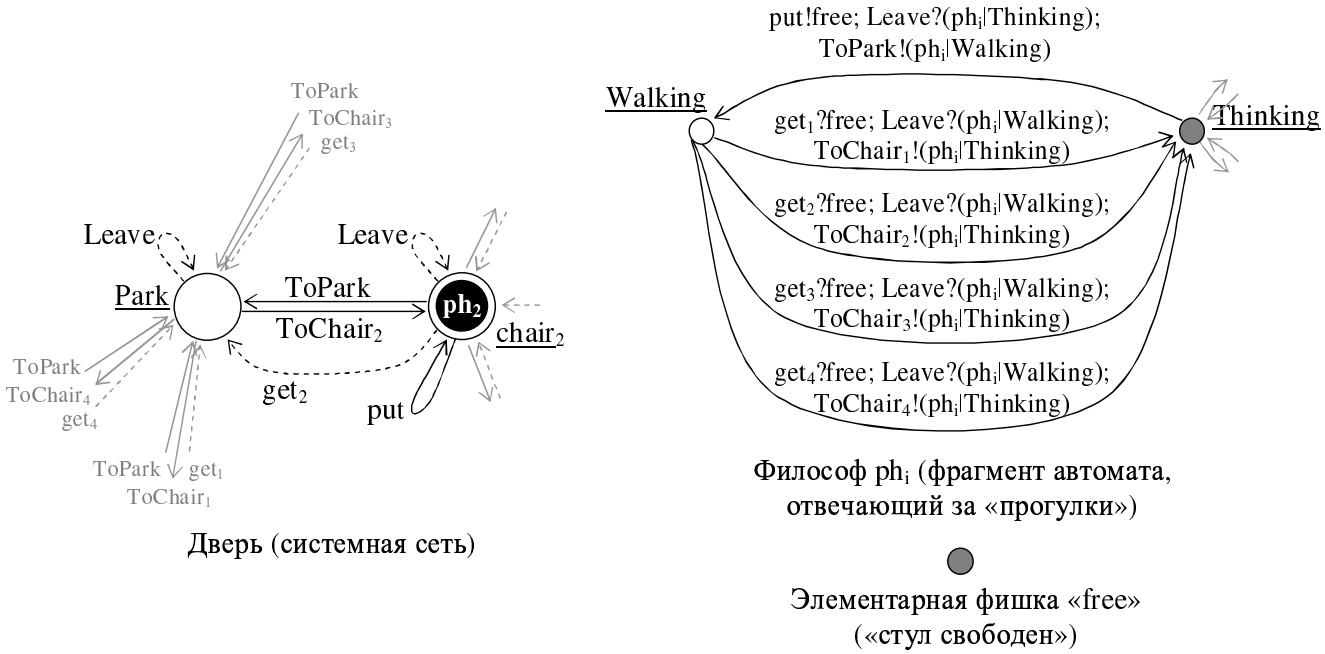


Рис. 4.37. Самостоятельные философы

Сети ресурсных автоматов моделируют явное и неявное взаимодействие агентов без введения каких-либо промежуточных слоёв и/или протоколов. Это позволяет достаточно адекватно отражать специфику предметной области при помощи единообразного и компактного синтаксиса. Агенты могут быть перемещены, созданы, скопированы и уничтожены другими агентами (внешняя мобильность). Они также могут выполнять все эти действия (над собой) по собственной инициативе (внутренняя мобильность).

Заметим, что выбор в качестве фишек автоматов, а не сетей Петри, нисколько не снижает выразительности формализма по сравнению с обыкновенными и высокоуровневыми сетями Петри (Теорема 4.20). При этом синтаксис языка получается максимально простым, что может позволить в дальнейшем строить различные гибридные и высокоуровневые расширения. В частности, использование вместо P-автоматов временных автоматов [59, 60] и их аналогов [96, 170, 183] уже позволило нам разработать ряд гибридных формализмов с моделью реального времени [89].

4.5. Клеточные сети с бесконечномерным ресурсом

4.5.1. Мотивация

Одной из интересных сторон P-сетей является возможность удобного моделирования пространственной динамики. Рассмотрим в качестве примера классическую задачу децентрализованной координации — модель “муравьиных колоний” (например, [130]).

Имеется конечное множество муравьев, произвольно блуждающих по конечному “лесу” (множество локаций, соединенных сетью тропинок) в поисках пищи. В отдельных локациях может находиться конечное число единиц пищи. Если муравей находит такую локацию (и что-то съедает, уменьшая количество доступной еды на единицу), то начинает в течение некоторого времени оставлять после себя след (феромон), помечающий пройденные локации. Если какой-то другой муравей находит этот след, то может двигаться по нему к пище.

Дополнительно усложним модель, считая, что муравьи способны определять стороны света: Север, Восток, Юг и Запад, и используют четыре типа феромонов в зависимости от направления движения (чтобы указывать точное направление к цели, т.е. помечать не только локации, но и тропинки). Кроме того, разрешим муравью “стирать” след, если этот след не привел его к пище (т.е. если локация истощилась).

Мы будем использовать:

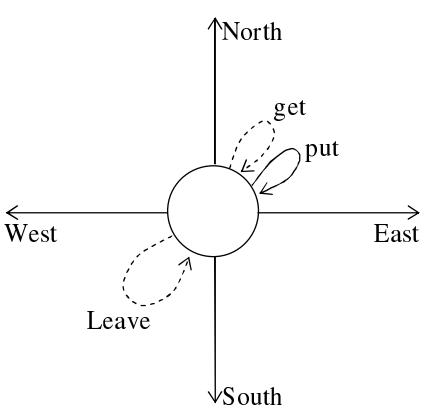
Типы:

item	(* что-то, находящееся на земле *)
ant	(* активный агент (автомат) *)

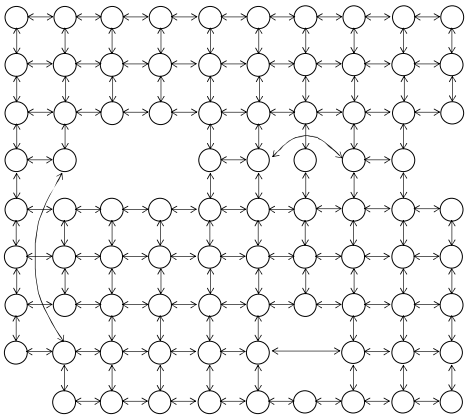
Константы:

fn, fe, fs, fw : item	(* феромоны: еда на севере/востоке/юге/западе *)
nfn, nfe, nfs, nfw : item	(* отсутствие соотв. феромонов *)
food : item	(* что-то съедобное *)
$a_1, a_2, \ldots, a_s$: ant	(* муравьи *)

Порты:



Локация (фрагмент системной сети)



“Лес” – схема системной сети (решетка локаций)

Рис. 4.38. Муравьиные колонии — пример системной сети

?get, !put : item (* “можно поднять/положить” *)

?Leave : ant (* “можно уйти” *)

!North, !East, !South, !West : ant (* “можно идти на север/восток/юг/запад” *)

Для простоты мы также используем четыре сокращенных обозначения:

MoveW ::= Leave?this; West!this MoveN ::= Leave?this; North!this

MoveS ::= Leave?this; South!this MoveE ::= Leave?this; East!this

Здесь *this* — также сокращение, которое обозначает текущий автомат в текущем состоянии. Например, надпись [*Leave*?*this*; *North*!*this*] на переходе из состояния *ate* в состояние *markingN* в автомате a_i обозначает выражение [*Leave*?(a_i|*ate*); *North*!(a_i|*markingN*)]. Другими словами, при помощи этого перехода автомат “переносит” сам себя из одного места в другое (через системные порты).

Системная сеть (Рис. 4.38) представляет собой топографическую карту “леса”. Она состоит из конечного числа узлов (локаций), объединенных рёбрами (тропинками) в подобие решетки (возможно, неполной). Каждый узел соединен не более чем с четырьмя “соседними” посредством выходных портов четырех типов: *North*, *East*, *South* и *West*. Через эти порты муравьи перемещаются в соседние локации, при этом для моделирования покидания текущей локации используется входной порт *Leave* . Порты *get* и *put* позволяют “брать” объекты (еду, феромоны) с земли и “класть” их обратно.

В начальном состоянии каждая локация содержит множество фишек *nfn*,

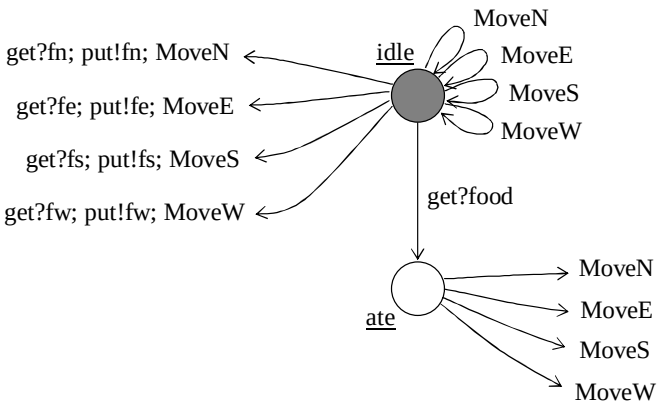


Рис. 4.39. Муравей — центральный фрагмент автомата

nfe, *nfs*, *nfw* (“нет никаких феромонов”), конечное (возможно, пустое) множество муравьёв и конечное (возможно, пустое) множество единиц пищи.

Муравей моделируется автоматом с 14 состояниями. Его ядро состоит всего из двух состояний: idle и ate. В состоянии idle возможны три варианта поведения: съесть пищу (если она есть в текущей локации); уйти из локации (недетерминированно) в одном из четырех направлений; обнаружив какой-нибудь феромон, уйти в соответствующем направлении. В состоянии ate муравей готовится начать новый феромонный след, выбирая (случайным образом) направление своего движения. Например, если он выбрал север, то так и продолжит двигаться на север, оставляя после себя след “sf-sf-sf-…” (“еда на юге”); если восток, то “wf-wf-wf-…” и т.д. На любом шаге (недетерминированно) он может вернуться в состояние idle, завершая тем самым пометку тропинки.

