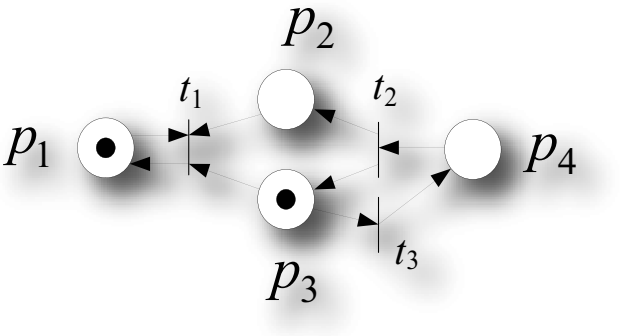
Мараховский В. Б., Розенблюм Л. Я., Яковлев А. В.

Моделирование параллельных процессов
Сети Петри



ПРОФЕССИОНАЛЬНАЯ ЛИТЕРАТУРА
Санкт-Петербург
2014

Рецензенты:

Доктор технических наук, профессор, заведующий кафедрой «Технологии программировании» Санкт-Петербургского национального исследовательского университета информационных технологий, механики и оптики А .А. Шалыто.

Доктор технических наук, профессор Санкт-Петербургского государственного электротехнического университета А. И. Водяхо.

Мараховский В. Б., Розенблюм Л. Я., Яковлев А. В.
Моделирование параллельных процессов. Сети Петри. — СПб.: Профессиональная литература, 2014. – 400 с.: ил.

Серия «Избранное Computer Science»

Данная книга представляет собой уникальное издание, написанное специалистами с мировыми именами в области моделирования параллельных процессов и сетей Петри. В книге излагаются вопросы построения формальных динамических моделей асинхронных параллельных процессов. При этом рассматриваемые процессы могут относиться к различным прикладным областям, например, к вычислениям, управлению, интерфейсам, программированию, робототехнике или искусственному интеллекту.

Подчеркивается неразрывная связь структурной модели, отражающей статические свойства объекта, и динамической (поведенческой) модели. Такой двуединый фундаментальный подход пригоден на всех этапах проектирования систем - спецификации, анализа, реализации и верификации.

Книга написана доступным языком, с большим количеством примеров, отличается хорошей организацией, структурой и подачей материала. Может использоваться в качестве учебного пособия. Рассчитана на широкую аудиторию.

Рекомендовано Учебно-методическим объединением по университетскому политехническому образованию в качестве учебного пособия для студентов высших учебных заведений, обучающихся по направлению подготовки «Системный анализ и управление».

© Мараховский В. Б., Розенблюм Л. Я., Яковлев А. В., 2014
© Профессиональная литература, 2014

Содержание

ОТ АВТОРОВ **11**

ВВЕДЕНИЕ **12**

СОКРАЩЕНИЯ, ПРИНЯТЫЕ В ТЕКСТЕ **15**

ГЛАВА 1. АСИНХРОННЫЙ ПРОЦЕСС **17**

1.1. ОПРЕДЕЛЕНИЕ АСИНХРОННОГО ПРОЦЕССА **19**

Понятие асинхронности 19

Концепция асинхронного процесса 21

Спецификация асинхронного процесса 22

Протокол асинхронного процесса 25

1.2. КЛАССИФИКАЦИЯ **26**

Автономный асинхронный процесс 26

Эффективный асинхронный процесс 27

Управляемый асинхронный процесс 29

Простой асинхронный процесс 30

1.3. РЕПОЗИЦИЯ **30**

Композиция асинхронного процесса и его репозиции 32

Приведенный асинхронный процесс 32

Конвейерный асинхронный процесс 33

1.4. СТРУКТУРИРОВАНИЕ **34**

1.5. РЕДУКЦИЯ **39**

Редукция по инициатору 39

Редукция по результанту 41

Редукция по входной компоненте ситуации 41

Редукция по выходной или внутренней компоненте 42

1.6. КОМПОЗИЦИЯ **46**

Параллельная композиция 46

Последовательная композиция 49

Замыкание 52

1.7. КОММЕНТАРИЙ. НЕДОСТАТКИ МОДЕЛИ АСИНХРОННОГО ПРОЦЕССА **55**

1.8. УПРАЖНЕНИЯ **58**

ГЛАВА 2. СЕТИ ПЕТРИ 63

2.1. РЕТРОСПЕКТИВА СЕТЕЙ ПЕТРИ 64

2.2. ОПИСАНИЕ МОДЕЛИ 70

2.3. ДИАГРАММА МАРКИРОВОК 76

Отношения между событиями 78

Сети Петри как модельная интерпретация асинхронных процессов 80

Два начала сетей Петри: статическое и динамическое 81

2.4. КЛАССИФИКАЦИЯ СЕТЕЙ ПЕТРИ 82

Зачем нужна классификация сетей Петри 82

Классификация по динамическим ограничениям 84

Классификация по статическим ограничениям 86

2.5. ЗАДАЧИ И АЛГОРИТМЫ АНАЛИЗА 88

2.5.1. Первая группа методов анализа 89

Задача достижимости 90

Модификации задачи достижимости 90

Алгоритм порождения диаграммы маркировок 92

Свойства диаграмм маркировок 95

Сводимость задач достижимости 95

Задача живости Сети Петри 96

2.5.2. Вторая группа методов анализа 97

2.5.3. Третья группа методов анализа 98

Дедлоки 98

Ловушки 99

2.5.4. Оценки вычислительной сложности проблем анализа сетей Петри 100

2.6. АНАЛИЗ: МАТРИЧНЫЙ ПОДХОД 101

Анализ на свойства структурной ограниченности и живости 101

Инвариантные и согласованные сети Петри 103

Решение задачи достижимости 110

2.7. ФУНКЦИОНАЛЬНЫЕ ВОЗМОЖНОСТИ 112

Сведения из теории формальных грамматик и языков 112

Генерирование формальных языков сетями Петри 116

2.8. ПРАКТИЧЕСКОЕ ИСПОЛЬЗОВАНИЕ СЕТЕЙ ПЕТРИ. ИНТЕРПРЕТАЦИЯ СЕТЕЙ ПЕТРИ 121

Пример спецификации конечного автомата сетью Петри 121
Использование сетей Петри в параллельном программировании 123
Пример моделирования аппаратных средств 125
Моделирование конвейерных процессов 126

2.9. УПРАЖНЕНИЯ 128

ГЛАВА 3. ДРУГИЕ СЕТЕВЫЕ МОДЕЛИ, РОДСТВЕННЫЕ СЕТЯМ ПЕТРИ 133

3.1. СЕТИ СО ВЗВЕШЕННЫМИ ДУГАМИ 136
Преобразование сети с взвешенными дугами в ординарную сеть Петри 137
Пример спецификации химической реакции сетью с взвешенными дугами 138
Проверка отсутствия маркера в некоторой позиции 139

3.2. РАСКРАШЕННЫЕ СЕТИ 140

3.3. СЕТИ С ТОРМОЗЯЩИМИ ДУГАМИ (ИНГИБИТОРНЫЕ СЕТИ ПЕТРИ) 143

3.4. ПРИОРИТЕТНЫЕ И ВРЕМЕННЫЕ СЕТИ 147

3.5.СИНХРОННЫЕ И САМОМОДИФИЦИРУЕМЫЕ СЕТИ 151

3.6. СТОХАСТИЧЕСКИЕ СЕТИ ПЕТРИ 155

3.7. ОЦЕНОЧНЫЕ СЕТИ 161
Примитивы оценочных сетей 162
Мотивы появления оценочных сетей 165

3.8. СЕТИ ПОТОКОВ ДАННЫХ 168

3.9. РАЗРЕШИМОСТЬ ЗАДАЧ АНАЛИЗА 173

3.10. УПРАЖНЕНИЯ 175

ГЛАВА 4. ДИНАМИЧЕСКАЯ ЛОГИКА 183

4.1. ДИАГРАММЫ ПЕРЕХОДОВ И ИХ КЛАССИФИКАЦИЯ 185
Состояния бинарной диаграммы переходов 187
Бифуркантное состояние 188
Классы диаграмм переходов 189

4.2. СХЕМЫ МАЛЛЕРА 191

Гипотезы о задержках в логических элементах и проводах ... 192
Самосинхронные схемы ... 193
Диаграммы Маллера ... 193
Понятие динамической логики ... 196
Задача построения рабочего цикла ... 196

4.3. МАРКИРОВАННЫЕ ГРАФЫ ... 199

4.4. СИГНАЛЬНЫЕ ГРАФЫ ... 203

Построение кумулятивной диаграммы по маркированному графу ... 206
Свойства кумулятивных диаграмм ... 208
Связь сигнальных графов с диаграммами переходов ... 208
Нормальность сигнального графа ... 209
Понятия одновременности и предшествования сигналов ... 211
Анализ сигнального графа на свойства регулярности и нормальности ... 212

4.5. УПРАЖНЕНИЯ ... 217

ГЛАВА 5. ПАРАЛЛЕЛЬНОЕ ПРОГРАММИРОВАНИЕ. ЗАДАЧИ КООРДИНАЦИИ И СИНХРОНИЗАЦИИ ВЗАИМОДЕЙСТВУЮЩИХ ПРОЦЕССОВ ... 219

5.1. АНОМАЛИИ ДИНАМИКИ ... 223

Поведенческий и структурный дедлоки ... 225
Эффект “каннибализма” ... 227
Эффект “голодной смерти” ... 227
Свойства “справедливости” и “голодания” ... 228

5.2. ЗАДАЧА ОБ ОБЕДАЮЩИХ ФИЛОСОФАХ ... 229

5.3. ЗАДАЧИ ВЗАИМНОГО ИСКЛЮЧЕНИЯ И СЕМАФОРЫ ... 231

Семафоры Дейкстры ... 232
Семафоры Цикритзиса и Бринч-Хансена ... 234
Семафоры Агервалы и Вентилборга ... 234

5.4. ЗАДАЧА “ПРОИЗВОДИТЕЛИ – ПОТРЕБИТЕЛИ” ... 235

Фрагмент спецификации Г. Дейтела ... 237
Программа Д. Цикритзиса и Ф. Бернстайна ... 239

5.5. ЗАДАЧА “ЧИТАТЕЛИ – ПИСАТЕЛИ” ... 240

5.6. ЗАДАЧА О КУРИЛЬЩИКАХ ... 245

5.7. МЕХАНИЗМ РАНДЕВУ....246

5.8. СЕМАНТИКА ВЗАИМОДЕЙСТВУЮЩИХ ПРОЦЕССОВ....250

Пошаговая детализация действий программ 254

Взаимные переходы между двумя формами семантик.... 255

Проверка условий справедливого решения конфликта 257

Выводы.... 258

5.9. ЗАКЛЮЧЕНИЕ....259

5.10. УПРАЖНЕНИЯ....259

ГЛАВА 6. ПРИНЦИПЫ АППАРАТНОЙ РЕАЛИЗАЦИИ 263

6.1. ОСОБЕННОСТИ САМОСИНХРОННОЙ СХЕМОТЕХНИКИ....265

6.2. СИНТЕЗ ПО МОДЕЛИ МАЛЛЕРА....268

Редукция диаграммы переходов.... 268

Кратные состояния 270

Проверка спецификации на полумодулярность.... 270

Проекции диаграмм переходов.... 271

Устранение противоречивости диаграмм переходов 272

Свойства логических переменных полумодулярных диаграмм переходов.... 273

Стандартные реализации самозависимых переменных 273

Метод совершенной реализации схемы Маллера 275

6.3. СТАНДАРТНЫЕ РЕАЛИЗАЦИИ ФРАГМЕНТОВ СЕТЕЙ ПЕТРИ....278

Реализация многовходовых С-элементов Маллера.... 281

Моделирование продвижения маркера в сети Петри элементом Давида.... 281

Прямая трансляция сетей Петри в логические схемы.... 284

Минимальный базис реализации сетей Петри методом прямой трансляции.... 287

6.4. ПРИНЦИПЫ КОМПОЗИЦИИ....288

6.4.1. Дополнительные модули 289

Модуль преобразования фазовой дисциплины 289

Модуль многократного вхождения оператора 290

Модуль управления циклом.... 290

6.4.2. Реализация неустойчивых сетей Петри.... 292

Реализация сетей Петри с вершинами свободного выбора..... 292

Явление электронного арбитража ... 292
Электронные арбитры ... 293
Ограниченные арбитры ... 295
Неограниченные арбитры ... 296
Реализация неустойчивых ограниченных сетей Петри ... 297

6.5. УПРАЖНЕНИЯ ... 298

ГЛАВА 7. ПРОТОКОЛЫ ИНФОРМАЦИОННОГО ОБМЕНА ... 301

7.1. КЛАССИФИКАЦИЯ ИНТЕРФЕЙСОВ ... 304

7.1.1. Классификация по топологии ... 304
Интерфейсы с кольцевой структурой ... 305
Интерфейсы с радиальной структурой ... 305
Интерфейсы с магистральной структурой ... 305
Интерфейсы со смешанной структурой ... 306

7.1.2. Классификация по способу передачи данных ... 306

7.1.3. Классификация по принципу реализации ... 306
Синхронные интерфейсы ... 306
Асинхронные интерфейсы ... 306
Самосинхронные интерфейсы ... 307

7.1.4. Классификация интерфейсов по назначению ... 307

7.2. ЗАДАНИЕ ПРОТОКОЛОВ ... 308

7.2.1. Протокол чтения данных в интерфейсе «Общая шина» ... 308

7.2.2. Примерная процедура проектирования протокола ... 312

7.2.3. Разработка описания протокола, его анализа и реализации. Оценка сложности аппаратной и программной поддержки протокола ... 313
Сервисный уровень протокола ... 313
Построение структурной модели интерфейсного протокола ... 314
Формализация описания интерфейсного протокола ... 315
Синтез средств сопряжения ... 315
Реализация протоколов ... 317

7.2.4. Формальная модель протокола информационного обмена физического уровня ... 317

7.2.5. Пример использования сети Петри для задания протокола обмена ... 320

7.3. ВЕРИФИКАЦИЯ ПРОТОКОЛОВ 322

7.3.1. Верификация эффектов взаимодействия 322

Выбор способа анализа протокола 322

Выбор совокупности желаемых эффектов 323

7.3.2. Выявление свойств протоколов 324

Анализ асинхронного протокола «чередующегося бита» 324

Модель с идеальным каналом 325

Модель с искажением сообщения в канале 326

Модель с потерей сообщений 327

Модель с ложным срабатыванием механизма таймаута 329

Модель без ограничений на емкость канала и число срабатываний таймаута 331

Модель с буферированным каналом 332

7.3.3. Описание протоколов сигнальными графами и диаграммами переходов 335

7.4. УПРАЖНЕНИЯ 339

ГЛАВА 8. ПРИМЕРЫ РЕШЕНИЯ ПРИКЛАДНЫХ ЗАДАЧ 343

8.1. ЛОГИЧЕСКОЕ УПРАВЛЕНИЕ 345

Первый пример — "Управление манипулятором" 346

Второй пример — "Управление движением вагонеток" 349

8.2. ПРОТОКОЛЫ АСИНХРОННЫХ СХЕМ 358

8.3 ИНФОРМАЦИОННЫЙ ОБМЕН 364

8.4. ИНТЕЛЛЕКТУАЛЬНЫЕ СИСТЕМЫ 369

Пример 1 369

Пример 2 374

8.5. УПРАЖНЕНИЯ 377

ПОСЛЕСЛОВИЕ АВТОРОВ (О В.И. ВАРШАВСКОМ И СТАНОВЛЕНИИ СОВЕТСКОЙ/РОССИЙСКОЙ НАУЧНОЙ ШКОЛЫ) 380

ПРЕДМЕТНЫЙ УКАЗАТЕЛЬ 390

СПИСОК ЛИТЕРАТУРЫ 395

От авторов

В 2013 году Виктору Ильичу Варшавскому, нашему шефу, научному лидеру и другу, исполнилось бы 80 лет. Его не стало в 2005 году. Нам хотелось бы посвятить эту книгу его памяти.

Авторы глубоко благодарны бывшим коллегам и соавторам – А. Г. Астановскому, В. Я. Володарскому, Л. В. Виноградовой, М. А. Кишиневскому, А. Ю. Кондратьеву, В.И.Красюку, О. В. Маевскому, Ю.В. Мамрукову, Н. А. Стародубцеву, Ю. С. Татаринову, А. Р. Таубину, И. В. Яценко, Р.Л.Финкельштейну, – совместная работа с которыми породила много идей, результатов, публикаций и критических замечаний. Большое спасибо рецензентам этой книги А. А. Шалыто и А. И. Водяхо за ценные советы. Особой признательности заслуживает издатель Марк Валерьевич Финков, чье неистребимое желание заставить авторов повысить удобочитаемость книги несомненно принесло ощутимые плоды и желание продолжить сотрудничество.

“Кто может – делает,
кто не может – учит”.
Закон Менкеса.
(Артур Блох. Законы Мерфи)

ВВЕДЕНИЕ

Предлагаемая книга представляет собой расширенное изложение части лекционного курса “Теория вычислительных процессов и структур”, читавшегося в течение многих лет в Санкт-Петербургском государственном электротехническом университете “ЛЭТИ” студентам, обучавшимся по специальности “Прикладная математика” и “Программное обеспечение вычислительной техники и автоматизированных систем”[1].

Пожалуй, следует для начала пояснить смысл не слишком употребительного сочетания терминов “вычислительный процесс” и “структура” в рамках одной теории. Дело в том, что любой объект, являющийся предметом изучения, с одной стороны, может быть представлен совокупностью компонентов и связей между ними. Такое описание носит статический характер и обычно соответствует структурной точке зрения на объект. Адекватные этой точке зрения модели обычно называются *структурными*. С другой стороны, объект может характеризоваться в терминах динамического поведения, т.е. поведения во времени. Такие модели принято называть *поведенческими* или *динамическими*. В этих случаях речь идет скорее о процессе, реализуемом объектом при его взаимодействии с внешней средой, чем собственно о структуре объекта.

Необходимо подчеркнуть неразрывный характер этих двух представлений объекта – **процесс предусматривает наличие определенной структуры, а структура порождает специфическое поведение объекта. Двуединый подход является фундаментальным инструментом проектирования дискретных вычислительных и управляющих систем.** Он используется на всех этапах проектной деятельности: спецификации, анализа, реализации и верификации проектного решения. Данная книга не претендует на полноту освещения всех возникающих здесь проблем и технологий проектирования. Материал, изложенный в книге, имеет целью приблизить читателя

1 Помимо представленного материала, в названном курсе читались такие разделы, как булева алгебра и теория переключательных схем, теория конечных автоматов, теория формальных языков и грамматик и выполнялась курсовая работа. Эти разделы не вошли в книгу

к навыкам построения формальной постановки (спецификации) конкретных задач и их эффективного решения.

Как всякая способная к эволюции теория, данная теория помимо методологических оснований должна иметь прагматические корни и связь с быстроразвивающимся окружением – вычислительными системами, их программными и аппаратными компонентами.

Тенденции развития вычислительной техники характеризуются отказом от архитектур традиционных вычислительных машин, опирающихся на модель фон Неймана, в пользу инновационных архитектурных подходов. Это объясняется не только успехами микроэлектронной технологии. Гораздо более весомым фактором является феномен "семантического разрыва", выражающегося в том, что машинные представления информационных единиц и машинные операции, реализуемые вычислительной системой, редко адекватны объектам и операциям, описываемым в языках программирования. Поэтому такой же "семантический разрыв" имеет место между вычислительной техникой и программированием. Они разделены на два слабо взаимодействующих направления – теорию автоматов и программирование. Эти направления развиваются настолько автономно, что специалисты по "софтверу" и "хардверу", призванные работать совместно, иногда с трудом понимают друг друга.

Выдающийся математик Д. Гильберт (1862 – 1943 гг.) утверждал, что всякая физическая или математическая теория проходит три фазы развития: наивную, формальную и критическую. Теория автоматов прошла две фазы развития и, по-видимому, находится в критической фазе. Причины этого лежат:

1. в резком увеличении размерности решаемых задач в связи с быстро растущим уровнем интеграции элементной базы (БИС, СБИС, системы на кристалле);

2. в необходимости реализации параллелизма и организации взаимодействия асинхронных процессов;

3. в установлении гарантоспособности проектных решений, т.е. обеспечении гарантированного выполнения возложенных на систему функций.

В теории автоматов наметилась явная тенденция к изучению моделей, занимающих по функциональным возможностям промежуточное место меж-

ду конечными автоматами и машинами Тьюринга. К таким моделям относятся сети Петри, которые оказались весьма удобным средством для моделирования динамики поведения вычислительных и управляющих систем, особенно многопроцессорных и распределенных. Определенные успехи в изучении сетей Петри породили другое научное направление – **общую теорию сетей** (сетевых моделей процессов). Одновременно развивалась **теория программирования**, базирующаяся в основном на программных нотациях символьного типа. Ее раздел под названием "**параллельное программирование**", уже давно вступивший в формальную фазу развития благодаря работам Э. Дейкстры и особенно Ч. Хоара (теория взаимодействующих последовательных процессов), ярко демонстрирует, что многие задачи параллельного программирования допускают интерпретации в терминах сетевого подхода, а аппарат сетей Петри удобен для верификации параллельных программ.

В данной книге делаются определенные шаги к сближению идей и подходов, присущих теории автоматов и программированию. Сокращению "семантического разрыва" в немалой степени должны способствовать новые принципы аппаратной реализации, присущие нетрадиционной, так называемой самосинхронной схемотехнике.

При ее использовании могут быть естественно реализованы спецификации на языках параллельных асинхронных моделей. Эти модели характеризуются одной важной особенностью – время как параметр в них не фигурирует, а привязка ко времени осуществляется с помощью причинно-следственных отношений между событиями, происходящими в моделируемой системе.

В пособии широко представлены абстрактные модели, которые могут быть реализованы как средствами параллельного программирования (его актуальность возрастает в связи с широким применением многоядерных процессоров), так и аппаратными средствами с параллельной работой и асинхронным взаимодействием (в частности в условиях нарастающих требований к экономии энергии).

Многие современные модели вычислений, ориентированные сугубо для программного обеспечения (например, объектно-ориентированные и модели с гипертрофированной иерархичностью), оказываются оторванными от реальных физических механизмов потребления ресурсов. "Физичность" представленного модельного подхода и его применимость в равной степени к схемам и программам, углубляет понимание динамики параллельных

процессов. Книга позволяет сохранить семантические “нити” между абстрактными процессами вычислений и физикой явлений, лежащих в основе их реализации.

Пособие написано так, что сначала вводится некоторая *метамодель*, носящая название *асинхронного процесса*, из которой путем различной интерпретации базовых понятий порождаются частные, объектные модели. Это дает возможность выработать единый взгляд на различные динамические модели.

Книга имеет следующую структуру. Глава 1 посвящена метамодели, главы 2 и 3 – специальной и общей теории сетей, глава 4 – моделям, объединяемым понятием динамической логики и удобным для спецификации и синтеза самосинхронных схем, глава 5 – классическим задачам параллельного программирования и механизмам синхронизации процессов, глава 6 – принципам самосинхронной реализации, глава 7 – протоколам обмена данными, глава 8 – прикладным задачам.

Каждая глава книги содержит некоторый набор упражнений, предлагаемых для самостоятельной работы.

Авторы сознают, что книга не свободна от недостатков, и полагают, что справедлив закон Купера: “Если вам непонятно какое-то слово в техническом тексте, не обращайте на него внимания. Текст полностью сохранит смысл и без него”. Тем не менее, замечания читателей будут с благодарностью приняты и, по мере возможности, учтены в дальнейшем.

Французский геометр Г. Монж (1746 – 1818) сказал когда-то, что очарование, сопровождающее науку, может помочь победить свойственное людям отвращение к напряжению ума. Будет приятно, если на экзаменах мы станем свидетелями победы студента, изучившего книгу, над самим собой, и будем приветствовать его попытки углубиться в проблематику, обратившись к литературе.

Сейчас наступил век новой информационной технологии, характеризующийся тем, что время удвоения объема накопленных научных знаний составляет 2 – 3 года, а материальные затраты на обработку национальных информационных ресурсов уже превысили расходы на энергетику. В этих условиях сколь-нибудь подробная библиография работ по тематике книги вряд ли окажет студенту помощь. По этой причине авторы уклонились от ссылок в тексте на литературу. Однако в кратком библиографическом спи-

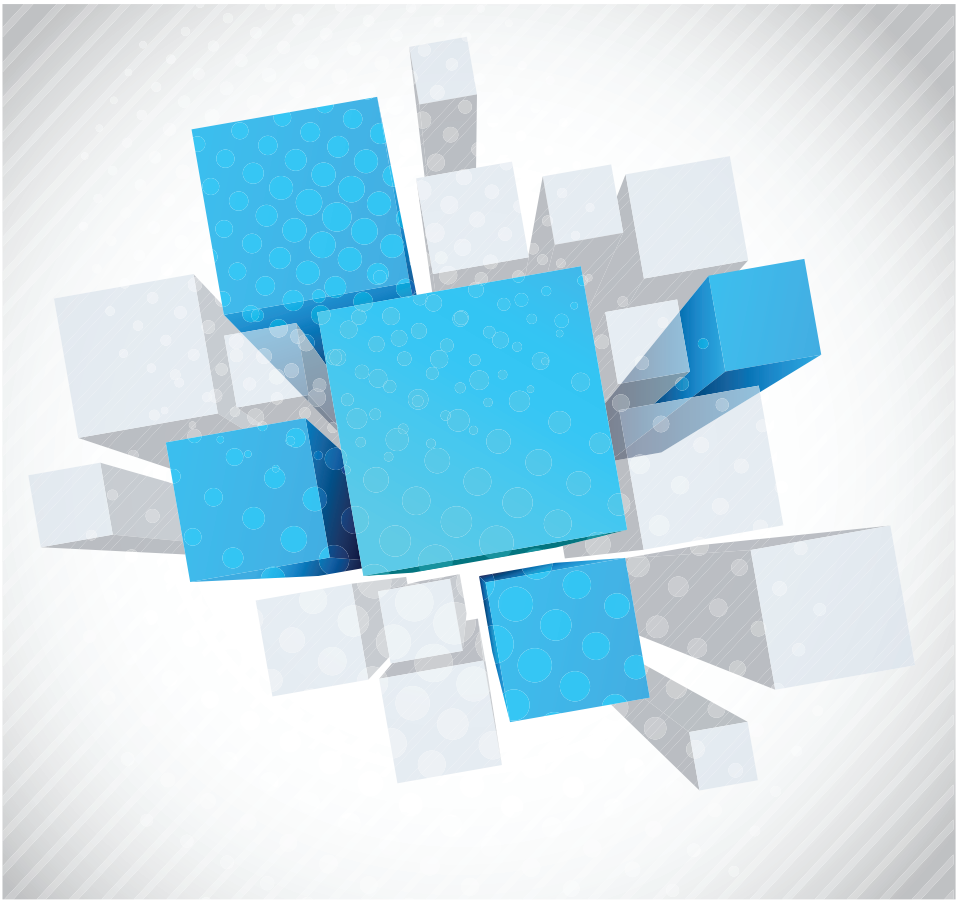
ске даны аннотации рекомендуемой основной и дополнительной литературы и указано, к каким главам она имеет наибольшее отношение.

Хотя основным объектом книги являются формальные модели, принятый уровень формализма не удовлетворит взыскательного читателя-студента, тяготеющего к строгим математическим выкладкам. Это сделано сознательно по двум причинам. Первая состоит в том, что авторам хотелось бы расширить читательскую аудиторию и не оттолкнуть от теоретических подходов инженеров и программистов, предпочитающих работать над практическими задачами. Вторая причина носит ресурсный характер: введение дополнительного формализма сильно увеличило бы объем книги и нанесло ущерб основной цели.

Наконец, следует отметить, что книга, претендующая на роль учебного пособия, носит компилятивный характер, хотя в значительной степени использует оригинальные результаты и точку зрения коллектива авторов и их ближайших коллег. Авторы выражают своим коллегам и рецензентам чувства своей искренней признательности.

СОКРАЩЕНИЯ, ПРИНЯТЫЕ В ТЕКСТЕ

АП — асинхронный процесс;
БИС — большая интегральная схема;
ВМ — вычислительные машины;
ДМ — диаграмма Маллера;
ДП — диаграмма переходов;
МГ — маркированный граф;
НМЛ — накопитель на магнитной ленте;
НСП — немаркированная сеть Петри;
СГ — сигнальный граф;
СБИС — сверхбольшая интегральная схема;
СП — сеть Петри;
СПД — сети потоков данных;
ССП — стохастические сети Петри;
ЦП — центральный процессор.



Глава 1.

АСИНХРОННЫЙ ПРОЦЕСС

Как правило, мячи, посланные головой, летят по траектории.
Из репортажа В. Маслаченко о футбольном матче "Спартак" – "Брюгге" 30.09.1981

Для проектирования дискретных вычислительных и управляющих систем, прежде всего, необходимы рабочие модели, позволяющие адекватно задавать статические (структурные) и динамические (поведенческие) свойства. Желательно, чтобы эти модели позволяли отразить возможный параллелизм и асинхронность взаимодействия подсистем. Естественно потребовать, чтобы такие модели были формальными (математическими), ибо выявление интересующих нас свойств (анализ систем) должно носить строгий характер. При этом **в первую очередь нас интересуют не количественные, а качественные особенности, позволяющие судить об адекватности проектируемой системы требованиям, заложенным в техническом задании**. Неадекватность может возникать, например, в случаях изменения порядка выполнения действий или порядка следования управляющих сигналов, нарушения предполагаемой параллельности действий или сигналов, возникновения тупиков и бесконечных циклов и других нарушений свойств системы, предусмотренных техническим заданием.

Под *математической моделью*, **как и обычно, понимается "приближенное описание какого-либо класса явлений внешнего мира, выраженное с помощью математической символики"** (Математическая энциклопедия, т. 3, с. 574). Каждая из этих моделей отражает те или иные аспекты поведения систем и в силу этого имеет ограниченное применение. Тем не менее, наличие общих свойств у таких моделей позволяет предложить некоторую удобную в методологическом отношении *метамодель*, которая порождает частные, объектные модели. **Термин "метамодель" понимается как модель, применяемая для исследования и описания некоторого класса моделей**, по аналогии с другими понятиями, образованными посредством приставки "мета". Механизм такого порождения определяется способами *интерпретации* – "задания (смысла) математических выражений (символов, формул и т.д.)" (там же, т. 2, с. 635) – основных понятий метамодели, в качестве которой здесь будет использована некоторая формализация, носящая название "асинхронный процесс".

1.1. Определение асинхронного процесса

Понятие асинхронности

Прежде, чем дать определение асинхронного процесса, раскроем содержание слова «*асинхронный*».

Любая вычислительная или управляющая система реализуется физическим устройством, функционирующим в реальном физическом времени. Время вводится на этапе структурного проектирования такого устройства. Реальное физическое время квантуется, т.е. представляется в виде последовательности квантов (тактов). Квантованное физическое время обычно называют *дискретным временем*. На выполнение системой шага алгоритма выделяется квант времени, или такт. Таким образом, последовательность шагов алгоритма выполняется за время прохождения соответствующего количества тактов. **В зависимости от того, как непрерывное физическое время отображается на последовательность тактов в дискретном времени, физические устройства делятся на два класса:** *синхронные автоматы* **и** *асинхронные автоматы*.

- **В синхронных автоматах физическое непрерывное время делится на такты сигналами от генератора тактовых импульсов.** Длительность такта постоянна и определяется частотой генератора (тактовой частотой). Генератор выполняет функцию часов. Для привязки события ко времени говорят о номере такта, когда произошло событие.

- **В асинхронных автоматах начало выполнения очередного шага алгоритма связано с изменением состояния входа (с событием на входе) и физическое время квантуется сигналами окончания переходных процессов в схеме автомата.** Такие асинхронные автоматы называются *согласованными* и отличаются от традиционных асинхронных автоматов, в которых смена входных наборов осуществляется с помощью тактирования (как и в синхронных), рассчитанного на худший случай. В дальнейшем под асинхронным автоматом будем понимать только согласованный автомат. В согласованных автоматах длительность такта не является постоянной, что приводит к сложности определения в реальном физическом времени моментов выполнения шагов алгоритма, реализуемого автоматом.

При построении устройства, содержащего много параллельно работающих блоков, возникает проблема организации требуемого взаимодействия блоков, т.е. проблема координации их функционирования, или проблема синхронизации. Когда говорят, что некоторые события происходят синхронно, то обычно имеют в виду, что эти события являются следствиями одних и тех же причин. Такое понимание синхронности можно использовать как в дискретном (логическом) времени, так и в физическом времени. Когда говорят о *синхронной системе*, то обычно подразумевают, что все события в системе могут происходить в том или ином такте. События происходят синхронно, если они происходят в одном такте. Такое понимание синхронности используется специалистами по аппаратным средствам при построении синхронных систем.

Термин *асинхронность* часто трактуется по-разному специалистами в области программного обеспечения и аппаратуры:

- **Специалисты в области программирования и архитектуры имеют дело с логическим временем, измеряемым в шагах алгоритма.** Они трактуют асинхронность как вариации (например, в зависимости от данных) числа шагов в процессе (алгоритме) от его инициализации до получения результата. Например, деление без восстановления остатка является синхронным процессом, в то время как деление с восстановлением остатка – асинхронным процессом.
- **Специалисты в области микроэлектроники и вычислительной техники имеют дело с физическими процессами, которые протекают в физических устройствах, и, следовательно, с аналоговым физическим временем.** Для них асинхронность связана с неуправляемыми вариациями времени распространения сигналов по сети элементов и перехода из одного дискретного состояния в другое. Некоторые факторы таких вариаций могут быть постоянными (например, разброс технологических параметров), некоторые – переменными (такие как изменения условий функционирования: температуры, питающих напряжений и др.), а также зависящими от данных (например, время завершения переносов в сумматоре).

Асинхронные модели являются альтернативой синхронного подхода к проектированию, в которой не требуется использования тактовых генераторов, а системное время (*логическое время*) определяется частичным порядком, накладываемым на дискретные события. Системное время вводится с помощью спецификации причинно-следственных отношений между событиями в системе.

Концепция асинхронного процесса

Само понятие *асинхронный процесс* будет далее использоваться как для задания описания (спецификации) поведения системы, так и для решения задач анализа и синтеза.

Назовем *асинхронным процессом* (АП) четверку $< S, F, I, R >$, в которой: S – непустое множество *ситуаций*, F – отношение непосредственного следования ситуаций, определяемое на множестве $S \times S$ ($F \subset S \times S$), I – множество *инициаторов* ($I \subset S$), R – множество *результантов* ($R \subset S$).

Прокомментируем это определение.

- Слова "ситуация" и "процесс" нужно понимать в общеупотребительном смысле. "**Ситуация (позднелат. Situati – положение) – сочетание условий и обстоятельств, создающих определенную обстановку, положение**" ("Советский энциклопедический словарь", 1980, с.1226). Конкретизация термина "ситуация" зависит от интерпретации понятия "асинхронный процесс" и будет дана ниже. К этому следует добавить, что **в любой момент наблюдения асинхронный процесс может находиться в одной и только одной ситуации**.

  "*Процесс* (от лат. *Processus* – продолжение) – 1) последовательная смена состояний ...; 2) совокупность последовательных действий для достижения какого-либо результата ..." (БСЭ, 3-е изд., т. 21, с.161). В данном случае процесс описывает динамику смены ситуаций. Термин "процесс" уточняется прилагательным "асинхронный". Это уточнение используется для того, чтобы показать, что *категория времени* формально не фигурирует в определении; привязка к сущности "время" осуществляется с помощью отношения F. Запись $s_i F s_j$ (которая равносильна записи $(s_i, s_j) \in F$), где $s_i, s_j \in S$, используется для задания того факта, что ситуация s_j непосредственно следует за ситуацией s_i. Время перехода из s_i в s_j не ограничивается – оно может быть произвольным, но конечным.

- Введенное отношение F может пониматься как *логическая возможность* (термин заимствован из *модальной логики*). Иначе: **запись $s_i F s_j$ означает, что пребывание процесса в ситуации s_i должно когда-нибудь вызвать его переход в последующую ситуацию s_j. Таким образом, ситуацию s_i можно считать причиной смены ситуации в асинхронном процессе, а s_j – ее следствием**. Между тем, наряду с $s_i F s_j$ может также иметь место $s_k F s_j$, $k \neq j$. Это означает, что существует и другая причина перехода АП

в s_j. Аналогично, из s_j при s_jFs_r и s_jFs_t следует переход либо в s_r, либо s_t ($s_r, s_t \in S, r \neq t$), и это означает, что одна и та же причина имеет два альтернативных следствия, т.е. локальное поведение асинхронного процесса носит *недетерминированный* характер. Этот недетерминизм может быть вызван "приблизительностью", отсутствием точных знаний о характере асинхронного процесса или желанием абстрагироваться от некоторых несущественных черт моделируемой системы.

- **Инициаторы – подмножество ситуаций, активирующих процесс.** Назначение инициаторов осуществляется на основе смыслового содержания (семантики) процесса.

- **Результанты – подмножество, состоящее из финальных ситуаций.** Их выбор также опирается на семантику процесса.

Спецификация асинхронного процесса

Асинхронный процесс можно задать в виде ориентированного графа (рис. 1.1), в котором $S = \{s_1, s_2, ..., s_9\}$ (число ситуаций равно 9), причем имеет место

$$s_1Fs_6,\ s_2Fs_4,\ s_2Fs_6,\ s_2Fs_5,\ s_5Fs_6,\ s_4Fs_6,\ s_6Fs_8,\ s_5Fs_9,\ s_3Fs_7,\ s_7Fs_8.$$

Таким образом, если s_iFs_j, то это изображается в орграфе дугой, исходящей из s_i и входящей в s_j. Пары ситуаций, не принадлежащих отношению F, дугами не связываются.

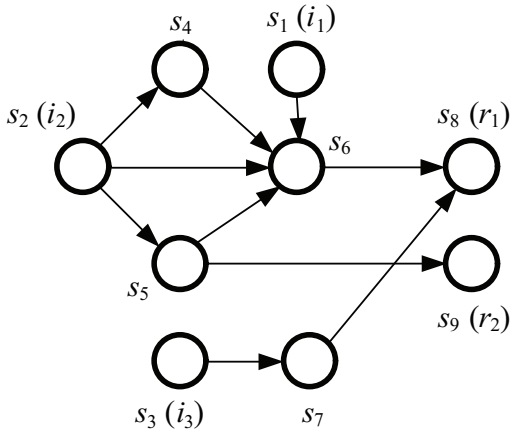


Рис. 1.1. Пример задания асинхронного процесса в виде ориентированного графа (далее — просто "пример асинхронного процесса")

Пусть $I = \{s_1, s_2, s_3\}$, $R = \{s_8, s_9\}$.

Наличие нескольких исходящих дуг из вершины графа в определенной мере отражает локальный недетерминизм процесса. Так, из s_2 асинхронный процесс может попасть в одну из трех ситуаций s_4, s_5 и s_6. Недетерминизм может носить и более глубокий, *глобальный* характер. Имеется в виду, что, начавшись в одном и том же инициаторе, процесс может попасть в разные результанты, т.е. реализоваться по-разному. Важно отметить, что для наблюдателя, отмечающего работу асинхронного процесса лишь на уровне инициаторов и результантов (это весьма типично на определенных этапах задания поведения описываемой дискретной системы), локальный недетерминизм недоступен в отличие от глобального недетерминизма. Изображение процесса в виде орграфа несет информацию о возможных вариантах его реализации, т.е. о возможных траекториях асинхронного процесса, ведущих от инициаторов в результанты. В данном случае возможны шесть таких траекторий асинхронного процесса:

Траектория 1): $s_1 \to s_6 \to s_8$,

Траектория 2): $s_2 \to s_4 \to s_6 \to s_8$,

Траектория 3): $s_2 \to s_6 \to s_8$,

Траектория 4): $s_2 \to s_5 \to s_6 \to s_8$,

Траектория 5): $s_2 \to s_5 \to s_9$,

Траектория 6): $s_3 \to s_7 \to s_8$.