Рассмотрим часть диаграммы переходов автомата, в которой оставлены центральные состояния idle и ate с инцидентными переходами (Рис. 4.39).

Оставшаяся часть автомата состоит из четырех одинаковых фрагментов, отличающихся только стороной света (в зависимости от направления первого шага из состояния idle).

На Рис. 4.40 изображен вариант шага на север. Он содержит три дополнительных состояния: checkingN, clearingN и markingN.

Муравей находится в состоянии checkingN, если перед этим он двигался на

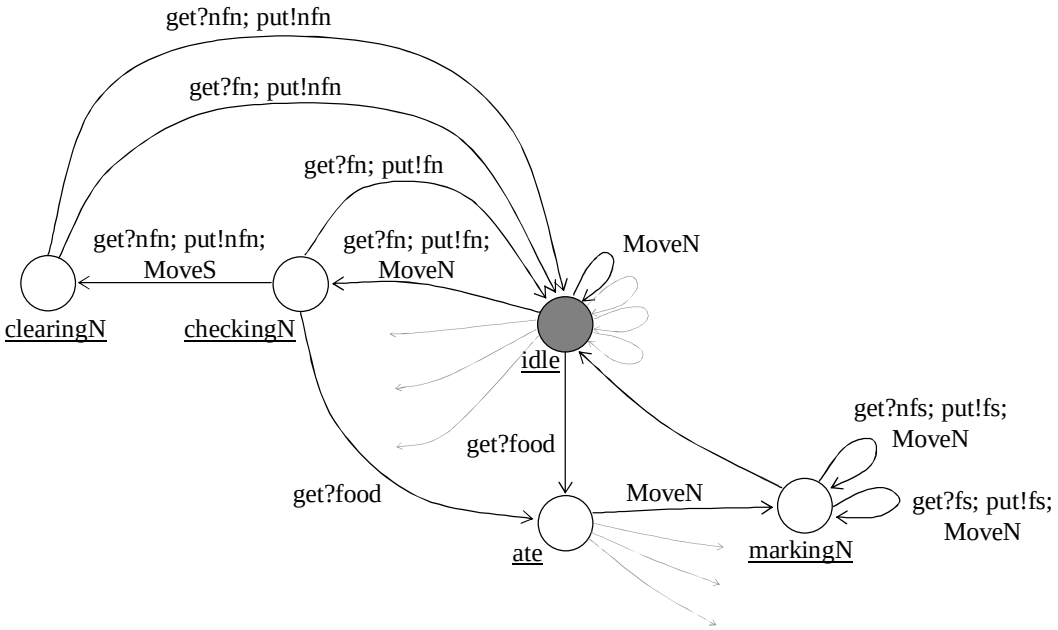


Рис. 4.40. Муравей — фрагмент “Идём на север”

север по следу *fn*. Если пища действительно имеется в данной локации, то она может быть съедена (переход в ate). Если феромонный след продолжается и в данной локации, то муравей может вернуться в состояние idle. Но если феромона *fn* нет (т.е. есть фишка *nfn*), то муравей может переместиться обратно на юг и “стереть” там след *fn* (состояние clearingN).

Муравей находится в состоянии markingN, если перед этим он сделал шаг на север в состоянии ate. В таком случае он делает случайное число шагов на север, помечая пройденные локации феромоном *fs*.

4.5.2. Клеточные P-сети

Реальный мир представляет собой достаточно большой (если не бесконечный) “лес”. Итак, рассмотрим конечные сообщества ресурсных автоматов, распределенных по узлам некоторой *бесконечной* (но регулярной) системной сети. Подобные структуры очень напоминают клеточные автоматы (КА) с их бесконечной решёткой и конечным набором правил эволюции для отдельной клетки, поэтому мы будем называть их клеточными P-сетями (Cellular Resource Driven Automata — CRDA). Фактически, наиболее близкий тип клеточных автоматов

— асинхронные КА, в которых клетки обновляются в произвольном (недетерминированном) порядке. Известно, что по выразительности асинхронные КА эквивалентны синхронным [175].

Однако имеется и существенная разница между CRDA и КА: у нас отдельная клетка содержит мультимножество фишек (активных P-автоматов или элементарных ресурсов), поэтому даже одноклеточная ресурсная сеть может обладать бесконечным множеством состояний. Кроме того, синтаксис организации межклеточных связей (входных и выходных портов) у нас более гибкий.

CRDA могут быть классифицированы в зависимости от способа начальной разметки:

1. *бесконечные CRDA:* все клетки бесконечной решетки, кроме конечного множества, помечены одним и тем же мультимножеством ресурсов;

2. *конечные CRDA:* все клетки бесконечной решетки, кроме конечного множества, пусты.

Известно, что клеточные автоматы обладают тьюринговой мощностью. В CRDA выразительная мощность отдельной ячейки выше, чем в классических КА — это уже не конечный автомат, а сеть Петри. Действительно, даже один-единственный нетривиальный “активный” P-автомат может работать, как счётчиковый автомат с потенциально неограниченными счётчиками (представленными количествами других “простых” фишек, находящихся с ним в одной ячейке). Следовательно, интуитивно ясно, что:

Теорема 4.21. *Бесконечные CRDA эквивалентны машинам Тьюринга.*

Доказательство: Рассмотрим некоторый асинхронный клеточный автомат. Правило срабатывания отдельной клетки в нём представляет собой конечный автомат M, “читающий” данные из соседних клеток. Мы “закодируем” это правило в P-автомат A^M и поместим копию A^M в каждую ячейку сети.

У автомата A^M два состояния — *dead* и *alive*. Пары соседних клеток соединены входными и выходными портами в обоих направлениях.

Рассмотрим правила срабатываний клетки. Возможны четыре типа срабатываний — рождение (*dead* → *alive*), умирание (*alive* → *dead*), выживание (*alive* → *alive*) и отсутствие жизни (*dead* → *dead*). Например, рассмотрим правило рождения. Мы будем моделировать его в A^M конечным множеством переходов из *dead* в *alive* — по одному переходу для каждого варианта подходящих значений соседних клеток (соответствующие значения проверяются посредством получения ресурсов из соседних клеток через входные порты и последующим возвращением их обратно через выходные порты). Оставшиеся три типа правил моделируются сходным образом. □

Конечные CRDA формализуют иной подход к моделированию (он, в определённом смысле, ближе по духу к сетям Петри): множество устройств-вычислителей (размеченных клеток) всегда конечно. В общем случае это множество может разрастаться, причём двумя способами: “в ширину” (количество размеченных клеток) и “в глубину” (количество фишек). Очевидно, что первый способ необходим для моделирования универсальных вычислений — в противном случае мы получаем конечную решетку и, следовательно, обычную P-сеть (сеть Петри). Но достаточен ли он для построения машин Тьюринга?

Далее мы оценим выразительную мощность нескольких простых классов конечных CRDA. Заметим, что клеточные автоматы универсально мощны даже в одномерном случае (гипотеза С. Вольфрама [208], доказанная М. Куком [112]), где решетка представляет собой цепочку клеток. Поэтому мы также будем рассматривать только одномерные сети.

Иерархия классов будет определена на основе ограничений топологии системной сети – множества доступных отдельной клетке портов. Наиболее общий случай изображен на Рис. 4.41 (соответствующий класс мы будем называть 1-dim CRDA). Активные фишки (P-автоматы), находящиеся в клетке, могут потреблять/производить ресурсы из трёх клеток — левой соседней (“West”), правой

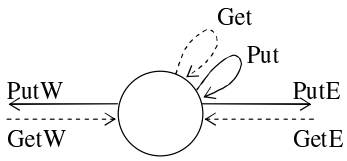
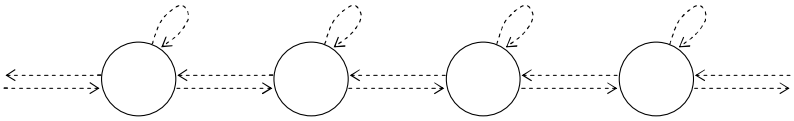
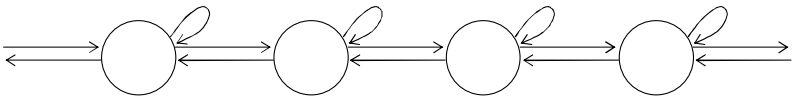


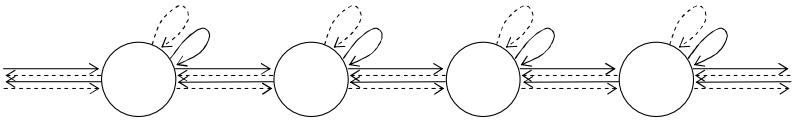
Рис. 4.41. Полный набор индивидуальных портов



a) 1-dim iCRDA (только входные порты)



b) 1-dim oCRDA (только выходные порты)



c) 1-dim CRDA (все порты)

Рис. 4.42. Типы решёток

соседней (“East”) и текущей. Ограничений на фишки нет.

Мы рассмотрим три ключевых класса иерархии, порождаемой топологией одномерной решётки:

1. сети, в которых есть только входные порты (1-dim iCRDA);
2. сети, в которых есть только выходные порты (1-dim oCRDA);
3. сети со всеми типами портов (1-dim CRDA).

Соответствующие решётки изображены на Рис. 4.42.

Теорема 4.22. *1-dim iCRDA эквивалентны конечным автоматам.*

Доказательство: Оценим количество различных достижимых состояний в 1-dim iRCDA. Заметим, что множество непустых клеток не может расти, поскольку

P-автоматы без выходных портов не могут ничего положить в соседние клетки. Более того, они не могут ничего положить даже в собственную клетку — поэтому её разметка также ограничена. Таким образом, любой 1-dim iRCDA представляет собой конечный набор P-автоматов, и, следовательно, может быть промоделирован некоторым конечным автоматом. □

Сетями Петри без коммуникаций (Communication-free nets, также BPP-nets) называются сети Петри, в которых каждый переход имеет ровно одну входную позицию (см., например, [119]). Они тесно связаны с алгеброй простых параллельных процессов (Basic Parallel Processes — BPP [104]), образующей важный подкласс милнеровской CSS.

Лемма 4.2. *Для любой 1-dim oCRDA существует бисимулярная сеть Петри без коммуникаций.*

Доказательство: Во-первых, заметим, что в ситуации без входных портов агент (активная фишка) не может наблюдать содержимое соседних клеток. Единственная информация, которой он обладает — его собственное состояние. Таким образом, его поведение не зависит от клетки, в которой он находится. Следовательно, мы можем заменить все *PutW*- и *PutE*-выражения на *put*-выражения, и это не нарушит наблюдаемое поведение системы в целом.

Таким образом, мы показали, что одноклеточные oCRDA бисимулярны одномерным oCRDA, и теперь нам нужно доказать, что для любой одноклеточной oCRDA существует бисимулярная сеть без коммуникаций.

Симуляция одноклеточной oCRDA сетью Петри без коммуникаций может быть построена следующим образом:

1. Для каждого состояния s каждого P-автомата x добавим в сеть новую позицию $(x|s)$.

2. Для каждого перехода $t = (s, s')$ каждого P-автомата x и каждого означивания переменных b добавим в сеть переход $t[b]$ с входной дугой $((x|s), t[b])$

и выходной дугой $(t[b], (x|s'))$. Дополнительно, для каждого выходного выражения $put!e$, приписанного переходу t P-автомата x, и для каждого $e[b] = (y|u)$ для некоторого P-автомата y и его состояния u, добавим выходную дугу $(t, (y|u))$.

3. Начальная разметка получается следующим образом: мы помещаем в каждую позицию $(x|s)$ в точности начальное количество P-автоматов x, находящихся в управляющем состоянии s.

Легко убедиться в том, что естественная биекция начальных разметок oCRDA и сети Петри является бисимуляцией: переход на одной стороне активен тогда и только тогда, когда активен соответствующий переход на другой стороне. □

Теорема 4.23. *1) Сети Петри без коммуникаций ⊆ 1-dim oCRDA;*
2) Сети Петри ⊈ 1-dim oCRDA;
3) 1-dim oCRDA ⊂ машины Тьюринга.

Доказательство: (1) Моделирование сети Петри одноклеточной oCRDA может быть организовано следующим образом. Построим P-автомат с управляющими состояниями, соответствующими позициям сети. Для каждого перехода сети добавим в P-автомат переход из состояния, соответствующего входной позиции, в состояние, соответствующее какой-нибудь из выходных позиций. Все остальные выходные дуги перехода сети заменяются выходными ресурсными выражениями вида $put!\cdot$, приписанными к моделирующему переходу P-автомата и производящими нужное количество P-автоматов в соответствующих управляющих состояниях.

В начальной разметке мы просто заменяем каждую фишку на P-автомат в соответствующем управляющем состоянии, помещенный в некоторую клетку решетки (можно даже поместить их все в одну и ту же клетку).

(2) Следует из разрешимости бисимуляции для сетей Петри без коммуникаций [105], неразрешимости бисимуляции для обыкновенных сетей Петри [140]

и Леммы 4.2.

(3) Следует из (2) и неуниверсальности сетей Петри. □

Теоремы 4.22 и 4.23 показывают, что:

1. В CRDA выходные порты в некотором смысле мощнее, чем входные.

2. Только входных портов недостаточно для эффективного использования бесконечности решётки: мы вполне можем ограничить рабочую область конечным числом клеток. В действительности достаточно даже одной клетки — легко заметить, что одноклеточные iRDA также эквивалентны конечным автоматам.

3. Только выходные порты также не могут в полной мере использовать бесконечность решётки, но теперь уже с точностью до бисимулярности: мы можем сузить рабочую область до одной клетки без нарушения наблюдаемого поведения.

С другой стороны, полный набор связей (портов) обладает универсальной мощностью:

Теорема 4.24. *1-dim CRDA = машины Тьюринга.*

Доказательство: Использовать здесь тот же подход, что и в доказательстве Теоремы 4.21, уже не получится. Действительно, решётка теперь “почти пуста” — это ключевое отличие от клеточных автоматов, в которых бесконечная решётка изначально “полна” — каждая клетка содержит конечный автомат (правило срабатывания).

Используем метод сведения к известной универсальной модели вычислений — докажем, что для любой машины Минского [172] (двухсчётчикового автомата с проверкой на ноль) можно построить эквивалентную CRDA.

Машина Минского со счётчиками c_1 и c_2 представляет собой последовательность команд

$$1:\ COMM_1;\quad 2:\ COMM_2;\quad \ldots;\quad n:\ COMM_n$$

где $COMM_n$ является завершающей $HALT$-командой, а $COMM_i$ ($i = 1, \ldots, n-1$) — командой одного из двух типов ($1 \le k, m \le n, 1 \le j \le 2$):

1) $i:\ c_j := c_j + 1;\ goto\ k$

2) $i:\ if\ c_j = 0\ then\ goto\ k\ else\ (c_j := c_j - 1;\ goto\ m)$

Симулирующая 1-dim CRDA строится следующим образом.

Решётку (цепочку клеток, ленту) мы символически делим на три части: бесконечную левую полуленту; центральную клетку; бесконечную правую полуленту. В центральной клетке находится специальная фишка `C` (признак центра). Значение первого счётчика (c_1) запомнено в левой полуленте: ровно c_1 клеток слева от центральной содержат по одной фишке вида • (и между заполненными клетками нет пустых). Слева от самой левой помеченной фишкой • клетки находится клетка, помеченная признаком левой (западной) границы — специальной фишкой `W`. Ещё левее все клетки пусты.

Аналогично, значение второго счётчика (c_2) хранится в правой (восточной) полуленте. Единственное отличие — граница помечена фишкой `E`, а не `W`.

Описанная структура будет сохраняться в течение всего процесса вычислений конструируемой CRDA. Например, увеличение счётчика c_1 будет моделироваться удлинением левой полуленты на единицу, уменьшение, соответственно, — укорачиванием.

Программа машины Минского моделируется единственным P-автоматом `Auto`, находящимся в центральной клетке. У него n управляющих состояний, соответствующих всем командам исходной программы (изначально он находится в первом состоянии), и набора переходов: по одному для каждой команды инкремента (увеличения счётчика на единицу) и по две для каждой команды условного декремента (усечённого вычитания единицы).

Рассмотрим более детально команды, изменяющие первый счётчик (в синтаксисе $COMM_i$, описанном выше).

1) $i:\ c_1 := c_1 + 1;\ goto\ k$

Эта команда моделируется переходом из состояния i в состояние k:

$$i \to [Get?(\texttt{Auto}|i);\ PutW!(\texttt{IncW}|k)] \to k.$$

При этом автомат `Auto` удаляется, а в ближайшую левую клетку помещается новый автомат `IncW`. Этот автомат "запоминает" значение k при помощи своего внутреннего состояния.

Автомат `IncW` состоит из n состояний, на каждое из которых "прицеплено" по две петли (перехода из k в k). Рассмотрим состояние k, соответствующие ему петли определяются следующим образом:

$$k \to [Get?\bullet;\ Get?(\texttt{IncW}|k);\ PutW!(\texttt{IncW}|k);\ Put!\bullet] \to k;$$
$$k \to [Get?\texttt{W};\ Get?(\texttt{IncW}|k);\ PutW!\texttt{W};\ Put!\bullet;\ PutE!(\texttt{MoveE}|k)] \to k.$$

Автомат `MoveE` также состоит из n состояний, на каждом теперь уже по три петли. Состоянию k соответствуют петли:

$$k \to [Get?\bullet;\ Get?(\texttt{MoveE}|k);\ Put!\bullet;\ PutE!(\texttt{MoveE}|k)] \to k;$$
$$k \to [Get?\texttt{W};\ Get?(\texttt{MoveE}|k);\ Put!\texttt{W};\ PutE!(\texttt{MoveE}|k)] \to k;$$
$$k \to [Get?\texttt{C};\ Get?(\texttt{MoveE}|k);\ Put!\texttt{C};\ Put!(\texttt{Auto}|k)] \to k.$$

Первая петля "перемещает" автомат `MoveE` от левой границы к центру. Третья петля может сработать только в центральной клетке: она стирает `MoveE` и восстанавливает `Auto` в нужном состоянии. Назначение второй петли станет ясным несколько позже.

2) $i:\ if\ c_1 = 0\ then\ goto\ k\ else\ (c_1 := c_1 - 1;\ goto\ m)$

Эта команда моделируется двумя переходами: из состояния i в состояние k и из состояния i в состояние m:

$$i \to [GetW?\bullet;\ Get?(\texttt{Auto}|i);\ PutW!\bullet;\ PutW!(\texttt{DecW}|m)] \to m;$$
$$i \to [GetW?\texttt{W};\ PutW!\texttt{W}] \to k.$$

Заметим, что второй переход может сработать только в том случае, когда первый счётчик пуст (фишка `W` “видна” автомату `Auto`).