Рассматривая, например, траектории 4 и 5 можно убедиться, что из одного и того же инициатора $s_2 \in I$ они ведут в разные результанты $s_8 \in R$ и $s_9 \in R$. Причины такого разного поведения не указаны в задании асинхронного процесса и могут быть объяснены лишь с помощью средств, находящихся вне модели.

Наличие в модели асинхронного процесса отношения F непосредственного следования ситуаций наталкивает на мысль о возможности использования более общего отношения их следования. В частности, можно использовать понятие степени отношения F. Пусть запись $s_i F^n s_k$ означает, что существуют $n-1$ промежуточных ситуаций s_a, s_b, ..., s_w, для которых имеет место $s_i F s_a$, $s_a F s_b$, ..., $s_w F s_k$, т.е. существует траектория, включающая $n+1$ вершину и n дуг и ведущая из s_i в s_k.

Возвращаясь к примеру АП, представленному на рис. 1.1, для его траекторий можно записать: 1) $s_1 F^2 s_8$, 2) $s_2 F^3 s_8$, 3) $s_2 F^2 s_8$, 4) $s_2 F^3 s_8$, 5) $s_2 F^2 s_9$, 6) $s_3 F^2 s_8$.

Транзитивное замыкание отношения *F* будем обозначать *F.** Запись $s_i F^* s_j$ можно понимать как факт наличия траектории, ведущей из s_i в s_j, длина которой (степень отношения), вообще говоря, несущественна, хотя и конечна.

Любой фрагмент асинхронного процесса , для которого имеет место $s_i F^* s_j$, будем называть переходом $s_i \to s_j$.

Поскольку понятия инициатора и результанта должны удовлетворять некоторым дополнительным ограничениям, вытекающим из их семантики, попытаемся явно ввести соглашения о правилах назначения инициаторов и результантов. Во-первых, очевидно, что инициатор не может быть только следствием некоторой ситуации, он обязательно должен быть причиной. Поэтому, если некоторая вершина в графе не содержит входящих дуг, то она обязана быть объявленной инициатором. Кроме того, для любого инициатора $i \in I$ должна существовать такая ситуация $s \in S$, что имеет место *iFs*. Это вовсе не означает, что обязательно для этой ситуации должно выполняться $s \in S \backslash I$, т.е. эта ситуация обязана не быть инициатором. Вполне допустимо $i_1 F i_2$, т.е. за одним инициатором может непосредственно следовать другой инициатор. Некоторое подмножество инициаторов может образовывать ориентированный цикл (контур) орграфа, но хотя бы один инициатор, входящий в этот контур, должен иметь исходящую дугу, являющуюся входящей дугой вершины (ситуации), принадлежащей множеству $S \backslash I$.

Из сказанного выше следует, что если из инициатора непосредственно следует ситуация *s*, не относящаяся к инициаторам, то непосредственно следующая за *s* ситуация не может быть инициатором. Иначе говоря, если имеет место iFs_1, $s_1 \notin I$, то из $s_1 F s_2$ следует, что $s_2 \notin I$.

Точно так же результант не может быть только причиной, он обязательно должен быть следствием какой-либо причины. По существу, при определении результантов подчеркивается тот факт, что если какая-либо ситуация из *S* объявлена результантом, то любая другая следующая за ней ситуация, если она существует, также является результантом. Поэтому, если некоторая вершина в орграфе не содержит исходящих дуг (является тупиковой), то она обязана быть объявленной результантом. Кроме того, если для некоторого $r \in R$ имеет место *rFs*, то должно выполняться $s \in R$. Что касается предшественников результанта, то ими могут быть любые ситуации из множества *S*, в том числе результант или инициатор. Некоторые подмножества результантов могут образовывать контуры, но хотя бы один из результантов

контура должен иметь входящую дугу, являющейся исходящей дугой вершины, принадлежащей ситуациям из множества $S\backslash R$ или I.

Сказанное выше означает, что в любом асинхронном процессе должны выполняться следующие два условия:

1) **все траектории из инициаторов ведут в результанты** (для любой ситуации $s \in S\backslash R$ найдется такой результант $r \in R$, что sF^*r);
2) **каждая из траекторий, приводящих к результанту, начинается в каком-либо инициаторе** (для любой ситуации $s \in S\backslash R$ найдется инициатор $i \in I$ такой, что iF^*s).

Далее будем полагать, что при спецификации (задании) АП приведенные соглашения выполнены. Таким образом, невыполнение любого из перечисленных выше ограничений означает, что рассматриваемая модель не является асинхронным процессом[1].

Протокол асинхронного процесса

Для задания АП на уровне его вход-выходного поведения удобно пользоваться понятием протокола АП. Протоколом асинхронного процесса будем называть отношение $Q \subseteq I \times R$.

Протокол АП можно рассматривать как простой асинхронный процесс, в котором за каждым инициатором непосредственно следует результант. Поэтому в протоколе АП множество ситуаций состоит лишь из инициаторов и результантов: $S = I \cup R$. При переходе от асинхронного процесса к его протоколу для каждой пары (i, r) пучки траекторий, ведущих из i в r, возможно через промежуточные ситуации, заменяются одной дугой.

Протокол АП, представленного на рис. 1.1, можно записать в виде:

$$i_1 Q r_1, i_2 Q r_1, i_2 Q r_2, i_3 Q r_1. \quad (1.1)$$

Отсюда ясно, что любой инициатор может вызвать попадание процесса в результант r_1, но только при инициаторе i_2 может быть достижим результант r_2. Такая упрощенная спецификация процесса полезна при рассмотрении АП как "черного ящика", когда детализация его внутреннего устройства не нужна наблюдателю (в протоколе АП исчезает необходимость рас-

[1] Внимательный читатель, вероятно, заметит, что для АП недопустимо как $I = \varnothing \ \& \ R \neq \varnothing$, так и $I \neq \varnothing \ \& \ R = \varnothing$.

смотрения и отображения параллелизма внутри АП). Наблюдателю доступна лишь реакция моделируемого устройства в ответ на определенные воздействия извне.

1.2. Классификация

Рассмотри четыре важных класса асинхронных процессов:

- **автономный,**
- **эффективный,**
- **управляемый,**
- **простой.**

Сразу заметим, что такая классификация не является разделительной. Далее мы подробно рассмотрим каждый из этих наиболее важных классов асинхронных процессов. А в следующих разделах книги будут введены некоторые другие классы асинхронных процессов.

Автономный асинхронный процесс

Говоря о классификации асинхронных процессов, необходимо, прежде всего, выделить **класс асинхронных процессов, для которого характерно отсутствие инициаторов и результантов.**

Если для некоторого процесса $I = R = \varnothing$, то такой процесс называется автономным. *Автономный* процесс есть двойка $< S, F >$, в которой, по определению АП с учетом принятого соглашения о назначении инициаторов и результантов, отсутствуют вершины–ситуации, не имеющие входящих или исходящих дуг.

Пример автономного асинхронного процесса представлен на рис. 1.2.

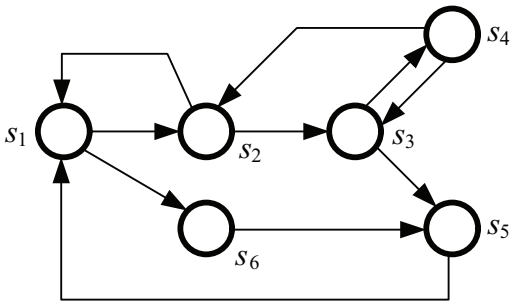


Рис. 1.2. Граф автономного асинхронного процесса

Эффективный асинхронный процесс

Асинхронный процесс будем называть эффективным, если в нем нет "непродуктивных" контуров, включающих ситуации, не являющиеся инициаторами или результантами. Более формально условие эффективности АП может быть сформулировано в виде: не найдется ни одной пары ситуаций s_i и s_j, для которой выполняется следующий предикат $(s_i \notin I \cup R) \& (s_j \notin I \cup R) \& (s_i F^* s_j) \& (s_j F^* s_i)$. Эффективность АП оставляет место недетерминированности, т.е. возможно, что из некоторого инициатора процесс попадает в разные результанты, но он не содержит ориентированных циклов вне ситуаций, принадлежащих I или R.

Рассмотрим асинхронный процесс, граф которого приведен на рис. 1.3.

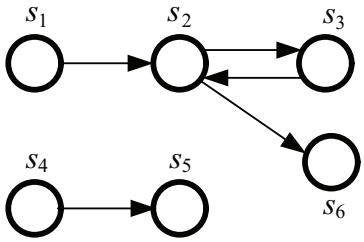


Рис. 1.3. Пример асинхронного процесса

Если для этого графа $I = \{s_1, s_4\}$, $R = \{s_5\}$, то данная модель не является эффективным АП, так указанный выше предикат выполняется для пары состояний (s_2, s_3). Если $I = \{s_1, s_4\}$, $R = \{s_5, s_6\}$, то предикат продолжает выполняться для той же пары состояний (s_2, s_3). Заметим, что если $I = \{s_1, s_4\}$, $R = \{s_3, s_5, s_6\}$, то процесс не является АП, так как имеется возврат из множества результантов: $s_3 F s_2$, $s_3 \in R$, а $s_2 \notin R$. Следующий вариант назначения инициаторов и результантов дает эффективный АП: $I = \{s_1, s_4\}$, $R = \{s_2, s_3, s_5, s_6\}$.

Из общих соображений ясно, что принадлежность АП классу эффективных процессов является положительным свойством. Наоборот, если АП неэффективен, то его спецификация некорректна в смысле возможности реализации и возникает необходимость ее уточнения и модификации.

Часто важно иметь способ разбиения множества ситуаций S на *классы эквивалентности*. С этой целью введем отношение E, удовлетворяющее двум условиям:

1) для $s_i, s_j \in S$ $s_i E s_j$, если $s_i F^* s_j$ и $s_j F^* s_i$;

2) для любого $s \in S$ sEs.

Условие 1) обеспечивает симметричность и транзитивность отношения E (транзитивность вытекает из транзитивности F^*), а условие 2) – рефлексивность E. Очевидно, что E является отношением эквивалентности.

Отношение E позволяет построить фактор-множество множества S, т.е. разбиение $\pi = (S_1, S_2, ..., S_p)$ множества S на классы эквивалентности. Для классов эквивалентности определим отношение F непосредственного следования классов.

Пусть имеется последовательность (возможно, бесконечная) ситуаций $s(1)$ ... $s(i)$ $s(i+1)$ ... , где $s(i)$ – элемент, стоящий в последовательности на i-м месте. Для АП можно составить множество последовательностей ситуаций, в которых для любого места i, кроме последнего, справедливо $s(i)Fs(i+1)$ и таких, что ни одна из последовательностей не является частью другой. Все такие последовательности назовем *допустимыми*. **Каждая допустимая последовательность ситуаций описывает возможный ход процесса смены ситуаций (траекторию АП) и соответствует реализации АП.**

Все допустимые последовательности классов эквивалентности, определяемые отношением F, конечны (в отличие от допустимых последовательностей для отношения F, определяемого на множестве $S \times S$, которые могут быть и бесконечными). **В допустимых последовательностях классов можно выделить начальные и конечные элементы, которые будем называть *начальными и заключительными классами эквивалентности*.**

На основании введенных понятий можно сформулировать следующие утверждения.

1. Если некоторая ситуация s является инициатором и $s \in S(j)$, где $S(j)$ – класс эквивалентности, стоящий на j-м месте в некоторой допустимой последовательности, то все ситуации классов $S(1)$, $S(2)$, ... , $S(j)$ в этой последовательности являются инициаторами.
2. Если некоторая ситуация s является результантом и $s \in S(j)$, то все ситуации из классов $S(j)$, $S(j+1)$, ... , $S(q)$, где $S(q)$ – заключительный класс, являются результантами.
3. Для эффективного АП любой начальный класс состоит только из инициаторов, а любой заключительный класс – только из результантов.
4. Для эффективного АП любой класс эквивалентности ситуаций, не принадлежащих к инициаторам и результантам, состоит из одной ситуации.

Управляемый асинхронный процесс

Если в эффективном асинхронном процессе каждая допустимая последовательность классов ведет из начального класса в один и только один заключительный класс, то такой процесс назовем управляемым. Таким образом, в управляемом АП вводится ограничение на степень недетерминизма: все траектории из любого инициатора оканчиваются одним заключительным классом. Принадлежность асинхронного процесса классу управляемых является положительным свойством, подчеркивающим детерминированный характер асинхронного процесса.

В качестве примера рассмотрим асинхронный процесс, граф которого представлен на рис. 1.4а. На рис. 1.4б показано отношение F для классов эквивалентности множества ситуаций этого асинхронного процесса.

Если $I = \{s_1, s_2, s_3, s_4\}$, $R = \{s_7, s_8, s_9, s_{10}\}$, легко убедиться, что этот процесс является управляемым. Если отношение F дополнить парой $s_2 F s_4$, то полученный асинхронный процесс становится неуправляемым вследствие того, что образуются два заключительных класса – $\{s_7\}$ и $\{s_8, s_9, s_{10}\}$, но остается эффективным.

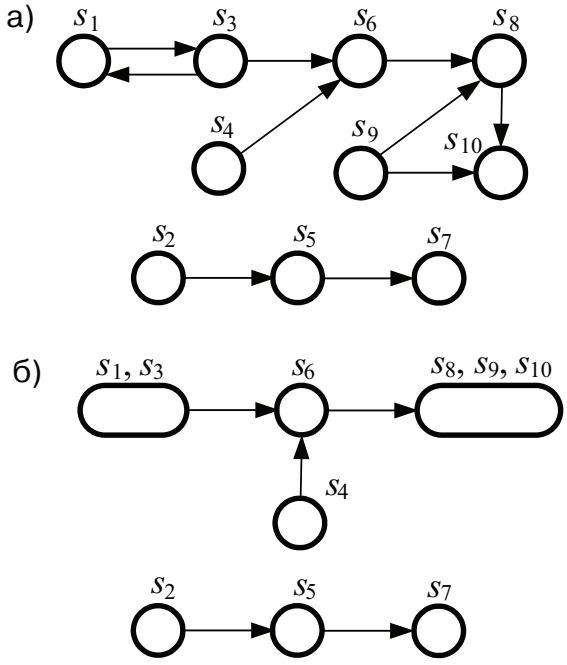


Рис. 1.4. а) Пример асинхронного процесса ;
б) разбиение ситуаций на классы эквивалентности.

Простой асинхронный процесс

В дальнейшем будет использоваться также следующий класс АП. Пусть в эффективном АП:

1) для любых $i \in I$ и $s \in S$ из iFs следует $s \notin I$;
2) для любых $s \in S$ и $r \in R$ из sFr следует $s \notin R$.

Это означает, что **из инициатора (результанта) нельзя попасть в другой инициатор (результант), т.е. каждая траектория содержит в точности по одному инициатору и результанту. Такой асинхронный процесс будем называть простым.**

Если рис. 1.1 задает отношение F для асинхронного процесса, у которого $I = \{s_1, s_2, s_3\}$, $R = \{s_8, s_9\}$, то такой процесс является простым. АП, представленный на рис. 1.4а, напротив, не принадлежит классу простых.

Ниже будут введены некоторые другие классы асинхронных процессов.

1.3. Репозиция

В неавтономных АП траектории, начинаясь в инициаторах, ведут в результанты, и в них "умирают", хотя при наличии в заключительном классе более одного результанта могут иметь место переходы внутри этого класса. Таким образом, АП не может быть возобновлен. По-видимому, необходим какой-то **механизм перехода от результантов к инициаторам, который позволил бы эффект "реанимации АП", т.е. его повторных активаций**. Такой механизм задается *репозицией* АП.

Репозицией асинхронного процесса $P = <S, F, I, R>$ **назовем эффективный асинхронный процесс** $P' = <S', F', I', R'>$ **такой, что** $S' \subseteq I \cup R \cup S^{\partial}, I' \subseteq R, R' \subseteq I$.

Ситуации S' репозиции могут содержать лишь те ситуации из исходного процесса, которые являются инициаторами и результантами и, кроме того, некоторые дополнительные ситуации S^{∂}, отсутствующие в описании исходного АП ($S^{\partial} \cap S = \varnothing$). Отношение F' задает траектории перехода от элементов из $I' \subseteq R$ к элементам из $R' \subseteq I$ (возможно, через *дополнительные ситуации* из S^{∂}).

Если $I' = R$, $R' = I$, то репозицию назовем *полной*, если $F' = \varnothing$, то репозиции не существует, в остальных случаях она называется *частичной*.

Объединение АП и его репозиции образует *автономный процесс* $P^a = <S^a, F^a>$, $S^a = S \cup S^\partial$, $F^a = F \cup F'$.

В качестве примера рассмотрим репозицию процесса, представленного на рис. 1.1.

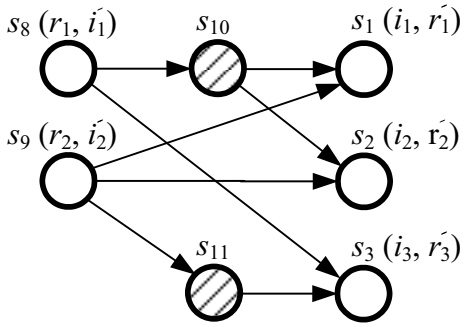


Рис. 1.5. Пример репозиции асинхронного процесса, показанного на рис. 1.1

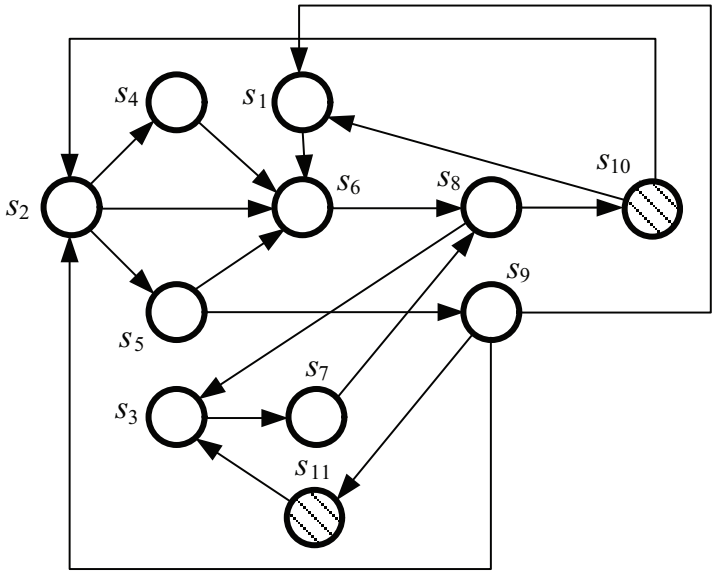


Рис. 1.6. Автономный процесс, образованный композицией асинхронного процесса и его репозицией, представленными на рис. 1.1 и рис. 1.5 соответственно

Один из вариантов репозиции показан на рис. 1.5. Имеет место: $I' = \{s_8, s_9\}$, $R' = \{s_1, s_2, s_3\}$, $S^\partial = \{s_{10}, s_{11}\}$. Дополнительные ситуации на рисунке заштрихованы. Поскольку отношение F', как видно из рисунка, таково, что существу-

ют траектории, ведущие из каждого инициатора репозиции к каждому ее результанту, то такая репозиция является полной. При исключении любой дуги, например, соответствующей отношению $s_8F's_3$ или $s_8F's_{10}$, репозиция становится частичной: из i'_1 (s_8) становится недостижим один результант r'_3 (s_3) в первом случае и два результанта r'_1 (s_1) и r'_2 (s_2) во втором случае. Объединение (композиция) АП, представленного на рис. 1.1, и его полной репозиции (рис. 1.5) приводит к автономному АП, показанному на рис. 1.6.

Композиция асинхронного процесса и его репозиции

При композиции асинхронного процесса и его репозиции склеиваются эквивалентные ситуации, так как попарно результанты (инициаторы) исходного процесса и инициаторы (результанты) его репозиции представляют собой одни и те же ситуации. Результат композиции является, как легко убедиться, автономным процессом. Появление дополнительных ситуаций диктуется соображениями семантического характера.

Приведенный асинхронный процесс

Возвращаясь к определению процесса и его репозиции, введем следующее определение.

Приведенным асинхронным процессом **назовем процесс** $P^n = <S^n, F^n, I^n, R^n>$ **такой, что он представляет собой композицию исходного асинхронного процесса** $P = <S, F, I, R>$ **и его репозиции** $P' = <S', F', I', R'>$, **в которой из отношения** F' **исключены дуги, входящие непосредственно в результанты** R'.

Очевидно, что для приведенного асинхронного процесса имеет место $S^n = S \cup S^\partial$, $F^n = F \cup (F' \setminus (S' \times I))$, $I^n \subseteq I$, $R^n = R \cup S^\partial$.

В зависимости от того, какую репозицию имеет процесс P, **соответствующий приведенный процесс** P^n **будем называть полностью или частично приведенным. Если для АП репозиции не существует, то он называется *неприводимым*.**

Возвращаясь к примеру, показанному на рис. 1.6, видим, что для построения приведенного АП достаточно исключить дуги, ведущие в инициаторы $\{s_1, s_2, s_3\}$. В результате получим АП, представленный на рис. 1.7. Он содержит дополнительные ситуации s_{10} и s_{11}, которые, естественно, должны быть отнесены к результантам приведенного АП, так как являются тупиковыми ситуациями. Ситуации s_8 и s_9 в этом АП по-прежнему остаются его результантами.

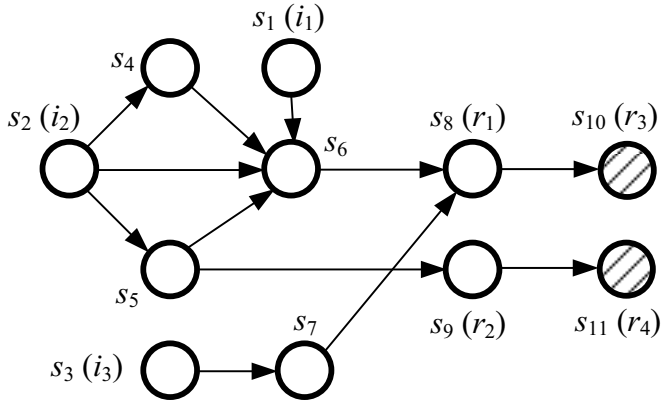


Рис. 1.7. Иллюстрация понятия ***приведенного асинхронного процесса***

Понятно, что приведенный асинхронный процесс представляет собой "заготовку" для осуществления композиции исходного АП и его репозиции, которая выполняется путем введения дуг, направленных от результантов приведенного АП (в том числе и от дополнительных ситуаций) к его инициаторам. По сути дела, приведение АП означает модификацию исходного АП за счет введения в его состав ситуаций из S^{∂}. Понятие приведенного процесса далее будет применяться при исследовании композиций АП.

Конвейерный асинхронный процесс

Введем еще один класс АП – конвейерные процессы.

Полные репозиции простого и непростого АП будем называть соответственно *тривиальной* и *нетривиальной*. При нетривиальной репозиции в качестве инициаторов АП репозиции могут служить результанты исходного АП, не обязательно принадлежащие заключительному классу эквивалентности.

Обозначим через R^3 объединение заключительных классов эквивалентности ситуаций некоторого АП, а через S^t – множество ситуаций АП, принадлежащих траектории t.

Пусть задан управляемый асинхронный процесс $P = <S, F, I, R>$ такой, что:

1) каждый его заключительный класс эквивалентности ситуаций состоит из одного результанта r_j ($r_j \in R^3$);

2) вместе с нетривиальной репозицией АП образует автономный процесс $P^a = <S^a, F^a>$, на множестве S^a ситуаций которого определена функция g: $S^a \rightarrow R^a$ такая, что если в исходном управляемом асинхронном процессе iF^*r, $i \in I$, $r \in R^3$, то в соответствующем автономном процессе из $i \in S^t$ следует, что существует $s \in S^t$, iF^*s и $g(s)=r$.

Автономный асинхронный процесс, удовлетворяющий свойствам 1) и 2), и имеющий фрагменты траекторий вида $...i_1...i_2...s_1...s_2...$, где $g(s_1)=r_1$, $g(s_2)=r_2$, называется *конвейерным* АП.

Введение понятия конвейерного АП позволяет изучать конвейерный принцип обработки информации, в основе которого лежит идея такой повторной инициации АП, при которой АП еще не достиг результанта из заключительного класса эквивалентности ситуаций. Для иллюстрации конвейерного принципа воспользуемся следующей аналогией. Представим себе наклонный желоб, по которому скатываются шарики. Когда пущен первый шарик, на верхний конец желоба можно поместить другой, потом третий и т.д. Шарик может догнать предыдущий либо отстать от него, но не может обогнать предыдущий, догнав его, он может катиться дальше не быстрее предыдущего. Примеры конвейерных процессов будут приведены ниже.

1.4. Структурирование

Понятие ситуации ранее не конкретизировалось. Говорилось только, что оно отражает некоторое глобальное состояние АП, т.е. состояние всех компонент системы, которую специфицирует АП. Поэтому часто возникает необходимость в уточнении, детализации ситуации или, как говорится, в структурировании ситуации. Структурирование в зависимости от решаемых задач может выполняться разными способами.

Способ I. Типовым вариантом структурирования является разбиение ситуаций на компоненты (события); при этом каждой компоненте ставится в соответствие некоторый предикат p_i, принимающий значение 1, если значение соответствующего логического условия истинно, и 0 в противном случае. Ситуация при таком структурировании представляется двоичным вектором, размерность которого равна числу задаваемых компонент, а число единиц вектора соответствует числу истинных (в этой ситуации) предикатов.

Подчеркнем, что попытки применить АП в приложениях обязательно связаны с семантикой и структурированием ситуаций.

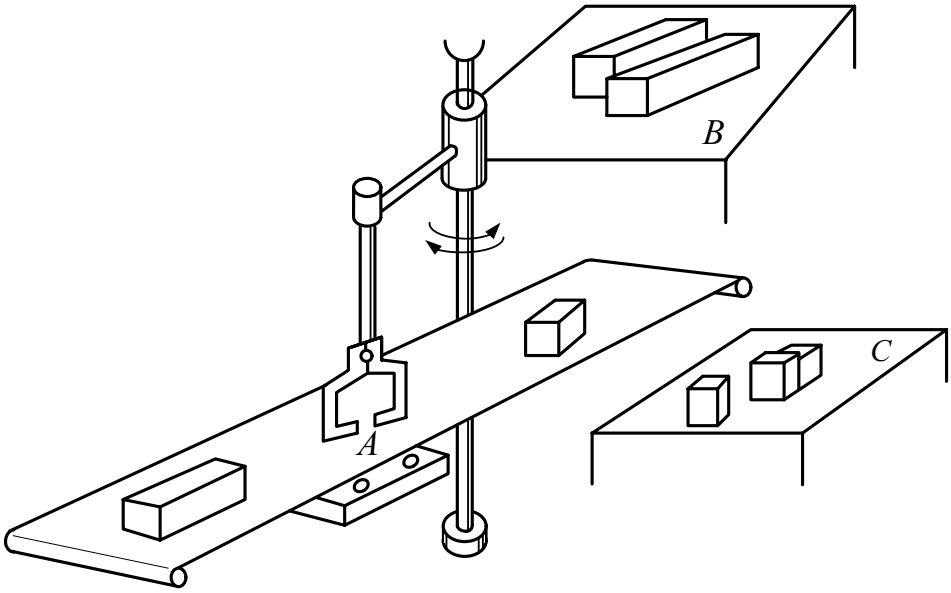


Рис. 1.8. Ленточный конвейер как объект, моделируемый на языке асинронных процессов

Пример. Пусть имеется горизонтальный ленточный конвейер (рис. 1.8), на который подаются детали двух сортов – тяжелые и легкие. В зоне *A* конвейера расположены весы, сигнализатор которых срабатывает от тяжелой детали и нечувствителен к прохождению легкой. Легкие детали транспортируются без обработки к концу конвейера, где сваливаются в бункер *C*. При появлении тяжелой детали в зоне *A* конвейер останавливается и рука манипулятора, находящаяся в исходном положении, производит захват детали. Затем привод манипулятора переносит тяжелую деталь в зону *B*; одновременно включается лента конвейера. В зоне *B* после срабатывания соответствующего сигнализатора отпускается захват, после чего осуществляется запуск привода манипулятора в направлении начального состояния. Далее выявляется новая тяжелая деталь и процесс повторяется.

Рассмотрим вопросы спецификации АП, ограничившись пока только той ее частью, которая связана с транспортировкой тяжелых деталей. В данном случае можно выделить 11 компонент (для организации управления процессом, некоторые из них могут оказаться избыточными), которым соответствуют следующие истинные значения предикатов:

$p_1 = 1$ – лента конвейера в движении,

$p_2 = 1$ – тяжелая деталь в зоне *A*;

$p_3 = 1$ – включен привод сжатия пальцев руки манипулятора;

$p_4 = 1$ – пальцы руки манипулятора сжаты;

$p_5 = 1$ – привод манипулятора работает в сторону зоны B;

$p_6 = 1$ – рука манипулятора в зоне B;

$p_7 = 1$ – включен привод разжимания пальцев руки манипулятора;

$p_8 = 1$ – пальцы руки манипулятора отпустили деталь;

$p_9 = 1$ – привод манипулятора работает в сторону зоны A;

$p_{10} = 1$ – пальцы руки манипулятора окончательно разжаты;

$p_{11} = 1$ – тяжелая деталь на ленте конвейера.

Ситуации в этом примере представляют собой двоичные векторы $p_1, p_2, \ldots, p_{11}$. Из $2^{11} = 2048$ возможных векторов ситуациями являются лишь следующие 9 (перечисленные компоненты равны 1, не перечисленные равны 0):

s_1: p_1, p_9, p_{10}, p_{11};

s_2: p_2, p_9, p_{10};

s_3: p_3;

s_4: p_4;

s_5: p_5;

s_6: p_6;

s_7: p_7;

s_8: p_1, p_8, p_{10}, p_{11};

s_9: p_2, p_8, p_{10}.

Асинхронный процесс, включающий операции по захвату и препровождению тяжелых деталей из зоны A в зону B, включает ситуации $\{s_1, \ldots, s_6\}$, отношение F задает последовательность $s_1, \ldots, s_6$, инициаторами естественно считать ситуации s_1 и s_2 ($I=\{s_1, s_2\}$), а результантом – s_6 ($R=\{s_6\}$). Ситуации из множества $\{s_6, s_7, s_8, s_9, s_1, s_2\}$ образуют процесс репозиции, инициатором которого является ситуация s_6 ($I'=\{s_6\}$), а результантами – ситуации s_1 и s_2 ($R'=\{s_1, s_2\}$). Композиция АП и его репозиции дает автономный АП, представленный на рис. 1.9. В нем ситуации, представляющие инициаторы и результанты исходного процесса, затемнены, а ситуации процесса репозиции, не являющиеся его инициаторами и результантами, заштрихованы. Понятно, что суть процесса репозиции состоит в возврате руки манипуля-

тора в исходное состояние (зону A) и разжимании пальцев его руки. Интересно заметить, что, в отличие от исходного АП, его репозиция является недетерминированным процессом.

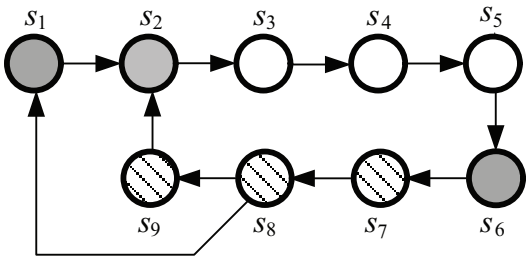


Рис. 1.9. Спецификация автономного асинхронного процесса транспортировки тяжелой детали на ленточном конвейере

Спецификацию АП, связанную с транспортировкой как тяжелых, так и легких деталей, читателю предлагается выполнить самостоятельно.

Способ II. Часто используемым вариантом структурирования является представление ситуаций АП в виде векторов, компоненты которых целочисленные (и, как правило, неотрицательные). Именно такое представление используется в диаграммах маркировок сетей Петри, речь о которых будет идти в гл. 2.

Способ III. В некоторых случаях структурирование ситуаций предполагает выделение входной, выходной и внутренней компонент или некоторых и них. Эти компоненты соответствуют тем событиям моделируемой системы, которые связаны с изменением значений сигналов на ее входах и выходах. При выделении как входной, так и выходной компоненты ситуация s_i представима упорядоченной тройкой $s_i = (x_j, y_k, z_l)$, где x_j – значение (символ) входной компоненты, $x_j \in X$, $1 \le j \le p$, y_k – значение (символ) выходной компоненты, $y_k \in Y$, $1 \le k \le q$, z_l – значение (символ) внутренней компоненты, не являющейся ни входной, ни выходной, $z_l \in Z$, $1 \le l \le n$.

В качестве примера рассмотрим таблицу переходов и выходов некоторого конечного автомата, скажем, заданного моделью Мили и имеющего трехсимвольный входной алфавит $\{x_1, x_2, x_3\}$, двухсимвольный выходной алфавит $\{y_1, y_2\}$ и два состояния $\{z_1, z_2\}$ (см. табл.1.1).

Задание конечного автомата. Таблица 1.1.

	x_1	x_2	x_3
z_1	z_1, y_1	z_2, y_2	z_2, y_2
z_2	z_1, y_1	z_1, y_2	z_1, y_1

Пусть исходно автомат находится в полном состоянии (ситуации) $x_1 z_1 y_1$. Это означает, что на вход его подан символ x_1, он находится во внутреннем состоянии z_1 и выдает на выход символ y_1. При подаче последовательности входных символов данный инициальный автомат осуществляет предписанные табл.1.1 переходы, которые должны следовать строго в следующем порядке:

1) изменение сигнала на входе (входной компоненты ситуации);
2) изменение сигнала на выходе автомата (выходной компоненты ситуации);
3) изменение внутреннего состояния автомата (внутренней компоненты ситуации).

При этом начало каждой фазы (изменение сигнала на входе) допускается после окончания предыдущей фазы (изменения внутреннего состояния).

В условиях этих ограничений последовательность входных символов $x_2 x_3 x_1$ должна в соответствии со спецификацией, заданной табл.1.1, порождать АП, показанный на рис. 1.10. В этом примере четко прослеживаются три фазы переходов при изменении каждого из входных сигналов. Моменты инициации изменений входных сигналов соответствуют пометкам на некоторых дугах рис. 1.10. Начальным состоянием автомата является s_0. В данном случае структурированный АП является автономным.

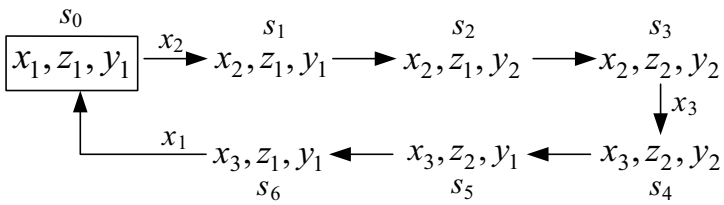


Рис. 1.10. Пример структурированного асинхронного процесса, порождаемого инициальным конечным автоматом, заданным табл. 1.1, при входной последовательности $x_2 x_3 x_1$

Асинхронная реализация конечного автомата сопряжена с рядом трудностей, которые требуют специального рассмотрения. Однако в данном пособии эти вопросы рассматриваться не будут. Тем не менее, приведенный пример свидетельствует, что выделение трех компонент структурированных ситуаций имеет определенный смысл.

В зависимости от семантики моделируемой системы вполне вероятны и другие способы представления ситуаций в структурированных АП. Так или иначе, **структурирование ситуаций предполагает выделение их компонент. Изменение значения компоненты является *событием* в АП. Возможность одновременного наступления нескольких событий в АП свидетельствует о наличии в нем параллелизма.** В предыдущем примере АП, порождаемый конечным автоматом, является чисто последовательным, но в общем случае параллелизм возможен. Примеры с параллелизмом будут рассмотрены далее. В неструктурированном АП параллелизм в явном виде не распознается, так как ситуации в нем представлены абстрактно без выделения (кодирования) событий. Тем не менее, наличие фрагментов типа *гамака*, образованного ситуациями s_2, s_4, s_5 и s_6 на рис. 1.1, косвенно свидетельствует о возможном параллелизме[2]. В этом гамаке возможны переходы s_2–s_4–s_6, s_2–s_5–s_6 наряду с "прямым" переходом s_2–s_6, в котором ситуации s_4 и s_5 не фиксируются, а значит, какие-то события происходят одновременно (параллельно).

1.5. Редукция

Если имеется полная спецификация моделируемой системы в виде АП, то обычно возникает задача усечения, т.е. выделение какой-либо интересующей нас части АП, или подпроцесса. Для этой цели используются различные виды операции редукции.

Редукция по инициатору

Пусть задан некоторый асинхронный процесс $P = <S, F, I, R>$ и некоторый инициатор $i^* \in I$, при активации которого функционирование процесса нас не интересует. Существенна только та часть АП (подпроцесс), которая имеет место при активации АП из инициаторов, принадлежащих множеству $I \setminus \{i^*\}$. Инициатор i^* будем в дальнейшем называть *несущественным*.

[2] Более подробно семантика гамаков, соответствующих либо параллелизму между переходами, либо случаю коммутативности таких переходов, изложена в гл.6.

Обратимся к АП, показанному на рис. 1.1. Пусть несущественным инициатором является i_3. Очевидно, что редукция по i_3 не должна содержать, прежде всего, эту вершину (ситуацию s_3) вместе с инцидентными ей входными и выходными дугами. Но исключение s_3 делает "висячей" вершину, соответствующую ситуации s_7. Она не является инициатором исходного АП, а поскольку после ее исключения АП не будет содержать входящих дуг, то по соглашению о назначении инициаторов, принятых при определении АП, s_7 должна быть объявлена инициатором. Это противоречие заставляет исключить s_7 из множества ситуаций редуцированного АП. Таким образом, редукция исходного АП, представленного на рис. 1.1, по несущественному инициатору i_3 приводит к АП, изображенному на рис. 1.11а.

Следовательно, редукция по несущественному инициатору сводится к исключению этого инициатора и всех отличных от инициаторов ситуаций, лежащих на ведущих из него в результанты траекториях, если эти ситуации не принадлежат траекториям, ведущим из оставшихся (существенных) инициаторов.

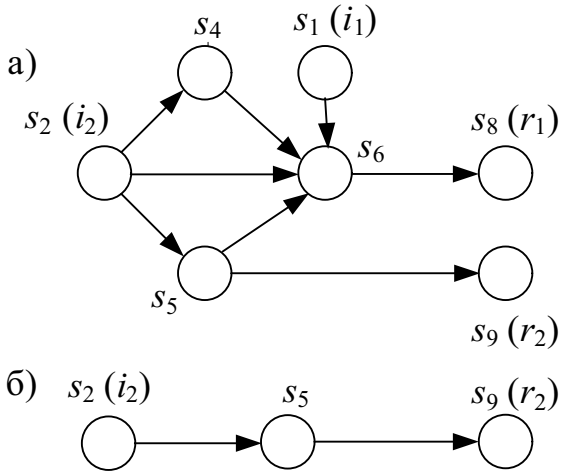


Рис. 1.11. Редукция АП, приведенного на рис. 1.1, а) по несущественному инициатору i_3, б) по несущественному результанту r_1

Если требуется выполнить редукцию по некоторому множеству несущественных инициаторов, то она может быть осуществлена последовательной редукцией по каждому инициатору, принадлежащему этому множеству, в любом порядке.