Автомат `DecW` состоит из n состояний, на каждом по две петли. Состоянию m соответствуют петли:

$$m \to [\mathit{Get}?\bullet;\ \mathit{Get}?(\texttt{DecW}|m);\ \mathit{PutW}!(\texttt{DecW}|m);\ \mathit{Put}!\bullet] \to m;$$
$$m \to [\mathit{Get}?\texttt{W};\ \mathit{Get}?(\texttt{DecW}|m);\ \mathit{PutE}!\texttt{W};\ \mathit{PutE}!(\texttt{MoveE}|m)] \to m.$$

Автомат `MoveE` уже определён выше. Теперь мы видим, для чего была нужна вторая петля, — она позволяет нам переместиться на “восток” от граничной клетки, помеченной фишкой `W`.

Второй счётчик моделируется тем же способом — достаточно заменить в определениях автоматов букву `W` на букву `E`, и наоборот. □

Заключение

Основные результаты диссертации:

1. Новые методы поиска эквивалентных ресурсов в сетях Петри. Два ресурса эквивалентны (подобны), если они взаимозаменяемы без нарушения видимого поведения системы (относительно бисимуляции). Предложены обладающие конструктивными свойствами расширения и сужения подобия ресурсов, а также методы бисимуляционной редукции систем и адаптивного управления процессами на основе отношений эквивалентности ресурсов.

2. Методы символьного анализа односчетчиковых сетей Петри — систем с одной неограниченной “переменной” (ресурсом). Показано, что для таких систем бесконечная часть пространства состояний описывается при помощи арифметических прогрессий, характеристики которых выражаются как числа Фробениуса от эффектов циклов сильно связных компонент сети. Предложены символьные методы анализа односчетчиковых сетей (глобальная достижимость, глобальная верификация темпоральной логики EF, аппроксимация бисимуляции, избыточность использования счетчика).

3. Развитие языка сетей Петри засчет явного синтаксического выделения ресурсов и наделения их расширенной семантикой: формализм сетей активных ресурсов (AP-сетей). Новые способы формализации семантических свойств систем со сложным поведением агентов, композициональные методы анализа систем, алгоритм нормализации модульной структуры. Предложен ряд формализмов, использующих концепцию AP-сетей для моделирования тех или иных аспектов распределенных систем: AP-сети с динамическими и ненадежными компонентами, вложенные AP-сети, управляемые ресурсами автоматы (P-сети) и др.

4. Формализм клеточных P-сетей — обобщение сетей Петри на случай бес-

конечной регулярной системной сети (“гибрид” сетей Петри и клеточных автоматов). Построена и исследована иерархия классов одномерных клеточных сетей (цепочек), основанная на ограничении топологии системной сети (начальный класс иерархии эквивалентен конечным автоматам, заключительный — машинам Тьюринга).

Благодарности

Автор глубоко признателен своему научному консультанту проф. И. А. Ломазовой за плодотворное сотрудничество, интересные обсуждения и полезные советы. Автор также благодарен проф. В. А. Соколову и всему коллективу кафедры теоретической информатики ЯрГУ за поддержку и внимание.

Литература

1. Алгоритмы, математическое обеспечение и архитектура многопроцессорных вычислительных систем / Под ред. Ершова А. П. Новосибирск: Наука, 1982.

2. *Ачасова С. М., Бандман О. Л.* Корректность параллельных вычислительных процессов. Новосибирск: Наука, 1990.

3. *Башкин В. А.* Методы насыщения сетей Петри с невидимыми переходами // *Моделирование и анализ информационных систем*. 1999. Т. 6. С. 16–20.

4. *Башкин В. А.* Бисимуляция ресурсов в сетях Петри с невидимыми переходами // *Современные проблемы математики и информатики: Сборник научных трудов молодых ученых, аспирантов и студентов*. 2002. Т. 5. С. 79–85.

5. *Башкин В. А.* Бисимуляция ресурсов в сетях Петри // Тез. доклада на IV ежегодной научно-практической конференции студентов, аспирантов и молодых ученых Ярославской области. 2003. С. 15.

6. *Башкин В. А.* Бисимуляция ресурсов в сетях Петри: Дис. канд. физ.-мат. наук / Ярославль: ЯрГУ. 2003.

7. *Башкин В. А.* Конечное представление отношений на мультимножествах // *Моделирование и анализ информационных систем*. 2003. Т. 10. С. 56–59.

8. *Башкин В. А.* Моделирование очереди (FIFO) рекурсивными вложенными сетями Петри // Современные проблемы математики и информатики. Ярославль: ЯрГУ, 2005. Т. 7.

9. *Башкин В. А.* Расширение бисимуляции разметок ограниченных сетей Петри // Дискретные модели в теории управляющих систем. Труды VII-ой Международной конференции. М.: Изд-во ВМиК МГУ, 2006.

10. *Башкин В. А.* Свойства бисимуляции разметок в ограниченных сетях Петри // *Моделирование и анализ информационных систем*. 2006. Т. 13, № 1. С. 35–40.

11. *Башкин В. А.* О междисциплинарных связях курса “Сети Петри” // Преподавание математики в классическом университете: Тезисы докладов научно-методической конференции преподавателей математического факультета ЯрГУ. Ярославль: ЯрГУ, 2007. С. 14–17.

12. *Башкин В. А.* Сети активных ресурсов // **Моделирование и анализ информационных систем**. 2007. Т. 14, № 4. С. 13–19.

13. *Башкин В. А.* Об одном способе моделирования мультиагентных систем с динамической структурой // Параллельные вычисления и задачи управления. Труды IV-й международной научной конференции PACO’2008. М.: ИПУ РАН, 2008. С. 1462–1482.

14. *Башкин В. А.* Специальные виды бисимуляции разметок ограниченных сетей Петри // Материалы зимних научных чтений ФСИТ РГСУ (1-4 февраля 2006 года). М.: Логос, 2008.

15. *Башкин В. А.* Модели потоков работ. Ярославль: ЯрГУ, 2009.

16. *Башкин В. А.* Построение дерева достижимых состояний в односчетчиковых сетях Петри // Компьютерные науки и технологии. Сборник трудов Первой Международной научно-технической конференции. Т. 1. Белгород: 2009. С. 19–22.

17. *Башкин В. А.* Сети активных ресурсов как обобщение двойственных сетей Петри // Дискретные модели в теории управляющих систем. Труды VIII-ой Международной конференции. М.: МАКС Пресс, 2009. С. 23–28.

18. *Башкин В. А.* Верификация на основе моделей с одним неограниченным

счетчиком // **Информационные системы и технологии**. 2010. Т. 4(60). С. 5–12.

19. *Башкин В. А.* Глобальная верификация свойств достижимости систем с одним неограниченным ресурсом // Информационные технологии в науке, образовании и производстве. Материалы IV Международной научно-технической конференции. Т. 2. Орел: 2010. С. 6–11.

20. *Башкин В. А.* Об использовании одноперiодических базисов для глобальной символьной верификации // Труды семинара “Семантика, спецификация и верификация программ: теория и приложения”. Казань: 2010. С. 26–31.

21. *Башкин В. А.* Формализация семантики систем с ненадежными агентами // **Программирование**. 2010. Т. 36, № 4. С. 3–15.

22. *Башкин В. А.* Построение приближений бисимуляции в односчетчиковых сетях // **Моделирование и анализ информационных систем**. 2011. Т. 18, № 4. С. 34–44.

23. *Башкин В. А.* Модульные сети активных ресурсов // **Автоматика и вычислительная техника**. 2012. Т. 1. С. 5–18.

24. *Башкин В. А.* Наследственные свойства модульных сетей // **Моделирование и анализ информационных систем**. 2012. Т. 19, № 6. С. 9–20.

25. *Башкин В. А.* Об эффективном моделировании неограниченного ресурса при помощи односчетчиковых контуров // **Моделирование и анализ информационных систем**. 2013. Т. 20, № 2. С. 139–156.

26. *Башкин В. А., Ломазова И. А.* О языках вложенных рекурсивных сетей Петри // Интеллектуальное управление: новые интеллектуальные технологии в задачах управления (ICIT’99). Труды международной конференции, Переславль-Залесский, 6–9 декабря 1999. М.: Наука. Физматлит, 1999. С. 9–13.

27. *Башкин В. А., Ломазова И. А.* Бисимуляция ресурсов в сетях Петри // **Известия РАН: Теория и системы управления**. 2003. Т. 4. С. 115–123.

28. *Башкин В. А., Ломазова И. А.* Подобие обобщенных ресурсов в сетях Петри // Труды второй всероссийской конференции “Методы и средства обработки информации” (МСО-2005). М.: Изд-во ВМК МГУ, 2005. С. 330–336.

29. *Башкин В. А., Ломазова И. А.* О параметризованном построении подобия ресурсов в сетях Петри // Интеллектуальные системы и компьютерные науки. Труды IX-ой Международной конференции. Т. 1. М.: Изд-во мехмата МГУ, 2006. С. 56–58.

30. *Башкин В. А., Ломазова И. А.* Эквивалентность ресурсов в моделях потоков работ // Программные системы: теория и приложения. Труды международной конференции. Т. 1. Переславль-Залесский: ИПС РАН, 2006. С. 323–338.

31. *Башкин В. А., Ломазова И. А.* О поиске эквивалентных ресурсов в сложных системах // **Известия ОрелГТУ. Серия “Фундаментальные и прикладные проблемы техники и технологии: информационные системы и технологии”**. 2008. Т. 1-2 / 269 (544). С. 33–39.

32. *Башкин В. А., Ломазова И. А.* Эквивалентность ресурсов в сетях Петри. М.: Научный мир, 2008.

33. *Башкин В. А., Ломазова И. А.* Двухуровневое моделирование мультиагентных систем на основе обобщенных сетей активных ресурсов // Труды семинара “Семантика, спецификация и верификация программ: теория и приложения”. Казань: 2010. С. 32–36.

34. *Башкин В. А., Ломазова И. А.* Моделирование мультиагентных систем с помощью обобщенных сетей активных ресурсов // **Кибернетика и системный анализ**. 2011. Т. 2. С. 31–39.

35. *ван дер Аалст В., ван Хей К.* Управление потоками работ: модели, методы и системы. М.: Научный мир, 2007.