Редукция по результанту

Эта операция аналогична предыдущей и отличается лишь исключением в эффективном АП несущественного результанта r^* и всех "висящих" ситуаций, начиная с "хвоста" АП. Результат редукции АП, изображенного на рис. 1.1, по несущественному результанту r_1 показан на рис. 1.11б.

Редукция по входной компоненте ситуации

Пусть задан неприведенный асинхронный процесс $P = <S, F, I, R>$, ситуации которого структурированы так, что множество ситуаций S представимо упорядоченной тройкой $S = (X, Y, Z)$, где X, Y и Z – соответственно множества значений (символов) входной компоненты, выходной компоненты и внутренней компоненты.

Пусть требуется редуцировать АП по несущественной входной компоненте x_j. Множество ситуаций, которые содержат эту компоненту, обозначим через S^*. В общем случае $S^* \subset S$. Исключим ситуации S^* из АП. Их исключение, естественно, с необходимостью приводит к исключению инцидентных им входных и выходных дуг. Далее следует исключить все ситуации, оказавшиеся "висячими" (изолированными). Наконец следует исключить те ситуации, которые не относятся к инициаторам, и вновь образовавшиеся тупиковые "висячие" ситуации.

Редукция по множеству несущественных входных компонент осуществляется последовательной редукций по каждой из компонент в любом порядке.

Пример. Пусть асинхронный процесс структурирован, и его ситуации заданы двоичными векторами длины 4. АП изображен на рис. 1.12.

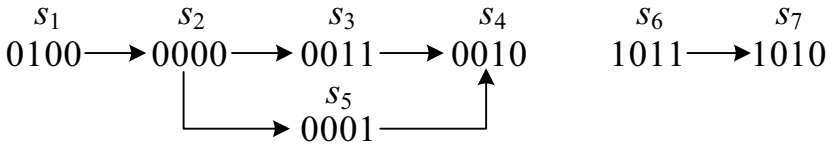


Рис. 1.12. Пример структурированного асинхронного процесса

Пусть $I = \{s_1, s_2, s_6\}$, $R = \{s_4, s_7\}$, а входной компоненте соответствуют два первых (слева) элемента векторов. Эта компонента принимает в данном случае три значения из четырех возможных {00, 01, 10}. Редукция по несущественной входной компоненте 00 осуществляется исключением ситуаций s_2, s_3, s_4 и s_5. При этом ситуация s_1 (инициатор) остается изолированной и по согла-

шению о правилах назначения инициаторов также должна быть исключена из состава редуцированного процесса. Поэтому искомой редукцией является АП, образованный единственным переходом s_6–s_7. Редукция по входной компоненте 01 приводит лишь к исключению инициатора s_1, поскольку s_2 также является инициатором. Если бы в исходном АП ситуация s_2 не являлась инициатором, то при редуцировании был бы исключен весь фрагмент, представленный ситуациями s_1, ..., s_5, и в результате остался бы единственный переход s_6–s_7.

Редукция по выходной или внутренней компоненте

Такого типа редукция структурированного АП производится аналогично редуцированию по несущественной входной компоненте. Пусть в приведенном выше примере (рис. 1.12) выходной компонентой исходного процесса объявлен крайний правый (четвертый) элемент y, принимающий значения из множества {0,1}. Редукция по несущественной выходной компоненте y=1, приводит, прежде всего, к исключению ситуаций s_3, s_5, s_6 и, как следствие, к исключению всех остальных ситуаций (изолированные ситуации не могут быть инициаторами и результантами). В результате получится “пустой” АП.

Если внутреннюю компоненту z образуют 2-й и 3-й разряды двоичного кода ситуаций (считая слева направо), то она принимает значения из множества {10, 00, 01}. Если несущественной компонентой объявлена компонента z^*= 01, то редукция по ней, прежде всего, исключает ситуации s_3, s_4, s_6 и s_7, а затем s_1, s_2 и s_5, так как s_5 не является результантом, вследствие чего траектория s_1–s_2–s_5 также должна быть исключена. Поэтому результатом редукции опять будет “пустой” АП. Если же, например, внутренней компонентой объявлены 2-й, 3-й и 4-й разряды кода ситуаций (в этом случае естественно считать, что входную компоненту образует 1-й разряд кода, а выходная компонента отсутствует) и редукция производится по компоненте z*=000, то исключение s_2 порождает необходимость исключения s_1, s_3, s_4 и s_5, и результатом редукции будет АП, содержащий единственную траекторию s_6–s_7.

До сих пор говорилось о редукции по компонентам для случая неприведенных процессов. Казалось бы, редукции приведенных и неприведенных процессов не отличаются друг от друга. Однако это не так. Дело в том, что редукция осуществляется с учетом некоторых семантических соображений, и поэтому для редукции приведенного процесса необходимо: 1) возвратиться к исходному процессу, 2) редуцировать его и только после этого 3) произвести приведение.

Поясним сказанное. Обратимся к примеру на рис. 1.1. Ситуации исходного (неприведенного) процесса закодируем двоичными векторами так, как показано на рис. 1.13.

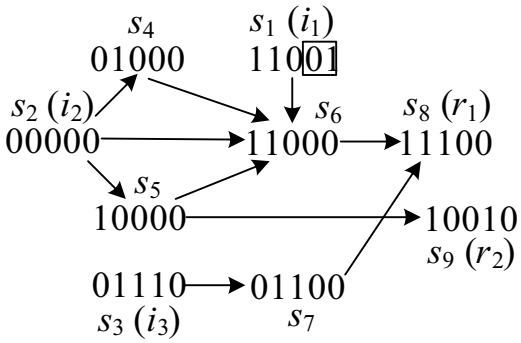


Рис. 1.13. Вариант структурированного АП, представленного на рис. 1.1

Пусть выходной компонентой y объявлены 4-й и 5-й разряды (слева направо) кодов ситуаций, а редукция осуществляется по выходной компоненте y^*=01 (отмечена квадратом на рис. 1.13). В результате этой редукции получается АП, который отличается от исходного отсутствием ситуации $s_1(i_1)$ и перехода s_1–s_6 (рис. 1.14).

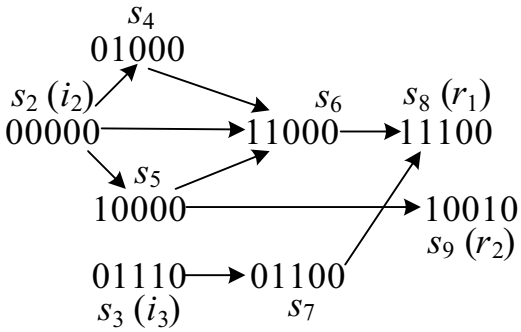


Рис. 1.14. Редукция асинхронного процесса на рис. 1.13 по выходной компоненте y= 01*

Полная репозиция редукции должна содержать траектории, ведущие из s_8 и s_9 в s_2 и s_3. Вариант полной репозиции редукции показан на рис. 1.15.

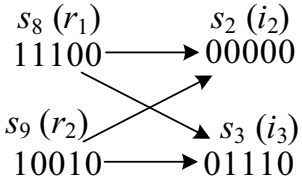


Рис. 1.15. Вариант репозиции редукции АП, представленной на рис. 1.14

Следует обратить внимание на то, что в АП и его редукции, изображенных на рис. 1.13 и рис. 1.14 соответственно, все непосредственные переходы кодируются соседними кодами, т.е. кодами, отличающимися лишь в одной позиции. Поэтому из семантических соображений вариант, представленный на рис. 1.15, является неудовлетворительным, так как имеет несоседние переходы. Поэтому желательно ввести в репозицию дополнительные ситуации, коды которых, во-первых, не должны совпадать с кодами ситуаций исходного процесса и, во-вторых, не должны содержать выходной компоненты y^*=01.

Выполнить такое кодирование в общем случае непросто, а иногда и невозможно, из-за чего приходится увеличивать число компонент исходного АП и таким образом модифицировать его. Введенные дополнительные компоненты соответствуют фиктивным событиям в АП. Не доводя пример до конца (это может служить упражнением), обратим внимание на еще одну возможность упрощения редукции. Она реализуется переназначением инициаторов и результантов. Объявим ситуации s_4, s_5, s_6 инициаторами i_4, i_5, i_6 соответственно. Это дает возможность использовать неполную репозицию, показанную на рис. 1.16.

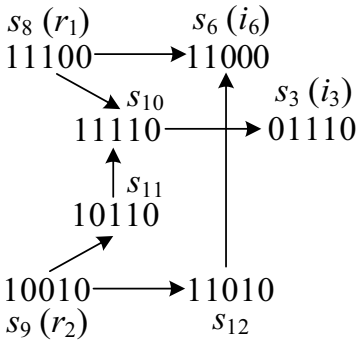


Рис. 1.16. Вариант неполной репозиции редукции АП, представленной на рис. 1.14

Репозиция здесь потребовала введения дополнительных ситуаций s_{10}, s_{11}, s_{12}, причем все переходы являются соседними. Результат приведения редукции исходного процесса (рис. 1.14) с его репозицией (рис. 1.16) изображен на рис. 1.17. Ситуации s_{10}, s_{11} и s_{12} должны быть отнесены к результантам r_3, r_4 и r_5 приведенного процесса.

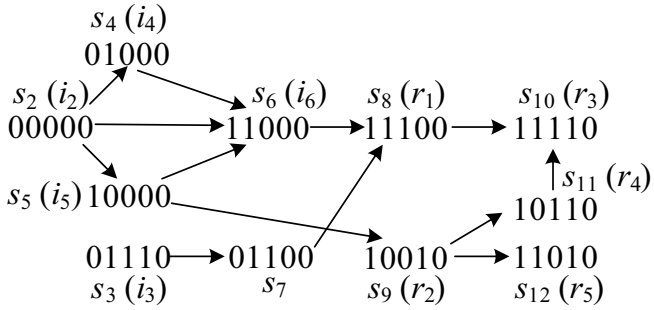


Рис. 1.17. Результат приведения редукции исходного АП, представленной на рис. 1.14

Если теперь обратиться к варианту приведения исходного асинхронного процесса (рис. 1.1), показанному на рис. 1.7, и закодировать все его ситуации, исключая дополнительные, теми же кодами, что и на рис. 1.13, а для дополнительных ситуаций s_{10} и s_{11} (рис. 1.7) использовать коды 11101 и 00010 соответственно, после чего произвести редукцию по той же выходной компоненте y^*=01, то ситуации s_1 и s_{10} будут исключены как содержащие эту компоненту. В результате будет получен асинхронный процесс, представленный на рис. 1.18, в котором все переходы имеют соседнее кодирование.

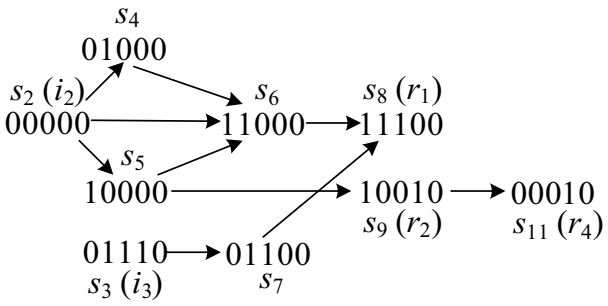


Рис. 1.18. Вариант "непосредственной редукции" приведенного АП, показанного на рис. 1.7

Сравнивая рис. 1.17 и рис. 1.18, можно убедиться в том, что они существенно отличаются друг от друга. Поэтому, во избежание ошибок, следует договориться о необходимости выполнения редукции приведенного процесса только путем использования перечисленных выше правил. Заметим, что применение этих правил все равно оставляет определенный произвол в выполнении репозиции, поскольку проектировщик должен пользоваться семантическими соображениями, которые не задаются спецификациями АП.

1.6. Композиция

Современные вычислительные машины и дискретные системы строятся на основе модульного (агрегатного) принципа. Аппаратные средства, равно как и программное обеспечение, выполняются в виде отдельных функциональных структурных единиц (модулей или IP-ядер). Последние объединяются (композируются, комплексируются, агрегатируются), образуя сложные системы.

Коль скоро введена модель асинхронного процесса, **необходим аппарат, который позволяет строить описания функционирования системы по описаниям ее фрагментов — аппарат композиции.** Такой аппарат основан на введении определенных операций над композируемыми асинхронными процессами и отличается от обычно используемых в автоматной проблематике методов композиции учетом динамических особенностей процессов.

Параллельная композиция

Говоря о параллельной композиции двух АП, имеем в виду, что заданы два асинхронных процесса P_1 и P_2 (составляющие композиции) и требуется построить такой асинхронный процесс P_3 (результат композиции), который воспроизводит (моделирует) **совместное функционирование** P_1 и P_2. Очевидно, что если составляющие композиции задаются четверками $P_1 = <S_1, F_1, I_1, R_1>$ и $P_2 = <S_2, F_2, I_2, R_2>$, то при их независимом функционировании результат композиции $P_3 = <S_3, F_3, I_3, R_3>$ должен быть задан таким образом, что:

а) $S_3 \subseteq S_1 \times S_2$, т.е. ситуации результата композиции представляются декартовым произведением ситуаций составляющих композиции, например, если S_1 и S_2 структурированы и содержат соответственно n и m компонент, то ситуации S_3 должны содержать $n+m$ компонент и иметь вид $s_k^3 : s_i^1 s_j^2$. Назовем s_i^1 и s_j^2 подситуациями s_k^3.

б) Если число ситуаций в P_1 и P_2 соответственно равно $|S_1|$ и $|S_2|$, то результат композиции P_3 должен иметь число ситуаций $|S_3| = |S_1| \times |S_2|$.

в) Инициаторами в $S_3 \subseteq S_1 \times S_2$ должны быть объявлены ситуации s_k^3, в которых обе подситуации s_i^1 и s_j^2 принадлежат инициаторам в P_1 и P_2.

г) Результантами в S_3 должны быть объявлены ситуации, в которых обе подситуации s_i^1 и s_j^2 принадлежат результантам в P_1 и P_2.

д) Пара ситуаций s_i^3 и s_j^3 процесса P_3 должна быть связана отношением $s_i^3 F s_j^3$, если для одной из подситуаций в составе АП P_1 или P_2 выполняется $s_i^1 F s_j^1$ или $s_i^2 F s_j^2$.

Пример "свободной" параллельной композиции P_3 процессов P_1 и P_2 представлен на рис. 1.19.

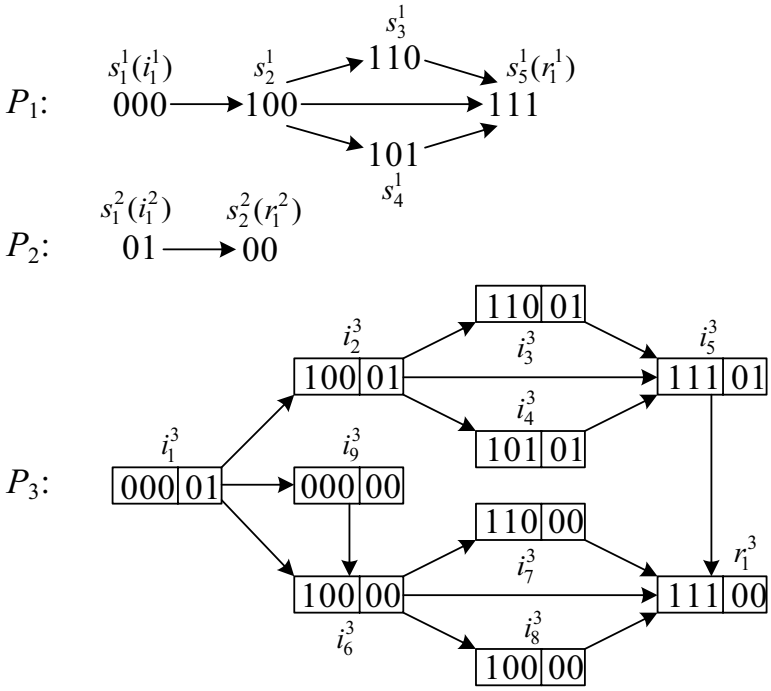


Рис. 1.19. Пример параллельной композиции двух асинхронных процессов

При параллельной композиции возможно, что композируемые асинхронные процессы пользуются общим ресурсом, в качестве которого может выступать, например, общая входная компонента ситуаций обоих процессов или общий инициатор. В этих случаях композируемые процессы функционируют уже не независимо, так как они связаны общим ресурсом.

Пусть ситуации S_1 и S_2 процессов P_1 и P_2 содержат входные компоненты X_1 и X_2 и некоторые значения X_1^{**} входной компоненты X_1 ($X_1^{**} \subseteq X_1$) объявлены эквивалентными (по семантике) значениям X_2^{**} входной компоненты X_2 ($X_2^{**} \subseteq X_2$), т.е. $X_1^{**} = X_2^{**} = X^{**}$.

Несущественные входные компоненты ситуаций S_1 и S_2 процессов P_1 и P_2 обозначим соответственно через X_1^* и X_2^*. Очевидно, что $X_i^* = X_i \setminus X^{**}$, $i = 1,2$. Допустим, что нас не интересует поведение композируемых АП при несущественных входных компонентах, и произведем редукции этих АП по X_1^* и X_2^* соответственно. Обозначим результаты таких редукций через $P_1(X^{**})$ и $P_2(X^{**})$. Далее можно произвести параллельную композицию процессов $P_1(X^{**})$ и $P_2(X^{**})$ по правилам, изложенным выше, имея в виду, что вместо ситуаций S_1 и S_2 в "свободной" композиции следует рассматривать ситуации $S_1(X^{**})$ и $S_2(X^{**})$. При этом, если ситуации $S_1(X^{**})$ и $S_2(X^{**})$ составляющих композиции представлены соответственно в виде $X^{**}Z_1Y_1$ и $X^{**}Z_2Y_2$, то ситуации S_3 результирующего АП представляются в виде $X^{**}Z_1Z_2Y_1Y_2$.

Пусть, например, на рис. 1.20а и рис. 1.20б заданы два процесса P_1 и P_2, причем входной компонентой в P_1 являются первые два левых разряда кодов ситуаций, а в P_2 – только первый разряд. Известно также, что входная компонента {01} ситуаций процесса P_1 эквивалентна входной компоненте {0} ситуаций процесса P_2. Осуществим редукцию P_1 по несущественным входным компонентам $X_1^* = \{00,10,11\}$ и сократим входную компоненту до одного разряда со значением {0}. Получим асинхронный процесс $P_1(X^{**})$, показанный на рис. 1.20в.

Редукция P_2 по несущественной входной компоненте $X_1^* = \{1\}$ дает асинхронный процесс $P_2(X^{**})$, представленный на рис. 1.20г. Результат композиции представлен на рис. 1.20д. Поскольку в ситуациях составляющих процессов не выделена выходная компонента, то ситуации результирующего процесса состоят из входной компоненты и конкатенации внутренних компонент ситуаций композируемых процессов.

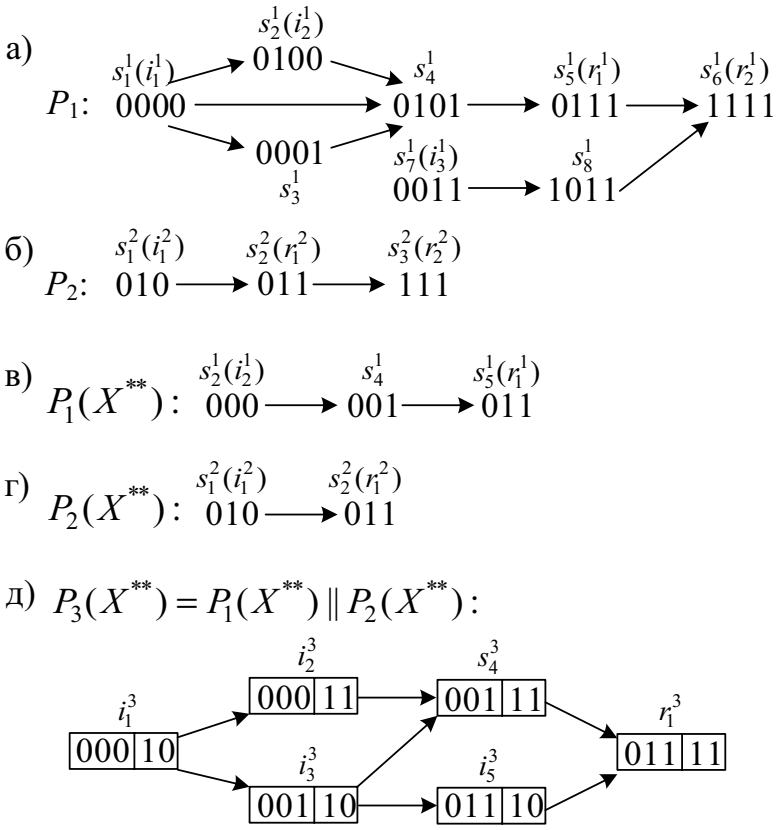


Рис. 1.20. Этапы параллельной композиции: а), б) исходные асинхронные процессы P_1 и P_2; в), г) редукции P_1 и P_2 по несущественным входным компонентам; д) параллельная композиция двух редуцированных асинхронных процессов

Последовательная композиция

Последовательная композиция двух процессов P_1 и P_2 предполагает их последовательное включение, при котором P_1 играет роль "головы" результата композиции P_3, а P_2 – роль его "хвоста". Процесс P_1 в составе композиции (после"отработки") инициирует P_2, а завершение функционирования последнего означает остановку P_3. Ясно, что особых требований к процессу P_1 в составе композиции не предъявляется, но P_2 должен "понимать" сообщения, выдаваемые P_1. Для этого требуется эквивалентность некоторых значений выходной компоненты ситуаций в P_1 соответствующим значениям входной компоненты ситуаций в P_2 или же эквивалентность некоторых результантов процесса P_1 соответствующим инициаторам процесса P_2. Однако эти требования не являются единственными. Дело в том, что для

активизации P_3 обе составляющие композиции P_1 и P_2 должны быть готовы к возбуждению своих инициаторов. Процесс P_1, отработав после инициации, окажется в некотором результанте, который должен быть воспринят процессом P_2 как инициатор. В свою очередь, процесс P_2, отработав, окажется в одном из результантов и, следовательно, оба процесса P_1 и P_2 не могут быть повторно активированы. Для того чтобы результирующий процесс P_3 был возобновляемым, его необходимо подготовить к повторной инициализации, что может быть сделано с помощью процедуры приведения процесса. Естественно еще одно требование: при функционировании P_2 в составе композиции в нем не должно возникать новых траекторий.

Итак, пусть заданы два асинхронных процесса $P_1 = <S_1, F_1, I_1, R_1>$ и $P_2 = <S_2, F_2, I_2, R_2>$. Ситуации моделирующего процесса $P_3 = <S_3, F_3, I_3, R_3>$ должны быть заданы таким образом, что $S_3 \subset S_1 \times S_2$, т.е. его ситуации также состоят из упорядоченных подситуаций, соответствующих ситуациям процессов P_1 и P_2.

Рассмотрим случай, когда результанты из некоторого подмножества R_1^{**} результантов R_1 процесса P_1 объявлены попарно эквивалентными ситуациям из соответствующего подмножества I_2^{**} инициаторов I_2 процесса P_2. В этом случае, очевидно, следует провести редукцию P_1 по результантам из множества $R_1^* = R_1 \setminus R_1^{**}$ и редукцию P_2 по инициаторам из множества $I_2^* = I_2 \setminus I_2^{**}$. Основанием для проведения этих операций является необходимость манипулирования только теми подпроцессами процессов P_1 и P_2, которые фактически участвуют в композиции. В результате будем иметь редукции $P_1(R_1^{**})$ и $P_2(I_2^{**})$ такие, что

$$P_1(R_1^{**}) = <S_1(R_1^{**}), F_1(R_1^{**}), I_1(R_1^{**}), R_1^{**}>,$$

$$P_2(I_2^{**}) = <S_2(I_2^{**}), F_2(I_2^{**}), I_2^{**}, R_2(I_2^{**})>.$$

Рассмотрим в качестве примера последовательную композицию процессов P_1 и P_2, заданных их протоколами. Оба асинхронных процесса, представленные на рис. 1.21, имеют по 3 инициатора и по 3 результанта. Там же приведен результат композиции. В этом примере предполагается, что r_2^1 и i_1^2, r_3^1 и i_2^2 попарно эквивалентны. Для построения протокола результата композиции произведены следующие операции:

1. Редукция P_1 по несущественному результанту r_1^1,
2. Редукция P_2 по несущественному инициатору i_3^2,
3. Склеивание пар эквивалентных ситуаций.

Теперь рассмотрим случай последовательной композиции двух АП $P_1 = <S_1, F_1, I_1, R_1>$ и $P_2 = <S_2, F_2, I_2, R_2>$. Пусть в ситуациях S_1 выделена выходная компонента Y_1, а в ситуациях S_2 – входная компонента X_2. И пусть все или некоторые значения Y_1^{**} выходной компоненты ситуаций из S_1 соответствуют всем или некоторым значениям X_2^{**} входной компоненты ситуаций из S_2. Тогда множества $Y_1^{*} = Y_1 \setminus Y_1^{**}$ и $X_2^{*} = X_2 \setminus X_2^{**}$ назовем несущественными значениями компонент Y_1 и X_2.

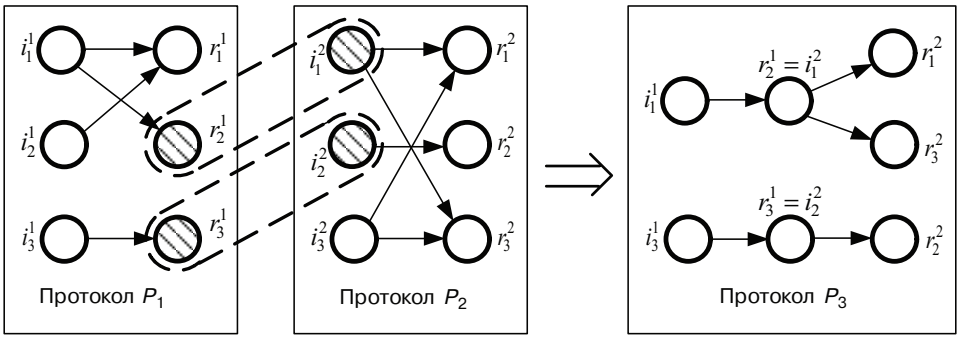


Рис. 1.21. Протокол последовательной композиции протоколов двух АП

Выполним редукции $P_1(Y_1^{**})$ и $P_2(X_2^{**})$ и построим процесс $P_3 = <S_3, F_3, I_3, R_3>$ такой, что:

1) его ситуации представимы в виде $S_3 \subset S_1(Y_1^{**}) \times S_2(X_2^{**})$, т.е. ситуации из S_3 представимы упорядоченными парами $s_k^{\,3} = (s_i^{\,1}, s_j^{\,2})$;
2) если в $s_k^{\,3}$ подситуация $s_j^{\,2} \in I_2(X_2^{**})$, то подситуация $s_i^{\,1} \in R_1(Y_1^{**})$.

Заключительным этапом последовательной композиции должно быть склеивание попарно эквивалентных значений компонент ситуаций S_3 процесса P_3. Если ситуации S_1 представлены в виде $X_1Z_1Y_1$, а ситуации S_2 – в виде $X_2Z_2Y_2$, причем $Y_1^{**} = X_2^{**}$, то ситуации S_3 процесса P_3 должны быть представлены в виде $X_1 Z_1 Y_1^{**} = X_2^{**} Z_2 Y_2$. Если ситуации S_1 и S_2 кодируются кодами длины n_1 и n_2 соответственно, выходная компонента ситуаций из S_1 содержит m_1 разрядов, а входная компонента ситуаций из S_2 содержит m_2 разрядов, то результирующие ситуации в S_3 содержат $n_1 + n_2 - max(m_1, m_2)$ разрядов.

Пример. На рис. 1.22а задан процесс P_1, в структурированных ситуациях которого выделена выходная компонента Y_1 (пятая и шестая позиция ситуаций слава направо), а на рис. 1.22б – процесс P_2 с выделенной входной компонентой X_2 (первая и вторая позиции ситуаций). Если $(Y_1^{**} \subseteq Y_1) = \{00,10\}$,

$(X_2^{**} \subseteq X_2) = \{00,10\}$ и $Y_1^{**} = X_2^{**}$, то последовательная композиция процессов P_1 и P_2 будет иметь вид, показанный на рис. 1.22в.

а) P_1: $0110\boxed{\overset{y_0^1}{00}} \longrightarrow 0110\boxed{\overset{y_1^1}{10}}$

б) P_2: $\boxed{\overset{x_1^2}{10}}0100 \longrightarrow \boxed{\overset{x_1^2}{10}}0110$

в) P_3: $0110\boxed{00}0100 \longrightarrow 0110\boxed{\overset{y_1^1 = x_1^2}{10}}0100 \longrightarrow 0110\boxed{\overset{y_1^1 = x_1^2}{10}}0110$

Рис. 1.22. Пример последовательной композиции двух процессов: а) первый процесс с выходной компонентой Y_1; б) второй процесс с входной компонентой X_2; в) результат последовательной композиции

Замыкание

Эта операция является одноместной и выполняется над приведенным процессом $P^n = <S^n, F^n, I, R>$, у которого по семантическим соображениям выделяются пары эквивалентных компонент ситуаций – входной и выходной – или пары эквивалентных инициаторов и результантов. Сначала рассмотрим замыкание по эквивалентным парам "результант–инициатор", принадлежащим соответственно множествам R^{**} и I^{**}. Тогда множества $I^* = I \setminus I^{**}$ и $R^* = R \setminus R^{**}$ являются несущественными инициаторами и результантами. Замыкание производится следующим образом:

1. осуществляется редукция P^n по несущественным инициаторам I^*,
2. проводится редукция P^n по несущественным результантам R^*,
3. склеиваются попарно эквивалентные ситуации из R^{**} и I^{**}.

Таким образом, **замыканием по эквивалентным парам "результант–инициатор" называется автономный асинхронный процесс** $P^3(I^{**}, R^{**}) = < S^3, F^3 >$, **удовлетворяющий ограничениям**:

1) $S^3 \subseteq S^n(I^{**}) \cap S^n(R^{**})$;

2) $F^3 = (F(I^{**}) \cup F'(I^{**})) \cap (F(R^{**}) \cup F'(R^{**}))$, где F и F' задают причинно-следственные связи для неприведенного процесса и его репозиции, соответствующих приведенному АП P^n;

3) пары эквивалентных ситуаций из I^{**} и R^{**} склеены.

Пример. Обратимся к асинхронному процессу и его приведенному виду, заданных соответственно на рис. 1.1 и рис. 1.7. Пусть $s_2 = s_{11}$. Для нахождения замыкания необходимо выполнить:

1) редукцию исходного процесса (рис. 1.1) по s_1 (i_1) и по s_3 (i_3) – см. рис. 1.23а;

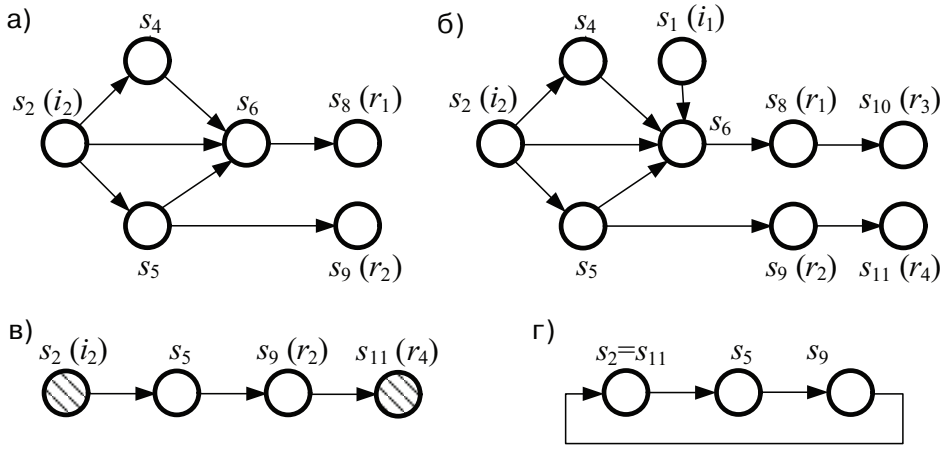


Рис. 1.23. Этапы замыкания приведенного процесса: а) редукция АП по несущественным инициаторам; б) приведение редуцированного процесса; в) редукция приведенного процесса по несущественным результантам; г) построение замкнутого (автономного) АП

2) построить репозицию редукции, осуществляемую через дополнительные результанты s_{10} (r_3) и s_{11} (r_4);

3) осуществить приведение объединения редуцированного процесса и его репозиции путем исключения дуг, ведущих из s_{10} и s_{11} в s_2 – рис. 1.23б;

4) редуцировать процесс, изображенный на рис. 1.23б, по несущественному результанту s_{10} – рис. 1.23в;

5) склеить эквивалентные ситуации s_2 и s_{11}, в результате чего получим автономный процесс, образующий замыкание – рис. 1.23г.

Замыкание по эквивалентным парам “входная компонента – выходная компонента” выполняется по существу аналогично предыдущей операции, хотя и имеет некоторые особенности. В приведенном АП выделяются множества

пар эквивалентных входных X^{**} и выходных Y^{**} компонент. Далее строится такой автономный процесс $P^3(X^{**}, Y^{**}) = < S^3, F^3 >$, что:

1) $S^3 \subseteq S^n(X^{**}) \cap S^n(Y^{**})$;
2) $F^3 = (F(X^{**}) \cup F'(X^{**})) \cap (F(Y^{**}) \cup F'(R^{**})) \cup F(X^{**}, Y^{**})$, где F и F' – те же обозначения, что были приняты ранее; X^* и Y^* – подмножества несущественных значений входных и выходных компонент, а через $F(X^{**}, Y^{**})$ обозначено множество дополнительных дуг, соединяющих ситуации–результанты из S^3 со значениями их Y^{**} выходной компоненты и ситуации–инициаторы из S^3 с эквивалентными значениями из X^{**} входной компоненты.

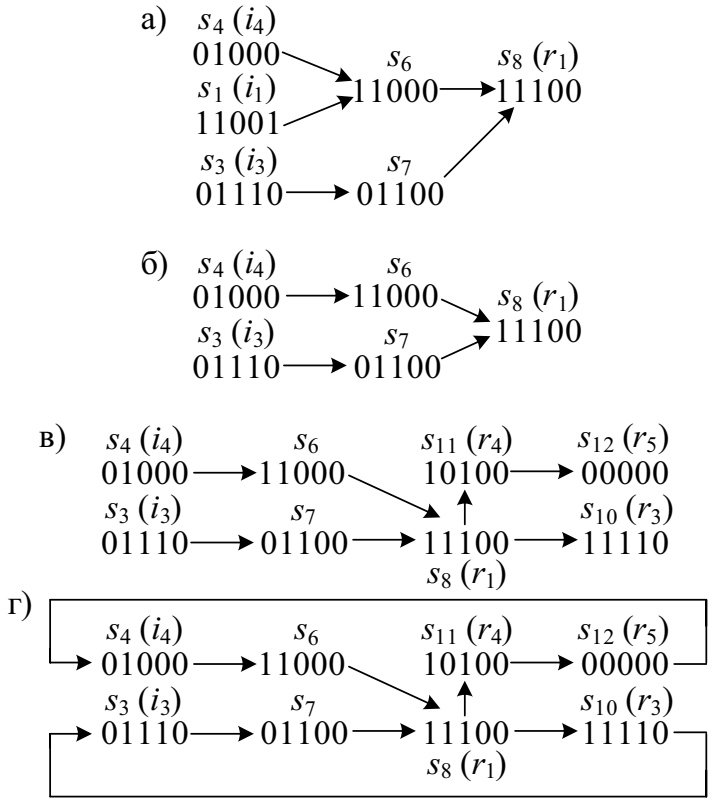


Рис. 1.24. Этапы замыкания АП, представленного на рис. 1.13 (ситуации s_4 и s_5 объявлены инициаторами): а) результат редукции по несущественным входным компонентам; б) результат редукции по несущественным выходным компонентам; в) построение приведенного процесса; г) построение автономного процесса путем замыкания

Обратимся для примера к структурированному асинхронному процессу, представленному на рис. 1.13. Объявим входной компонентой X первые два разряда кода ситуаций, а выходной компонентой Y – последние два разряда кода. Пусть по семантике X^{**} = {11---, 01---}, Y^{**} = {---10, ---00}, где прочерки расставлены, чтобы отличить входную компоненту от выходной, причем установлена эквивалентность следующих пар значений компонент: {11---}={---10}, {01---}={---00}. Пусть также ситуации s_4 и s_5 объявлены инициаторами АП. Редукция АП по несущественным значениям входной компоненты из X^*= {00---, 10---} приводит к асинхронному процессу, показанному на рис. 1.24а. Далее редуцируем этот процесс по несущественным значениям из Y^* = {---01, ---11} выходной компоненты. Получим АП, показанный на рис. 1.24б. Пусть его приведение выполнено так, как это показано на рис. 1.24в (появляются дополнительные ситуации–результанты s_{10}, s_{11}, s_{12}).

Заключительная фаза (рис. 1.24г) осуществляется проведением дуг, образующих переходы s_{10}–s_3 и s_{12}–s_4.

Введенные двуместные операции параллельной и последовательной композиции и одноместная операция замыкания образуют функционально полный базис, позволяющий манипулировать асинхронными процессами. Естественно, этот базис можно расширить за счет введения новых операций, но здесь не имеет особого смысла заниматься этими вопросами. Основная причина этого – универсальность модели АП, рассчитанной на возможность спецификации на ее языке широкого класса процессов различной физической природы и, что для нас самое важное, дискретных вычислительных и управляющих процессов.

1.7. Комментарий. Недостатки модели асинхронного процесса

Асинхронный процесс по существу является общей моделью описания динамики поведения параллельно функционирующих асинхронных систем. Эта модель задает допустимые последовательности действий над некоторыми объектами системы, каждой из которых соответствует некоторая траектория АП. В этом смысле можно говорить о том, что АП есть *модель управляющей структуры* системы. АП может служить инструментом исследования весьма общих свойств и закономерностей параллельных систем, независимо от семантики их работы. Поэтому АП можно считать *неинтерпре-*

тированной моделью. С другой стороны, асинхронный процесс можно понимать как *метамодель*, порождающую различные конкретные динамические модели (в том числе рассматриваемые далее).

Порождение частных моделей предусматривает использование механизма интерпретации асинхронных процессов, который может быть двухступенчатым:

1. **Первый этап** (назовем его *модельной интерпретацией*) состоит в том, что основным понятиям асинхронного процесса (ситуации, отношению следования, инициаторам и результантам) ставятся в соответствие понятия частных моделей, причем эта система соответствий базируется на введении дополнительных ограничений на АП. Примеры модельных интерпретаций АП рассмотрены далее.

2. **Второй этап.** Модельные интерпретации сами по себе могут оставаться неинтерпретированными моделями. Однако если асинхронному процессу или его модельным интерпретациям при решении конкретных задач поставить в соответствие некоторые метки, обозначающие символы, операторы, атрибуты или предикаты из заданного множества (пользуясь семантическими соображениями), то такие модели становятся интерпретированными. **Это вторая,** *предметная интерпретация* АП, которая должна быть согласована с приложениями и зависит от специфики решаемых задач. Конкретный случай предметной интерпретации, когда компонентам структурированных ситуаций ставились в соответствие некоторые предикаты, рассматривался в примере раздела 1.4.

Материал этой главы практически не содержит каких-либо строгих утверждений и носит скорее дескриптивный (т.е. описательный) характер. Это можно поставить в упрек авторам, которые, ратуя за строгую формализацию, по существу не используют ее в своих построениях. Тем не менее, мы ввели базовые понятия и определения, которые далее будут необходимы при изучении модельных интерпретаций асинхронных процессов. Таким образом, **основной задачей этой главы явилось задание метаязыка, с помощью которого методологически легче понять особенности частных моделей, их специфику, достоинства и недостатки.** Достоинства модели асинхронного процесса уже обсуждались. Возникает вопрос, каковы недостатки этой модели?

Недостатками модели асинхронного процесса являются, во-первых, ориентация на описание структур управления в ущерб описанию информационных потоков моделируемых систем и, во-вторых, то, что структурированные АП являются моделью в глобальных состояниях, а не событийной моделью, в которой взаимодействие компонент системы должно задаваться локально. Эти вопросы обсуждаются подробнее в последующих главах.

Перечисление недостатков модели АП можно продолжить. Остается пока совершенно неясным, как аппаратно или программно реализовать спецификацию в виде АП. Не обсуждался еще вопрос и о том, как установить, является ли спецификация, представленная на языке ситуаций и причинно-следственных отношений между ними, асинхронными АП. Не обсуждались алгоритмы характеризации АП, т.е. установления принадлежности АП определенным классам (скажем, эффективным, управляемым и простым АП). Не ставились задачи, связанные с установлением факта возможности одновременной (параллельной) реализации событий в структурированном АП.

Теоретической базой для решения большинства упомянутых выше задач является теория графов, но некоторые из задач решаются специфическими методами, которые будут описаны ниже. Некоторые из этих задач приведены в качестве самостоятельных упражнений. Если читатель не сможет выполнить эти упражнения сейчас, то не беда. Можно вернуться к ним после ознакомления с материалом последующих глав, где эти задачи, так или иначе, обсуждаются применительно к частным модельным интерпретациям АП. Процесс обучения не обязан быть ациклическим, и хотя структура курса (пособия) предполагает дедуктивную технологию обучения (от общего – к частному), индуктивный подход (от частного – к общему) может оказаться не менее продуктивным.

1.8. Упражнения

1. Сформулируйте минимально необходимый набор формальных правил назначения инициаторов и результантов.

2. Объясните, почему в графе АП отсутствуют петли, т.е. дуги, являющиеся одновременно входящими и исходящими для какой-либо вершины–ситуации.

3. На рис. 1.25 задан АП с тремя инициаторами и тремя результантами.

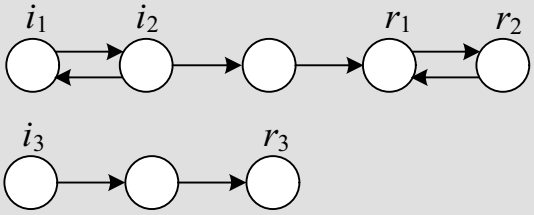


Рис. 1.25. Пример асинхронного процесса

Можно ли сказать, что процесс после инициации раньше окажется:

а) в r_1, чем в r_2;
б) в r_2, чем в r_1;
в) в r_3, чем в r_1;
г) в r_3, чем в r_2.

Обоснуйте свой ответ.

4. Укажите, выполняется ли свойство наблюдаемости заданного АП, т.е. может ли внешний наблюдатель указать, через какой промежуток времени после инициации АП он достигнет результанта.

5. На рис. 1.26 задан АП с одним инициатором и двумя результантами. Может ли АП одновременно оказаться в r_1 и r_2? Обоснуйте свой ответ.

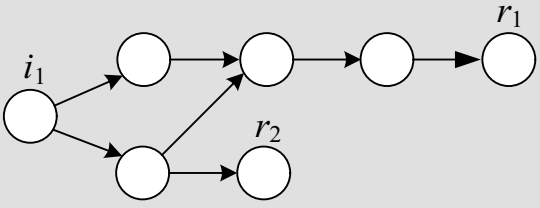


Рис. 1.26. Пример асинхронного процесса

6. Приведите пример автономного АП, описываемого несвязным орграфом. (*Связным* называют орграф, в котором существует путь из любой вершины во все остальные).

7. *Диссипативным* называют процесс, который, начавшись в некотором инициаторе, затем функционирует так, что его траектории охватывают некоторое подмножество его ситуаций. Приведите пример эффективного и управляемого диссипативного АП.

8. Докажите, что композиция АП и его полной репозиции представляет собой связный орграф. Имеет ли место это утверждение, если репозиция не является полной? Приведите пример.

9. Приведите пример АП, неполная и полная репозиция которого имеют один и тот же протокол.

10. Постройте спецификацию автономного АП, описывающего процесс согласованного переключения ламп уличного светофора на перекрестке двух магистралей. Циклическая последовательность переключения цветов в каждом из направлений должна быть такой: красный – желтый – зеленый – желтый – красный – ... Учтите требования правил дорожного движения.

11. Имеются два стержня I и II, вращающиеся соответственно вокруг осей O_1 и O_2 против и по часовой стрелке и приводимые в движение двигателями, имеющими разные угловые скорости вращения (рис. 1.27). Расстояние между осями O_1 и O_2 меньше суммарной длины стержней и поэтому существует опасная зона, в которой возможно столкновение стержней и, следовательно, аварийная ситуация, сопровождаемая разрушением стержней или выходом из строя двигателей. Опасная зона на рисунке заштрихована. Для безопасной работы положение стержней фиксируется контактами 1, 3 до входа в опасную зону и 2, 4 после выхода из нее. При одновременном замыкании контактов 1 и 3 один из двигателей должен быть остановлен, а при замыкании контакта 2 или 4 стержнем, вышедшим из опасной зоны, вновь включен.

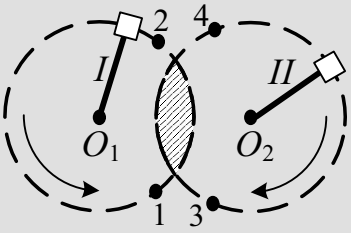


Рис. 1.27. Электромеханическая система, для которой следует построить спецификацию на языке асинхронных процессов

Постройте спецификацию асинхронного процесса для управления безопасной работой системы. Начальное положение стержней примите произвольным, либо введите дополнительные контакты, фиксирующие определенное начальное положение стержней.

12. Может ли иметь место, чтобы в ситуациях структурированного процесса не выделялась:

а) входная компонента,

б) выходная компонента,

в) внутренняя компонента,

г) как входная, так и выходная компоненты,

д) как выходная, так и внутренняя компоненты,

е) входная и внутренняя компоненты.

Укажите семантику таких АП и их соответствие классам конечных автоматов.

13. Приведите пример АП, ситуации которого (по семантике примера) заданы как целочисленные векторы.

14. Укажите, в каком случае приведенный процесс совпадает с исходным асинхронным процессом.

15. Выполните редукцию АП, показанного на рис. 1.1:

а) по несущественному инициатору i_2,

б) по несущественному результанту r_2,

в) по несущественному инициатору i_3 в предположении, что ситуация s_7 объявлена инициатором i_4 исходного АП.

16. Пусть в АП, показанном на рис. 1.4, $I = \{s_1, s_2, s_3, s_4\}$, $R = \{s_7, s_8, s_9, s_{10}\}$.

Выполните редукцию этого АП: а) по несущественным инициаторам $\{s_1, s_2\}$, б) по несущественным результантам $\{s_7, s_9\}$.

17. Докажите или опровергните, что редукция эффективного процесса является эффективным процессом.

Докажите или опровергните, что редукция управляемого процесса является управляемым процессом.

Докажите или опровергните, что редукция частично приведенного процесса может быть полностью приведенным процессом, а редукция полностью приведенного процесса может оказаться частично приведенным или даже неприводимым процессом.

18. Выполните приведение АП, показанного на рис. 1.13, в предположении, что ситуации s_4, s_5 и s_6 не объявлены инициаторами.

19. Предложите приемлемый вариант назначения инициаторов и результантов АП, графы которых приведены на рис. 1.28 и рис. 1.29. Выполните для этих двух процессов:

а) параллельную композицию,

б) последовательную композицию, считая “головой” композиции АП, изображенный на рис. 1.28,

в) последовательную композицию, считая “головой” композиции АП, изображенный на рис. 1.29,

г) замыкание одного из АП.

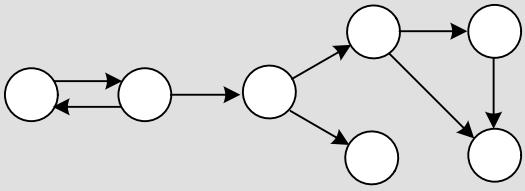


Рис. 1.28. Пример графа асинхронного процесса

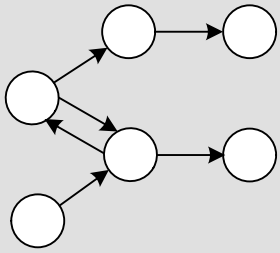


Рис. 1.29. Пример графа асинхронного процесса

20. Структурируйте ситуации АП в предыдущем упражнении и выполните указанные в нем пункты для случая композиций по компонентам ситуаций.

21. Попытайтесь ввести понятие отношения включения одного асинхронного процесса другим.

22. Приведите примеры операций над АП, приводящие к его расширению.

23. Формализуйте операцию "реверса" исходного АП, после выполнения которой процесс функционирует от результантов к инициаторам. Приведите примеры. Как определить реверс на автономном АП?

24. По определению АП и в соответствии с принятыми соглашениями о назначениях инициаторов и результантов каждая из траекторий АП должна начинаться в некотором инициаторе и заканчиваться в некотором результанте. Сформулируйте задачу определения принадлежности АП данной спецификации как модификацию задачи достижимости: ситуация s_j достижима из s_i, если выполняется $s_i F^* s_j$, т.е. существует траектория, ведущая из s_i в s_j. Предложите (или найдите в литературе) алгоритм решения задачи.

25. Предложите алгоритм проверки данного АП на принадлежность классу эффективных АП.

26. Предложите алгоритм проверки данного АП на принадлежность классу управляемых АП.



Глава 2.

СЕТИ ПЕТРИ

Тятя, тятя, сети Петри притащили мертвеца.
(Из студенческого фольклора).

Среди множества моделей, ориентированных на решение разнообразных задач моделирования, анализа и синтеза процессов различной природы (в том числе вычислительных процессов и процессов управления), большим вниманием пользуются так называемые сети Петри (ударение на первом слоге). Исследователей привлекают такие достоинства сетей Петри, как возможность представления динамики функционирования процессов, отражения свойств недетерминированности, асинхронности и параллелизма процессов, простота синтаксиса и наглядность модели.

В настоящее время теория сетей Петри (СП) превратилась в самостоятельную научную дисциплину и области ее применения непрерывно расширяются. Развитие некоторых направлений теории сетей Петри мотивируется ее внутренними потребностями, но основным стимулом являются запросы практики и, в первую очередь, практики построения вычислительных, информационных и управляющих систем и комплексов.

Теория СП во многих своих аспектах теснейшим образом связана с теорией автоматов и, по существу, является развитием последней. Это обстоятельство позволяет с единых методологических позиций рассмотреть вопросы моделирования процессов и явлений, присущих программному обеспечению, аппаратной поддержке и информационному обмену, при проектировании вычислительных систем. В данном разделе такие вопросы будут центральными. В то же время следует заметить, что область возможного применения аппарата сетей Петри значительно шире и не исчерпывается проблематикой этой книги.

2.1. Ретроспектива сетей Петри

Научная традиция часто называет фундаментальное понятие, концепцию, или теорию именами своих выдающихся представителей. Для примера сошлемся на Абелевы группы, постоянную Планка или числа Фиббоначи. В информатике эта традиция встречается реже (как то, например, машина Тьюринга, алгоритмы Дейкстра или Флойда или АВЛ-деревья (аббре-

виатуры фамилий Адельсона-Вельского и Ландиса, который на западе называют “алгоритмом трех русских”). Немецкий ученый **Карл Адам Петри** (*Carl Adam Petri, 1926 - 2010*) является одной из немногих исключительных личностей нашего времени: вероятно, каждый профессионал знаком с понятием предложенной Карлом Петри парадигмы, получившей название “теория сетей Петри”. Начала этой теории были заложены в диссертации Петри, представленной в Дармштадтский Технический Университет в 1961 году и защищенной в следующем году. Кстати, Дармштадт получил в 1997 году официальный статус “Города наук”. После публикации диссертации (вскоре переведенной на английский и использованной во многих проектах, в частности, в проекте MAC Массачусетского Технологического института) , она цитировалась часто, хотя (вероятно) была прочитана много реже. Это была не обычная диссертация доктора философии, содержащая решение открытой проблемы или развивающая старую теорию. Вместо этого, как и во многих более поздних публикациях, Петри представил новые фундаментальные идеи и предложения о пересмотре основ информатики. Они начали реализоваться на практике намного позже, а многие из них только начнут питать новые открытия.

Мотивируя новый подход, Петри начинает с конкретной практической проблемы, касающейся вычисления рекурсивных функций. К тому времени было уже известно, что для общей рекурсивной функции f и аргумента n, количество необходимых ресурсов для вычисления $f(n)$, не может быть оценено заранее. Поэтому надо начать вычисления с имеющимся набором ресурсов. Если их хватает, вычисление $f(n)$ удачно заканчивается. В противном случае надо запросить больше ресурсов и начать снова. Петри с самого начала бросил вызов потребности “перезапуска”. Обычная компьютерная архитектура терпит неудачу, потому что каждый новый запрос ресурсов увеличивает время выполнения операций за счет увеличения необходимого числа тактовых импульсов генератора (часов), что, в свою очередь, требует увеличения тактовой частоты и расхода энергии. Частота часов и расход энергии не могут быть изменены по желанию и могут привести к отказу системы.

Поэтому возникает вопрос, существует ли какая-нибудь “растяжимая” архитектура, которая свободна от принципиально синхронных архитектур. Петри доказал, что это может быть достигнуто: надо, грубо говоря, чтобы каждый компонент системы функционировал автономно, причем система должна работать в асинхронном режиме. В качестве технико-

экономического обоснования, Петри спроектировал асинхронное запоминающее устройство магазинного типа, в котором каждый модуль общается лишь с его двумя соседями. Известно, что два таких устройства достаточны, чтобы смоделировать машину Тьюринга. Свои результаты Петри представил на первой компьютерной конференции IFIP в Мюнхене в 1962 году. Петри намеревался показать, что асинхронные системы мощнее синхронных систем. Отсюда следует, что общая теория обработки информации в принципе должна базироваться отнюдь не на последовательных моделях, а на *асинхронных* моделях.

Петри сформулировал три важнейших требования для моделирования асинхронных распределенных систем:

- **Первое требование.** Модель должна отвечать законам физики. В частности, нет смысла рассматривать модели, оперирующие с глобальными состояниями. Дискретное событие в системе обычно не затрагивает все компоненты системы, а только некоторые из них. События не должны происходить в соответствии с вымышленной, идеализированной шкалой времени, а должны отвечать локальным причинно-следственным отношениям.
- **Второе требование.** Элементарные дискретные действия должны подчиняться законам сохранения, в том числе обратимости процессов. В информатике это означает, что переход из состояния S к состоянию S' предполагает, что от состояния S' в результате вычислений можно перейти обратно к S. Простой пример: оператор присваивания $x := x+1$ обратим, тогда как $x := 1$ не обратим.
- **Третье требование.** Техника моделирования должна быть удобной для пользователей вычислительных систем.

Развитие проблематики сетей Петри с 1960-х годов до 1980 года

В начале 60-х годов компьютерная техника была представлена огромными универсальными ЭВМ для численных вычислений, в которых в качестве устройств ввода-вывода информации использовались сначала перфорированная бумажная лента, а потом перфокарты. Глобальная тактовая система казалась естественной. Никто не думал о компьютерных сетях. Прикладное программное обеспечение ориентировалось на числовые проблемы. Базовое программное обеспечение было главным образом написано

в ФОРТРАНЕ и АЛГОЛЕ 60. Началось систематическое исследование в области теории последовательных программ и последовательных процессов. Предложение Петри сделать ввод и вывод параллельными процессами не было замечено. Теоретическая информатика сосредоточилась на последовательных моделях для автоматов и компьютеров, формальных языках и теории компиляторов. Спекулятивные статьи по кибернетике и искусственному интеллекту циркулировали, главным образом, без каких-либо реалистических идей относительно их реализации. Предложения Петри были слишком ранними для прикладной информатики. И теория сосредоточилась на других темах.

Первым крупным достижением было использование некоторых из идей Петри в проекте MAC в конце 60-х. Это стимулировало начало исследований по характеризации классов Сетей Петри с помощью формальных языков. Множество теоретических результатов в области линейно-алгебраических методов для проблематики сетей Петри появилось в 1970-х. В работах по распределенным системам уделялось больше внимания сетям Петри и методам их анализа, включая временную (темпоральную) логику.

Развитие проблематики сетей Петри с 1980-х годов до 2000 года

Число печатных изданий по сетям Петри резко возросло, начиная с ранних 1980-х. По существу, все теоретические результаты и применения, доступные в поздних 1970-х, были собраны и представлены в первом курсе о сетях Петри в Гамбурге, изданном в начале 1980-х.

Что принесет будущее?

В своей речи на 26-й международной конференции по приложениям и теории сетей Петри в Майами, 2005, Петри оценил разнообразие и качество приложений его теории и призвал к существенному их расширению. Он признал, однако, что долгосрочная программа исследований связана с разработкой новых аппаратных средств, а также абстрактных и программных средств обеспечения интерфейсов. Петри предложил работать над ревизией понятия информации. Ранее, по его мнению, информатика развилась бурно и без плана, будучи ведомой ее технологическим и экономическим потенциалом. Очевидно, что прогресс в информатике затрагивает всю структуру научного развития. Это мнение истинного реформатора, отмеченного мно-

гими наградами, в том числе наградой “Пионер компьютерной отрасли“ общества IEEE в 2009 году.

Изложенный выше в этой главе материал в некоторой степени является намеренным плагиатом. Авторы предпочли “предоставить слово” двум современникам Карла Адама Петри, являющимися одновременно свидетелями и соучастниками развития проблематики сетей Петри. Им виднее. То, что вы прочли выше, не просто перевод с английского на русский статьи “Wilfried Brauer, Wolfgang Reisig. Carl Adam Petri and Petri Nets”, опубликованной в 2006 году. Скорее это краткое авторизованное изложение части тезисов оригинальной статьи.

Столь краткое введение в парадигму сетей Петри читатель может расширить, проштудировав нижеследующую главу и выполнив упражнения. Нырнуть в мир связанных с СП вопросов можно с помощью Интернета. Но будьте осторожны: запрос поиска “Petri nets” в Google меньше чем за минуту выдаст 3 миллиона 650 тысяч ссылок. Для облегчения поиска ссылок на интересующую вас информацию ориентируйтесь на следующее расписание.

60-е годы – основы событийных моделей, связь с теорией автоматов.

70-е годы:

- первые попытки использования сетей Петри для моделирования компьютерных архитектур и схем,
- создание базовой теории структурных подходов, алгебр сетей Петри.

80-е годы:

- расширение сетей Петри до высокого уровня, развитие цветных (раскрашенных) сетей,
- создание программного инструментария для анализа достижимости и других свойств сетей Петри,
- сигнальные графы переходов (STGs), новые результаты в рамках семантических теорий.

90-е годы:

- существенное улучшение инструментария анализа СП, основанное на бинарных диаграммах решений (binary decision diagrams),
- развитие model checking (метод верификации программных систем),

- разработка инструментария стохастических сетей Петри,
- теория регионов (регион – это множество состояний в графе состояний, которые соответствуют помеченному месту в сети Петри), позволяющая извлекать параллелизм из последовательных поведенческих описаний,
- новые подходы к проектированию программного обеспечения, протоколов информационного обмена, автоматического управления,
- развитие гибридных сетей Петри, создание специализированного пакета моделирования сетей Петри (CPN Tool),
- установление связей сетей Петри с моделью параллельных процессов CSP Хоара и алгебрами процессов,
- сведение графов достижимости к сжатым графам с неявным отображением частичных порядков с целью уменьшения вычислительной сложности процедур анализа,
- развитие теории асинхронных схем, создание программного инструмента Petrify для анализа сетей Петри и синтеза самосинхронных схем.

00-е годы характеризуются массовым использованием сетей Петри в новых областях:

- моделирование задач бизнеса,
- process mining (спецификация, стандартизация и улучшение процессов, увеличение их продуктивности),
- в химии, биологии, медицине и др.,
- новые идеи в подходах к синтезу сетей Петри, интерактивные средства для новейших графических пакетов и т.д.

В настоящее время сети Петри вносят свой вклад в проекты, связанные с созданием программного обеспечения. Примером может служить удачное применение компонентов сетей Петри в UML 2 (унифицированный язык моделирования), где наиболее заметны использование диаграмм деятельности и их автоматическая трансляция в сети Петри и обратно.

Для исследования моделирующей способности (точнее, функциональных возможности) сетей Петри (СП) и сравнения возможностей СП с возможностями других известных в теории автоматов объектов (машин Тьюринга, конечных автоматов и т.д.) прибегают к интерпретации СП.

Функционирование сетей Петри порождает некоторые последовательности (траектории) срабатывания событий (называемых в СП переходами). Если поставить в соответствие каждому переходу СП некий символ из заданного алфавита, данная сеть порождает некоторые строки, каждая из которых соответствует одной возможной траектории, начинающейся с начального символа. Говорят, что множество строк, соответствующих данной СП, образует ее язык. Тогда множество всевозможных СП, удовлетворяющих указанным требованиям, определяет языки Петри, короче, СП-языки.

Мощность класса СП-языков в определенной степени позволяет судить о функциональных и моделирующих возможностях сетей Петри. Конечно, этот подход по сути носит чисто последовательный характер, не учитывая существенные особенности сетей Петри – параллелизм и асинхронность.

Результаты исследований показали, что **СП-языки по своей мощности не уступают контекстно-свободным языкам, класс контекстно-зависимых языков включает класс СП-языков, а язык регулярных выражений и автоматных языков суть подклассы СП-языков.** Более подробно о классификации формальных языков, носящей название иерархии Хомского (по имени ее изобретателя), посвящен раздел 2.7.

Таким образом, **говоря обиходным языком, можно утверждать, что сети Петри закрывают некоторую теоретическую «брешь» между конечными автоматами и машиной Тьюринга.**

2.2. Описание модели

Сеть Петри – это двудольный ориентированный граф с двумя типами вершин (позициями и переходами, изображаемыми соответственно кружками и полочками), дугами, соединяющими позиции с переходами и переходы с позициями, и начальной маркировкой, которая представляется вектором.

Пример такого двудольного ориентированного графа приведен на рис. 2.1.

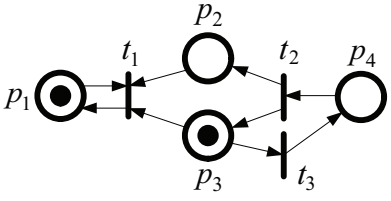


Рис. 2.1. Пример сети Петри

На этом рисунке позиции обозначены символами p_i, а переходы символами t_j. Направленные дуги могут соединять позиции только с переходами, а переходы только с позициями, так что любая позиция может быть входной или (и) выходной позицией одного или нескольких переходов. Некоторые позиции помечаются маркерами, или фишками, которые изображаются в виде жирных точек, помещаемых внутри позиции. В общем случае позиция может содержать несколько точек. Такая разметка позиций называется начальной маркировкой сети.

Число компонент вектора начальной маркировки равно числу позиций, а значение каждой компоненты представляется либо нулем, если в соответствующей позиции отсутствует маркер, либо целыми неотрицательным числом, равным числу маркеров – жирных точек, помещаемых внутри позиции.

Динамика в сети Петри описывается перемещением маркеров по сети в соответствии с соглашением о правиле срабатывания перехода, носящем локальный характер. Переход считается возбужденным, если каждая из его входных позиций содержит, по крайней мере, по одному маркеру. Через некоторое время (заранее не заданное) возбужденный переход срабатывает. Срабатывание перехода считается неделимой операцией, в результате которой одномоментно выполняется изъятие по одному маркеру из каждой входной позиции перехода и добавление по одному маркеру в каждую его выходную позицию. Сеть Петри функционирует, переходя от одной маркировки к другой посредством срабатывания возбужденных переходов. Поэтому ее поведение характеризует диаграмма маркировок – орграф, вершинами которого являются маркировки, а дуги помечены наименованиями срабатывающих переходов.

Формально сеть Петри (СП) определяется как четверка вида

$$< P, T, E, \mu^0 >,$$

где: P – конечное множество *позиций* (иначе – *мест* или *условий*); T – конечное множество *переходов* (или *событий*); E – конечное множество *дуг*, $E \subseteq P \times T \cup T \times P$; μ^0 – *начальная маркировка* (*вектор разметки*), $\mu^0 : P \to N$, где $N = \{0,1,2,...\}$ – множество неотрицательных целых чисел.

Множества входных и выходных позиций перехода $t_j \in T$ будем обозначать $I(t_j)$ и $O(t_j)$ соответственно. Аналогично, записи $I(p_j)$ и $O(p_j)$ суть обозначе-

ния множеств переходов, являющихся соответственно входом и выходом позиции $p_j \in P$.

Если обозначить через I и O объединения $I(t_j)$ и $O(t_j)$ по всем переходам $t_j \in T$, т.е. $I = \bigcup_T I(t_j)$, $O = \bigcup_T O(t_j)$, $P = I \bigcup O$, то сеть Петри может быть задана также в виде

$$< P, T, I, O, \mu^0 >.$$

Представление сети Петри в виде двудольного графа позволяет задать структуру сети Петри *статически*. Как было сказано ранее, *динамика* вносится в модель с помощью механизма смены маркировки (разметки) позиций и соглашением о правиле срабатывания (реализации) переходов.

Начальная маркировка, как она была определена выше, приписывает позициям сети Петри некоторые целые числа (в том числе нуль). **На графе маркировка задается так: в соответствующих кружках рисуют определенное число точек (жирных), носящих название маркеров (фишек)**.

Еще раз подробно проговорим принцип функционирования сетей Петри: передвижение маркеров по сети Петри осуществляется посредством *срабатывания ее переходов*. Сработать может только *возбужденный* (возможный) переход, т.е. такой переход $t_k \in$ T , во всех входных позициях $I(t_k)$ которого содержится хотя бы по одному маркеру (для каждого такого перехода и для каждой позиции $p_s \in I(t_k)$ имеет место $\mu_s \geq 1$, где через $\mu_s = \mu(p_s)$ обозначено число маркеров в позиции p_s). Срабатывание перехода может произойти через *любой конечный промежуток времени* после возбуждения перехода. Результатом срабатывания является изъятие из всех входных позиций сработавшего перехода по одному маркеру и добавление во все его выходные позиции по одному маркеру. Срабатывание перехода считается *неделимым актом*, т.е. предполагается, что изъятие маркеров из всех входных позиций и их перемещение во все выходные позиции осуществляется мгновенно, с нулевой задержкой. Однако в некоторых приложениях соглашение о неделимости может быть нарушено.

Соглашение о правиле срабатывания переходов отражает *концепцию причинно-следственной* (казуальной) связи: **после того, как известные нам объективные причины (условия) какого-либо события созрели, это событие может наступить.** Однако время его наступления заранее неизвестно, поскольку возможно существование других, неизвестных нам или несущественных для нас причин, которые, в конечном счете, вместе с существенными причинами неизбежно обусловят наступление интересующего

нас события. После наступления события возникнут новые условия – следствия, которые, в свою очередь, могут стать причинами последующих событий. При этом первоначальные причины могут либо исчезать, либо сохраняться. Последнее возможно как за счет того, что ресурс маркеров полностью не исчерпан, так и за счет регенерации маркеров (когда позиции являются одновременно входными и выходными).

На рис. 2.2 показаны четыре фрагмента, иллюстрирующие правила изменения маркировок для каждой позиции сети Петри (еще раз напомним, что эти правила являются локальными).

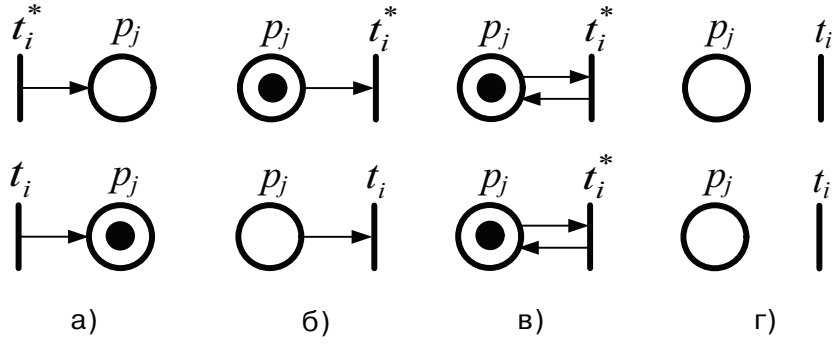


Рис. 2.2. Правила смены маркировок

На рис. 2.2а позиция p_j является выходной позицией возбужденного перехода t_i^* (возбужденный переход отмечен звездочкой), т.е. $p_j \in O(t_i)$. В этом случае текущая маркировка $\mu(p_j)$ позиции p_j (т.е. маркировка, имеющая место до возбуждения перехода t_i) в результате срабатывания возбужденного перехода t_i^* заменяется маркировкой $\mu'(p_j)$, причем имеет место $\mu'(p_j) = \mu(p_j) + 1$ (т.е. добавляется в точности один маркер). На рис. 2.2б позиция p_j является входной позицией возбужденного перехода t_i^*, т.е. $p_j \in I(t_i)$. В результате срабатывания t_i^* $\mu'(p_j) = \mu(p_j) - 1$ (из p_j изымается один маркер).

На рис. 2.2в позиция p_j является одновременно входной и выходной позицией возбужденного перехода t_i^*, т.е. $p_j \in O(t_i) \bigcap I(t_i)$, и после его срабатывания $\mu'(p_j) = \mu(p_j) - 1 + 1 = \mu(p_j)$ (сохраняется прежняя маркировка позиции). Наконец, на рис. 2.2г позиция p_j не инцидентна переходу (не связана с неоходом) t_i, т.е. имеет место $p_j \notin O(t_i) \bigcup I(t_i)$. Переход t_i не является возбужденным, сработать не может, и маркировка тоже сохраняется: $\mu'(p_j) = \mu(p_j)$.

Во фрагменте, представленном на рис. 2.3,

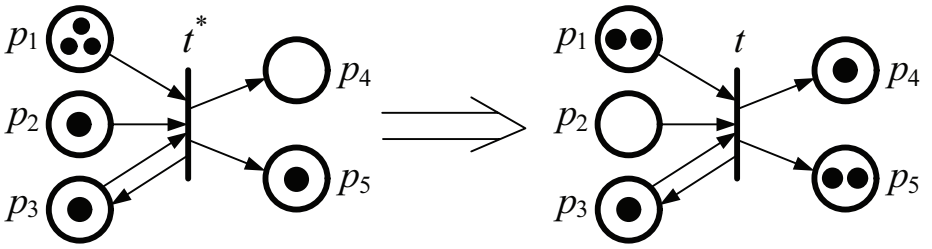


Рис. 2.3. Результат срабатывания перехода

$$I(t) = \{p_1, p_2, p_2\}, \quad O(t) = \{p_4, p_5, p_3\}, \quad i(p_1) = 3,$$

$$i(p_2) = 1, \quad i(p_3) = 1, \quad i(p_4) = 0, \quad i(p_5) = 1,$$

переход t возбужден (t^*), потому что все его входные позиции содержат, по крайней мере, по одному маркеру, и после его срабатывания исходная маркировка перейдет в маркировку, показанную в правой части рисунка:

$$i'(p_1) = 3 - 1 = 2, \quad i'(p_2) = 1 - 1 = 0, \quad i'(p_3) = 1 - 1 + 1 = 1,$$

$$i'(p_4) = 0 + 1 = 1, \quad i'(p_5) = 1 + 1 = 2.$$

Переход t не может сработать вторично, так как одна из его входных позиций (p_2) пуста (не содержит маркеров).

Еще раз подчеркнем: **срабатывание какого-либо возбужденного перехода в общем случае вызывает смену маркировки сети Петри.** При этом может измениться маркировка входных и выходных позиций только возбужденного перехода. Маркировка позиций, являющихся одновременно входными и выходными для данного перехода, не изменяется после его срабатывания.

Текущая маркировка сети Петри может пониматься как *состояние* сети Петри в данный момент времени. Срабатывание возбужденного перехода, являющееся *локальным актом*, в общем случае ведет к изменению маркировки сети и, следовательно, ее глобального (полного) состояния.

Сеть Петри функционирует, переходя от одной маркировки к другой в результате срабатывания возбужденных переходов. Таким образом, можно считать, что динамика поведения СП (ее функционирование во времени) может быть описана тройкой $< \mu^0, \rightarrow, M >$, где μ^0 – начальная марки-

ровка; $\rightarrow$ – отношение непосредственного следования маркировок (запись $\mu^u \rightarrow \mu^v$ означает, что за маркировкой μ^u непосредственно следует маркировка μ^v); M – множество допустимых маркировок (состояний) сети Петри, достижимых из μ^0.

Маркировка может изображаться вектором $\mu = (\mu_1, \mu_2, ..., \mu_n)$, число компонент которого равно числу позиций СП, а значение i-й компоненты, $1 \le i \le n$, есть натуральное число $\mu_1 = \mu(p_i)$ – число маркеров в i-й позиции.

Рассмотрим пример сети Петри, представленный на рис. 2.4, с семью позициями $P = \{p_1, p_2, ..., p_7\}$ и пятью переходами $T = \{t_1, t_2, t_3, t_4, t_5\}$. Эта сеть Петри может быть описана следующим образом: $I(t_1) = \{p_1, p_2\}$, $O(t_1) = \{p_3, p_4\}$, $I(t_2) = \{p_3\}$, $O(t_2) = \{p_5\}$, $I(t_3) = \{p_3, p_6\}$, $O(t_3) = \{p_5, p_6\}$, $I(t_4) = \{p_4\}$, $O(t_4) = \{p_7\}$, $I(t_5) = \{p_7\}$, $O(t_5) = \{p_2\}$.

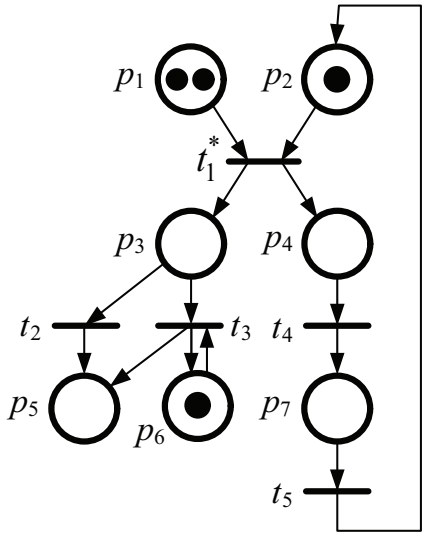


Рис. 2.4. Пример сети Петри

В векторе ее начальной маркировки $\mu^0 = \{\mu_1^0, \mu_2^0, ..., \mu_7^0\}$ только три ненулевых компоненты: $\mu_1^0 = 2$, $\mu_2^0 = \mu_6^0 = 1$. Таким образом, начальная маркировка представлена вектором $\mu^0 = (2100010)$. При такой начальной маркировке имеется лишь один возбужденный переход t_1^*. После его срабатывания изымается по одному маркеру из входных позиций p_1 и p_2 и в его выходные позиции p_3 и p_4 помещается по одному маркеру. Таким образом, из начальной маркировки $\mu^0 = (2100010)$ непосредственно следует мар-

кировка $\mu^1 = (1011010) : \mu^0 \to \mu^1$. При этой маркировке возбуждены переходы t_4, t_2 и t_3. Из двух последних может сработать только один, так как срабатывание любого из них снимает возбуждение другого. Время срабатывания возбужденного перехода в модели не оговаривается: оно считается произвольным, хотя и конечным. При срабатывании t_2 или t_3 в данном примере будет достигнута одна и та же маркировка $\mu^2 = (1001110)$, а при срабатывании t_4 образуется маркировка $\mu^3 = (1010011)$. Возможно одновременное срабатывание t_2 и t_4, t_3 и t_4. В этом случае состояние сети Петри будет определяться маркировкой $\mu^4 = (1000111)$.

Некоторые авторы отвергают возможность одновременного срабатывания двух и более возбужденных переходов, ссылаясь на его малую вероятность, однако такая точка зрения, будучи удобной в некоторых отношениях, противоречит концепции независимости поведения сетей Петри от времен срабатываний переходов.

2.3. Диаграмма маркировок

Для задания динамики поведения сетей Петри обычно используют так называемые диаграммы достижимых состояний (маркировок), для краткости именуемые просто диаграммами маркировок.

Диаграмма маркировок **представляет собой орграф, вершинами которого являются маркировки из множества M достижимых маркировок, а дуги направлены из маркировки μ^a в маркировку μ^b, если $\mu^a, \mu^b \in M$ и существует непосредственный переход от μ^a к μ^b ($\mu^a \to \mu^b$).**

Заметим, что $\mu^a \to \mu^a$ означает наличие петли в вершине диаграммы маркировок. Дуги, если это необходимо, помечаются обозначениями тех переходов, срабатывание которых является причиной смены маркировок.

Диаграмма на рис. 2.5 соответствует сети Петри, изображенной на рис. 2.4. Пометки вида $(t_2 \oplus t_3)\cdot t_4$ в данном случае означают, что одновременно срабатывает одна из пар (t_2, t_4) или (t_3, t_4) переходов. Таким образом, диаграмма отражает возможный параллелизм в соответствующем ей процессе, представимым СП. Множество допустимых маркировок (состояний) в данном случае состоит из 16 маркировок, причем из выделенной начальной марки-

ровки μ^0 достижима лишь одна результирующая (тупиковая) маркировка $\mu^r = (0100210)$, из которой СП не может выйти, что гарантирует однозначное завершение процесса. По диаграмме, в частности рис. 2.5, можно определить множество последовательностей срабатываний переходов (траекторий процесса). Одна из возможных траекторий такова: $t_1t_3t_4t_5t_1t_4t_5t_2$. Нетрудно видеть, что развитие рассматриваемого процесса не является детерминированным, так как возможны и другие его реализации (другие траектории).

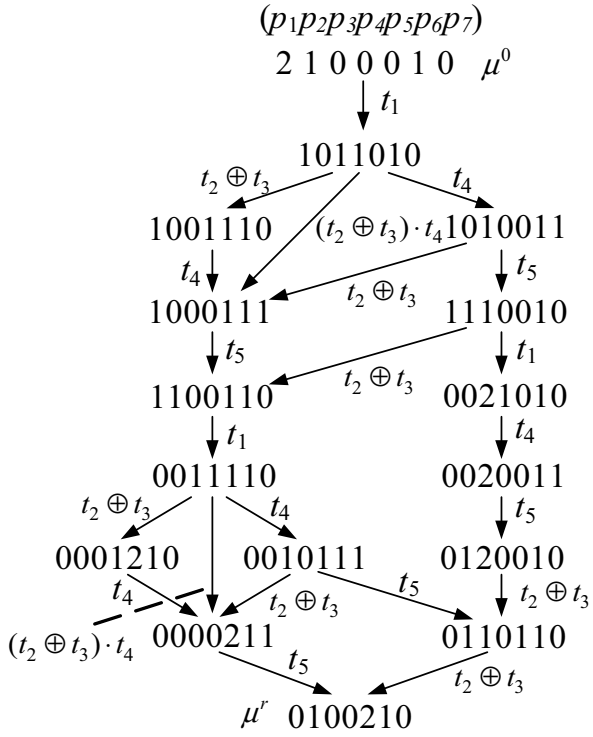


Рис. 2.5. Диаграмма маркировок для сети рис. 2.4

В общем случае одной начальной маркировке могут соответствовать несколько результирующих маркировок. Достижение каждой из них возможно при соответствующем распределении времён срабатываний переходов. Именно *асинхронность* процессов, отражаемая в модели отсутствием каких-либо указаний о времени срабатывания переходов (предполагается лишь, что они конечны), гарантирует недетерминизм модели. Этот *недетерминизм* может проявляться двояко: в общем случае неизвестно, по какой траектории будет идти развитие процесса (как он будет реализован) и в каком заключительном состоянии он окажется в финале. Недетерми-

низм является весьма полезным свойством модели, поскольку он может отражать наше незнание отдельных несущественных деталей рассматриваемого процесса и, более того, позволяет игнорировать их.