36. *Верещагин Н. К., Шень А.* Лекции по математической логике и теории алгоритмов. Ч. 2: Языки и исчисления. М.: МЦНМО, 2002.

37. *Гашков С. Б., Чубариков В. Н.* Арифметика. Алгоритмы. Сложность вычислений. М.: Высшая Школа, 2000.

38. *Диль Ю. В.* Исследование массовых алгоритмических проблем для супердвойственных сетей Петри и сетей активных ресурсов // Сборник научных работ студентов и аспирантов. Ярославль: ЯрГУ, 2008.

39. *Захаров В. А.* Проверка эквивалентности программ при помощи двухленточных автоматов // *Кибернетика и системный анализ*. 2010. Т. 4. С. 39–48.

40. *Коннов И. В.* Применение ослабленных отношений симуляции в методе сетевых инвариантов для верификации параметризованных асинхронных моделей // *Моделирование и анализ информационных систем*. 2010. Т. 15, № 3. С. 3–13.

41. *Котов В. Е.* Сети Петри. М.: Наука, 1984.

42. *Летичевский А. А.* Практические методы распознавания эквивалентности дискретных преобразователей и схем программ // *Кибернетика*. 1973. Т. 4. С. 15–26.

43. *Ломазова И. А.* Моделирование мультиагентных динамических систем вложенными сетями Петри // Программные системы: Теоретические основы и приложения. М.: Наука, 1999. С. 143–156.

44. *Ломазова И. А.* Некоторые алгоритмы анализа для многоуровневых вложенных сетей Петри // *Известия РАН: Теория и системы управления*. 2000. Т. 6. С. 965–974.

45. *Ломазова И. А.* Рекурсивные вложенные сети Петри: анализ семантических свойств и выразительность // *Программирование*. 2001. Т. 4. С. 21–35.

46. *Ломазова И. А.* Объектно-ориентированные сети Петри: формальная семантика и анализ // *Системная информатика*. 2002. Т. 8, № 8. С. 143–205.

47. *Ломазова И. А.* Вложенные сети Петри: моделирование и анализ распределенных систем. М.: Научный мир, 2004.

48. *Петровский А. Б.* Основные понятия теории мультимножеств. М.: Едиториал УРСС, 2002.

49. *Питерсон Д.* Сети Петри и моделирование систем. М.: Мир, 1984.

50. *Подловченко Р. И.* О проблеме эквивалентных преобразований программ // *Программирование*. 1986. Т. 6. С. 3–14.

51. *Подловченко Р. И.* Эквивалентные преобразования схем программ с коммутирующими и монотонными операторами // *Программирование*. 2002. Т. 6. С. 301–313.

52. *Подловченко Р. И., Хачатрян В. Е.* Об одном подходе к разрешению проблемы эквивалентности // *Программирование*. 2004. Т. 3. С. 1–17.

53. *Сидорова Н. С.* Преобразования сетей Петри: Дис. канд. физ.-мат. наук / Ярославль: ЯрГУ. 1998.

54. *Сидорова Н. С., Соколов В. А.* О редукции сетей Петри // Тез. докладов на Третьей международной конференции по алгебре памяти М.И.Каргаполова. Красноярск. 1993. С. 305–306.

55. *Соколов В. А., Кушнаренко О. Б.* Рекурсивно-параллельное программирование и сети Петри: моделирование, анализ и верификация программ // *Моделирование и анализ информационных систем*. 1994. Т. 2. С. 98–102.

56. *Тарасюк И. В.* Эквивалентностные понятия для моделей параллельных и распределенных систем: Дис. канд. физ.-мат. наук / Новосибирск: ИСИ СО РАН. 1997.

57. *Харари Ф.* Теория графов. М.: Мир, 1973.

58. *Abdulla P. A., Čerans K.* Simulation is decidable for one-counter nets // CONCUR'98 Concurrency Theory / Ed. by D. Sangiorgi, R. Simone. Springer Berlin Heidelberg, 1998. Vol. 1466 of *Lecture Notes in Computer Science*. P. 253–268.

59. *Alur R., Dill D.* Automata for modeling real-time systems // Automata, Languages and Programming / Ed. by M. S. Paterson. Springer Berlin Heidelberg, 1990. Vol. 443 of *Lecture Notes in Computer Science*. P. 322–335.

60. *Alur R., Dill D. L.* A theory of timed automata // *Theoretical Computer Science*. 1994. Vol. 126, no. 2. P. 183–235.

61. *Autant C., Pfister W., Schnoebelen P.* Place bisimulations for the reduction of labeled Petri nets with silent moves // Proc. 6th Int. Conf. on Computing and Information, Peterborough, Canada. 1994.

62. *Autant C., Schnoebelen P.* Place bisimulations in Petri nets // Application and Theory of Petri Nets 1992 / Ed. by K. Jensen. Springer Berlin Heidelberg, 1992. Vol. 616 of *Lecture Notes in Computer Science*. P. 45–61.

63. *Baeten J. C. M., Bergstra J. A., Klop J. W.* Decidability of bisimulation equivalence for process generating context-free languages // *J. ACM*. 1993. Vol. 40, no. 3. P. 653–682.

64. *Barkaoui K., Ben Ayed R., Sbaï Z.* Workflow Soundness Verification based on Structure Theory of Petri Nets // *International Journal of Computing and Information Sciences*. 2007. Vol. 5, no. 1. P. 51–61.

65. *Barkaoui K., Petrucci L.* Structural Analysis of Workflow Nets with Shared Resources // Proceedings of Workflow Management: Net-based Concepts, Models, Techniques and Tools (WFM'98) / Ed. by W. M. P. van der Aalst, G. Michelis, C. A. Ellis. Vol. 98/7 of *Computing Science Reports*. Eindhoven University of Technology, Eindhoven, 1998. P. 82–95.

66. *Bashkin V.* Approximating bisimulation in one-counter nets // *Automatic Control and Computer Sciences*. 2012. Vol. 46, no. 7. P. 317–323.

67. *Bashkin V. A.* Applications of Marking Bisimulation for Adaptive Workflow Management // Concurrency, Specification and Programming. Proc. of CS&P 2005 / Ed. by L. Czaja. Warsaw University, Warsaw, 2005. P. 41–49.

68. *Bashkin V. A.* Nets of active resources for distributed systems modeling // *Joint Bulletin of NCC & IIS, Comp. Science*. 2008. Vol. 28. P. 43–54.

69. *Bashkin V. A.* On the single-periodic representation of reachability in one-counter nets // Concurrency, Specification and Programming. Proc. of CS&P 2009 / Ed. by L. Czaja, M. Szczuka. Warsaw University, Warsaw, 2009. P. 60–71.

70. *Bashkin V. A.* Formalization of semantics of systems with unreliable agents by means of nets of active resources // *Programming and Computer Software*. 2010. Vol. 36, no. 4. P. 187–196.

71. *Bashkin V. A.* Approximating bisimulation in one-counter nets // Proc. of "Program semantics, specification and verification: theory and applications". St.Petersburg, 2011. P. 10–17.

72. *Bashkin V. A.* On the modularity in Petri Nets of Active Resources // Proc. Of Petri Nets Compositions (Petri Nets 2011). Newcastle-upon-Tyne. Vol. 726 of *CEUR*. 2011. P. 33–47.

73. *Bashkin V. A.* Modular nets of active resources // *Automatic Control and Computer Sciences*. 2012. Vol. 46, no. 1. P. 1–11.

74. *Bashkin V. A.* On the Hereditary Properties of Modular Nets // Proc. of "Program semantics, specification and verification: theory and applications". N.Novgorod: 2012. P. 24–31.

75. *Bashkin V. A.* One-counter Circuits // Concurrency, Specification and Programming. Proc. of CS&P'2012. Vol.1. Humboldt-Universitat zu Berlin, Berlin, 2012. P. 25–36.

76. *Bashkin V. A.* One-dimensional Resource Workflow Nets // Proc. of "Program semantics, specification and verification: theory and applications". Yekaterinburg: 2013. P. 11–20.

77. *Bashkin V. A., Lomazova I. A.* Reduction of Coloured Petri nets based on resource bisimulation // *Joint Bulletin of NCC & IIS, Comp. Science*. 2000. Vol. 13. P. 12–17.

78. *Bashkin V. A., Lomazova I. A.* Resource bisimulations in Nested Petri Nets // Concurrency, Specification and Programming. Proc. of CS&P'2002. Vol.1. Informatik-Bericht 161. Humboldt-Universitat zu Berlin, Berlin, 2002. P. 39–52.

79. *Bashkin V. A., Lomazova I. A.* Petri nets and resource bisimulation // **Fundamenta Informaticae**. 2003. Vol. 55, no. 2. P. 101–114.

80. *Bashkin V. A., Lomazova I. A.* Resource Similarities in Petri Net Models of Distributed Systems // Parallel Computing Technologies / Ed. by V. E. Malyshkin. Springer Berlin Heidelberg, 2003. Vol. 2763 of **Lecture Notes in Computer Science**. P. 35–48.

81. *Bashkin V. A., Lomazova I. A.* Similarity of Generalized Resources in Petri

Nets // Parallel Computing Technologies / Ed. by V. Malyshkin. Springer Berlin Heidelberg, 2005. Vol. 3606 of **Lecture Notes in Computer Science**. P. 27–41.

82. *Bashkin V. A., Lomazova I. A.* Resource equivalence in workflow nets // Concurrency, Specification and Programming. Proc. of CS&P'2006. Vol.1. Humboldt-Universitat zu Berlin, Berlin, 2006. P. 80–91.

83. *Bashkin V. A., Lomazova I. A.* Resource Driven Automata Nets // Concurrency, Specification and Programming. Proc. of (CS&P 2010). Berlin, Humboldt-Universitat zu Berlin, 2010. Vol. 1. P. 37–48.