Возможность *синхронизации* и параллелизм также являются неотъемлемыми особенностями СП. Так, уже из рис. 2.4 видно, что только при синхронизации условий p_1 и p_2 (наличие хотя бы одного маркера как в позиции p_1, так и в позиции p_2) возможно срабатывание перехода t_1, а далее события из пар (t_2, t_4) или (t_3, t_4) могут произойти одновременно (параллельность). Заметим, что в диаграмме синхронизм проявляется в фрагментах типа "гамаков", пример которых демонстрирует четверка маркировок (0011110), (0001210), (0010111), (0000211) в левом нижнем углу рис. 2.5, а параллельность – в фрагментах, замыкаемых дугами с пометками типа $(t_2 \oplus t_3)\cdot t_4$.

Срабатывание какого-либо перехода в сети Петри может трактоваться как *событие*. Событие является локальным поведенческим актом. В таком случае маркировка входных позиций какого-либо перехода, вызывающая его возбуждение, может пониматься как *условие (предусловие)* потенциального наступления соответствующего события, а маркировка выходных позиций перехода после его срабатывания – как *следствие (постусловие)* его наступления.

Отношения между событиями

Если говорить о срабатывании какой-либо пары переходов t_i и t_j сети Петри, то оно подчиняется одному из трех фундаментальных отношений:

1) переход t_j следует за переходом t_i (записывается как $t_i \rightarrow t_j$, см. рис. 2.6а), что отражает *последовательность* соответствующих событий, их причинно–следственную связь;

2) может сработать либо переход t_i, либо переход t_j, но не оба вместе (отношение типа исключительного ИЛИ, записывается как $t_i \oplus t_j$, см. рис. 2.6б), такое отношение отражает свободный *выбор* (*конфликт* срабатывания двух переходов, или *недетерминизм*);

3) могут сработать оба перехода, как переход t_i, так и переход t_j, в любой последовательности или оба вместе (записывается как $t_i \parallel t_j$, см. рис. 2.6в), упорядочения этих событий нет; такое отношение выражает *непоследовательность*, возможно *одновременность* или *параллельность*, предполагающую перекрытие времен срабатываний переходов.

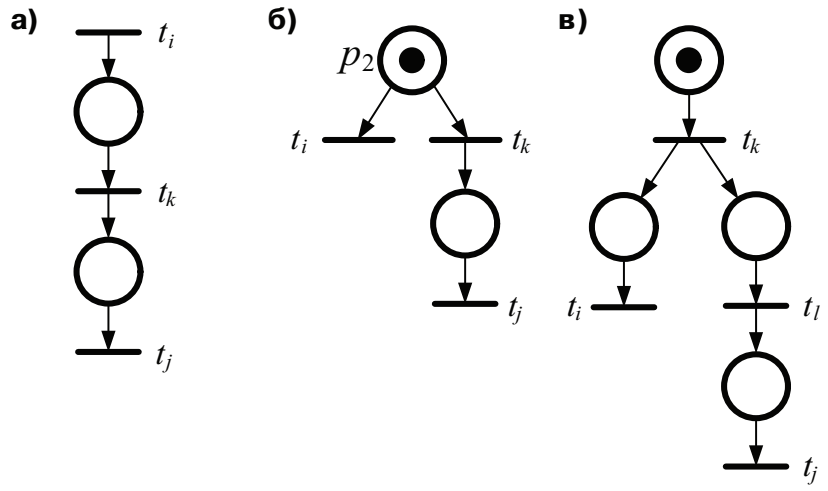


Рис. 2.6. Примеры различных отношений между переходами сети Петри

Таким образом, базовые понятия недетерминизма и параллельности могут быть выявлены в рассматриваемой модели и на локальном уровне.

Сделаем попутно два замечания.

1. Представление маркировок сети Петри в виде целочисленного вектора с числом компонент, равным числу позиций, естественно, не является единственным. На практике часто представляют маркировки в виде списка позиций, в которых число маркеров не равно нулю. Если позиция содержит $k > 1$ маркеров, то в этом списке для компактности перед обозначением позиции помещают множитель k. В этой форме записи выполнена диаграмма, показанная на рис. 2.7, множество достижимых маркировок которой совпадает с множеством маркировок в эквивалентной диаграмме рис. 2.5.

2. Дуги диаграммы маркировок, изображенной на рис. 2.5, которые помечены обозначениями вида $(t_2 \oplus t_3)\cdot t_4$, очевидно, являются избыточными: они не несут никакой полезной информации, так как новые маркировки при одновременном срабатывании нескольких (в данном случае пары) переходов появиться не могут. Вследствие этого обычно в диаграммах маркировок такие дуги опускают, что и сделано в диаграмме на рис. 2.7.

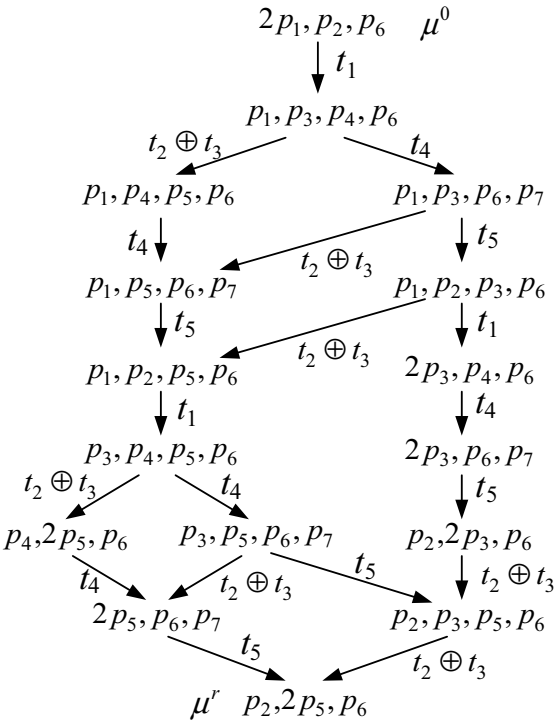


Рис. 2.7. Альтернативное представление диаграммы маркировок

Сети Петри как модельная интерпретация асинхронных процессов

Сеть Петри, порождающая конечное множество M маркировок, представляет собой модельную интерпретацию асинхронного процесса (см. гл.1). Покажем это. Ситуациями в сети являются начальная маркировка μ^0 и все достижимые из нее маркировки. Отношение F толкуется как отношение непосредственного следования маркировок. Очевидно, что на множестве M можно, как и ранее, определить отношение эквивалентности маркировок и задать отношение для классов эквивалентных маркировок. Если для некоторой сети существует единственный класс эквивалентности, то соответствующая СП является автономной, в которой отсутствуют входные и выходные позиции. Если классов эквивалентности больше одного, то можно (из семантических соображений) выделить подмножество маркировок–инициаторов, в которое всегда входит начальный класс эквивалентности (ему, в частности, принадлежит начальная маркировка μ^0), и подмно-

жество маркировок–результантов, в которое обязательно входит хотя бы один заключительный класс.

Работа реальных объектов часто описывается не единственной сетью Петри, а некоторым множеством сетей, имеющих одинаковую структуру (заданную тройкой $< P, T, E >$), **но разные начальные маркировки**. Способ сведения множества таких сетей Петри к одной понятен из рис. 2.8. При срабатывании одного из дополнительных переходов t_0' или t_0'' устанавливается маркировка $p_1, 2p_2, p_3$ или p_3, p_4, остальные позиции пусты. Читатель, видимо, обратил внимание на допущенную авторами некорректность: изображенная на рис. 2.8 сеть, строго говоря, не является сетью Петри, так как переход t_0' и его выходную позицию p_2 связывает не одна дуга, а две дуги. Однако в разд. 3.3 будет показано, что использование для задания СП мультиграфа вместо графа допустимо. В качестве упражнения читателю предлагается изобразить модификацию этой конструкции, предполагающую использование обычных (ординарных) СП, т.е. сетей без мультидуг.

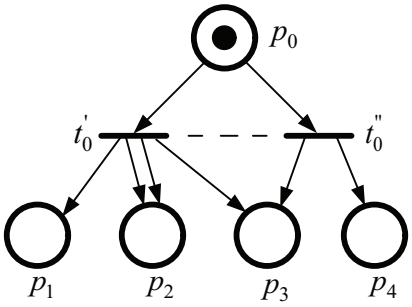


Рис. 2.8. Правило склеивания сетей Петри с одинаковыми графами

Два начала сетей Петри: статическое и динамическое

Очевидно, что класс сетей Петри, порождающих бесконечное множество маркировок, мощнее в функциональном отношении, чем АП, ибо АП определяется на конечном множестве ситуаций. Оценка функциональных возможностей сетей Петри будет дана ниже, в разд. 2.7.

Здесь важно подчеркнуть следующее. **Сеть Петри как модель несет в себе два начала – статическое и динамическое. Статическая компонента (двудольный орграф с начальной маркировкой пози-**

ций) аккумулирует в себе (содержит в "закодированном" виде) исчерпывающую информацию о динамике (поведении) описываемой ею системы. Динамическая компонента (диаграмма маркировок) представляет собой модель в полных (глобальных) состояниях и порождается непосредственно статической компонентой (срабатываниями возбужденных переходов сети в соответствии с принятыми соглашениями). Диаграмма достижимых маркировок – одна из возможных интерпретаций АП со структурированными ситуациями. Интерес специалистов по вычислительным наукам к сетям Петри вызван тем, что спецификация на языке сетей Петри, как правило, более компактна по сравнению с моделями в полных состояниях и, кроме того, она составляется на основе локальных причинно-следственных отношений, присущих исследуемому объекту. Полная информация о поведении системы может быть извлечена из сети Петри путем порождения диаграммы маркировок и анализа положительных реакций объекта и/или поведенческих аномалий, свидетельствующих о неблагоприятных эффектах, вызываемых некорректностью спецификации. Этот анализ может выполняться формально (о чем будет сказано ниже). Он является необходимым этапом проектирования управляющих и вычислительных структур (или программ).

2.4. Классификация сетей Петри

Зачем нужна классификация сетей Петри

Каждый человек, даже не обязательно причастный к информатике, знаком с такими понятиями, как техническое задание и спецификация. Если кому-то потребуются их уточнение, рекомендуем обратиться к Википедии. Здесь мы остановится лишь на некоторых моментах, предваряющих понимание необходимости классификации сетей Петри.

Техническое задание (ТЗ) – это исходный документ на проектирование любого сколь-нибудь сложного объекта независимо от его природы. ТЗ регулирует отношения между заказчиком и исполнителем и по существу является юридическим документом, устанавливающим ответственность сторон. Грамотное ТЗ позволяет значительно сократить время и средства для выполнения проекта, избежать существенных ошибок с обеих сторон. Отсутствие ТЗ приводит к казусам.

Известен анекдотичный случай, когда некая фирма пригласила выдающегося изобретателя Эдисона с просьбой починить электрогенератор. Он вооружился молотком и выстукивал генератор по всему периметру, пока не указал место, где нужно заменить обмотку. После замены машина заработала, и счастливый владелец фирмы вручил Эдисону чек, попросив его самому написать запрашиваемую сумму. По тем времена 100 долларов было чуть ли не целым состоянием, и заказчик попросил исполнителя объяснить, почему его услуги так дороги. На этот вопрос Эдисон ответил просто: сама процедура стоит 1 доллар, а факт, что я знаю, как починить агрегат – 99 долларов.

Надо отметить, что великий Томас Алва Эдисон (1847-1931) был самоучкой, обладал несравненной интуицией, но никогда не использовал математики, к которой он потерял вкус, не осилив в юности “Математические начала натуральной философии” Ньютона. Вот слова выдающегося изобретателя: “... у меня перебывало множество математиков. Все они оказались никчемными специалистами... Математика всегда шла следом за экспериментами, а не перед ними.... Я нанимаю математиков, а не они меня».

В наше время трудно следовать такому подходу вообще, и в информатике в частности. Обладая математическими знаниями, Эдисон предложил бы миру в сотню раз больше изобретений. Но не все родились гениями.

В информатике чаще используется не ТЗ, а так называемая *формальная спецификация* – математическое описание программной или аппаратной системы, которая может быть реализована в соответствии с этим описанием и является моделью системы. Использование (формальных) методов *верификации* и *валидации* (обоснованности) исходной модели позволяет определить, удовлетворяет ли она или нет требованиям спецификации.

Поскольку в данном контексте речь идет о сетях Петри, естественно возникает вопрос: как верифицировать исходное задание данной сети Петри на отсутствие нежелательных эффектов поведения реализуемой системы *до ее реализации* или убедиться в их отсутствии? Пожалуй, самым эффективным ответом на этот вопрос является попытка классификации сетей Петри. Почему? Идея простая: определить принадлежность модели (спецификации) тем или иным классам сетей Петри. Кроме того, поскольку в этой главе речь идет о теории сетей Петри, то может ли существовать такая (как и всякая другая) теория без классификации? Более того, как показывает история наук, правильная классификация часто порождает новые теории.

Итак, о каких классах идет речь? В приложениях если все входные позиции данного перехода содержат, по крайней мере, по одному маркеру, то этот факт интерпретируются как соблюдение определенного условия возбуждения этого перехода, а его срабатывание – как событие. Поэтому каждый фрагмент сети Петри соответствует некоторой причинно-следственной связи в спецификации.

Наличие специфических свойств модели в виде СП порождает разнообразие видов поведения (множество различных диаграмм). Задача типизации СП связана со следующими эффектами их поведения. Поскольку в некоторых позициях сетей возможно накопление бесконечного числа маркеров, то множество допустимых состояний (маркировок) может не быть конечным. В сети Петри может не существовать тупиковых маркировок (такого рода сетями моделируются циклические или автономные процессы). Наконец, в цикл могут входить состояния только из некоторого подмножества допустимых состояний. Эти особенности также вызывают необходимость классификации сетей Петри.

Различные СП могут отличаться друг от друга не только структурами графов, но и диаграммами смены маркировок (состояний, а также множествами допустимых состояний. Исследование модели, какой является СП, связано с изучением ее статических и динамических свойств. **Классификация СП базируется на качественных и количественных ограничениях, накладываемых на допустимые конфигурации (статические ограничения) и возможные типы маркирования (динамические ограничения)**.

Классификация по динамическим ограничениям

Дадим определения некоторых, представляющих особый интерес классов сетей Петри, порождаемых динамическими ограничениями.

Сеть Петри называется:

- **k-ограниченной** (k ≥ 1 – целое число), если на множестве ее достижимых состояний не найдется ни одной позиции $p_i \in P$, для которой $\mu(p_i) > k$ (в которой при функционировании сети Петри появилось бы более k маркеров);
- **безопасной**, если она 1-ограничена (ни в одной ее позиции не может появиться более одного маркера);

– **ограниченной**, если найдется такое k, для которого она k-ограничена;

– **1-консервативной**, если в процессе функционирования сети Петри общее число маркеров в ней остается постоянным, т.е. для любого $t_r \in T$ имеет место $\sum_{p_i \in I(t_r)} \mu(p_i) = \sum_{p_j \in O(t_r)} \mu(p_j)$;

– **консервативной**, если существует положительная целочисленная функция $f : P \to N$ такая, что для любого перехода $t_r \in T$ имеет место $\sum_{p_i \in I(t_r)} f(p_i) = \sum_{p_j \in O(t_r)} f(p_j)$ (другими словами, СП консервативна, если для любого перехода $t_r \in T$ существуют такие положительные целые коэффициенты a_i и a_j, что $\sum_{p_i \in I(t_r)} a_i \mu(p_i) = \sum_{p_j \in O(t_r)} a_j \mu(p_j)$); таким образом, консервативные СП 1-консервативны, если $a_i = 1$ для всех $p_i \in P$, в этом случае в СП всегда присутствует одно и то же число маркеров;

– **живой (активной)**, если каждый переход $t \in T$ является потенциально срабатывающим при любой маркировке из M;

– **устойчивой**, если для всех пар $t_i, t_j \in T$, $i \neq j$ и любой допустимой маркировки, при которой t_i и t_j возбуждены, срабатывание одного из них не может снять возбуждения другого.

Прокомментируем практический смысл приведенных ваше определений. Свойство ограниченности СП отражает конечность информации, хранимой в «памяти» СП. В самом деле, если СП ограничена каким-либо параметром k, то из этого седует, что число возможных целочисленных векторов марикировок конечно, а следовательно и конечно число сосотяний в системе, моделируемой СП. В таком случае можно всегда найти верхнюю границу объему памяти для представления всех достижимых маркировок СП. Безопасность СП соответствует возможности нахождения прямого соответствия между сетью Петри и некоторой логической схемой, состояния которой могут быть представлены в виде бинарных векторов, максимальное количество которых ограничено числом 2^n, где n – это число позиций в СП. Консервативность СП может, например, показывать, что в моделируемой системе ограничен объем ресурсов. В современных постановках задач часто необходимо моделировать потоки потребляемой энергии (важнейшего ресурса в мобильных системах с ограниченной емкостью батарейки). Моделируя маркерами кванты энергии, выделяемой на выполнение операций (перехо-

дов СП), таким образом можно связать ограничения по энергии со свойствами модели системы.

Легко понять прагматику живости сетей Петри – **если мы хотим, чтобы все органы в моделируемой системе оставлись активными в течение всего цикла ее работы, соответствующая сеть Петри должна быть живой**. Это свойство очень важно для встроенных систем, доступ к элементам которых после их введения в эксплуатацию затруднен или же вообще не возможен. Интересно свойство устойчивости. Ему могут соответствовать разные практические трактовки. Например, в системе с нарушением устойчивости в ее модели возможны поведенческие конфликты. А именно, при возбуждении некоторого действия (например, подаче стартового сигнала на линии), в случае его конфликта с другим возбужденным действием, может произойти нежелательная «гонка», и в случае срабатывания одного из действий, другое действие должно быть остановлено. Поскольку физические процессы (даже в самой быстрой электронной системе) инерционны, наличие такого конфликта может привести к неопределенности, а именно к появлению состояния метастабильности на линии сигнала или же проскакиванию нежелательного импульса (glitch). С другой стороны, свойство наличия конфликта может отражать просто наличие ветвления или выбора между различными вариантами в поведении, напрмер, в случае, когда переходы-соперники соответствуют входным событиям, а конфликт – это просто выбор, выполняемый за пределами системы (т.е. все решает внешняя среда).

Советуем проследить связь поведенческой устойчивости со структурной бесконфликтностью, определенной ниже.

Классификация по статическим ограничениям

Перечислим некоторые важные в приложениях классы сетей Петри, порождаемые статическими ограничениями, накладываемыми на СП. (При этом, через $|A|$ будем обозначать *мощность множества* A).

Сеть Петри называется:

– **сетью свободного выбора**, если для любых $t_j \in T$ и $p_i \in I(t_j)$ позиция p_i является либо единственной входной позицией перехода t_j, т.е. $|O(p_i)| = 1$, либо этот переход имеет единственную входную позицию, т.е. $|I(t_j)| = 1$ (другими словами, если два перехода имеют общую входную позицию, то эта позиция единственна для каждого перехода);

- **маркированным графом**, если каждая позиция имеет в точности по одному входному и выходному переходу, т.е. если $|O(p_i)| = |I(p_i)| = 1$;
- **автоматной**, если каждый переход t_j имеет не более одного входа и не более одного выхода, т.е. если $|O(t_j)| = |I(t_j)| = 1$;
- **бесконфликтной**, если либо для каждой ее позиции $p_i \in P$ существует не более одной исходящей дуги, т.е. $|O(p_i)| \le 1$, либо для всех $t_j \in O(p_i)$ выполняется $t_j \in I(p_i)$ (любая позиция, являющаяся входной для более, чем одного перехода, является одновременно и выходной для каждого такого перехода). Бесконфликтные СП устойчивы, хотя обратное справедливо не всегда.

Прежде всего, смысл таких сетей связан с возможностью представления в поведении параллельных вычислительных процессов определенных «черт», таких как ветвлений, обусловленных выбором, и параллельных «нитей», связанных с независимыми действиям на разных ресурсах. В этом ракурсе, например, марикированные графы – это класс СП, в которых невозможен выбор по условиям. В них может лишь присутствовать параллельность, что полезно для моделирования структур управления стабильными и неизменяющимися (от данных) поледовательностями срабатываний, например, в конвейерных структурах. В автоматной СП, напротив, не может быть параллельности, зато может быть выбор. Эти классы в принципе дуальны, и дополняют друг друга, что отражено в сетях свободного выбора, которые покрывают оба этих класса по своим возможностям.

Кроме того, введение вышеуказанных структурных ограничений чрезвычайно полезно при анализе сложных моделей путем их рассмотрения по частям, т.е. с применением декомпозиции. Известны методы анализа СП свободного выбора путем покрытия их конечным набором либо марикированных графов, либо конечных автоматов. Поскольку и те и другие структурно просты – монохроматичны (представимы одним типом вершин) – алгоритмы их анализа (например, на живость и ограниченность) имеют значительно меньшую вычислительную сложность, чем алгоритмы общего класса СП.

Интересно, что бесконфликтные СП могут иметь такой структурный выбор, который не приводит к динамической борьбе за ресурс со снятием возбуждения с одного из переходов в результате срабатывания другого перехода. Такое поведение может быть полезно для представления процессов, в которых исходная спецификация с неупорядоченными действиями, например A и B, должна быть реализована на ограниченном ресурсе. В этом

случае неважно, в каком порядке выполняются действия *A* и *B*, сначала *A*, а потом *B* или наоборот. Главное, что оба действия выполнить одновременно нельзя из-за единственности ресурса (например, процессорного блока или шины для обмена данными).

Рис. 2.9 иллюстрирует общепринятую классификацию.

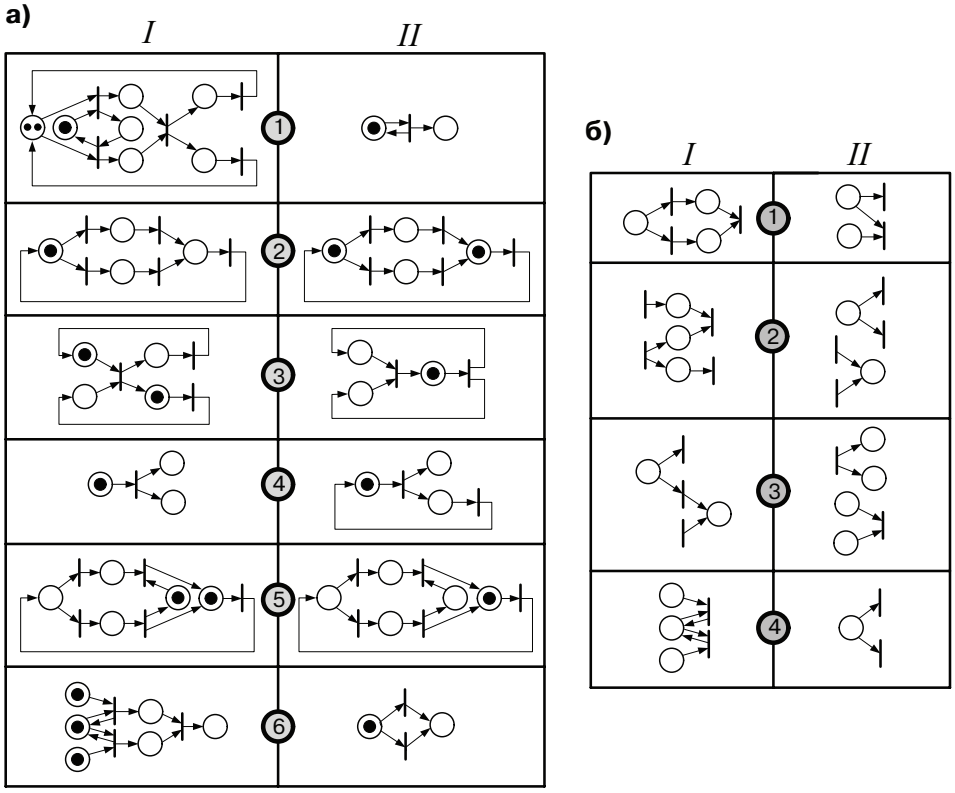


Рис. 2.9. К классификации сетей Петри: а) примеры (I) и контрпримеры (II) сетей следующих типов: ограниченные (1), безопасные (2), 1-консервативные (3), консервативные (4), живые (5), устойчивые (6); б) допустимые фрагменты (I) и недопустимые фрагменты (II) сетей следующих типов: свободного выбора (1), маркированные графы (2), автоматные (3), бесконфликтные (4)

2.5. Задачи и алгоритмы анализа

Под анализом конкретной сети Петри понимают совокупность методов, алгоритмов и приемов, применяемых с целью исследования ее статических (структурных) и динамических (поведенческих) свойств. В ходе анализа устанавливается наличие некоторых ее положительных свойств или анома-

лий, характеризующих нежелательные эффекты поведения. Трактовка тех или иных свойств как позитивных или негативных зависит от семантики процесса, моделируемого сетью Петри.

Классическими массовыми задачами анализа в теории сетей Петри являются:

- **достижимость** (возможно ли достижение некоторой исходно заданной маркировки из начальной в данной сети, т.е. содержится ли эта заданная маркировка в диаграмме маркировок);
- **живость** (возможно ли в принципе срабатывание любого перехода данной сети);
- **одновременность** (возможно ли параллельное срабатывание нескольких переходов);
- **ограниченность** (может ли в процессе функционирования сети в какой-либо ее позиции наблюдаться число маркеров, не превышающее заданное; в неограниченной сети в позициях может накапливаться бесконечное число маркеров);
- **безопасность** (частный случай ограниченности, когда элементы векторов достижимых маркировок, соответствующие позициям, содержат не более одного маркера (представляют собой булевы векторы);
- **устойчивость** (когда срабатывание одного перехода не может вызвать снятие возбуждения другого).

Доказана алгоритмическая разрешимость этих задач, в общем случае являющихся экспоненциальными по сложности. В приложениях чаще всего используются определенные подклассы сетей Петри, допускающие естественную интерпретацию: автономные (живые, в которых из любой маркировки достижима любая другая), безопасные и бесконфликтные.

Множество методов анализа сетей Петри можно условно разбить на три группы.

2.5.1. Первая группа методов анализа

К этой группе методов анализа сетей Петри относятся множество задач, сводимых к задачам достижимости и живости.

Задача достижимости

Ключевой задачей анализа сетей Петри является задача достижимости, при решении которой выясняется, достижима ли в данной сети некоторая исходно заданная маркировка из начальной, т.е. содержится ли эта заданная маркировка в диаграмме маркировок.

Ее стандартная постановка такова.

Заданы сеть Петри $< P, T, E, \mu^0 >$ и некоторая (целевая) маркировка μ^* (целочисленный вектор той же длины, что и начальная маркировка μ^0). Требуется установить факт принадлежности μ^* множеству M маркировок, достижимых из начальной, или установить противное. Иначе говоря, требуется определить значение некоторого предиката α такого, что

$$\alpha = \begin{cases} 1, \text{ если } \mu^* \in M, \\ 0, \text{ если } \mu^* \notin M. \end{cases}$$

Модификации задачи достижимости

Вышеприведенная формулировка может быть детализирована введением дополнительных требований, например, следующих.

- Решить задачу достижимости в предположении о том, что маркировка μ^* является пустой, т.е. μ^* = (00...0). Эта модификация известна как задача о достижимости пустой маркировки.
- Построить диаграмму (граф) маркировок, вершины которого – маркировки из M, а дуги помечены наименованиями переходов, срабатывание которых порождает изменение текущей маркировки (полная задача достижимости).
- Определить (построить) множество M достижимых маркировок.

Другие постановки задачи достижимости могут заключаться в следующем.

- Если достижимость имеет место, то указать хотя бы одну траекторию срабатываний переходов, которая ведет из начальной маркировки μ^0 в заданную маркировку μ^*.

- Если достижимость имеет место, то указать все траектории, ведущие из μ^0 в μ^* (при условии, что ни одна из траекторий не является частью другой).
- Может ли появиться маркер в заданной позиции $p^* \in P$ при заданной начальной маркировке μ^0 (задача о *достижимости позиции*).
- Найти маркировку, получающуюся из исходной при заданной последовательности переходов, если она существует.
- Решить задачу достижимости для подмножества P' ⊂ P позиций СП (задача *достижимости подмаркировки*).
- Решить задачу о достижимости позиции в предположении, что ее маркировка пуста (задача о *достижимости пустой позиции*).

Само перечисление модификаций задач достижимости наталкивает на мысль, что некоторые из них являются более общими, чем другие, или эквивалентны другим. Эти факты можно установить посредством конструктивных приемов сведения одной задачи к другой. Однако прежде, чем заняться этими вопросами, следует получить ответ на главный вопрос – разрешима ли задача достижимости. Прежде всего, из интуитивных соображений следует обратить внимание на формулировку полной задачи разрешимости. Понятно, что диаграмма маркировок с дугами, нагруженными наименованиями переходов, несет всю информацию, полезную для решения остальных модификаций задачи достижимости. Однако сразу же возникают сомнения в возможности получить такую диаграмму, и эти сомнения основываются на том, что число маркировок, достижимых из начальной, может быть бесконечным.

Действительно, фрагмент сети Петри, приведенный на рис. 2.9а в верхнем правом углу, порождает бесконечную диаграмму маркировок вида

$$10 \xrightarrow{p_1 p_2} 11 \to 12 \to \ldots \to 1\infty,$$

если через p_1 и p_2 обозначить соответственно левую и правую позиции фрагмента. Эту диаграмму можно искусственно привести к виду 10→1ω, где символ ω обозначает сколь угодно большое число. Ту же идею можно использовать и в общем случае, но сначала обсудим вопрос о том, как строить диаграмму маркировок.

Алгоритм порождения диаграммы маркировок

Прежде, чем сформулировать алгоритм порождения диаграммы маркировок, введем некоторые дополнительные определения.

Множество переходов, возбужденных при текущей маркировке μ^i, будем называть *селектором* и обозначать его через $S(\mu^i)$.

Маркировку μ^d, для которой $S(\mu^d) = \varnothing$, назовем *тупиковой*.

Введем отношение *покрытия* для пары маркировок μ^i и μ^j : $\mu^i \geq \mu^j$, которое выполняется, если в каждой компоненте $\mu^i(p_k) \geq \mu^j(p_k)$. Отметим, что если маркировке μ^j соответствует селектор $S(\mu^j)$, то для покрывающей маркировки μ^i $S(\mu^i) \geq S(\mu^j)$. При этом очевидно, те переходы, которые были возбуждены при μ^j, будут также возбуждены при μ^i, так как наличие "лишних" маркеров не меняет локального условия возбуждения.

Пусть в процессе функционирования СП из маркировки μ достигнута строго покрывающая ее маркировка μ', μ' > μ, т.е. μ' – μ > 0. При этом переход от μ к μ' связан с добавлением некоторого числа маркеров в определенные позиции. Поскольку при μ' сохраняются те же возможности для срабатывания переходов, что и при μ, то дальнейшая работа СП приводит к неограниченному наращиванию числа маркеров в указанных позициях. Таким образом, обобщение эффекта накопления маркеров в позициях при циклическом функционировании СП может быть задано с помощью символа ω, для которого справедливы следующие алгебраические операции: ω + a = ω и ω – a = ω при a < ω. Эти операции являются базисными для построения диаграммы маркировок.

Сформулируем соответствующий алгоритм. В процессе построения каждая вершина диаграммы может быть отнесена к одному из следующих типов:

- *тупиковая* (для которой $S(\mu) = \varnothing$),
- *внутренняя* (включенная в диаграмму),
- *граничная* (еще не обработанная алгоритмом, по которой требуется принять решение о включении ее в число тупиковых или внутренних).

Алгоритм порождения диаграммы маркировок:

0. Алгоритм начинает работу с объявления начальной маркировки граничной вершиной. Пусть x – граничная вершина. Она обрабатывается в соответствии со следующими правилами.

1. Если в диаграмме уже имеется некоторая (обработанная) вершина y и с ней связана та же маркировка $\mu^y = \mu^x$, то вершина x исключается из рассмотрения, а все дуги, ведущие в x, заводятся в y.

2. Если для маркировки μ^x селектор $S(\mu^x) = \varnothing$, то x – тупиковая вершина.

3. Для каждого перехода $t_j \in T$ такого, что $t_j \in S(\mu^x)$, создать новую вершину z диаграммы с маркировкой μ^z, которая определяется для каждой позиции p_i следующим образом:

а) если $\mu^x(p_i) = \omega$, то $\mu^z(p_i) = \omega$;

б) если на пути от вершины, соответствующей начальной маркировке, к x найдется вершина y такая, что $\mu^y < \mu^r$, где μ^r удовлетворяет $\mu^x \xrightarrow{t_j} \mu^r$, и $\mu^y(p_i) < \mu^r(p_i)$, то $\mu^z(p_i) = \omega$;

в) в противном случае $\mu^z(p_i) = \mu^r(p_i)$.

Дуга, помеченная символом t_j, направлена от вершины x к вершине z. Переопределить вершину x как внутреннюю, а вершину z определить как граничную и добавить ко множеству граничных вершин.

4. Если множество граничных вершин не пусто, то выбрать любую граничную вершину и перейти к п.1, иначе завершить процедуру.

Алгоритм иллюстрируется построением диаграммы маркировок (рис. 2.10б) для сети Петри, представленной на рис. 2.10а.

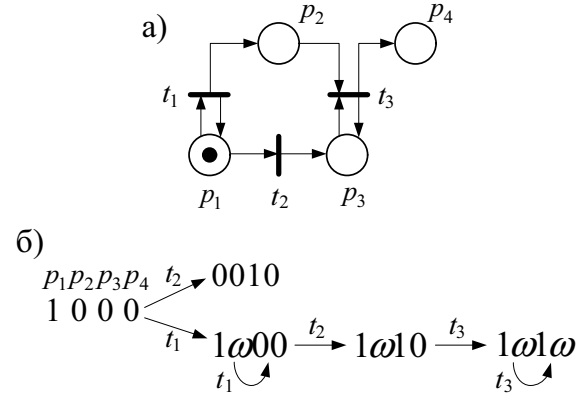


Рис. 2.10. а) Сеть Петри и б) ее диаграмма маркировок

К сожалению, “свернутые” диаграммы маркировок, содержащие символ ω в позициях неограниченного накопления маркеров, теряют часть информа-

ции о поведении СП. Чтобы подтвердить этот тезис, рассмотрим СП, приведенную на рис. 2.11а. Она порождает диаграмму, показанную на рис. 2.11в. Теперь модифицируем СП на рис. 2.11а так, чтобы переход t_2 был связан с позицией p_2 двумя дугами (см. рис. 2.11б). (Мы несколько забегаем вперед, потому что такого рода объект, вообще говоря, не является сетью Петри, как она была определена выше, но в разд. 3.1 будет показано, что такая модель равномощна сети Петри.) В этом случае переход t_2 при каждом срабатывании забрасывает в p_2 два маркера.

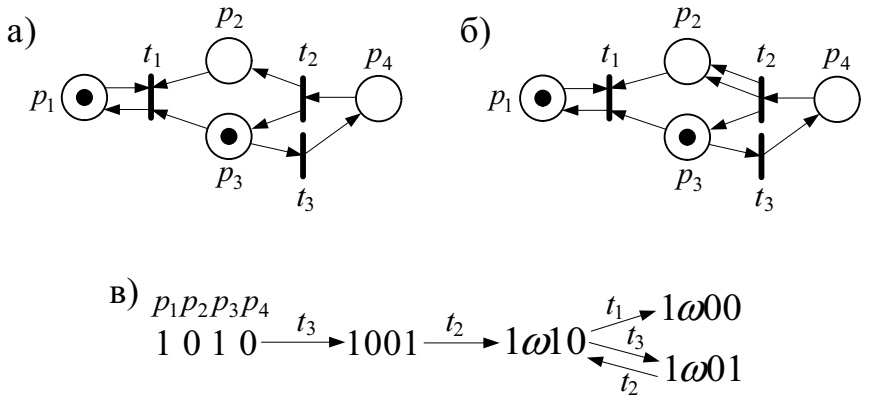


Рис. 2.11. а), б) Примеры сетей Петри и в) соответствующая им диаграмма маркировок

Легко убедиться, что эта модификация СП не отражается на диаграмме маркировок, которая и для данного случая совпадает с диаграммой, изображенной на рис. 2.11в. Однако множества маркировок обеих сетей не совпадают: в СП на рис. 2.11а число маркеров в позиции p_2 есть произвольное целое, а в СП на рис. 2.11б количество маркеров в этой позиции является только четным числом при условии задержки срабатывания перехода t_1, ведущего на диаграмме рис. 2.11в в тупиковую маркировку 1ω00. Наличие символов ω в маркировках 1ω10 и 1ω01 скрывает эту особенность второй сети, в которой, например, маркировки 1110 и 1101 являются недостижимыми. Потеря информации, которая иллюстрируется этим примером, невосполнима. Поэтому сжатие диаграммы маркировок, достигаемое посредством использования символа ω, не позволяет корректно решить задачу достижимости в ее классической постановке. Приведенный пример свидетельствует и о том, что диаграммы маркировок, содержащие символ ω, искажают решение задачи о допустимых трейсах (траекториях) в СП и, по-видимому, не имеют практического значения.

Свойства диаграмм маркировок

Несмотря на то, что диаграмма маркировок не несет в себе исчерпывающей информации о поведении СП, по ней легко устанавливаются характеристические свойства. Так, СП принадлежит классу ограниченных тогда и только тогда, когда символ ω отсутствует в диаграмме маркировок. Наличие этого символа в маркировке указывает, в каких позициях может иметь место бесконечное накопление маркеров. Верхняя граница $\mu(p_i)$ для всех $\mu \in M$ указывает, для какого k СП является k-ограниченной. Сеть является безопасной, если в любой маркировке μ диаграммы $\mu(p_i) \le 1$ для всех $\mu \in M$. Иными словами, в каждой позиции безопасной сети может появиться только один маркер. Можно также определить позиции, по отношению к которым СП безопасна.

Свойство 1-консервативности проверяется по ограниченной диаграмме путем суммирования компонент каждой маркировки μ по числу позиций, т.е. $|\mu| = \sum_{i=1}^{n} \mu(p_i)$. Если $|\mu| = const$ для всех $\mu \in M$, то СП обладает этим свойством (при функционировании такой СП в ней всегда присутствует постоянное число маркеров). Более общее свойство консервативности, когда позиции ограниченной СП взвешиваются посредством целых неотрицательных чисел, лучше всего определять матричным методом, речь о котором будет идти ниже. Если СП не ограничена, т.е. содержит символ ω, то консервативной она будет лишь тогда, когда вес маркера в соответствующей позиции принимается равным нулю. Такой вариант взвешивания представляет интерес в том случае, когда свойство консервативности устанавливается относительно некоторого подмножества позиций СП (консервативная составляющая).

Сводимость задач достижимости

Выше уже говорилось о сводимости одних модификаций задачи достижимости к другим. Граф сводимости таких задач приведен на рис. 2.12. В этом графе дуга ориентирована от задачи *A* к задаче *B*, если *A* сводится к *B*.

Из сказанного выше ясно, что построение свернутых диаграмм маркировок не позволяет ответить на вопрос о достижимости заданной маркировки μ^*, поэтому возникает вопрос о разрешимости задачи достижимости в ее классической постановке.

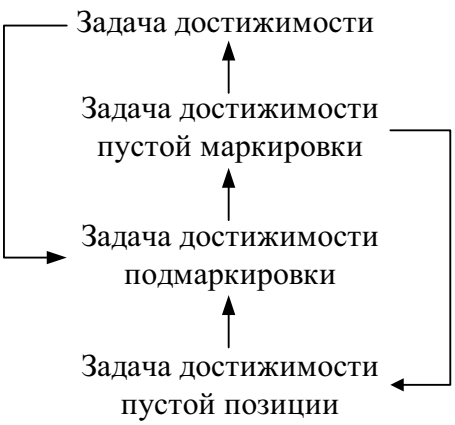


Рис. 2.12. Сводимость задач достижимости

Выяснение разрешимости этой задачи потребовало немалых усилий специалистов. В настоящее время разрешимость этой задачи доказана, хотя доказательство является весьма громоздким.

В практических применениях часто приходится иметь дело с ограниченными СП, в которых достижимость устанавливается в ходе построения диаграммы маркировок.

Между тем теория сетей Петри дает примеры задач, неразрешимость которых установлена. В частности, неразрешимыми оказываются задачи о том, выполняется ли $M_1 \subset M_2$ или $M_1 = M_2$ для двух заданных СП.

Задача живости Сети Петри

Теперь перейдем к обсуждению **задач живости СП, при решении которой выясняется принципиальная возможность срабатывания любого перехода анализируемой сети.**

Напомним, что СП называется *живой* (активной), если при любой маркировке из M каждый ее переход потенциально возбужден (может сработать).

Задача живости СП имеет две модификации.

- Для данной СП $< P, T, E, \mu^0 >$ определить, все ли ее переходы $t_i \in T$ живы (могут сработать) – *общая задача живости.*
- Для данной СП определить, является ли живым некоторый ее переход $t_i \in T$ – *задача живости одного перехода.*

Было показано, что задача живости сводится к задаче достижимости пустой позиции, которая, в свою очередь, сводится к задаче достижимости. Доказана и обратная сводимость.

Можно показать, что задача живости одного перехода также сводится к задаче достижимости.

Для решения задачи живости СП свернутая посредством использования символа ω диаграмма маркировок также не годится, так как может привести к потере существенной информации. Так, изображенная на рис. 2.13а сеть Петри, которая порождает диаграмму на рис. 2.13в, является живой. Траектория $t_1t_2t_3$ в ней возможна и при этом возбужден переход t_4, так что все переходы сети Петри активны.

Сети на рис. 2.13б соответствует та же диаграмма, но при той же траектории СП попадает в тупиковую маркировку 010, при которой переход t_4 мертв.

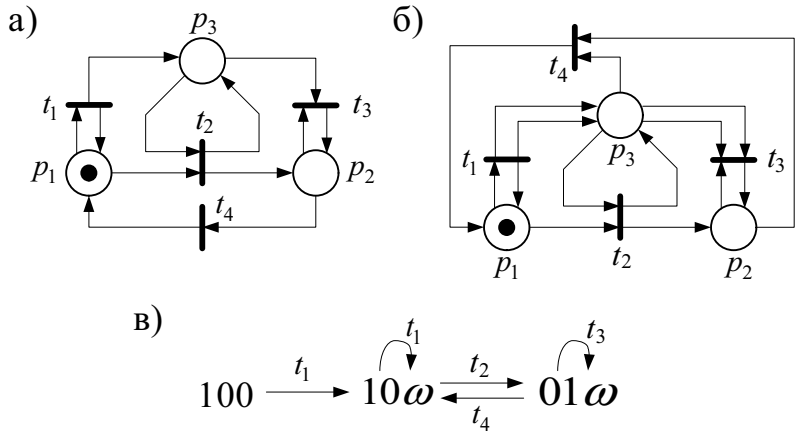


Рис. 2.13. а),б) Примеры сетей Петри и в) порождаемая ими диаграмма маркировок

2.5.2. Вторая группа методов анализа

Вторая группа методов анализа касается задач на установление принадлежности СП классам сетей свободного выбора, бесконфликтным сетям, маркированным графам и автоматным сетям (определения таких сетей даны в разд. 2.4). Задачи этой группы здесь не рассматриваются в силу того, что они решаются методами теории графов на основе рассмотрения локальных признаков на структурном уровне.

2.5.3. Третья группа методов анализа

Третья группа методов анализа ориентирована на выявление специфических конфигураций сети, в которых при функционировании возможен дефицит маркеров или, наоборот, их накопление. При этом сами методы анализа также основаны на использовании структурного подхода и не связаны с построением диаграмм маркировок.

Дедлоки

***Дедлоком* в сети Петри называют подмножество позиций сети, обусловливающих попадание сети в тупиковое состояние (маркировку), в котором не возбуждается ни один переход.** Используя формальный язык, *дедлоком* в сети Петри $< P, T, E, \mu^0 >$ называют подмножество ее позиций $D \subseteq P$ такое, что $I(D) \subseteq O(P)$.

Иначе говоря, каждый переход, имеющий принадлежащую дедлоку выходную позицию, должен иметь своими входными позициями только те из них, которые принадлежат дедлоку.

Помещая маркер в одну из позиций дедлока, сеть обязательно забирает маркер из некоторой позиции дедлока. Это означает, что если все позиции дедлока становятся пустыми (без маркеров), то они далее будут оставаться пустыми для любой возможной последовательности срабатываний переходов. Интуитивно ясно, что понятие дедлока полезно для определения живости СП. Однако **дедлок – это не обязательно тупиковая маркировка, скорее это потенциальный тупик, т.е. маркировка, обещающая стать тупиковой вследствие возможного дефицита маркеров**.

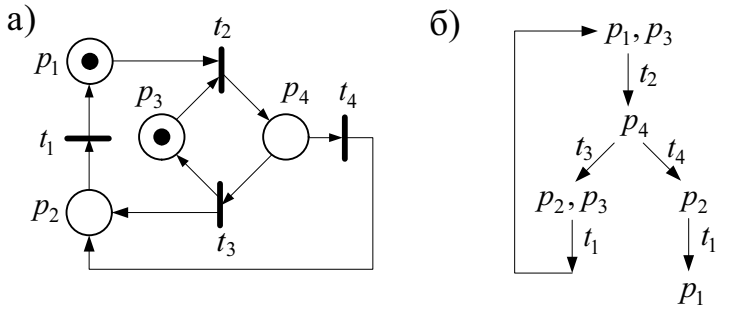


Рис. 2.14. а) Сеть Петри, содержащая дедлок и б) ее диаграмма маркировок

Пример сети Петри, содержащей дедлок, приведен на рис. 2.14а. Дедлок образуют позиции $D = \{p_3, p_4\}$. Действительно, $I(p_3) = \{t_3\}$, $I(p_4) = \{t_2\}$,

$I(\{p_3, p_4\}) = I(p_3) \cup I(p_4) = \{t_2, t_3\}$, $O(p_3) = \{t_2\}$, $O(p_4) = \{t_3, t_4\}$, $O(\{p_3, p_4\}) = O(p_3) \cup O(p_4) = \{t_2, t_3, t_4\}$, откуда $I(\{p_3, p_4\}) \subset O(\{p_3, p_4\})$, так что позиции p_3 и p_4 образуют дедлок. Диаграмма маркировок для рассматриваемой сети приведена на рис. 2.14б. Из нее видно, что из начальной маркировки возможны две траектории. Одна ($t_2t_3t_1$) возвращает сеть в исходное состояние, а вторая ($t_2t_4t_1$) заканчивается тупиком: сказывается дефицит маркеров в дедлоке.

Ловушки

Ловушкой **в сети Петри называется такое множество позиций, что каждый переход, входом для которого является одна из позиций множества, имеет выходом другую позицию этого же множества. Иными словами, ловушка представляет собой цикл в сети Петри, из которого нет выхода.** Более формально, *ловушкой* в сети Петри называют подмножество ее позиций $L \subseteq P$ такое, что $O(L) \subseteq I(L)$.

Если из ловушки забирается маркер, то СП обязательно забрасывает в нее, по крайней мере, один маркер обратно. Если в позициях ловушки оказался хотя бы один маркер, то никогда все позиции ловушки не окажутся пустыми.

Пример СП, содержащей ловушку, приведен на рис. 2.15а. Ловушку образуют позиции $L = \{p_3, p_4\}$. Действительно, $I(p_3) = \{t_2\}$, $I(p_4) = \{t_3, t_4\}$, $I(L) = \{t_2, t_3, t_4\}$, $O(p_3) = \{t_3\}$, $O(p_4) = \{t_2\}$, $O(L) = \{t_2, t_3\}$, откуда $I(L) \supset O(L)$.

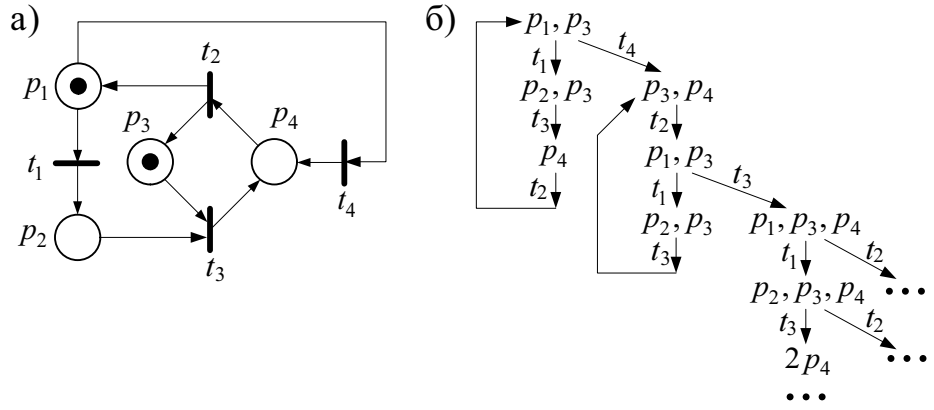


Рис. 2.15. а) Пример сети Петри, содержащий ловушку и б) начальный фрагмент ее бесконечной диаграммы маркировок

Из диаграммы маркировок на рис. 2.15б видно, что при функционировании сети Петри число маркеров в ловушке может расти. Дедлоки и ловушки не обязательно исключают друг друга. Например, любая сильно связная СП содержит как ловушку, так и дедлок.

Известно, что **сеть Петри свободного выбора жива тогда и только тогда, когда каждый ее дедлок содержит непустую (маркированную) ловушку**. Этот факт известен как *теорема Коммонера*.

2.5.4. Оценки вычислительной сложности проблем анализа сетей Петри

Раздел о методах анализа сетей Петри, очевидно, следует завершить *оценками вычислительной сложности* проблем достижимости, живости, покрываемости, ограниченности и консервативности для различных классов СП. Результаты этих исследований сведены в таб.2.1. В этой таблице обозначения *NSE*, *DSP*, *DSE*, *NSL*, *NTP*, *NSP* расшифровываются следующим образом:

- **Первый символ** в них указывает на тип распознающей машины Тьюринга – детерминированная (*D*) или недетерминированная (*N*).
- **Второй символ** – емкостную (*S*) или временную (*T*) сложность распознавания (число требуемых ячеек ленты и число шагов соответственно).
- **Третий символ** – собственную оценку сложности: *P* – полиномиальная, *E* – экспоненциальная, *L* – логарифмическая.

Таблица оценок вычислительной сложности проблем. Таблица 2.1

	Проблемы				
Класс СП	Достижимости	Живость	Покрываемость	Ограниченность	Консервативность
Автоматные	*NSL*-полная				Тривиальная
Маркированные графы	*NSL*-трудная*	*NSL*-полная*	*NSL*-трудная*		*NSL*-полная
1-консервативные со свободным выбором	*DSP*-полная	*NTP*-полная*	*DSP*-полная		Тривиальная
Произвольные со свободным выбором	*DSE*-трудная	*NTP*-полная	*DSE*-трудная	*DSP*-полная	*NTP*-полная
1-консервативные	*DSP*-полная	*DSP*-полная	*DSP*-полная		Тривиальная
Произвольные	*DSE*- трудная		*DSP*-полная		*NTP*-полная

Известно, что DSP = NSP, DSE = NSE, а также что $NSL \subseteq DTP \subseteq NTP \subseteq DSP$. Имеют ли место другие включения – неизвестно.

В этой таблице символом * помечены оценки, справедливые для случаев, когда сети не обладают соответствующим свойством.

2.6. Анализ: матричный подход

Исследование динамики функционирования сети Петри с заданной начальной маркировкой может осуществляться, как было показано выше, по диаграмме маркировок. Однако в большинстве случаев с ростом размерности сети сложность выявления таких фундаментальных свойств, как ограниченность и живость, становится слишком большой, что делает такой лобовой подход неприемлемым с практической точки зрения. В теории сетей Петри активно развивалась иная стратегия анализа – *матричный (линейно-алгебраический) подход*, основанный, прежде всего, на исследовании структурных свойств сети. В частности, со "структурной точки зрения" (т.е. независимо от начальной маркировки) можно ввести ряд определений, базирующихся на понятии графа (диграфа) сети Петри, или *немаркированной сети Петри* (НСП), заданной четверкой $N = < P, T, \alpha, \beta >$, где P – непустое множество позиций, T – непустое множество переходов, $\alpha : P \times T \to N^+$, $N^+ = \{0,1,2,...\}$, – функция инцидентности по входу, $\beta : T \times P \to N^+$ – функция инцидентности по выходу.

Легко заметить, что в данном определении функции α и β имеют целочисленные значения, а это означает наличие весов на соответствующих дугах диграфа сети Петри. (Здесь авторы сознательно "забегают вперед" для внесения необходимой общности в анализируемый объект – сеть Петри – с целью получения больших иллюстративных возможностей линейно-алгебраического подхода. Более подробно сети Петри с кратными или взвешенными дугами рассмотрены в следующей главе.)

Анализ на свойства структурной ограниЧенности и живости

Немаркированная сеть Петри N является *структурно ограниченной*, если для любой маркировки μ^0 сеть Петри $< N, \mu^0 >$, образованная этой НСП и маркировкой μ^0, является ограниченной. Кроме того, немаркированная сеть Петри N является *структурно живой*, если существует, по крайней мере, одна маркировка μ^0, для которой сеть Петри $< N, \mu^0 >$ является живой.

Для проведения анализа указанных свойств обычно принимается ограничение на структуру – рассматриваются так называемые *чистые* СП, в которых отсутствуют петли, имеющие место в случае $I(t) \cap O(t) \neq \varnothing$. Такое ограничение не является принципиальным, поскольку, во-первых, для каждой СП можно построить более сложную чистую СП и, во-вторых, на практике при построении СП петель можно избежать.

Для чистой НСП можно определить *матрицу инцидентности* $C = \| c_{ij} \|_{n \times m}$, где

$$c_{ij} = \begin{cases} \beta(t_j, p_i), \text{ если } \beta(t_j, p_i) \neq 0, \\ -\alpha(p_i, t_j), \text{ если } \alpha(p_i, t_j) \neq 0, \\ 0 \quad , \text{ в противном случае,} \end{cases}$$

причем $1 \le i \le n$, $1 \le j \le m$, где $n = |P|$, $m = |T|$. Например, для НСП, показанной на рис. 2.16, матрица инцидентности приведена в табл. 2.2.

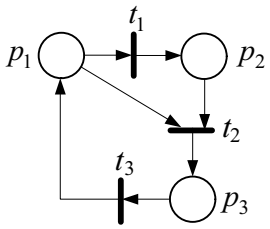


Рис. 2.16. Немаркированная сеть Петри

Матрица инцидентности. Таблица 2.2

	p_1	p_2	p_3
t_1	–1	1	0
t_2	–1	–1	1
t_3	1	0	–1

Чистая СП и чистая НСП, имеющие матрицу инциденций C, могут быть представлены соответственно как $< C, \mu^0 >$ и $< C, - >$.

Пусть в чистой СП $< N, \mu^0 >$ имеет место последовательность $\sigma = t_{j1} t_{j2} ... t_{jk}$ срабатываний переходов, приводящая к маркировке μ^k. Поставим ей в соответствие характеристический вектор $\overline{\sigma} = (\sigma_1, ..., \sigma_m)$, каждая компонента которого σ_j $(1 \le j \le m)$ равна числу вхождений t_j в σ. Известно, что для каж-

дого возбужденного при маркировке μ перехода t_j, срабатывание которого приводит к маркировке μ', справедливо

$$\mu' = \mu + C_j, \qquad (2.1)$$

где C_j –j-й вектор-строка матрицы инциденций C, соответствующий переходу t_j.

Легко видеть, что (2.1) обобщается на случай последовательности срабатываний переходов σ в виде:

$$\mu^k = \mu^0 + \overline{\sigma} \cdot C. \qquad (2.2)$$

Это уравнение принято называть фундаментальным уравнением СП.

Оно позволяет эффективно использовать для анализа СП аппарат линейной алгебры. Совмещение в одной матричной форме маркировки СП и структуры (матрицы инцидентности) СП дает возможность исследовать поведенческие свойства как множества достижимых маркировок (ситуаций АП), характеризующих ограниченность или, в более общей форме, емкостные свойства СП, так и множества допустимых последовательностей срабатываний (следов АП), характеризующих живость или, вообще говоря, свойства активности СП.

Заметим, что, помимо ограниченности (напомним, что безопасность СП является частным случаем ограниченности), к емкостным свойствам СП обычно относят консервативность, невозвратность, наличие дедлоков и ловушек. К свойствам активности, помимо живости, относятся также беступиковость (так называемая слабая живость), повторяемость и справедливость.

Инвариантные и согласованные сети Петри

Фундаментальными понятиями в матричном подходе к анализу СП являются понятия p- и t-инвариантов и соответствующие им понятия инвариантной и согласованной СП.

Целочисленный вектор $Y = (y_1, y_2, ..., y_n)$, являющийся решением линейной системы

$$CY^t = 0, \qquad (2.3)$$

называется p-инвариантом немаркированной сети Петри $< C, - >$.

Примечание. Здесь и далее верхний индекс t означает операцию транспонирования вектора или матрицы.

Целочисленный вектор $X = (x_1, x_2,..., x_m)$, **являющийся решением линейной системы**

$$XC = 0, \qquad (2.4)$$

называется *t*-**инвариантом** НСП $< C, - >$. Соответственно, НСП называется *инвариантной* (*согласованной*), если для нее существует *p*-инвариант $Y > 0$ (*t*-инвариант $X > 0$), все компоненты которого y_i (x_j) положительны.

Перечислим набор утверждений, связывающих понятия инвариантной и/или согласованной НСП с ее емкостными и активностными свойствами.

1. НСП является согласованной тогда и только тогда, когда существует маркировка μ^0 и последовательность σ срабатываний переходов такие, что $\mu^0 \xrightarrow{\sigma} \mu^0$.
2. Если НСП ограничена и жива, то она согласована.
3. Если НСП инвариантна, то она ограничена.
4. Множество позиций инвариантной НСП образует дедлок и ловушку.
5. Если для НСП $< C, - >$ существует целочисленный положительный вектор Y, такой что $CY^t < 0$, то $< C, - >$ неживая и несогласованная, а множество позиций $P^* = \{p_i \mid y_i > 0\}$ образует дедлок.
6. Если для живой СП $< C, \mu^o >$ целочисленный вектор Y, $Y > 0$, такой что $CY^t > 0$, то $< C, \mu^o >$ не ограничена.
7. Если для НСП $< C, - >$ существует вектор Y, $Y > 0$, такой что $CY^t > 0$, то:
   а) при условии того, что $< C, - >$ жива, она не является ограниченной;
   б) $< C, - >$ не согласована;
   в) множество позиций $P^* = \{p_i \mid y_i > 0\}$ является ловушкой.
8. НСП $< C, - >$ является неограниченной тогда и только тогда, когда существует целочисленный вектор $X = (x_1, x_2,..., x_m)$, $X > 0$, для которого $XC > 0$.
9. Если для НСП $< C, - >$ существует целочисленный вектор $X > 0$ такой, что $XC < 0$, то:
   а) при условии своей ограниченности $< C, - >$ является неживой;
   б) $< C, - >$ не инвариантна.

10. Если НСП $< C, - >$ ограниченная и согласованная НСП, то она инвариантна и для любой маркированной СП $< C, \mu^o >$ множество достижимых маркировок состоит из попарно несравнимых векторов. Следует заметить, что для заданной ограниченной и согласованной НСП $< C, - >$ можно вычислить верхнюю границу мощности множества маркировок. Например, для безопасной $< C, \mu^o >$ легко установить, что мощность множества достижимых маркировок ограничена C_k^n, где $k = n/2$ для четных n и $k = (n-1)/2$ для нечетных n.

11. Если НСП $< C, - >$ является ограниченной и живой, то она также является согласованной и инвариантной.

12. Если НСП $< C, - >$ является инвариантной и несогласованной, то она неживая.

13. Если НСП $< C, - >$ является согласованной и неинвариантной, то она не ограничена.

Прокомментируем некоторые из приведенных утверждений (большинство из них являются достаточно прозрачными с содержательной точки зрения и читателю будет полезно рассмотреть иллюстративные примеры в качестве самостоятельного упражнения). Прежде всего, следует обсудить связь понятий инвариантности и согласованности со свойствами ограниченности и живости НСП. В частности, утверждение 3 дает достаточное условие, а утверждение 8 необходимое и достаточное условие ограниченности.

Попытаемся вычислить p-инвариант для НСП, заданной с помощью рис. 2.16 и табл. 2.2. Исходя из условия (2.3), система уравнений будет иметь следующий вид:

$$\begin{cases} -y_1 + y_2 = 0 \\ -y_1 - y_2 + y_3 = 0 \\ y_1 - y_3 = 0 \end{cases}.$$

Нетрудно видеть, что, так как множество ее ненулевых решений пусто, то она не инвариантна. Следовательно, условие утверждения 3 не выполняется. Однако для того, чтобы принять заключение относительно неограниченности этой НСП, следует проверить условие утверждения 8, т.е. попытаться найти целочисленный вектор $X > 0$, для которого $XC > 0$. Легко видеть, что система неравенств

$$\begin{cases} -x_1 - x_2 + x_3 > 0 \\ x_1 - x_2 > 0 \\ x_2 - x_3 > 0 \end{cases}$$

не имеет решения $X > 0$, так как $(X > 0) \wedge (x_1 > x_2 > x_3) \Rightarrow x_3 < x_1 + x_2$, что противоречит первому неравенству: $x_3 > x_1 + x_2$. Следовательно, НСП является ограниченной.

В том, что данная НСП неживая, легко убедиться из проверки условия утверждения 5: существует решение неравенства $CY^t < 0$ для $Y > 0$, т.е. решение, удовлетворяющее условию $(y_2 < y_1) \wedge (y_3 < y_1 + y_2) \wedge (y_1 < y_3)$, например, $Y = (3, 2, 4)$.

Рассмотрим теперь НСП, которая отличается от предыдущей лишь наличием веса 2 на дуге из t_3 в p_1. Соответствующая матрица инциденций будет иметь вид:

$$C = \begin{Vmatrix} -1 & 1 & 0 \\ -1 & -1 & 1 \\ 2 & 0 & -1 \end{Vmatrix}.$$

Ее p-инвариант Y вычисляется по (2.3) как решение системы:

$$\begin{cases} -y_1 + y_2 = 0 \\ -y_1 - y_2 + y_3 = 0 \\ 2y_1 - y_3 = 0 \end{cases}.$$

Данная система имеет бесконечное множество решений $Y > 0$, которые удовлетворяют условию: $y_1 = y_2$, $y_3 = 2y_1$. Например, $Y = (1, 1, 2)$. Следовательно, НСП инвариантна и ограничена.

Легко убедиться, что для этой НСП условие утверждения 8 не выполняется: система неравенств

$$\begin{cases} -x_1 - x_2 + 2x_3 > 0 \\ x_1 - x_2 > 0 \\ x_2 - x_3 > 0 \end{cases}$$

не имеет решения при $X > 0$, поскольку из второго и третьего неравенств следует $x_1 > x_2 > x_3$, что несовместимо с первым неравенством.

Свойство согласованности может быть установлено решением системы уравнений вида $XC = 0$, т.е.

$$\begin{cases} -x_1 - x_2 + 2x_3 = 0 \\ x_1 - x_2 = 0 \\ x_2 - x_3 = 0 \end{cases},$$

которая имеет бесконечное множество целочисленных решений $X > 0$, удовлетворяющих $x_1 = x_2 = x_3$, например $X = (1, 1, 1)$. Все эти решения являются t-инвариантами НСП.

Внимательный читатель, вероятно, заметил, что в приведенных выше утверждениях присутствуют лишь достаточные условия отсутствия живости, в связи с чем проверка свойства живости не всегда может быть выполнена в рамках данного подхода. Действительно, последний пример иллюстрирует "коварство" свойства живости, а именно: несмотря на то, что НСП инвариантна, ограничена и согласована, она не является (структурно) живой, т.е. не существует маркировки μ^0, для которой соответствующая маркированная сеть Петри $< C, \mu^o >$ была бы живой. В этом можно убедиться, построив диаграмму маркировок этой сети Петри для $\mu^0 = (2\ 0\ 0)$. Из диаграммы будет видно, что ввиду возможности неограниченного оттока маркеров из p_1 и p_2 всегда достижима маркировка, при которой $\mu(p_1) = \mu(p_3) = 0$ и переход t_2 является мертвым.

Обратим также внимание на следующий факт. Несмотря на то, что в данном примере не выполняется условие существования такого Y > 0, что $CY^t < 0$, данная сеть содержит дедлок $\{p_1, p_3\}$. Следовательно, из наличия дедлока, равно как и из отсутствия живости, не следует существование $Y > 0$, для которого $CY^t < 0$.

Особого внимания заслуживает **свойство консервативности сетей Петри,** которое неразрывно связано с понятием p-инварианта НСП. Дело в том, что уравнения вида (2.3) соответствуют стационарным режимам функционирования сети. Именно поэтому вектор Y и получил наименование инварианта. Необходимую ясность в данные рассуждения вносит следующее преобразование фундаментального уравнения сетей Петри (2.2). Помножив его на Y^t, получим скалярное уравнение

$$\mu \cdot Y^t = \mu^0 Y^t + \overline{\sigma} \cdot C \cdot Y^t,$$

откуда, ввиду (2.3), следует условие постоянства маркерного потока в сети Петри: $\mu \cdot Y^t = const$, т.е. для любой маркировки сети Петри, имеющей инвариант Y, произведение этого инварианта на вектор-маркировку есть постоянная величина.

Данный закон является прямым аналогом закона Кирхгофа о постоянстве потока зарядов в электрической цепи. Элементы вектора Y играют роль ве-

сов маркеров в соответствующих позициях СП. Следовательно, произведение $\mu \cdot Y^t$ – это взвешенная сумма маркеров сети.

Можно заметить, что в определении инвариантности НСП оговаривалось, что компоненты вектора Y должны быть положительны. Действительно, если для любого i верно $y_i > 0$, то свойство консервативности, т.е. постоянства маркерного потока, охватывает всю сеть Петри.

В общем случае, для НСП может существовать множество инвариантов, среди которых могут быть, естественно, такие, у которых не все компоненты положительны. Для таких инвариантов принято говорить о так называемой *консервативной составляющей сети*, а именно: если вектор Y_r является одним из p-инвариантов НСП, то он выделяет некоторое подмножество позиций $P_r \subset P$ такое, что $p_i \in P_r$, если i-я компонента вектора Y_r больше нуля. Это подмножество называется *суппортом инварианта* Y_r и образует консервативную составляющую сети, для которой взвешенная сумма маркеров в ее позициях в процессе функционирования постоянна:

$$\sum_{p_i \in P_r} \mu(p_i) \cdot y_i = \sum_{p_i \in P_r} \mu^0(p_i) \cdot y_i = const . \qquad (2.5)$$

Легко понять, что по значению p-инварианта можно найти верхнюю границу числа маркеров k для каждой из позиций консервативной составляющей сети. Так, для позиции $p_i \in P_r$

$$k_i \leq \left\lfloor \frac{\mu^0 \cdot Y_r^t}{y_i^r} \right\rfloor,$$

где $\lfloor x \rfloor$ – ближайшее целое, меньшее или равное x.

Например, для НСП с матрицей инцидентности

$$C = \begin{Vmatrix} -1 & 1 & 0 \\ -1 & -1 & 1 \\ 2 & 0 & -1 \end{Vmatrix}$$

минимальным положительным p-инвариантом является Y = (1 1 2). Поскольку все компоненты вектора положительны, то вся сеть инвариантна, т.е. консервативна.

Подставив значение Y в (2.5) и взяв в качестве начальной маркировки вектор $\mu^0 = (2\ 0\ 0)$, можно найти характеристическое уравнение маркировки сети.

Запишем:

$$(m_1\ m_2\ m_3)\cdot\begin{pmatrix}1\\1\\2\end{pmatrix} = (2\,0\,0)\cdot\begin{pmatrix}1\\1\\2\end{pmatrix},$$

где m_i, $1 \le i \le 3$, являются компонентами вектора μ, откуда следует $m_1 + m_2 + 2m_3 = 2$. Целочисленными решениями этого уравнения при условии $m_i \ge 0$, $1 \le i \le 3$ будут маркировки: 200, 110, 020 и 001.

Знание характеристического уравнения множества достижимых маркировок инвариантной СП может помочь также в проверке наличия тупиковых маркировок. Условие тупиковой маркировки тождественно условию невозможности срабатывания ни одного из переходов СП. С целью нахождения такого условия используется матричное представление функции α (функции инцидентности входных позиций и переходов). Так, для нашего примера

$$\alpha = \begin{Vmatrix}1 & 0 & 0\\1 & 1 & 0\\0 & 0 & 1\end{Vmatrix}.$$

Легко видеть, что строки этой матрицы задают минимальные векторы маркировок для срабатывания переходов. Очевидно, что любая нетупиковая маркировка должна покрывать хотя бы одну из строк данной матрицы. Следовательно, условие тупика для данной матрицы может быть задано следующим логическим выражением:

$$(m_1 < 1) \wedge ((m_1 < 1) \vee (m_2 < 1)) \wedge (m_3 < 1),$$

которое легко преобразуется к виду:

$$(m_1 < 1) \wedge (m_3 < 1).$$

Решив совместно систему:

$$\begin{cases} m_1 + m_2 + 2m_3 = 2 \\ m_1 < 1 \\ m_3 < 1 \end{cases},$$

получим единственное значение $\mu = (0\,2\,0)$, которое соответствует тупиковой маркировке.

Рассмотренные примеры показывают эффективность и наглядность метода инвариантов для решения задачи структурного анализа сетей Петри. К настоящему времени появилось достаточно много алгоритмов поиска инвариантов, большинство из которых основаны на методах целочисленного линейного программирования. Однако ввиду своей сложности они не получили дальнейшего развития, поскольку на их фоне появились более эффективные эвристические методы. Коротко характеризуя их, можно отметить, что они представляют собой варианты циклической процедуры преобразования строк и столбцов расширенной матрицы инцидентности сети. Процедура выполняется до тех пор, пока не останется матрица, строки которой представляют собой порождающее семейство множества инвариантов. Любой инвариант далее может быть получен как линейная комбинация инвариантов из порождающего семейства. В случае работы с *p*-инвариантами выявляются консервативные составляющие, если же это *t*-инварианты, то они определяют повторяющиеся последовательности срабатываний переходов (бесконечные следы). Результаты поиска инвариантов могут далее использоваться для установления более тонких свойств (например, различных видов справедливости).

Решение задачи достижимости

В заключение остановимся на решении задачи достижимости матричным методом, базирующимся на использовании фундаментального уравнения (2.2), которое соответствует утверждению о том, что если маркировка μ^k достижима из μ, то уравнение (2.2) имеет решение в виде вектора $\overline{\sigma}$ в неотрицательных целых числах. Обратное, к сожалению, справедливо лишь тогда, когда (2.2) не имеет решения в неотрицательных целых числах, что означает, μ^k недостижима из μ. Сказанное проиллюстрируем двумя примерами.

В качестве первого примера рассмотрим сеть Петри, приведенную на рис. 2.11б. Для нее имеем:

$$\mu^k = (1010) + \overline{\sigma} \cdot \begin{Vmatrix} 0 & -1 & -1 & 0 \\ 0 & +2 & +1 & -1 \\ 0 & 0 & -1 & +1 \end{Vmatrix}.$$

Для определения того, является ли достижимой маркировка $\mu^k = (1801)$, необходимо подставить μ^k в левую часть этого уравнения и найти $\overline{\sigma}$, значение которого интерпретируется как набор чисел срабатываний переходов СП. Данное уравнение имеет решение $\overline{\sigma} = (045)$. Оно, в частности, соответствует последовательности переходов

$$\sigma = t_3 t_2 t_3 t_2 t_3 t_2 t_3 t_2 t_3 .$$

Если же взять $\mu^k = (1701)$, то оказывается, что уравнение не имеет решения и, следовательно, μ^k недостижима.

Данный подход, к сожалению, не универсален по двум причинам. Первая причина – неизвестно, как в общем случае восстанавливать последовательность срабатываний переходов σ по вектору $\overline{\sigma}$. Вектор $\overline{\sigma}$ представляет собой характеристику множества возможных последовательностей, среди которых могут быть как допустимые, так и не допустимые для данной СП. Вторая причина состоит в том, что существование решения уравнения является лишь необходимым, но не достаточным условием достижимости.

Например, для СП, представленной на рис. 2.17, решением уравнения

$$(0001) = (1000) + \overline{\sigma} \cdot \begin{vmatrix} -1 & +1 & -1 & 0 \\ 0 & -1 & +1 & +1 \end{vmatrix}$$

является $\overline{\sigma} = (1\ 1)$. Это решение соответствует двум последовательностям $t_1 t_2$ и $t_2 t_1$. Однако ни та, ни другая не являются допустимыми, поскольку СП не может выйти из начальной маркировки (1000).

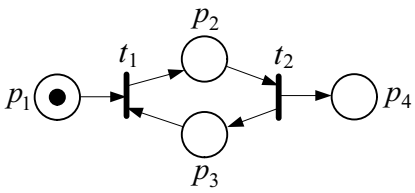


Рис. 2.17. Сеть Петри, для которой решение матричного уравнения существует, но не выполняется достижимость позиции p_4

Приведенные примеры демонстрируют одновременно как привлекательность, так и ограниченность матричного подхода.

2.7. Функциональные возможности

Теория формальных грамматик и языков является относительно независимой частью курса «Теория вычислительных структур» либо читается в составе курса «Формальные языки и автоматы». Полностью эта теория здесь не описана, но некоторые сведения из нее понадобятся, чтобы выяснить вопрос о функциональных возможностях сетей Петри (с целью сохранения "автономности" материала, приведенного в этом разделе).

Сведения из теории формальных грамматик и языков

Определим *формальную грамматику* как четверку $G = < N, T, S, P >$, **где:**

- N – **конечное множество нетерминальных символов (нетерминалов), обозначаемых обычно прописными латинскими буквами;**
- T – **конечное множество терминальных символов (терминалов), обозначаемых строчными латинскими буквами,**
- S – **начальный нетерминал**, $S \in N$;
- P – **множество *продукций* (правил) вида** $\alpha \to \beta$, $\alpha, \beta \in (N \cup T)^*$ (здесь * используется для обозначения множества, возможно бесконечного, слов, или цепочек символов, в алфавите $N \cup T$) .

Может потребоваться также введение обозначений: $\varnothing$ – пустое слово и Λ – пустое множество.

Пример формальной грамматики: $N = \{A, B, S\}$, $T = \{a, b\}$, S – начальный нетерминал, $P = \{S \to AB, A \to aA, A \to a, B \to Bb, B \to b\}$.

Продукция $\alpha \to \beta$ является правилом преобразования строки. Она содержит левую часть α как образец для распознавания преобразуемой цепочки и правую часть β, показывающую подцепочку, которой можно заменить образец α . Порядок таких замен (подстановок) и их число могут быть произвольными, за исключением необходимости инициации процесса путем использования продукций с начальным нетерминалом в левой части. Говорят, что цепочка γ *непосредственной выводится* из цепочки μ (обозначается $\mu \Rightarrow \gamma$) в грамматике G, если применение продукции $(\alpha \to \beta) \in P$ к цепочке $\mu = \sigma\alpha\tau$ приводит к цепочке $\gamma = \sigma\beta\tau$, где σ и τ – произвольные, возможно пустые, цепочки. Говорят также, что цепочка γ *выводится* из цепочки μ $(\mu \Rightarrow \gamma)$ в грамматике G, если имеется последовательность цепочек

$\omega_0, \omega_1, ..., \omega_{n-1}, \omega_n$ такая, что $\mu = \omega_0$, $\gamma = \omega_n$ и для каждого $0 \le i \le n$ имеет место $\omega_i \Rightarrow \omega_{i+1}$ (т.е. $\mu \Rightarrow \omega_1 \Rightarrow ... \Rightarrow \omega_{n-1} \Rightarrow \gamma$) с $\omega_i \in (N \bigcup T)^* \setminus \Lambda$ (вывод из цепочки, принадлежащей пустому множеству, запрещен). Список $\{\omega_i\}$ и есть вывод γ в грамматике G. Таким образом, отношение $\overset{*}{\Rightarrow}$ является транзитивным замыканием отношения $\Rightarrow$.

Типичный вывод в приведенном выше примере:

$$S \Rightarrow AB \Rightarrow aAB \Rightarrow aABb \Rightarrow aaBb \Rightarrow aabb,$$

т.е. $S \overset{*}{\Rightarrow} aabb$. Цепочку вида *aabb* будем записывать как a^2b^2, т.е. запись вида q^i означает i-кратную конкатенацию символа q (последовательность вида $\underbrace{q...q}_{i}$).

Множество цепочек, которые могут быть порождены при выводе, образуют язык L(G), определяемый грамматикой G, т.е. множество предложений (сентенций), выводимых в G из S и содержащих только терминалы:

$$L(G) = \{\omega \in T^* \mid S \overset{*}{\Rightarrow} \omega\}.$$

Цепочки, образуемые в процессе вывода и содержащие нетерминалы, называются сентенциальными формами. Так, *aAB* и *aabb* в рассмотренном примере – *сентенциальные формы*, а a^2b^2 – *сентенция*.

Предложенная Хомским классификация языков опирается на ограничения, накладываемые на порождающие их грамматики. Она приведена в табл. 2.3. В этой таблице $|\alpha|$ обозначает длину цепочки, т.е. число содержащихся в ней символов (длина пустой цепочки равна 0).

- **В языках типа 0** никакие ограничения на виды продукций не накладываются.
- **В языках типа 1** (контекстно-зависимые) единственным ограничением является невозможность уменьшения длины цепочки после применения продукционного правила.
- **В языках типа 2** (контекстно-свободные) левая часть цепочки представляет собой только один нетерминал.
- **В регулярных языках (выражениях) – языки типа 3** – соблюдается то же ограничение, что и в КС–языках плюс дополнительно правая часть цепочки может быть лишь одним терминалом или конкатенацией одного терминала и одного нетерминала (праволинейная грамматика), или конкатенацией одного нетерминала и одного терминала (леволинейная грамматика).