84. *Bashkin V. A., Lomazova I. A.* Cellular Resource-Driven Automata // Concurrency, Specification and Programming. Proc. of CS&P 2011 / Ed. by L. Czaja, M. Szczuka, A. Skowron, M. Kacprzak. Poland, Bialystok University of Technology, 2011. P. 29–41.

85. *Bashkin V. A., Lomazova I. A.* Modelling multiagent systems with the help of generalized nets of active resources // *Cybernetics and System Analyses*. 2011. Vol. 47, no. 2. P. 202–209.

86. *Bashkin V. A., Lomazova I. A.* Resource Driven Automata Nets // **Fundamenta Informaticae**. 2011. Vol. 109, no. 3. P. 223–236.

87. *Bashkin V. A., Lomazova I. A.* Cellular Resource Driven Automata Nets // **Fundamenta Informaticae**. 2012. Vol. 120, no. 3–4. P. 245–259.

88. *Bashkin V. A., Lomazova I. A.* Soundness of Workflow Nets with an Unbounded Resource is Decidable // Joint Proc. of Petri Nets and Software Engineering (PNSE'13) and Modeling and Business Environments (ModBE'13). Milano. Vol. 989 of *CEUR*. 2013. P. 61–75.

89. *Bashkin V. A., Lomazova I. A., Novikova Y. A.* Timed Resource Driven Automata Nets for Distributed Real-Time Systems Modelling // Parallel Computing Tech-

nologies / Ed. by V. E. Malyshkin. Springer Berlin Heidelberg, 2013. Vol. 7979 of **Lecture Notes in Computer Science**. P. 13–25.

90. *Bednarczyk M. A., Bernardinello L., Pawlowski W., Pomello L.* Modelling Mobility with Petri Hypernets // Recent Trends in Algebraic Development Techniques / Ed. by J. L. Fiadeiro, P. D. Mosses, F. Orejas. Springer Berlin Heidelberg, 2005. Vol. 3423 of *Lecture Notes in Computer Science*. P. 28–44.

91. *Bernardinello L., Bonzanni N., Mascheroni M., Pomello L.* Modeling Symport/Antiport P Systems with a Class of Hierarchical Petri Nets // Membrane Computing / Ed. by G. Eleftherakis, P. Kefalas, G. Păun et al. Springer Berlin Heidelberg, 2007. Vol. 4860 of *Lecture Notes in Computer Science*. P. 124–137.

92. *Berthelot G.* Transformations and decompositions of nets // Petri Nets: Central Models and Their Properties / Ed. by W. Brauer, W. Reisig, G. Rozenberg. Springer Berlin Heidelberg, 1987. Vol. 254 of *Lecture Notes in Computer Science*. P. 359–376.

93. *Best E., Devillers R., Koutny M.* Petri Net Algebra. Monographs in Theoretical Computer Science. An EATCS Series. Springer Berlin Heidelberg, 2001.

94. *Best E., Fraczak W., Hopkins R. P. et al.* M-nets: An algebra of high-level Petri nets, with an application to the semantics of concurrent programming languages // *Acta Informatica*. 1998. Vol. 35, no. 10. P. 813–857.

95. *Boigelot B., Wolper P.* Symbolic verification with periodic sets // Computer Aided Verification / Ed. by D. L. Dill. Springer Berlin Heidelberg, 1994. Vol. 818 of *Lecture Notes in Computer Science*. P. 55–67.

96. *Bolognesi T., Lucidi F., Trigila S.* From Timed Petri Nets to Timed LOTOS // Proceedings of the IFIP WG 6.1 Tenth International Symposium on Protocol Specification, Testing and Verification (Ottawa 1990) / Ed. by L. Logrippo, R. Probert, H. Ural. North-Holland, Amsterdam, 1990. P. 1–14.

97. *Borowiecki M., Broere I., Frick M. et al.* A survey of hereditary properties of graphs // *Discussiones Mathematicae Graph Theory*. 1997. Vol. 17, no. 1. P. 5–50.

98. *Bouajjani A., Habermehl P.* Symbolic reachability analysis of FIFO-channel systems with nonregular sets of configurations // *Theoretical Computer Science*. 1999. Vol. 221, no. 1–2. P. 211 – 250.

99. *Brand D., Zafiropulo P.* On Communicating Finite-State Machines // *Journal of ACM*. 1983. Vol. 30, no. 2. P. 323–342.

100. *Brauer A.* On a Problem of Partitions // *American Journal of Mathematics*. 1942. Vol. 64, no. 1. P. 299–312.

101. *Bultan T., Gerber R., Pugh W.* Symbolic model checking of infinite state systems using Presburger arithmetic // Computer Aided Verification / Ed. by O. Grumberg. Springer Berlin Heidelberg, 1997. Vol. 1254 of *Lecture Notes in Computer Science*. P. 400–411.

102. *Chang L., He X., Lian J., Shatz S.* Applying a Nested Petri Net Modeling Paradigm to Coordination of Sensor Networks with Mobile Agents // Proc. of Workshop on Petri Nets and Distributed Systems 2008. Xian, 2008. P. 132–145.

103. *Cherubini A., Citrini C., Crespi Reghizzi S., Mandrioli D.* QRT FIFO automata, breadth-first grammars and their relations // *Theoretical Computer Science*. 1991. Vol. 85, no. 1. P. 171 – 203.

104. *Christensen S.* Decidability and decomposition in process algebras: Phd thesis / Department of Computer Science, University of Edinburgh. 1993.

105. *Christensen S., Hirshfeld Y., Moller F.* Bisimulation equivalence is decidable for basic parallel processes // CONCUR'93 / Ed. by E. Best. Springer Berlin Heidelberg, 1993. Vol. 715 of *Lecture Notes in Computer Science*. P. 143–157.

106. *Christensen S., Hirshfeld Y., Moller F.* Decomposability, decidability and axiomatisability for bisimulation equivalence on basic parallel processes // Logic in Computer Science, 1993. LICS '93., Proceedings of Eighth Annual IEEE Symposium on Logic in Computer Science. 1993. P. 386–396.

107. *Christensen S., Hüttel H., Stirling C.* Bisimulation equivalence is decidable for all context-free processes // CONCUR'92 / Ed. by W. R. Cleaveland. Springer Berlin Heidelberg, 1992. Vol. 630 of *Lecture Notes in Computer Science*. P. 138–147.

108. *Christensen S., Petrucci L.* Modular Analysis of Petri Nets // *The Computer Journal*. 2000. Vol. 43, no. 3. P. 224–242.

109. *Chrobak M.* Finite automata and unary languages // *Theoretical Computer Science*. 1986. Vol. 47, no. 2. P. 149–158.

110. *Chrząstowski-Wachtel P.* Sound Markings in Structured Nets // Concurrency, Specification and Programming. Proc. of (CS&P 2005) / Ed. by L. Czaja. Warsaw University, Warsaw, 2005. P. 71–85.

111. *Comon H., Jurski Y.* Multiple counters automata, safety analysis and Presburger arithmetic // Computer Aided Verification / Ed. by A. J. Hu, M. Y. Vardi. Springer Berlin Heidelberg, 1998. Vol. 1427 of *Lecture Notes in Computer Science*. P. 268–279.

112. *Cook M.* Universality in Elementary Cellular Automata // *Complex Systems*. 2004. Vol. 15, no. 1. P. 1–40.

113. CPN Tools Reference Manual. URL: http://cpntools.org/.

114. *Dickson L. E.* Finiteness of the Odd Perfect and Primitive Abundant Numbers with n Distinct Prime Factors // *American Journal of Mathematics*. 1913. Vol. 35, no. 4. P. 413–422.

115. *Dufourd C., Finkel A., Schnoebelen P.* Reset nets between decidability and undecidability // Automata, Languages and Programming / Ed. by K. G. Larsen, S. Skyum, G. Winskel. Springer Berlin Heidelberg, 1998. Vol. 1443 of *Lecture Notes in Computer Science*. P. 103–115.

116. *Dworzanski L. W., Lomazova I. A.* On Compositionality of Boundedness and Liveness for Nested Petri Nets // *Fundamenta Informaticae*. 2012. Vol. 120, no. 3–4. P. 243–257.

117. *Erdös P., Graham R. L.* On a linear diophantine problem of Frobenius // *Acta Arithmetica*. 1972. Vol. 21. P. 399–408.

118. *Esparza J.* Decidability of model checking for infinite-state concurrent systems // *Acta Informatica*. 1997. Vol. 34, no. 2. P. 85–107.

119. *Esparza J.* Petri nets, commutative context-free grammars, and basic parallel processes // *Fundamenta Informaticae*. 1997. Vol. 31. P. 13–26.

120. *Esparza J.* Decidability and complexity of Petri net problems — An introduction // Lectures on Petri Nets I: Basic Models / Ed. by W. Reisig, G. Rozenberg. Springer Berlin Heidelberg, 1998. Vol. 1491 of *Lecture Notes in Computer Science*. P. 374–428.

121. *Esparza J., Nielsen M.* Decidability Issues for Petri Nets — a Survey // *Bulletin of the European Association for Theoretical Computer Science*. 1994. Vol. 52. P. 245–262.

122. *Esser R.* An Object Oriented Petri Net Approach to Embedded System Design: Phd thesis / Zurich: Swiss Federal Institute of Technology. 1996.

123. *Farwer B., Lomazova I.* A Systematic Approach towards Object-Based Petri Net Formalisms // Perspectives of System Informatics / Ed. by D. Bjørner, M. Broy,

A. V. Zamulin. Springer Berlin Heidelberg, 2001. Vol. 2244 of *Lecture Notes in Computer Science*. P. 255–267.

124. *Finkel A., Leroux J.* How to compose Presburger-accelerations: applications to broadcast protocols // FST TCS 2002: Foundations of Software Technology and Theoretical Computer Science / Ed. by M. Agrawal, A. Seth. Springer Berlin Heidelberg, 2002. Vol. 2556 of *Lecture Notes in Computer Science*. P. 145–156.