Иерархия Хомского для формальных языков. Таблица 2.3

Типы	Наименование типа языка	Вид продукции и ограничения	Распознающее устройство
0	Общерекурсивный	$\alpha \to \beta$ $\alpha, \beta \in (N \bigcup T)^*$ $\alpha \notin \Lambda$	Машина Тьюринга
1	Контекстно-зависимый (КЗ)	$\sigma\alpha\tau \to \sigma\beta\tau$ $\sigma, \tau \in (N \bigcup T)^*$ $\alpha, \beta \in (N \bigcup T)^* \setminus \Lambda$ $\lvert\alpha\rvert \le \lvert\beta\rvert$	Линейно-связный автомат
2	Контекстно-свободный (КС)	$A \to B$ $A, B \in (N \bigcup T)^* \setminus \Lambda$ $A \in N$	Магазинный автомат
3	Регулярный	(праволинейная) $\begin{cases} A \to aB \\ A \to a \end{cases}$ или (леволинейная) $\begin{cases} A \to Ba \\ A \to a \end{cases}$ $A \in T, B \in N$	Конечный автомат

В последнем столбце табл.2.3 указан тип *абстрактной машины*, которая может *распознавать* (*интерпретировать*) язык соответствующего типа.

Пример грамматики, порождающей язык 1:

$N = \{B, C, S\}, T = \{a, b, c\},$

$S \to aSBC \mid aBC,$

$CB \to DC,$

$aB \to ab,$

$bB \to bb,$

$bC \to bc,$

$cC \to cc.$

Связка | в первом правиле (продукции) является металингвистической и интерпретируется как ИЛИ. Такая запись, принятая в нормальной форме Бекуса-Наура (БНФ), используется для сокращенной записи двух продукций: $S \to aSBC$ и $S \to aBC$.

Эта грамматика порождает цепочки вида $a^n b^n c^n$, $n \geq 1$. Следует заметить, что тот же язык порождается и другой (эквивалентной) грамматикой с нетерминальным алфавитом $N = \{A, B, S\}$, тем же терминальным алфавитом и системой продукций:

$S \to Abc$,

$Ab \to aAbB$,

$Bb \to bB$,

$Bc \to bcc$,

$A \to a$.

Пример грамматики, порождающей язык типа 2:

$a^n b^n$, $n \geq 1$, был приведен ранее.

Пример регулярной грамматики:

$N = \{B, S\}$, $T = \{a, b\}$,

$S \to aS \mid aB$,

$B \to bB \mid b$.

Она порождает язык $a^m b^n$, $m,n \geq 1$.

Ограничения, приведенные в табл. 2.3, носят иерархический характер: они усиливаются по мере возрастания номера типа языка. *Иерархия Хомского* классифицирует языки по типу порождающих их грамматик: язык типа i может порождаться грамматикой типа i, но не типа i+1, i = 0, 1, 2. Иначе, язык типа i является строгим подмножеством языка типа i–1, что отражено в диаграмме, изображенной на рис. 2.18.

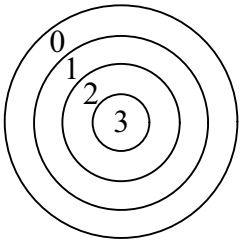


Рис. 2.18. Иерархия языков по Хомскому

Генерирование формальных языков сетями Петри

Для исследования моделирующей способности СП и сравнения ее с возможностями других, известных в теории автоматов объектов (машин Тьюринга, конечных автоматов и др.), вновь прибегают к интерпретации СП.

Через $S^{\Sigma} =< P, T, E, \mu^0, \mu^r >$ **обозначим класс** ***помеченных*** **СП, удовлетворяющий следующим ограничениям:** 1) начальная маркировка μ^0 представляет собой вектор, одна компонента которого равно единице, а остальные нулю; 2) выделена некоторая финальная маркировка μ^r из множества *M* допустимых маркировок. Образуем двойку $< S^{\Sigma}, \Psi >$, в которой Ψ – *интерпретация* S^{Σ}, а также двойку $< \Sigma, \sigma >$, в которой Σ – *конечный алфавит* (множество символов, включающее в общем случае пустой символ $\varnothing$), а σ – *функция наименований*, отображающая множество переходов СП в алфавит Σ, $\sigma : T \to \Sigma$.

Как уже известно, функционирование СП порождает некоторые последовательности (траектории) срабатываний переходов. Поскольку функция наименований ставит в соответствие каждому переходу СП некоторый символ из Σ, данная сеть порождает некоторые строки (последовательности символов), каждая из которых соответствует одной возможной траектории. В силу принятого ограничения все строки должны начинаться символами, именующими переходы, возбужденные при начальной маркировке. Говорят, что множество строк, соответствующих данной СП, *образует ее язык*. Тогда множество всевозможных СП, удовлетворяющих ограничениям, определяет *языки сетей Петри* или, короче, СП-*языки*.

Рассмотрим сеть Петри, представленную на рис. 2.19а. Примем, что $\mu^0 = 1000$ (первоначально маркер находится в позиции p_1) и что $\mu^r = 0001$ (результирующий маркер должен оказаться в позиции p_4). Диаграмма маркировок этой сети Петри бесконечна и часть ее показана на рис. 2.19б. Начальная и финальная маркировки выделены прямоугольниками.

Поскольку нас интересуют только траектории, начинающиеся в μ^0 (инициатор) и кончающиеся в μ^r (результант), то следует отсечь те участки траектории, которые ведут в тупиковые маркировки из множества $R^* = \{0101, 1201, 0301, ...\}$ (они нас не интересуют). Иными словами, нужно произвести редукцию по результантам из R^*. В результате получим рис. 2.19в. Пусть функция наименований имеет вид: $\sigma(t_1) = a, \sigma(t_2) = c, \sigma(t_3) = b, \sigma(t_4) = d$. Легко убедиться в том, что рассматриваемая СП порождает язык $a^n c b^n d$, $n \geq 0$.

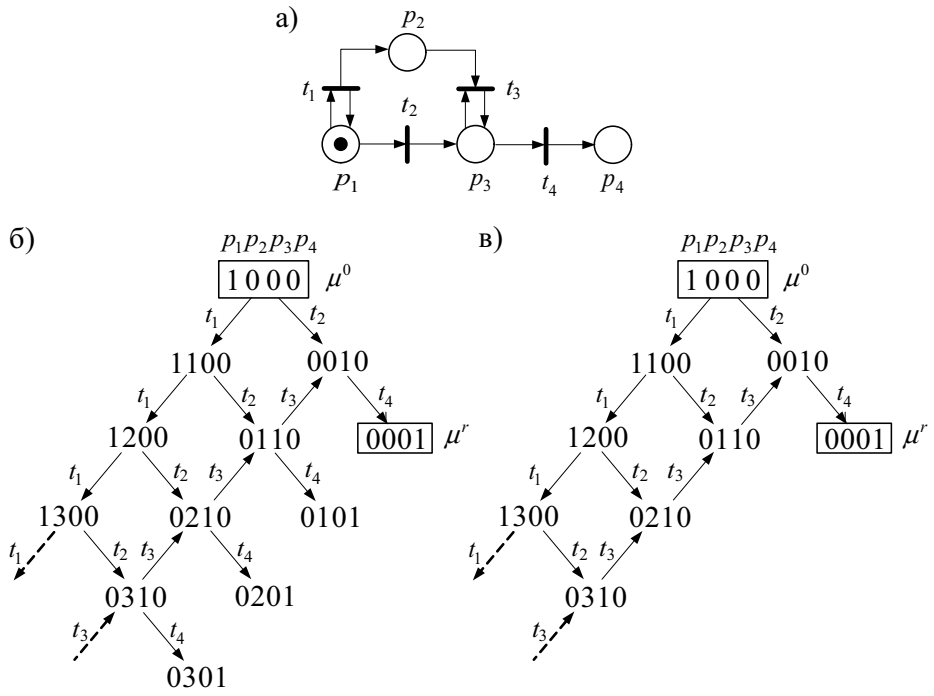


Рис. 2.19. а) Сеть Петри, б) порождаемая ею диаграмма маркировок, в) редукция этой диаграммы по всем тупиковым маркировкам, кроме μ^r

Теперь выясним, как влияет выбор функций наименований на порождаемые СП-языки. Прежде всего, заметим, что **можно выделить три подкласса функций наименований:**

- ***бесповторные*** – когда каждый переход именуется новым символом, т.е. такие наименования, что $\sigma(t_i) \neq \sigma(t_j)$ для $i \neq j$;
- ***повторные*** – когда разные переходы могут быть обозначены одинаково;
- и наименования, в которых некоторые переходы обозначаются пустым символом ∅ (в этом случае символ ∅ входит в алфавит Σ, но не входит ни в одну из порождаемых СП строк).

Очевидно, что разные подклассы функций наименований для одной и той же СП могут приводить к разным СП-языкам.

Действительно, пусть для СП, изображенной на рис. 2.19а, с диаграммой маркировок на рис. 2.19б функции наименований повторны, причем $\sigma(t_1) = \sigma(t_2) = a$ и $\sigma(t_3) = \sigma(t_4) = b$. Тогда данная СП порождает язык $\{a^n b^n \mid n > 0\}$. Если же функции наименований имеют вид: $\sigma(t_1) = \varnothing$ (переход t_1 именует-

ся пустым символом), $\sigma(t_2) = c$, $\sigma(t_3) = b$ и $\sigma(t_4) = d$, то СП порождает язык, представленный регулярным выражением cb^*d.

В приведенных примерах единственная маркировка 0001 является результирующей. Вообще говоря, **к результирующим маркировкам могут быть отнесены маркировки, определяемые следующими четырьмя способами:**

1. любая одна маркировка из множества допустимых;
2. множество маркировок, не меньших любой маркировки из множества заданных;
3. все тупиковые маркировки (т.е. маркировки, при которых не возбужден никакой переход и недостижима никакая другая маркировка);
4. все множество допустимых маркировок.

Возвратившись к рис. 2.19а и рис. 2.19б и рассматривая бесповторную функцию наименований, положим, что результирующими маркировками являются:

а) единственная маркировка 0010;

б) любая маркировка, не меньшая 0010;

в) все тупиковые маркировки;

г) все маркировки, достижимые из 1000.

Тогда получим соответствующие языки:

а) $\{a^n cb^n \mid n \geq 0\}$,

б) $\{a^m cb^n \mid m \geq n \geq 0\}$,

в) $\{a^m cb^n d \mid m \geq n \geq 0\}$,

г) $\{a^m cb^n \mid m \geq n \geq 0\} \bigcup \{a^m cb^n d \mid m \geq n \geq 0\}$.

В случае а) требуется редуцировать диаграмму маркировок по всем тупиковым маркировкам, поскольку результантом объявляется маркировка 0010, а попадание в любую тупиковую маркировку делает результант недостижимым.

В случае б) редукция аналогична и вследствие того, что результантами должны быть объявлены все маркировки, не меньшие 0010, т.е. маркировки 0010, 0110, 0210, 0310,..., число повторений m символа a может превысить число повторений n символа b.

В случае в) результантами объявлены тупиковые ситуации {0001, 0101, 0201, 0301, ...}, поэтому опять же имеет место $m \geq n$.

В случае г) редукция не производится, так как результантами объявляются все маркировки, кроме начальной, и множество порождаемых языков состоит из объединения языков, порождаемых в случаях а), б) и в), причем язык б) включает в себя язык а) (поэтому язык а) в явном виде не присутствует в объединении).

Мощность класса СП-языков в определенной степени позволяет судить о функциональных возможностях СП и их моделирующей способности. С другой стороны, узость языкового подхода состоит в том, что языковые представления по сути дела носят чисто последовательный характер (если не рассматривать языки, порождаемые графовыми грамматиками), ибо они не учитывают параллелизма. Эта неадекватность существенна, но не отвергает попыток использования языковых средств для анализа функциональных возможностей СП.

Результаты исследований показали, что СП-языки по своей мощности не уступают бесконечным языкам. Класс контекстно-зависимых языков включает класс СП-языков, а язык регулярных выражений и автоматные языки – суть подклассы СП-языков. Иерархия различных формальных языков показана на рис. 2.20а, где дуги означают отношение нестрогого включения. Точнее, если i-й язык включает в себя j-й, то на рисунке это изображается дугой, направленной от i к j.

Возможно, более наглядным является представление, показанное на рис. 2.20б, в котором цифрами 0 и 3 обозначены языки типа 0 и регулярные языки соответственно, а аббревиатурами КЗ, КС и СП – соответственно контекстно-зависимые языки, контекстно-свободные языки и языки, порождаемые сетями Петри.

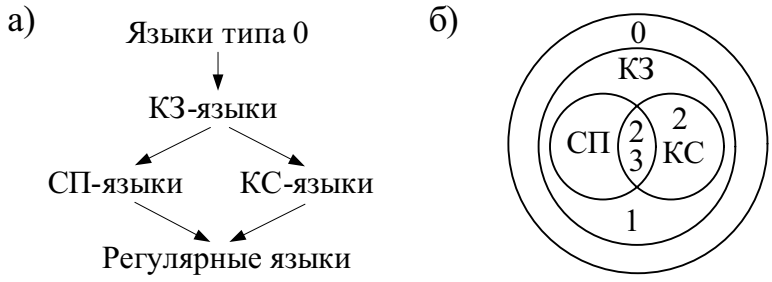


Рис. 2.20. Место СП-языков в иерархии Хомского

Из приведенных диаграмм ясно, что **существуют:**

1) СП-языки, которые не являются КС языками

2) КС языки, не порождаемые ни одной помеченной СП.

Примером языка первого типа является язык $\{a^n b^n c^n \mid n \geq 0\}$, который не относится к контекстно-свободным языкам. Этот язык порождается сетью Петри, представленной на рис. 2.21, при $\mu^r = 000001$.

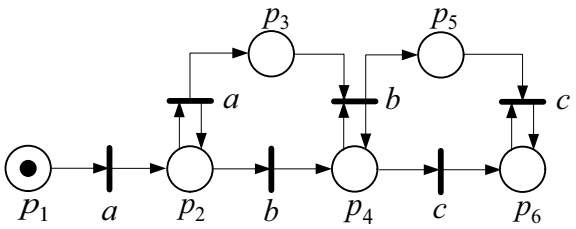


Рис. 2.21. Сеть Петри, порождающая язык $a^n b^n c^n \mid n \geq 0$, *не являющийся контекстно-свободным*

Примером второго языка является контекстно-свободный язык (КС язык), порождаемый следующей грамматикой:

$N = \{S, A\}$, $T = \{a, b, c\}$,

$S \to ScS$,

$S \to A$,

$A \to AA$,

$A \to aAb$,

$A \to \varepsilon$.

Из сказанного выше следует утверждение о том, что **класс помеченных сетей Петри строго мощнее класса конечных автоматов, несравним с классом магазинных автоматов и строго менее мощен, чем класс машин Тьюринга**.

Это утверждение важно тем, что указывает на большие функциональные возможности СП по сравнению с конечными автоматами. Этот факт, безусловно, привлекателен: применение аппарата СП является шагом вперед в проблематике теории автоматов. Однако ряд вопросов этой проблематики остается невыясненным. В частности, неизвестно, как по заданной СП-грамматике конструировать соответствующую помеченную СП. Зато ин-

тенсивно исследуются массовые алгоритмические проблемы, поставленные в теории формальных языков. Например, установлено, что *неразрешимой* проблемой является ответ на вопрос: изменится ли язык, порождаемый СП, если из нее удалить некоторый переход или из некоторой ее позиции удалить маркер? Это указывает на то, что увеличение функциональной мощности формальной системы с неизбежностью сопровождается неразрешимостью некоторых задач.

2.8. Практическое использование сетей Петри. Интерпретация сетей Петри

Сети Петри широко используются как средство моделирования поведения параллельных процессов различной природы. Объекты исследования могут изучаться посредством СП с разными целями – классификации, выявления полезных и аномальных свойств, реализации и т.п.

Описание реального объекта обязательно сопровождается *интерпретацией* СП, под которой понимается установление соответствия между специфическими понятиями определенной предметной области с базовыми понятиями СП.

Способы интерпретации могут быть различными и этому, собственно, посвящена большая часть пособия. В этом разделе ограничимся только простейшими примерами применения аппарата СП к различным областям инженерной деятельности, сильно сузив круг этих областей.

Пример спецификации конечного автомата сетью Петри

Сначала рассмотрим простейший пример использования СП как средства спецификации дискретных (логических) систем. Подробнее эта проблема затронута в гл.8, а в этом разделе продемонстрируем ее на примере описания автомата для продажи газированной воды[1].

Правила работы автомата: если он исправен (горит индикатор готовности), то при опускании в монетоприемник монеты достоинством в 1 коп. и 3 коп. наливается в стакан соответственно вода без сиропа и с сиропом. После отпуска воды автомат через некоторое время вновь приходит в состояние готовности.

[1] Пример взят из далекого прошлого. Современный читатель вряд ли когда-либо видел автомат по продаже газированной воды, тем более за такую смехотворную цену. Хотя в последнее время аппараты по продаже "газировки" все чаще стали появляться в торговых центрах.

Решение задачи, конечно, не должно вызывать затруднений. Оно приведено на рис. 2.22а, причем указанная на рисунке маркировка соответствует случаю, когда покупатель опустил в монетоприемник однокопеечную монету. Семантика позиций и переходов такова:

p_1 (p_3) – условие наличия или отсутствия в монетоприемнике монеты достоинством в 1 (3) копейки ;

p_2 – условие готовности автомата к работе;

p_4 – условие выдачи чистой газированной воды;

p_5 – условие выдачи воды с сиропом;

p_6 – окончание налива воды (с сиропом или без);

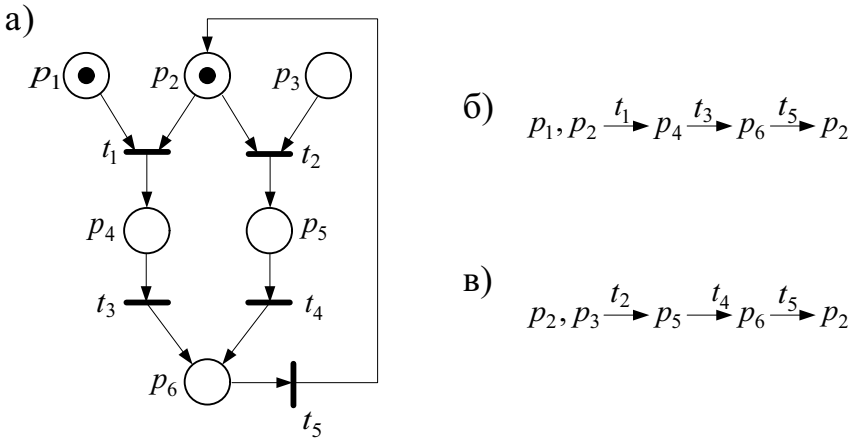


Рис. 2.22. а) Сеть Петри, моделирующая работу автомата для продажи газированной воды, и графы достижимых маркировок для случаев налива воды б) без сиропа и в) с сиропом

t_1 (t_2) – событие, срабатывание которого моделирует команду на налив чистой воды (воды с сиропом);

t_3 (t_4) – событие, срабатывание которого означает окончание налива воды (воды с сиропом);

t_5 – событие, осуществляющее частичную репозицию процесса (при срабатывании этого перехода автомат вновь готов к работе).

На рис. 2.22б и рис. 2.22в показаны диаграммы маркировок для случаев 1-копеечной и 3-копеечной монет соответственно. Таким образом, в этом примере рассматриваются две разные сети Петри, имеющие одинаковый граф, но отличающиеся двумя разными начальными маркировками: 110000 и 011000.

Использование сетей Петри в параллельном программировании

Теперь покажем, как сети Петри используются в программировании. Прежде, чем формально определить параллельную программу, условимся через $S^{П} = <P, T, E, \mu^0>$ обозначать класс СП, обладающий следующими свойствами: свободного выбора, живости, безопасности, наличия двух особых позиций (входной и выходной), не имеющих соответственно входов и выходов, соответствия μ^0 наличию маркера во входной позиции сети.

Тогда ***параллельная программа*** есть двойка $П = <S^{П}, \Psi>$, где Ψ – интерпретация $S^{П}$, порождающая программу $П$. Интерпретация Ψ, в свою очередь, есть четверка $\Psi = < A, \delta, R, \gamma >$, где $A = \{a_1,...,a_s\}$ – множество *операторов*, включающее *пустой оператор* (ε); δ – функция распределения операторов, $\delta : T \to A$; $R = \{r_1, ..., r_m\}$ – множество предикатов; γ – функция распределения предикатов из R по множеству позиций $P' \in P$, обладающих свойством свободного выбора, т.е. таких $p \in P'$, которые являются единственной входной позицией нескольких переходов $t_i, t_j, ..., t_u$, $i \neq j \neq ... \neq u$.

Пример параллельной программы приведен на рис. 2.23. Позиции p_1 и p_6 соответствуют началу и концу программы, а переходы поименованы названиями операторов, причем нижний индекс указывает номер оператора, а верхний – номер его выполнения. Позиция свободного выбора p_3 соответствует некоторому предикату, при истинности которого выполняется, скажем, a_3^1, а в противном случае – a_2^1.

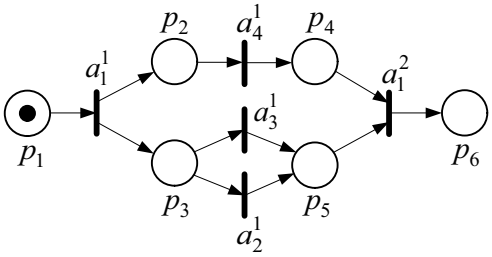


Рис. 2.23. Пример параллельной программы

Программный сегмент для вычисления арифметического выражения $z = (a + b) \times (c - d) - e / f$ приведен на рис. 2.24. Он не содержит предиката, моделирующего условный переход, а переходы именуются следующими операторами: $t_1 \to z_1 = a + b$, $t_2 \to z_2 = c - d$, $t_3 \to z_3 = e / f$, $t_4 \to z_4 = z_1 \times z_2$, $t_5 \to z = z_4 - z_3$. Переход t_0 служит для распараллелива-

ния процесса. Ему соответствует пустой оператор ε. Обратим внимание на то, что вычисление величин z_1, z_2 и z_3 производится параллельно (одновременно), промежуточный результат z_4 синхронизируется с результатом z_3, после чего появление маркера в позиции p_9 свидетельствует об окончании процесса вычисления.

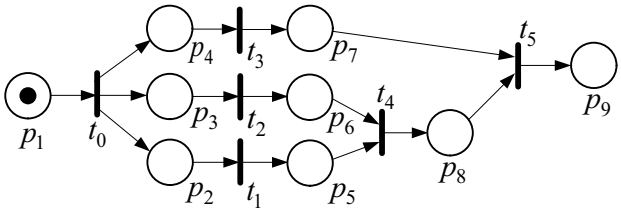


Рис. 2.24. Параллельная программа для вычисления по формуле $z = (a + b) \times (c - d) - e / f$

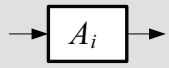
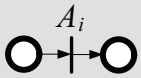
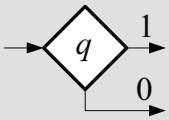
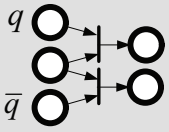
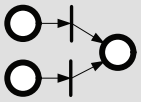
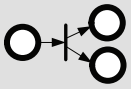
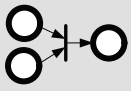
Номер рис.	Тип вершины в ПАБС	Название вершины	Фрагмент сети Петри
а)		Оператор	
б)		Условный переход	
в)		Сборка	
г)		Бифуркатор	
д)		Синхронизатор	

Рис. 2.25. Соответствие между фрагментами параллельных асинхронных блок-схем и сетей Петри

Параллельная программа, заданная сетью Петри, изображается достаточно простыми средствами и, по-видимому, такое представление программы

ничуть не хуже традиционной формы представления в виде параллельной асинхронной блок-схемы (ПАБС), имеющей пять типов операторных вершин: оператор, условный переход, сборка, бифуркатор и синхронизатор. На рис. 2.25 этим типам вершин блок-схем поставлены в соответствие фрагменты СП. Дополнительных пояснений требуют лишь две операторные вершины.

На рис. 2.25б условный переход представлен не позицией свободного выбора с инцидентными переходами (как это было сделано на рис. 2.23), а специальным фрагментом СП, в котором маркировка каждой из позиций q и $\overline{q}$ управляет засылкой маркера в общую позицию двух переходов, разрешающего возбуждение того перехода, во второй входной позиции которого находится маркер. Таким образом, свободный выбор, отражающий недетерминированность, заменен детерминированным выбором, зависящим от выполнения или невыполнения условия ($q = 1$, $\overline{q} = 0$ или $q = 0$, $\overline{q} = 1$). Синхронизатор (рис. 2.25д) необходим в параллельных асинхронных программах и не нужен в последовательных.

Вопросы, связанным с представлением параллельных программ сетями Петри, будут рассматриваться также в главах 3 и 5.

Пример моделирования аппаратных средств

Возможность моделирования аппаратных средств, которая будет обсуждаться более подробно в гл.6, может быть проиллюстрирована следующим тривиальным примером. Рассмотрим сеть Петри, представленную на рис. 2.26а.

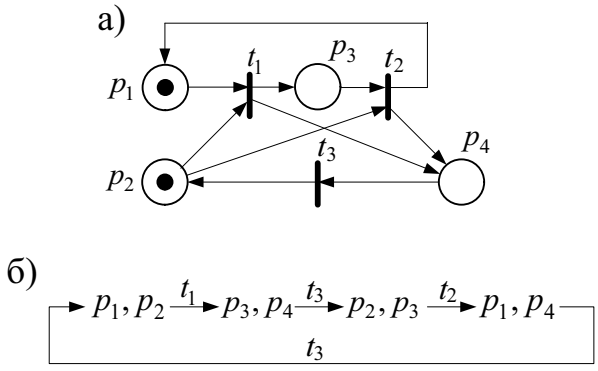


Рис. 2.26. а) Сеть Петри, моделирующая счетный триггер, б) диаграмма переходов для этой сети

Построим ее диаграмму маркировок (рис. 2.26б). Эта диаграмма является последовательной и описывается единственным трейсом $t_1t_3t_2t_3$..., из которого видно, что переходы t_1 и t_2 срабатывают вдвое реже, чем t_3. Следовательно, СП на рис. 2.26а моделирует поведение *счетного триггера*. Композиция такого типа фрагментов позволяет построить СП для счетчика, считающего по модулю 2^n, где n – число разрядов счетчика. Более подробно проблемы синтеза схем управляющих и вычислительных устройств будут рассмотрены в главах 4, 6 и 8.

Моделирование конвейерных процессов

Еще один вопрос, который здесь имеет смысл затронуть, касается *моделирования конвейерных процессов* сетями Петри.

Если природа моделируемого процесса такова, что его единственному начальному состоянию однозначно соответствует определенное финальное состояние, то в представляющей такой процесс СП также должна быть единственная результирующая маркировка. В этих случаях имеет смысл говорить о подклассе СП, задаваемых пятеркой $S = <T, P, E, \mu^0, \mu^r>$. Поскольку μ^r является тупиковой маркировкой, описание посредством СП может отражать лишь поведение однократно запускаемого процесса. Для возобновления процесса необходимо ввести механизм перехода от результирующей маркировки μ^r к начальной μ^0 или, иначе, механизм репозиции. Механизм репозиции сети $S = <T, P, E, \mu^0, \mu^r>$ естественно моделировать с помощью некоторой СП $S_1 = <T_1, P_1, E_1, \mu_1{}^0, \mu_1{}^r>$, для которой $\mu_1^r = \mu^0$ и, кроме того, либо $\mu_1^0 = \mu^r$, либо $\mu_1^0 = \mu^i$, где μ^i – текущая (промежуточная между μ^0 и μ^r) маркировка СП S из множества M допустимых маркировок. В первом случае репозицию назовем *простой*. Второй случай (*непростая* репозиция) отличается от первого тем, что в качестве маркировки μ_1^0 может быть использована текущая маркировка, отличающаяся от μ^r.

Объединение исходной СП и ее репозиции приводит к автономной СП S^A, в которой нет смысла выделять начальную и результирующие маркировки (вернее, таковыми могут являться любые маркировки из M). Для простой репозиции множество маркировок сети S^A совпадает с множеством маркировок сети S, для непростой репозиции $M^A \supset M$, так как появляются новые маркировки, не фигурирующие ни в S, ни в S_1.

В проекции S^A на S можно выделить подмножество маркировок, совпадающих с маркировками исходной СП S и среди них μ^0 и μ^r. Если на каж-

дое достижение μ^0 сеть будет отвечать достижением μ^r, то процесс в сети S^A будем называть *конвейерным*. Это определение не является формальным; необходимые детали читатель уяснит из приведенного ниже примера. Конвейерный процесс моделирует не только процесс обработки данных, но и механизм их подкачки.

Конвейерные процессы и соответствующие им СП моделируют один из эффективных методов введения параллелизма, широко используемый как на уровне программного обеспечения, так и на уровне аппаратной поддержки.

В основе конвейерного принципа обработки информации лежит идея такой загрузки задачами последовательности операторов (модулей системы), при которой каждая следующая задача поступает на решение еще до того, как завершилось решение предыдущей задачи.

При конвейерной обработке каждый оператор может быть инициирован каждый раз, когда непосредственно предшествующий и непосредственно следующий за ним операторы завершили выработку результатов. При этом предшествующий оператор подготовил информацию для обработки данным оператором очередной задачи, а последующий оператор использовал информацию, выработанную данным оператором при решении предыдущей задачи.

В качестве примера, доказывающего возможность повышения эффективности за счет конвейеризации, приведем модель конвейера для реализации последовательной работы трех операторов (см. рис. 2.27а).

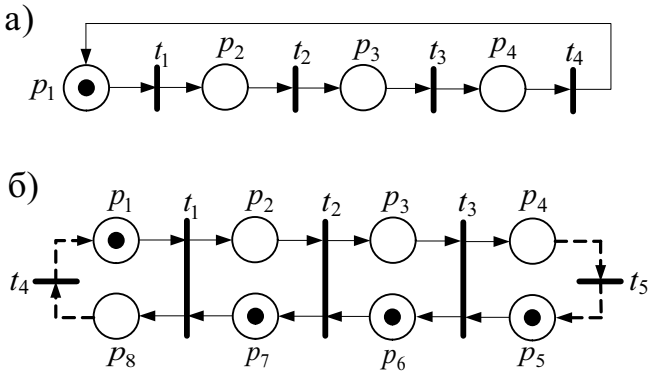


Рис. 2.27. Пример репозиции сети Петри: а) последовательный процесс с простой репозицией; б) его конвейерный аналог

В ней операторы моделируются переходами. Расстановка маркеров по позициям соответствует начальной маркировке. СП на рис. 2.27а моделирует процесс решения одной задачи. Каждая следующая задача может быть инициирована после последовательности $t_1t_2t_3t_4$ срабатывания переходов, т.е. после того, как будет решена предыдущая задача (маркер при этом находится в позиции p_4). Репозиция осуществляется фрагментом СП, включающим переход t_4 и инцидентные ему позиции p_4 и p_1.

На рис. 2.27б уже моделируется решение двух задач, а в различных фазах решения могут находиться одновременно до трех задач. Следующая задача может быть инициирована после срабатывания переходов t_1, t_2 и t_4, когда результатов решения еще нет (отсутствует маркер в позиции p_4). Здесь непростая репозиция осуществляется фрагментом СП, состоящим из переходов t_4 и t_5 с инцидентными позициями p_8, p_1 и p_4, p_5 соответственно.

Системы с конвейерным способом обработки информации находят все большее применение. В литературе на русском языке используются синонимические названия – *конвейерные*, *магистральные*, *желобковые*, *водопроводные* и, наконец, *пайплайновые* (калька с англ. *pipeline* – трубопровод).

2.9. Упражнения

1. Изобразите граф СП, соответствующий следующему описанию: $P = \{p_1, ..., p_5\}$, $T = \{t_1, ..., t_4\}$, $I(t_1) = \{p_1\}$, $O(t_1) = \{p_2, p_3, p_5\}$, $I(t_2) = \{p_2, p_3, p_5\}$, $O(t_2) = \{p_3\}$, $I(t_3) = \{p_4\}$, $O(t_3) = \{p_2, p_5\}$, $I(t_4) = \{p_5\}$, $O(t_4) = \{p_4\}$, $\mu^0 = \{p_1, p_5\}$. Постройте диаграмму маркировок для этой СП. Указание: в качестве маркеров используйте монеты, достоинства которых совпадают с числом маркеров в позициях.

2. Определим понятие *реверсной* СП, т.е. сети, которая образована изменением направления всех дуг исходной СП на противоположное. Приведите пример СП и постройте для нее реверсную сеть.

3. Определим понятие СП, *двойственной* заданной. Это такая сеть, которая образована из исходной заменой переходов позициями и позиций переходами. Изобразите СП, двойственную описанной в упражнении 1.

4. Определите понятия *антидвойственной* и *самодвойственной* СП по аналогии с соответствующими понятиями для булевых функ-

ций. Напомним, что для функции $f(x_1, x_2, ..., x_n)$ двойственная определяется как $\bar{f}(\bar{x}_1, \bar{x}_2, ..., \bar{x}_n)$, для антидвойственной выполняется $f(x_1, x_2, ..., x_n) = f(\bar{x}_1, \bar{x}_2, ..., \bar{x}_n)$, для самодвойственной – $f(x_1, x_2, ..., x_n) = \bar{f}(\bar{x}_1, \bar{x}_2, ..., \bar{x}_n)$. Приведите примеры таких СП.

5. Покажите на примерах, какие из введенных в упражнениях 2 – 4 операций над СП подчиняются *закону инволюции* (когда двукратное применение операции приводит к исходной СП).

6. Покажите, что:

а) сеть, двойственная исходной сети свободного выбора, является сетью свободного выбора,

б) сеть, двойственная маркированному графу, является автоматной СП,

в) сеть, двойственная автоматной, является маркированным графом,

г) сеть, двойственная сильно связной, является сильно связной.

7. Установите, принадлежат ли сети Петри, показанные на рис. 2.28, к классу эффективных асинхронных процессов.

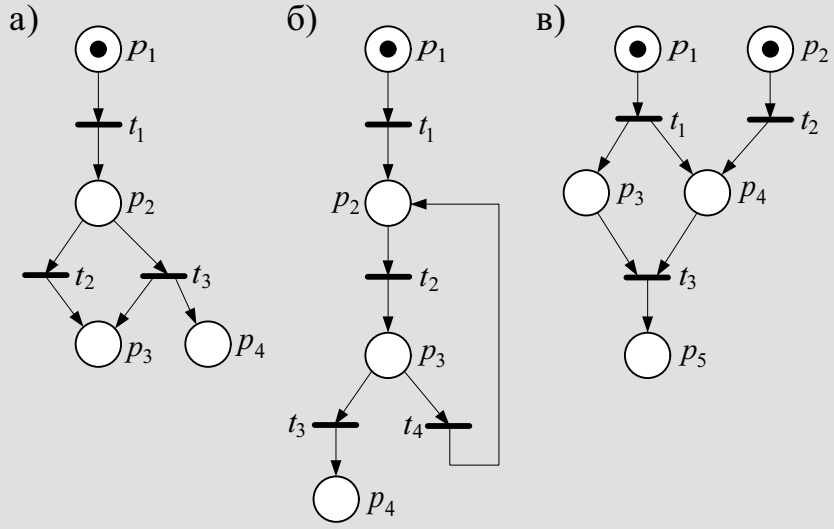


Рис. 2.28. Примеры сетей Петри

8. *Диссипативной* СП (по аналогии с понятием, используемым в теории управления) назовем сеть с множеством M достижимых маркировок,

которая, начиная функционировать из начальной маркировки, после попадания в некоторую маркировку погружается в фиксированное множество маркировок $G \in M$. Приведите пример диссипативной СП.

9. Постройте диаграмму маркировок для СП, изображенной на рис. 2.29а.

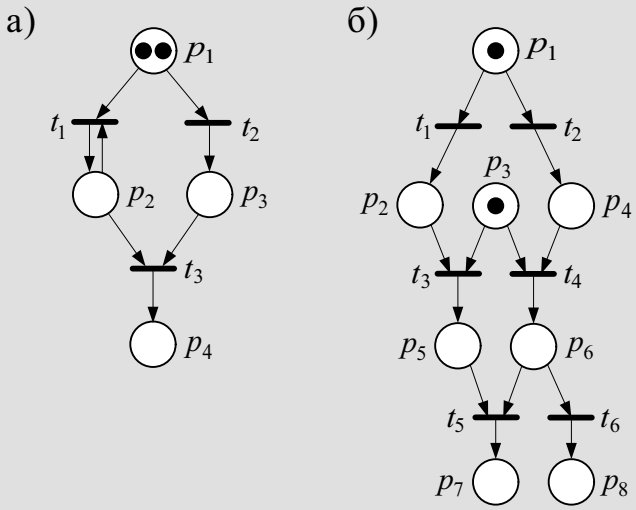


Рис. 2.29. Примеры сетей Петри

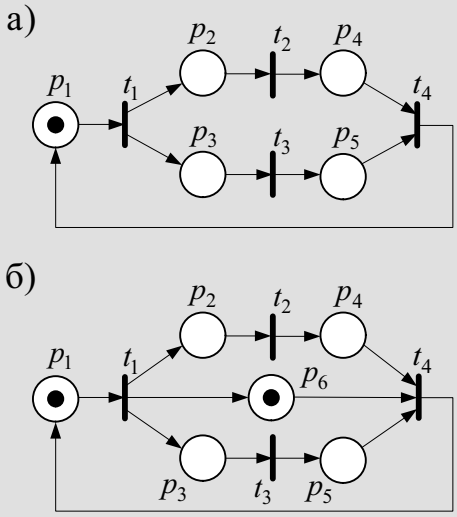


Рис. 2.30. "Почти одинаковые" сети Петри

10. Воспользуйтесь понятием протокола АП, изобразите на языке СП протокол сети, представленной на рис. 2.29б.

11. Модифицируйте СП, приведенную на рис. 2.26, таким образом, чтобы ее переход срабатывал в точности пять раз.

12. Сравните рис. 2.30а и рис. 2.30б и прокомментируйте их.

13. Сеть Петри задана матрицей инциденций:

$$C = \begin{Vmatrix} -1 & 0 & 1 \\ 1 & -1 & 0 \\ 0 & 1 & -1 \end{Vmatrix}.$$

Вычислите ее p- и t-инварианты. Для начальной маркировки $\mu^0 = 100$ установите характеристическое уравнение достижимых маркировок сети.

14. Для НСП, заданной матрицей инциденций:

$$C = \begin{Vmatrix} -1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & -1 & 1 \end{Vmatrix}$$

Проверить условия согласованности и инвариантности, живости и ограниченности.

15. Для СП из упражнения 14 определить консервативные составляющие и ловушки.

16. Определите, для каких значений k переходы t_1 и t_4 НСП, представленной на рис. 2.31, живы при начальной маркировке $\mu^0 = \{k\,0\,0\}$.

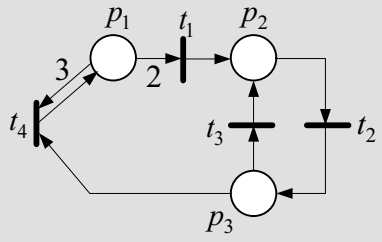


Рис. 2.31. Пример, иллюстрирующий "коварство" свойства живости

17. Исследуйте матричным методом достижимость маркировок 11110000 и 10100101 для сети Петри, показанной на рис. 2.27б.

18. Постройте диаграмму маркировок для сети Петри, показанной на рис. 2.32. Запишите реализуемое ею регулярное выражение.

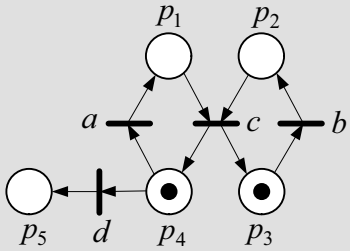


Рис. 2.32. Сеть Петри, порождающая регулярный язык

19. Для СП, моделирующей конвейерный АП (рис. 2.27б), привести оценку мощности множества M достижимых маркировок.