125. *Finkel A., Sutre G.* Decidability of reachability problems for classes of two counters automata // STACS 2000 / Ed. by H. Reichel, S. Tison. Springer Berlin Heidelberg, 2000. Vol. 1770 of *Lecture Notes in Computer Science*. P. 346–357.

126. *Fribourg L., Olsén H.* A decompositional approach for computing least fixed-points of Datalog programs with Z-counters // *Constraints*. 1997. Vol. 2, no. 3-4. P. 305–335.

127. *Genrich H. J., Lautenbach K.* System modelling with high-level Petri nets // *Theoretical Computer Science*. 1981. Vol. 13, no. 1. P. 109–135.

128. *Ginsburg S.* The mathematical theory of context-free languages. New York, NY, USA: McGraw-Hill, Inc., 1966.

129. *Ginsburg S., Spanier E. H.* Semigroups, Presburger formulas and languages // *Pacific Journal of Mathematics*. 1966. Vol. 16. P. 285–296.

130. *Goldin D., Keil D.* Indirect Interaction in Environments for Multi-agent Systems // Environments for Multi-Agent Systems II / Ed. by D. Weyns, H. Dyke Parunak, F. Michel. Springer Berlin Heidelberg, 2006. Vol. 3830 of *Lecture Notes in Computer Science*. P. 68–87.

131. *Göller S., Haase C., Ouaknine J., Worrell J.* Model checking succinct and parametric one-counter automata // Automata, Languages and Programming / Ed. by

S. Abramsky, C. Gavoille, C. Kirchner et al. Springer Berlin Heidelberg, 2010. Vol. 6199 of *Lecture Notes in Computer Science*. P. 575–586.

132. *Göller S., Mayr R., To A. W.* On the computational complexity of verifying one-counter processes // Proceedings of the 2009 24th Annual IEEE Symposium on Logic In Computer Science. LICS '09. Washington, DC, USA: IEEE Computer Society, 2009. P. 235–244.

133. *Hack M.* Petri net languages: Computation Structures Group Memo 124: MIT, 1975.

134. *Haddad S., Poitrenaud D.* Theoretical Aspects of Recursive Petri Nets // Application and Theory of Petri Nets 1999 / Ed. by S. Donatelli, J. Kleijn. Springer Berlin Heidelberg, 1999. Vol. 1639 of *Lecture Notes in Computer Science*. P. 228–247.

135. *Hirshfeld Y.* Congruences in commutative semigroups: Research Report EC-S-LFCS-94-291: Edinburgh, University of Edinburgh, Department of Computer Science, 1994.

136. *Hirshfeld Y., Jerrum M.* Bisimulation Equivalence Is Decidable for Normed Process Algebra (Extended abstract) // Automata, Languages and Programming / Ed. by J. Wiedermann, P. van Emde Boas, M. Nielsen. Springer Berlin Heidelberg, 1999. Vol. 1644 of *Lecture Notes in Computer Science*. P. 412–421.

137. *Hirshfeld Y., Jerrum M., Moller F.* A Polynomial-Time Algorithm for Deciding Bisimulation Equivalence of Normed Basic Parallel Processes // *Mathematical Structures in Computer Science*. 1996. Vol. 6, no. 3. P. 251–259.

138. *Hopcroft J. E., Pansiot J.-J.* On the Reachability Problem for 5-Dimensional Vector Addition Systems. // *Theoretical Computer Science*. 1979. Vol. 8. P. 135–159.

139. *Hüttel H.* Decidability, Behavioural Equivalences and Infinite Transition Graphs: Phd thesis / Edinburgh: University of Edinburgh. 1991.

140. *Jančar P.* Decidability questions for bisimilarity of Petri nets and some related problems // STACS 94 / Ed. by P. Enjalbert, E. W. Mayr, K. W. Wagner. Springer Berlin Heidelberg, 1994. Vol. 775 of *Lecture Notes in Computer Science*. P. 581–592.

141. *Jančar P.* Bisimulation equivalence is decidable for one-counter processes // Automata, Languages and Programming / Ed. by P. Degano, R. Gorrieri, A. Marchetti-Spaccamela. Springer Berlin Heidelberg, 1997. Vol. 1256 of *Lecture Notes in Computer Science*. P. 549–559.

142. *Jančar P., Kučera A., Moller F.* Simulation and bisimulation over one-counter processes // STACS 2000 / Ed. by H. Reichel, S. Tison. Springer Berlin Heidelberg, 2000. Vol. 1770 of *Lecture Notes in Computer Science*. P. 334–345.

143. *Jančar P., Kučera A., Moller F., Sawa Z.* DP lower bounds for equivalence-checking and model-checking of one-counter automata // *Information and Computation*. 2004. Vol. 188, no. 1. P. 1–19.

144. *Jančar P., Moller F.* Checking regular properties of Petri nets // CONCUR '95: Concurrency Theory / Ed. by I. Lee, S. A. Smolka. Springer Berlin Heidelberg, 1995. Vol. 962 of *Lecture Notes in Computer Science*. P. 348–362.

145. *Jančar P., Moller F.* Techniques for Decidability and Undecidability of Bisimilarity // CONCUR 99 Concurrency Theory / Ed. by J. C. Baeten, S. Mauw. Springer Berlin Heidelberg, 1999. Vol. 1664 of *Lecture Notes in Computer Science*. P. 30–45.

146. *Jantzen M.* Language theory of Petri nets // Petri Nets: Central Models and Their Properties / Ed. by W. Brauer, W. Reisig, G. Rozenberg. Springer Berlin Heidelberg, 1987. Vol. 254 of *Lecture Notes in Computer Science*. P. 397–412.

147. *Jensen K.* Coloured Petri Nets and the invariant-method // *Theoretical Computer Science*. 1981. Vol. 14, no. 3. P. 317 – 336.

148. *Jensen K.* Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use. Monographs in Theoretical Computer Science. An EATCS Series. Springer Berlin Heidelberg, 1996.

149. *Kindler E.* A compositional partial order semantics for Petri net components // Application and Theory of Petri Nets 1997 / Ed. by P. Azéma, G. Balbo. Springer Berlin Heidelberg, 1997. Vol. 1248 of *Lecture Notes in Computer Science*. P. 235–252.

150. *Klai K., Haddad S., Ilié J.-M.* Modular Verification of Petri Nets Properties: A Structure-Based Approach // Formal Techniques for Networked and Distributed Systems - FORTE 2005 / Ed. by F. Wang. Springer Berlin Heidelberg, 2005. Vol. 3731 of *Lecture Notes in Computer Science*. P. 189–203.

151. *Köhler M., Heitman F.* On the Expressiveness of Communication Channels for Object Nets // *Fundamenta Informaticae*. 2009. Vol. 93. P. 205–219.

152. *Köhler M., Rölke H.* Super-Dual Nets // Concurrency, Specification and Programming. Proc. of (CS&P 2005) / Ed. by L. Czaja. Warsaw University, Warsaw, 2005. P. 271–280.

153. *Köhler M., Rölke H.* Properties of Super-Dual Nets // *Fundamenta Informaticae*. 2006. Vol. 72. P. 245–254.

154. *Köhler M., Rölke H.* Web Service Orchestration with Super-Dual Object Nets // Petri Nets and Other Models of Concurrency ICATPN 2007 / Ed. by J. Kleijn, A. Yakovlev. Springer Berlin Heidelberg, 2007. Vol. 4546 of *Lecture Notes in Computer Science*. P. 263–280.

155. *Köhler-Bussmeier M.* Hornets: Nets within Nets Combined with Net Algebra //

Applications and Theory of Petri Nets / Ed. by G. Franceschinis, K. Wolf. Springer Berlin Heidelberg, 2009. Vol. 5606 of *Lecture Notes in Computer Science*. P. 243–262.

156. *Kučera A.* Efficient verification algorithms for one-counter processes // Automata, Languages and Programming / Ed. by U. Montanari, J. D. Rolim, E. Welzl. Springer Berlin Heidelberg, 2000. Vol. 1853 of *Lecture Notes in Computer Science*. P. 317–328.

157. *Kuzmin E. V., Chalyy D. J.* On the Reachability Set of Automaton Counter Machines // *Automatic Control and Computer Sciences*. 2011. Vol. 45, no. 7. P. 444–451.

158. *Lautenbach K.* Duality of Marked Place/Transition Nets: Research Report 18: Universitat Koblenz-Landau, Institut fur Informatik, 2003.

159. *Leroux J.* Vector addition system reachability problem: a short self contained proof // Proceedings of the 5th international conference on Language and automata theory and applications. LATA'11. Berlin, Heidelberg: Springer-Verlag, 2011. P. 41–64.

160. *Leroux J., Sutre G.* On flatness for 2-dimensional vector addition systems with states // CONCUR 2004 - Concurrency Theory / Ed. by P. Gardner, N. Yoshida. Springer Berlin Heidelberg, 2004. Vol. 3170 of *Lecture Notes in Computer Science*. P. 402–416.

161. *Leroux J., Sutre G.* Flat counter automata almost everywhere! // Automated Technology for Verification and Analysis / Ed. by D. Peled, Y.-K. Tsay. Springer Berlin Heidelberg, 2005. Vol. 3707 of *Lecture Notes in Computer Science*. P. 489–503.

162. *Liubicz U. I.* Bounds for the optimal determinization of nondeterministic autonomic automata // *Sibirskii Matemat. Journal*. 1964. Vol. 2. P. 337–355.

163. *Lomazova I. A.* Nested Petri nets — a Formalism for Specification and Verification of Multi-Agent Distributed Systems // *Fundamenta Informaticae*. 2000. Vol. 43. P. 195–214.

164. *Lomazova I. A.* Nested Petri Nets: Multi-level and Recursive Systems // *Fundamenta Informaticae*. 2001. Vol. 47, no. 3-4. P. 283–293.

165. *Lomazova I. A.* Communities of Interacting Automata for Modelling Distributed Systems with Dynamic Structure // *Fundamenta Informaticae*. 2004. Vol. 60, no. 1-4. P. 225–235.

166. *Lomazova I. A.* Nested Petri Nets for Adaptive Process Modeling // Pillars of Computer Science: Essays Dedicated to Boris (Boaz) Trakhtenbrot on the Occasion of His 85th Birthday / Ed. by A. Avron, N. Dershowitz, A. Rabinovich. Springer Berlin Heidelberg, 2008. Vol. 4800 of *Lecture Notes in Computer Science*. P. 460–474.

167. *Mascheroni M.* Generalized Hypernets and Their Semantics // Proceedings of the Fith International Workshop on Modelling of Objects, Components and Agents, MOCA'09, Bericht 290, 2009. 2009.

168. *Mayr E. W.* An algorithm for the general Petri net reachability problem // *Siam Journal on Computing*. 1984. Vol. 13, no. 3. P. 441–460.

169. *Mayr R.* Decidability of model checking with the temporal logic EF // *Theoretical Computer Science*. 2001. Vol. 256, no. 1-2. P. 31 – 62.

170. *Merlin P. M.* A Study of the Recoverability of Computing Systems: Phd thesis / University of California, Irvine, CA, USA. 1974.

171. *Milner R.* A Calculus of Communicating Systems. Springer Berlin Heidelberg, 1980. Vol. 92 of *Lecture Notes in Computer Science*.

172. *Minsky M.* Computation: Finite and Infinite Machines. Prentice Hall, 1967.

173. *Moller F.* Infinite results // CONCUR'96: Concurrency Theory / Ed. by U. Montanari, V. Sassone. Springer Berlin Heidelberg, 1996. Vol. 1119 of *Lecture Notes in Computer Science*. P. 195–216.

174. *Moore C., Lakdawala P.* Queues, stacks, and transcendentality at the transition to chaos // *Physica D: Nonlinear Phenomena*. 2000. Vol. 135, no. 1-2. P. 24–40.

175. *Nehaniv C. L.* Asynchronous Automata Networks Can Emulate any Synchronous Automata Network. // *International Journal of Algebra and Computation*. 2004. Vol. 14, no. 5-6. P. 719–739.

176. *Park D.* Concurrency and automata on infinite sequences // Theoretical Computer Science / Ed. by P. Deussen. Springer Berlin Heidelberg, 1981. Vol. 104 of *Lecture Notes in Computer Science*. P. 167–183.

177. *Patil S. S.* Limitations and Capabilities of Dijkstra's Semaphore Primitives for Coordination among Processes: CSG Memo 57: MIT, Project MAC, 1971.

178. *Pawlowski W.* Petri Hypernets with Constraints // Concurrency, Specification and Programming. Proc. of (CS&P 2009) / Ed. by L. Czaja, M. Szczuka. Warsaw University, Warsaw, 2009. P. 467–479.

179. *Petri C.* "Forgotten topics" of Net Theory // Petri Nets: Applications and Relationships to Other Models of Concurrency / Ed. by W. Brauer, W. Reisig, G. Rozenberg. Springer Berlin Heidelberg, 1987. Vol. 255 of *Lecture Notes in Computer Science*. P. 499–514.

180. *Petri C. A.* Kommunikation mit Automaten: Phd thesis / Bonn: Institute für Instrumentelle Mathematik. 1962.

181. *Podlovchenko R. I., Rusakov D. M., Zakharov V. A.* On the Equivalence Problem for Programs with Mode Switching // Implementation and Application of

Automata / Ed. by J. Farré, I. Litovsky, S. Schmitz. Springer Berlin Heidelberg, 2006. Vol. 3845 of *Lecture Notes in Computer Science*. P. 351–352.

182. *Presburger M.* Über die Vollständigkeit eines gewissen Systems der Arithmetik ganzer Zahlen, in welchem die Addition als einzige Operation hervortritt // Comptes Rendus du I congrès de Mathématiciens des Pays Slaves. Warszawa: 1929. P. 92–101.

183. *Ramchandani C.* Analysis of asynchronous concurrent systems by timed Petri nets: Research report: MIT, 1974.

184. *Ramirez Alfonsin J. L.* The Diophantine Frobenius Problem. Oxford University Press, 2006.

185. *Redei L.* The theory of finitely generated commutative semigroups. Oxford University Press, 1965.

186. *Reisig W.* Petri Nets. Springer Berlin Heidelberg, 1985. Vol. 4 of *Monographs in Theoretical Computer Science. An EATCS Series*.

187. *Reisig W.* Petri nets and algebraic specifications // *Theoretical Computer Science*. 1991. Vol. 80, no. 1. P. 1–34.

188. *Reisig W.* Petri Net models of distributed algorithms // Computer Science Today / Ed. by J. Leeuwen. Springer Berlin Heidelberg, 1995. Vol. 1000 of *Lecture Notes in Computer Science*. P. 441–454.

189. *Reutenauer C.* The mathematics of Petri nets. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1990.

190. *Schnoebelen P., Sidorova N.* Bisimulation and the Reduction of Petri Nets // Application and Theory of Petri Nets 2000 / Ed. by M. Nielsen, D. Simpson. Springer Berlin Heidelberg, 2000. Vol. 1825 of *Lecture Notes in Computer Science*. P. 409–423.

191. *Serre O.* Parity games played on transition graphs of one-counter processes // Foundations of Software Science and Computation Structures / Ed. by L. Aceto, A. Ingólfsdóttir. Springer Berlin Heidelberg, 2006. Vol. 3921 of *Lecture Notes in Computer Science*. P. 337–351.

192. *Sidorova N., Stahl C.* Soundness for Resource-Contrained Workflow Nets is Decidable: Research Report BPM-12-09: BPM Center, 2012.

193. *Smith E.* Principles of high-level net theory // Lectures on Petri Nets I: Basic Models / Ed. by W. Reisig, G. Rozenberg. Springer Berlin Heidelberg, 1998. Vol. 1491 of *Lecture Notes in Computer Science*. P. 174–210.

194. *Stirling C.* Decidability of bisimulation equivalence for normed pushdown processes // *Theoretical Computer Science*. 1998. Vol. 195, no. 2. P. 113 – 131.

195. *Sylvester J. J.* Question 7382 // *Mathematical Questions with their Solutions, Educational Times*. 1884. Vol. 41. P. 21.

196. *Tarasyuk I. V.* τ-Equivalences and Refinement for Petri Net Based Design: Research Report FI00-11: Dresden: Technische Universität zu Dresden, 2000.

197. *Tarjan R.* Depth-First Search and Linear Graph Algorithms // *SIAM Journal on Computing*. 1972. Vol. 1, no. 2. P. 146–160.

198. *Tiplea F. L., Marinescu D. C.* Structural soundness of workflow nets is decidable // *Information Processing Letters*. 2005. Vol. 96, no. 2. P. 54–58.

199. *Valiant L. G., Paterson M. S.* Deterministic one-counter automata // *Journal of Computer and System Sciences*. 1975. Vol. 10, no. 3. P. 340–350.

200. *Valk R.* Petri Nets as Token Objects: An Introduction to Elementary Object Nets // Application and Theory of Petri Nets 1998 / Ed. by J. Desel, M. Silva. Springer Berlin Heidelberg, 1998. Vol. 1420 of *Lecture Notes in Computer Science*. P. 1–24.

201. *Valk R.* Object Petri Nets // Lectures on Concurrency and Petri Nets / Ed. by J. Desel, W. Reisig, G. Rozenberg. Springer Berlin Heidelberg, 2004. Vol. 3098 of *Lecture Notes in Computer Science*. P. 819–848.

202. *van der Aalst W. M. P.* The Application of Petri Nets to Workflow Management. // *Journal of Circuits, Systems, and Computers*. 1998. Vol. 8, no. 1. P. 21–66.

203. *van der Aalst W. M. P., van Hee K. M.* Workflow Management: Models, Methods and Systems. MIT Press, 2002.

204. *van der Aalst W. M. P., van Hee K. M., ter Hofstede A. H. M. et al.* Soundness of workflow nets: classification, decidability, and analysis // *Formal Aspects of Computing*. 2011. Vol. 23, no. 3. P. 333–363.

205. *van Hee K., Oanea O., Serebrenik A. et al.* Checking Properties of Adaptive Workflow Nets // *Fundamenta Informaticae*. 2007. Vol. 79, no. 3-4. P. 347–362.

206. *van Hee K., Serebrenik A., Sidorova N., Voorhoeve M.* Soundness of Resource-Constrained Workflow Nets // Applications and Theory of Petri Nets 2005 / Ed. by G. Ciardo, P. Darondeau. Springer Berlin Heidelberg, 2005. Vol. 3536 of *Lecture Notes in Computer Science*. P. 250–267.

207. *van Hee K., Sidorova N., Voorhoeve M.* Generalised Soundness of Workflow Nets Is Decidable // Applications and Theory of Petri Nets 2004 / Ed. by J. Cortadella, W. Reisig. Springer Berlin Heidelberg, 2004. Vol. 3099 of *Lecture Notes in Computer Science*. P. 197–215.

208. *Wolfram S.* Universality and complexity in cellular automata // *Physica D: Nonlinear Phenomena*. 1984. Vol. 10, no. 1-2. P. 1–35.

209. *Zakharov V. A.* The Equivalence Problem for Computational Models: Decidable and Undecidable Cases // Machines, Computations, and Universality / Ed. by

M. Margenstern, Y. Rogozhin. Springer Berlin Heidelberg, 2001. Vol. 2055 of *Lecture Notes in Computer Science*. P. 133–152.