20. Модифицируйте СП, показанную на рис. 2.22а, таким образом,чтобы отразить работу неисправного автомата, "глотающего" монеты без выдачи воды и/или сиропа. Обратите внимание на необходимость репозиции АП. Предложите вариант полной репозиции.

21. Изобразите интерпретированные СП, реализующие следующие формулы: а) $z = a \cdot b - d / c$, б) $z = a \cdot b - a / b$.

22. Прокомментируйте изображенный на рис. 2.33 вариант реализации булевой функции $f = (x_1 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee x_2)$, представленной в конъюнктивной нормальной форме. Обобщите этот подход для случая функции $\varphi = (x_1 \vee \bar{x}_2 \vee x_3) \wedge (x_1 \vee \bar{x}_3) \wedge (x_2 \vee \bar{x}_3)$.

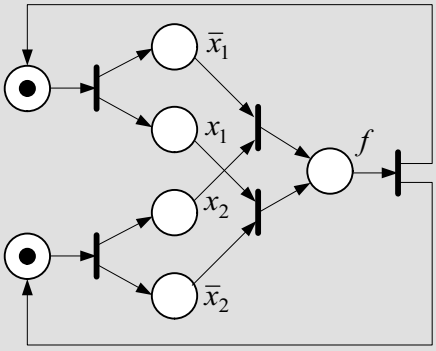


Рис. 2.33. Реализация булевой функции сетью Петри



Глава 3.

ДРУГИЕ СЕТЕВЫЕ МОДЕЛИ, РОДСТВЕННЫЕ СЕТЯМ ПЕТРИ

Какой бы проблемой не занялся ученый, в конце концов, у него все равно получается оружие.

Курт Воннегут

С момента своего появления в 1962 г. теория сетей Петри, как и любая другая теория, начала развиваться самостоятельно, причем часто направления развития были обусловлены не столько практической необходимостью, сколько внешней привлекательностью. Такие достоинства моделей СП, как возможность отражения асинхронности и параллелизма, недетерминированности процессов, динамики их функционирования, простой синтаксис и наглядность в сочетании с достаточно широкими функциональными возможностями, явились причиной популярности теории СП, что, в свою очередь, стимулировало ее развитие.

Несмотря на то, что теория СП развивалась самостоятельно, она, тем не менее, может рассматриваться не только как автономная область науки, но и как ветвь теории автоматов в ее широком понимании, занимающая ранее не исследованную нишу между конечными автоматами и машинами Тьюринга. Эта ниша крайне важна по той простой причине, что если многие массовые задачи анализа в рамках машин Тьюринга являются либо алгоритмически неразрешимыми, либо разрешимы с полиномиальной сложностью, в то время как те же задачи в рамках ординарных сетей Петри являются разрешимыми и разрешимы либо с экспоненциальной сложностью, либо, для некоторых классов сетей Петри, с полиномиальной сложностью. Полиномиальная сложность решения массовых задач анализа моделей СП чрезвычайно важна для практических применений при проектировании аппаратных и программных средств.

Появление формализма сетей Петри, изучение их функциональных возможностей, с одной стороны, позволили выявить их большую "вычислительную мощность" по сравнению с конечными автоматами и использовать этот аппарат для моделирования параллельных и распределенных вычислительных устройств. С другой стороны, возникла тенденция поиска усилений модели СП, т.е. поиска таких "петриподобных" моделей, которые являются обобщениями СП и имеют бо́льшие функциональные возможности. Все модели такого сорта получили название *сетевых моделей* из-за близости к модели СП. Был предложен целый спектр таких моделей и число исследований в этом

направлении продолжает расти. Честно говоря, нам не очень понятно стремление к расширению функциональных возможностей ординарных сетей Петри. Дело в том, что такое расширение, или обобщение, сетей Петри почти сразу же приводит к функциональным возможностям машин Тьюринга и соответственно к проблемам, связанными с массовыми задачами их анализа. Почти единственным стимулом создания новых сетевых моделей может служить увеличение компактности получаемых спецификаций. Методы анализа в таких сетях, как правило, сводятся к симуляции (моделированию) поведения и работают при малых размерностях задач.

В рамках этой книги нет нужды подробно анализировать все существующие сетевые модели. Важнее другое: проанализировать некоторые из них, получившие наиболее широкую известность, и попытаться выяснить, в каких аспектах они эквивалентны исходной модели (ординарным СП), а в каких аспектах ее превосходят, если таковые существуют. Такая попытка делается ниже, и авторы полагают, что читатель после ознакомления с материалом этой главы самостоятельно сможет проанализировать любую из встретившихся ему или новоиспеченных сетевых моделей. Однако следует предостеречь от применения чисто интуитивного подхода при сравнительной оценке моделей. Он может привести к ошибочным заключениям, что иногда имеет место. Строгие доказательства эквивалентностей и морфизмов – вещь сложная, требующая специальной подготовки и, более того, как правило, громоздкая. Как писал Марвин Минский: “Доказать эквивалентность двух или нескольких определений всегда трудно, если определения вытекают из различных подходов и диктуются разными побудительными мотивами”. Поэтому авторы не имели возможность строго доказывать некоторые результаты, тем более что для этого необходим экскурс в теорию вычислимости, с которой читатель может ознакомиться дополнительно (что весьма желательно).

Еще раз подчеркнем, что сколько-нибудь существенные обобщения СП приводят к конструкциям, эквивалентным по функциональным возможностям машинам Тьюринга. Сетевые модели с универсальными моделирующими возможностями, конечно, интересны с теоретической точки зрения. Но возможности их практического использования ограничены в силу того, что некоторые задачи анализа спецификаций, заданных на языках сетевых моделей, являются алгоритмически неразрешимыми.

Перейдем к описанию некоторых ставших уже традиционными сетевых моделей, часть из которых равномощны ординарным сетям Петри.

3.1. Сети со взвешенными дугами

Сеть со взвешенными дугами **представляет собой сеть Петри, в которой допускаются мультидуги. Для того, чтобы не изображать несколько дуг вместо одной, эта дуга помечается целым числом, указывающим ее кратность, или вес.**

Формальное определение ординарной СП как четверки $< P, T, E, \mu^0 >$ дополним весовой функцией W такой, что $W : E \to N^+$, где N^+ – множество положительных целых чисел. Смысл введения этой функции в данном случае состоит в задании кратности дуг. **Для такой модификации СП, получившей название сетей со взвешенными дугами, правило возбуждения перехода** t **таково: если каждая из входящих дуг, инцидентных переходу, имеет вес (кратность)** W_i, **то соответствующая позиция** $p_i \in I(t),\ t \in O(p_i)$, **должна содержать не менее** W_i **маркеров**. Пример сети с взвешенными дугами представлен на рис. 3.1. Предполагается, что вес непомеченных дуг равен единице.

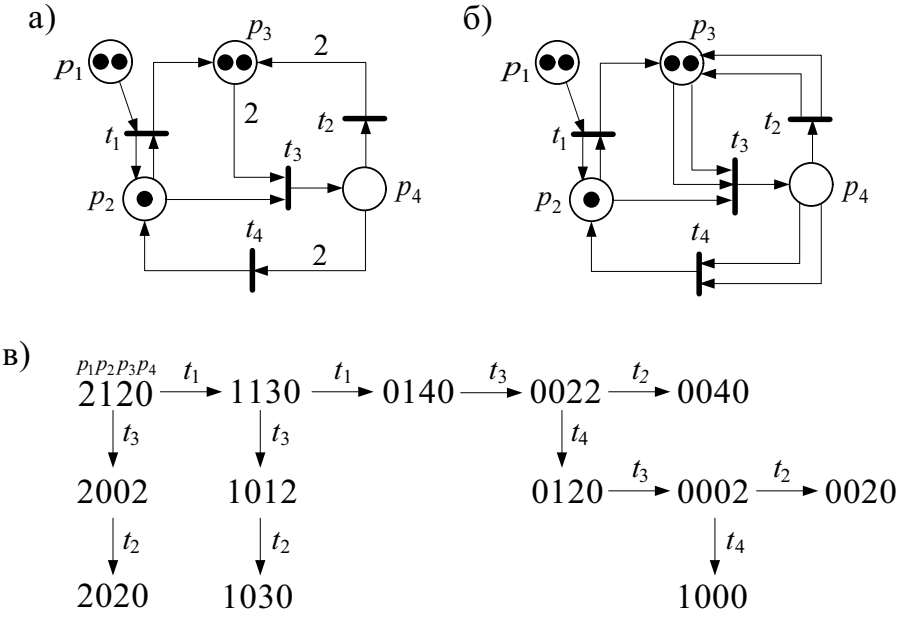


Рис. 3.1. а) Пример сети Петри со взвешенными дугами, б) альтернативное представление сети, в) диаграмма маркировок сети

Очевидно, что сети с взвешенными дугами могут быть представлены двудольным ориентированным *мультиграфом*, т.е. графом, в котором позиция с переходом или переход с позицией связан не одной, а несколькими дугами с весом единица, число которых равно весу дуги. Таким образом, граф на рис. 3.1а допускает альтернативное представление, показанное на рис. 3.1б. Диаграмма маркировок этих графов представлена на рис. 3.1в.

Преобразование сети с взвешенными дугами в ординарную сеть Петри

Менее очевиден способ преобразования произвольной сети с взвешенными дугами в ординарную сеть Петри. Он состоит в следующем:

1. Для каждой позиции $p \in P$ определяется максимальная кратность $n(p)$ дуг, инцидентных этой позиции (так, для позиции p на рис. 3.2а $n(p) = 3$).

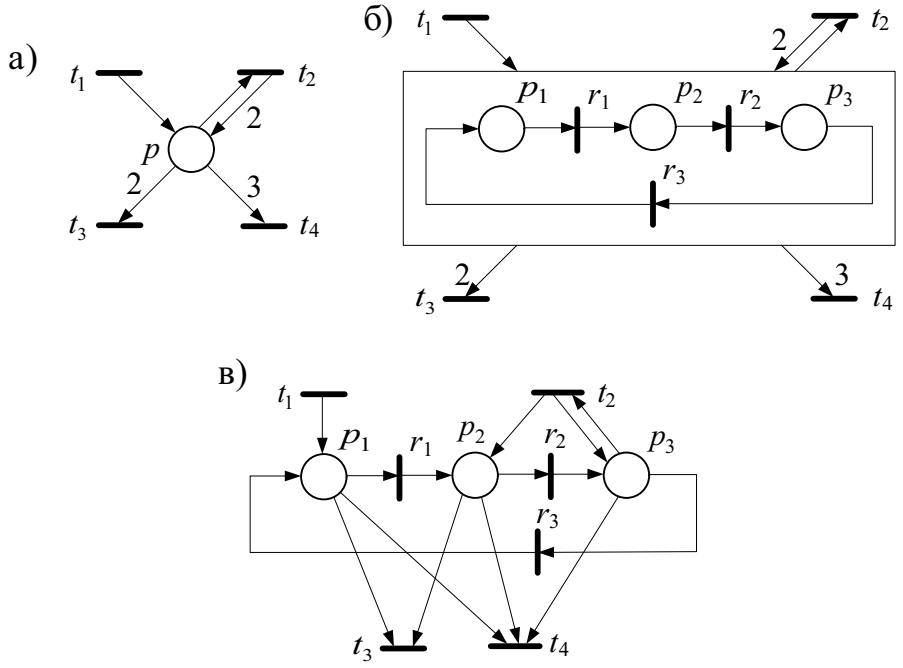


Рис. 3.2. Сведение сети с взвешенными дугами к ординарной сети Петри: а) исходный фрагмент, б) промежуточное представление, в) ординарная сеть Петри

2. Каждая позиция p с максимальной кратностью $n(p)$ представляется $n(p)$ позициями $p_1, p_2, ..., p_{n(p)}$ ординарной сети (для фрагмента рис. 3.2а это позиции p_1, p_2 и p_3).

3. Каждому переходу $t \in T$ исходной сети соответствует одноименный переход ординарной СП, но в последней при $n(p) > 1$ вводится дополнительно $n(p)$ переходов $r_1, r_2, ..., r_{n(p)}$, которые вместе с позициями $p_1, p_2, ..., p_{n(p)}$ (в данном случае r_1, r_2, r_3) объединяются в кольцо, как показано на рис. 3.2б.

4. Для каждой дуги, связывающей позицию p с входным или выходным переходом t и имеющей кратность w, заводятся w дуг, связывающих t с позициями $p_1, p_2, ..., p_{n(p)}$ (с какими именно позициями – несущественно, лишь бы t и p_i были связаны единственной дугой в одном направлении), как, например, показано на рис. 3.2в.

5. При начальной маркировке $\mu(p)$ позиции исходной сети позиция p_1 маркируется $\mu(p)$ маркерами, при этом $\mu(p_j) = 0$ для всех позиций $p_j, j \neq 1$.

Приведенный алгоритм преобразования устанавливает (конструктивно) эквивалентность сетей с взвешенными дугами и ординарных сетей Петри. Отсюда, в частности, следует, что эти модели равномощны. Однако понятно, что сети с взвешенными дугами позволяют задавать спецификации моделируемых систем более компактно, чем ординарные сети, что часто бывает желательно. Анализ спецификации может выполняться либо путем использования соответствующих алгоритмов после сведения сети с взвешенными дугами к ординарной, либо путем использования модификаций этих алгоритмов, не требующих такого сведения.

Пример спецификации химической реакции сетью с взвешенными дугами

На рис. 3.3 показан кажущийся экзотическим пример задания химической реакции между фосфором и хлором, приводящей к получению треххлористого фосфора ($2P + 3Cl_2 = 2PCl_3$). Веса дуг в этом случае интерпретируются как множители инградиентов, используемых для уравновешивания. В результате реакции при заданной начальной маркировке изымаются три маркера из позиции Cl_2 и два маркера из позиции P и два маркера заносятся в позицию PCl_3.

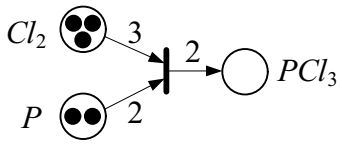


Рис. 3.3. Пример задания химической реакции на языке сетей с взвешенными дугами

Проверка отсутствия маркера в некоторой позиции

Для дальнейшего изложения будет полезно обсудить вопрос о возможности так называемой проверки на нуль, т.е. проверке отсутствия маркера в некоторой позиции. Один из подходов к решению этой задачи на СП с взвешенными дугами иллюстрируется посредством рис. 3.4. При начальной маркировке, показанной на рисунке, может сработать только переход t_3, после чего маркер появится в позиции p_4, сигнализируя, что $\mu(p) = 2$. Поскольку $\mu(p) + \mu(p') = 2$, то при симметричной начальной маркировке $\mu(p_1) = 1$, $\mu(p') = 2$ сработать может только t_1 и появление маркера в p_2 означает, что $\mu(p) = 0$.

При начальной маркировке $\mu(p) = \mu(p') = \mu(p_1) = 1$ возбужден только переход t_2 и после его срабатывания появление маркера в p_3 сигнализирует, что $\mu(p) = 1$.

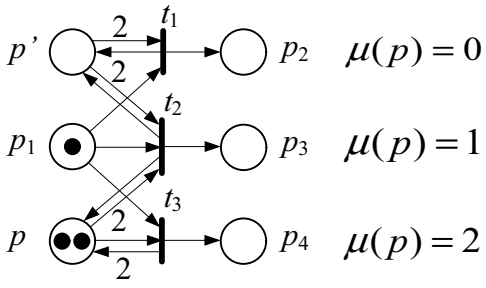


Рис. 3.4. Сеть Петри со взвешенными дугами, осуществляющая проверку на отсутствие и наличие одного или двух маркеров в позиции p.

Консервативность пары 2-инверсных позиций p и p' (всегда выполняется $\mu(p) + \mu(p') = 2$) обеспечивается тем, что эти позиции являются как входными, так и выходными для переходов t_1 и t_3. Маркер в p_1 интерпретируется как запрос проверки на 0, 1 или 2.

Такая конструкция, к сожалению, требует использования дуг с кратностью k для k-инверсных позиций, что делает этот подход малоэффективным в общем случае. Таким образом, для неограниченных СП с взвешенными дугами проверку на нуль выполнить не удается, хотя известны соответствующие конструкции с ограниченными кратностями дуг, позволяющие осуществить проверку на нуль для сетей, позиции которых могут накапливать до 2^{2^n} маркеров.

3.2. Раскрашенные сети

Желание приблизить формализм сетей Петри к некоторым реальным моделируемым системам оправдывает попытку введения маркеров разных цветов. Цвет маркера является атрибутом, позволяющим интерпретировать с его помощью типы данных, ресурсов и пр.

Раскрашенная (цветная) сеть Петри **определяется как тройка** $< PN, C, \Psi >$, **где** PN – **ординарная сеть Петри**, C – **конечное множество цветов, а функция** Ψ **задает правила возбуждения и срабатывания переходов**. Точнее, для перехода $t_i \in T$

$$\Psi(I(t_i) \times C, t_i) \subseteq O(t_i) \times C,$$

т.е. функция для заданного распределения цветов входных позиций перехода указывает распределение цветов в выходных позициях этого перехода после его срабатывания.

Пример раскрашенной СП приведен на рис. 3.5а; обозначение 1 (2) в начальной маркировке указывает на то, что изначально в позиции p_1 (p_5) находится по одному маркеру цвета 1 (2), например, красного (зеленого). С переходами t_i, $1 \le i \le 4$, связаны функции

$$\Psi(p_1^1, t_1) = p_2^1,$$

$$\Psi(p_2^1, p_4^2, t_2) = p_3^2,$$

$$\Psi(p_3^2, t_3) = p_1^1, p_5^2,$$

$$\Psi(p_5^2, t_4) = p_4^2.$$

Обозначение p_i^j говорит о том, что в позиции p_i находится маркер цвета j. Диаграмма маркировок раскрашенной СП (рис. 3.5а) приведена на рис. 3.5б.

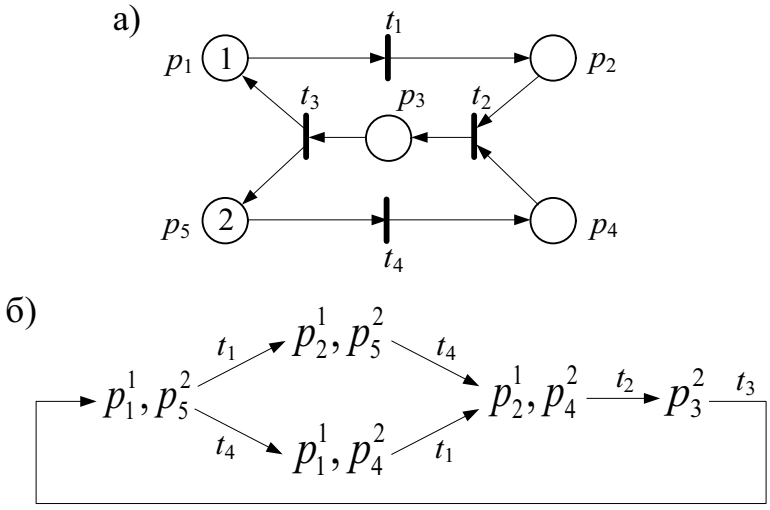


Рис. 3.5. а) Раскрашенная сеть Петри и б) ее диаграмма маркировок

Таким образом, условия срабатывания переходов раскрашенных СП индивидуальны и это несколько усложняет алгоритмы анализа. Однако **доказано, что раскрашенные СП с конечным числом цветов маркеров эквивалентны ординарным СП** и при соответствующем преобразовании описательная сложность ординарной сети может существенно возрасти.

По-видимому, здесь следует упомянуть, что развитые для ординарных СП матричные методы анализа (инварианты) сравнительно несложно обобщаются для случая раскрашенных СП.

Идея раскраски маркеров несомненно расширяет возможности компактного представления СП, хотя как указано выше, сама раскраска не приводит к расширению функциональных возможностей, свойственных ординарным СП. Тем не менее, двигаясь дальше в плане расширения описательных возможностей цветных СП, соблазнительным оказывается расширение их на случай неограниченного числа цветов, что связано с представлением данных в программировании. Кроме того, вместе с введением весьма разнообразных типов данных на маркерах, возникла потребность во введении всевозможных механизмов задания условий на переходах и дугах, определенных на этих типах данных. Все эти дальнейшие синтаксические расширения привели исследователей цветных СП к модели цветных сетей типа CP-сетей или CPN (Coloured Petri Nets), которая активно развивалась в 90-ые годы (в основном под руководством Курта Йенсена в Университете Орхуса в Дании). Развитие CPN шло в ногу с созданием и непрерывным

усовершенствованием мощнейшего пакета программ поддержки под названием Design-CPN, а с 2000 г. CPN Tools (в настоящее время поддерживаемого Технологическим Университетом Эйндховена в Нидерландах).

Итак, по своей сути, современные **Цветные Сети Петри (CP-сети) – это графический язык для построения моделей параллельных систем и анализа их свойств. CP-сети позволяют моделировать параллелизм, причинно-следственные связи и синхронизацию.** Функциональный язык программирования Standard ML предоставляет примитивы для определения типов данных, для описания обработки данных, а также для создания компактных и параметризуемых моделей.

С точки зрения технологии программирования, язык CP-сетей является дополнением к существующим языкам моделирования и методологии проектирования (например, UML) и может быть использован вместе с ними или встраиваться в них. Типичными областями применения являются протоколы связи, сетей передачи данных, распределенные алгоритмы, встроенные системы, бизнес-процессы, документооборот, производственные системы, и многоагентные системы.

Для анализа свойств моделируемых систем программы в пакете CPN Tools позволяют проводить имитационное моделирование (так называемую симуляцию). В последние годы для моделей с ограниченным множеством значений цветов, были также разработаны средства анализа достижимости, проверки на тупиковые состояния, ограниченность и т.д. Тем не менее, основное достоинство CP-сетей и пакета CPN Tools – это возможность проводить интерактивное моделирование, т.е. «прогонять» модель, подробно исследуя различные ее сценарии функционирования, и проверять, работает ли модель так, как ожидалось.

CP-сети включают понятие времени, например время, затрачиваемое на срабатывание события в системе как в детерминированном, так и вероятностном виде. Благодаря этому CP-сети могут использоваться для анализа производительности систем, исследуя такие показатели деятельности, как длительность процессов, их пропускную способность и длину очередей в системе.

При всех привлекательных качествах модельных расширений CP-сетей, платой за это являются ограниченные возможности анализа при аналогичных, по сравнению с ординарными СП, функциональных возможностях. Например, такие эффективные методы анализа как те, что базируются на развертках и частичных порядках, не дают желаемого эффекта на CP-сетях. Поэтому, использование CP-сетей для формальной верификации спецификаций не рекомендуется.

3.3. Сети с тормозящими дугами (ингибиторные сети Петри)

Существенное обобщение сетей Петри достигается введением тормозящих дуг, направленных только от позиции к переходам. Такие дуги изображаются с кружочком на конце (вместо стрелки). Тормозящий вход перехода t реализует функцию запрета его срабатывания, если в позиции, из которой исходит тормозящая дуга, имеется, по крайней мере, один маркер. В противном случае (если позиция пуста) этот вход игнорируется.

Рассмотрим простой пример сети Петри с тормозящими дугами (иначе их называют ингибиторными дугами), показанный на рис. 3.6а.

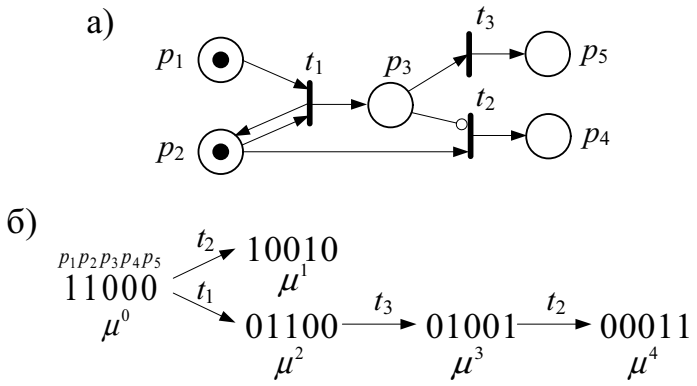


Рис. 3.6. а) Ингибиторная сеть Петри и б) ее диаграмма маркировок

В этой СП имеется только одна тормозящая дуга, связывающая p_3 с t_2. При начальной маркировке возбуждены два перехода (t_1 и t_2). Если первым сработает t_2 (а это возможно, так как позиция p_3 пуста), то СП оказывается в тупиковой маркировке μ^1. Если же первым сработает t_1, то при маркировке μ^2 наличие маркера в p_3 запрещает срабатывание t_2 до тех пор, пока не сработает t_3 и не установится маркировка μ^3. После этого разрешается срабатывание перехода t_2 и СП оказывается в тупиковой маркировке μ^4.

Попытаемся выяснить, каковы функциональные возможности СП с ингибиторными дугами (ингибиторных СП) сравнительно с одинарными СП. Рассмотрим типичный фрагмент СП с тормозящими дугами, показанный на рис. 3.7 слева, предположив, что эта СП безопасна (в единственной позиции p этой сети не может появиться более одного маркера). Заметим, что

при наличии маркера в p (в результате срабатывания перехода t_1) сначала может сработать только переход t_3 (при соответствующих условиях), а затем переход t_2. Если же позиция p пуста (переход t_1 не сработал), то наоборот, переход t_2 может быть возбужден, а t_3 нет. Ясно, что переходы t_2 и t_3 находятся в состоянии конфликта, управляемого переходом t_1.

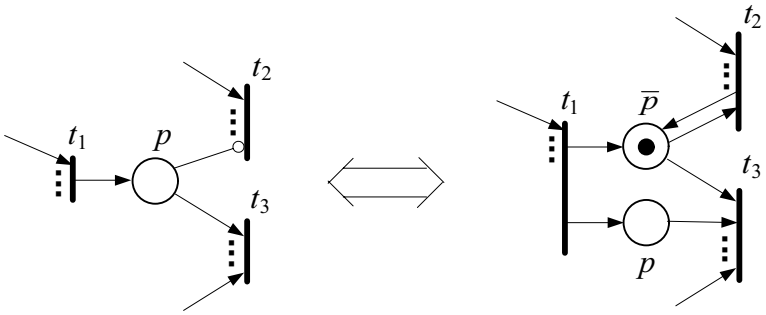


Рис. 3.7. Эквивалентное преобразование фрагмента безопасной сети Петри с тормозящими дугами в ординарную сеть Петри

Траектории срабатываний переходов в рассматриваемом фрагменте могут быть такими: t_2, t_1, t_3 или t_1, t_3, t_2. В правом фрагменте на рис. 3.7 позиция p заменена парой позиций p и $\overline{p}$ такой, что маркер может находиться либо в p, либо в $\overline{p}$, но не в обеих вместе. Позиция $\overline{p}$ (будем называть ее позицией, *инверсной* позиции p) является одновременно входящей и выходящей для перехода t_2 и выходящей для перехода t_3. Остальные связи в этом фрагменте повторяют связи исходного фрагмента. Легко заметить, что траектории срабатываний переходов правого фрагмента такие же, как в левом фрагменте, из чего следует сделать вывод об эквивалентности фрагментов.

Приведенные выше соображения позволяют сформулировать следующее утверждение:

Безопасные сети Петри с тормозящими дугами эквивалентны (равномощны) ординарным безопасным сетям Петри.

Обратим внимание на прием, состоящий в замене позиций парой взаимно инверсных позиций. Этот прием будет достаточно широко использоваться далее в других моделях СП. Очевидно, что такой прием не работает для случая небезопасных СП с тормозящими дугами, потому что при наличии $k > 1$ маркеров в исходной позиции понятие инверсной позиции теряет свой

первоначальный смысл. Оставляя без обсуждения вопрос о возможности построения ординарной сети Петри, эквивалентной ограниченной ингибиторной сети Петри, обратимся к примеру, показывающему возможность реализации так называемого оператора условного вычитания. Этот оператор можно представить как

ЕСЛИ k = 0, ТО перейти к метке m_1, ИНАЧЕ k = k – 1 и перейти к метке m_2.

На рис. 3.8 показан фрагмент ингибиторной сети Петри, реализующий *оператор условного вычитания*. При числе маркеров $k \geq 1$ в позиции p_2 переход t_1 срабатывает ровно k раз, каждый раз осуществляя переход к метке $m2$, соответствующей позиции p_3, и только при $k = 0$ открывается возможность возбуждения перехода t_2, в результате срабатывания которого маркер оказывается в позиции p_4, т.е. происходит переход к метке m_1.

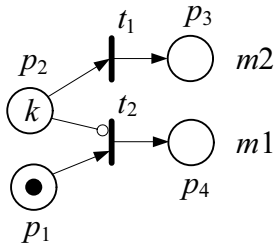


Рис. 3.8. Ингибиторная СП, осуществляющая проверку на пустую маркировку позиции

Таким образом, при реализации оператора условного вычитания срабатывание некоторого перехода (t_2) оказывается возможным только в том случае, когда инцидентная этому переходу по входящей тормозной дуге позиция (p_2) пуста (имеет нулевую маркировку). Попутно осуществляется механизм проверки на нуль.

Замечание. Если для безопасных сетей Петри для инверсных позиций *p* и *p'* имеет место $\mu(p) + \mu(p') = 1$, то для *k*-ограниченных СП можно ввести обобщающее понятие *k*-инверсных позиций, т.е. такой пары позиций *p* и *p'*, что $\mu(p) + \mu(p') = k$. Проверку на нуль в позиции *p'* для пары *k*-инверсных позиций *p* и *p'* тогда можно выполнить на основании того, что $\mu(p') = 0$, если $\mu(p) = k$. Естественно, такая проверка возможна лишь в том случае, если СП обладает свойством консервативности относительно этой пары позиций.

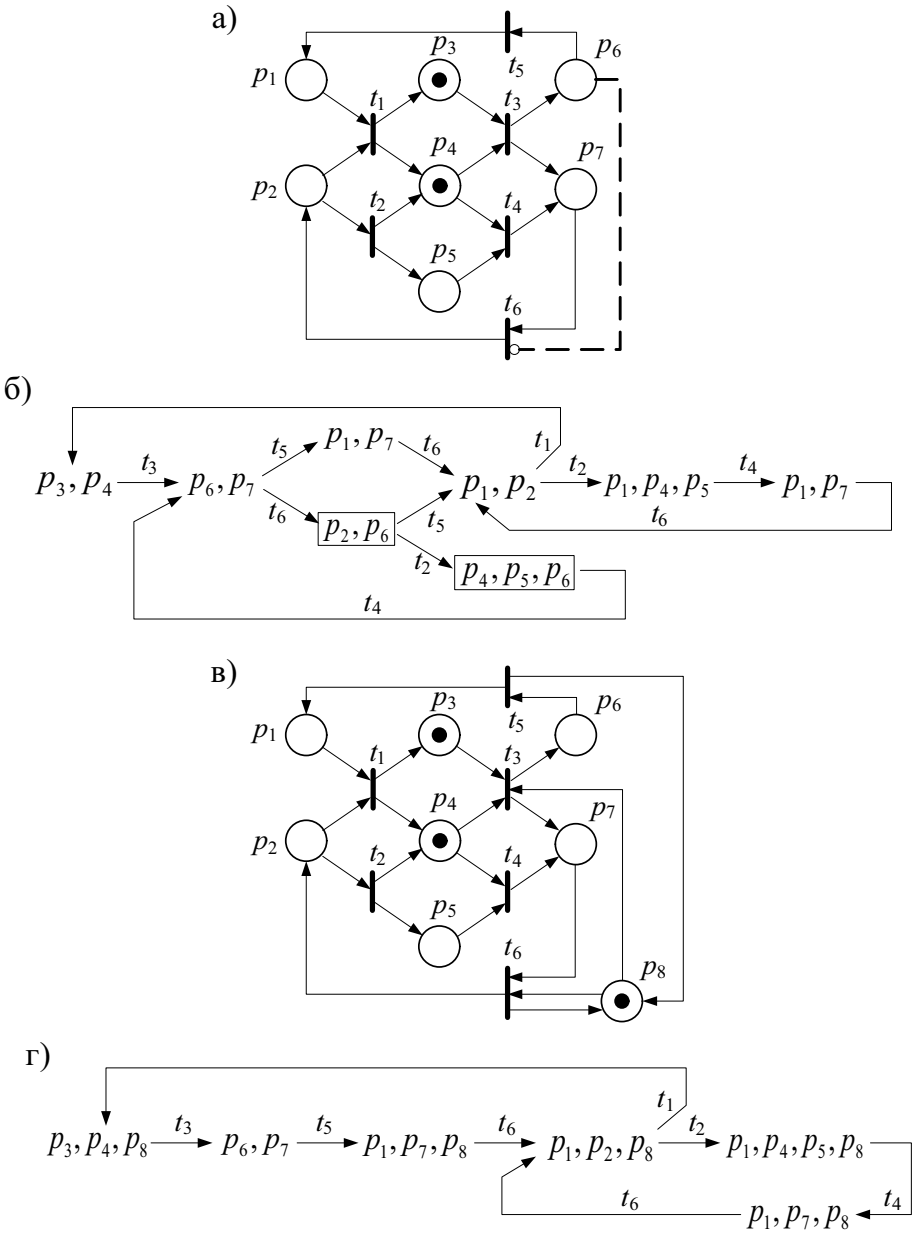


Рис. 3.9. а) Исходная ординарная СП; тормозящая дуга, ведущая из p_6 в t_6, отсутствует; б) диаграмма маркировок исходной СП; в) эквивалентная ординарная СП и г) ее диаграмма маркировок

Введение дополнительных тормозящих дуг, т.е. модификация исходных спецификаций на языке ординарных сетей Петри, часто дает возможность проектировщику "исправлять" допущенные ошибки. Например, пусть некоторая моделируемая система задана спецификацией, изображенной на рис. 3.9а. В ней первоначально отсутствует обозначенная пунктиром дуга, ведущая из p_6 в t_6. Диаграмма маркировок этой СП приведена на рис. 3.9б. Пусть также по причинам семантического характера должно быть запрещено срабатывание сети Петри в маркировке p_2, p_6. Тогда требуется запретить срабатывание t_6 в этой маркировке. Это и осуществляется введением тормозящей дуги, обозначенной пунктиром. Однако следует заметить, что при такой модификации оказывается недостижимой не только маркировка p_2, p_6, но также и маркировка p_4, p_5, p_6. На рис. 3.9б эти маркировки помечены прямоугольниками.

Использование приема, иллюстрируемого рис. 3.7, позволяет введением дополнительной позиции p_8 с инцидентными ей дугами перейти от СП с тормозящей дугой (рис. 3.9а) к ординарной СП, изображенной на рис. 3.9в. Диаграмма маркировок последней приведена на рис. 3.9г. Очевидно, что траектории, порождаемые этой диаграммой, совпадают с траекториями, порождаемыми диаграммой на рис. 3.9б, не содержащей маркировок, заключенных в прямоугольники.

Методически модификация исходной СП путем перехода от ординарной безопасной СП к СП с дополнительно вводимыми тормозящими связями, а затем – вновь к ординарной СП, весьма удобна. Причина предпочтительности этого подхода над альтернативным способом построения СП по модифицированной (расширенной) диаграмме переходов состоит в том, что пока в рамках второго подхода не известны общие алгоритмы построения СП по диаграмме маркировок, т.е. альтернативный способ не является алгоритмическим. Иллюстрируемый рис. 3.9 подход может рассматриваться как одна из процедур стандартного преобразования СП методом "дифференциального проектирования" (проектирования по образцу).

3.4. Приоритетные и временные сети

Распределение ресурсов является одной из важных проблем в вычислительных системах, вследствие чего необходимы средства описания процесса распределения. К сожалению, ординарные СП не способны детерминированным образом разрешать конфликты. Так, в СП на рис. 3.10 воз-

буждены при заданной начальной маркировке оба перехода и срабатывание каждого из них снимает возбуждение другого (конфликтная ситуация). Первым может сработать как t_1, так и t_2, т.е. процесс недетерминированный.

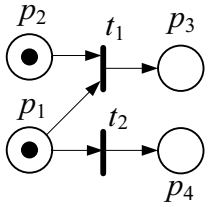


Рис. 3.10. Пример сети Петри

Можно предложить такую интерпретацию данной сети Петри. Два устройства (t_1 и t_2) запрашивают ресурсы. Первому необходимы ресурсы p_1 и p_2, а второму – только p_2. Какова обычная тактика принятия решения в том случае, когда запрос на ресурсы производится двумя устройствами одновременно? Она состоит в установлении приоритетов. Применительно к нашему случаю каждому конфликтующему переходу из некоторого подмножества $T' \subseteq T$ следует поставить в соответствие некоторое положительное число, называемое его *приоритетом*. На рис. 3.10, например, переходу t_1 можно приписать приоритет $pr(t_1) = 1$, а переходу t_2 – $pr(t_2) = 2$. Тогда при конфликтной маркировке первым должен сработать t_2, т.е. ресурс будет предоставлен переходу с наибольшим приоритетом. Соответствующее обобщение СП названо **приоритетными сетями**.

Навешивание приоритетов на конфликтующие переходы СП чем-то "нефизично", потому что оно не раскрывает механизма, обеспечивающего выход из коллизии. Без такого механизма поведение, задаваемое приоритетными СП, не может быть реализовано ни на аппаратном, ни на программном уровне. Гораздо более естественно задать временные параметры переходов. Очевидно, что если в рассмотренном примере переход t_2 быстрее перехода t_1, то первым обязательно сработает t_2. Такая интерпретация более естественна.

Временной сетью Петри **называют пару** $< PN, f >$, **где** $PN =< P, T, E, \mu^0 >$ **– ординарная сеть Петри, а** f **–** ***функция задержки*** **переходов** T, $f : T \to N^+$, N^+ **– множество целых неотрицательных чисел.**

Таким образом, каждому переходу t_i сети Петри приписывается некоторое число $\tau_i = f(t_i)$, смысл которого состоит в следующем. После момента времени, когда переход оказался возбужденным, он срабатывает ровно через τ_i единиц времени. Если при данной маркировке несколько переходов конфликтуют, то срабатывает тот из них, задержка которого минимальна. Если при данной маркировке возбуждены два или более перехода с одинаковой задержкой, то они сработают одновременно.

Рассмотрим пример временной СП, представленной на рис. 3.11а. На этом рисунке каждый переход помечен не только своим наименованием, но и величиной задержки. Диаграмма маркировок ординарной СП, соответствующей данной временной СП, в которой удалены задержки переходов, показана на рис. 3.11б. Дуги на этой диаграмме помечены, как обычно, обозначениями переходов. Рядом с обозначением каждого перехода через запятую указана величина его задержки. Диаграмма для временной СП (рис. 3.11в) строится путем выбора той из исходящих дуг текущей маркировки, которая соответствует минимальной задержке возбужденных переходов, причем, если возбужденные переходы имеют одинаковые задержки, то такие переходы должны присутствовать в диаграмме в виде параллельных ветвей.

В рассматриваемой модели хорошо видна идея синхронизации. Одновходовые переходы работают как элементы задержки, а многовходовые – как синхронизаторы. Поэтому естественным способом представления поведения временных СП, кроме диаграмм маркировок, могут служить традиционные временные диаграммы. Для сети на рис. 3.11а соответствующая временная диаграмма показана на рис. 3.12. Дуги в ней указывают на синхронизируемые события.

Полезно отметить, что практический интерес, по-видимому, представляет только класс безопасных временных СП, в которых в каждый момент их функционирования маркировка является безопасной. Небезопасные временные СП особого смысла в данной постановке не имеют. Подход временных СП может оказаться полезным при расчетах характеристик производительности систем, однако не будем здесь на этом останавливаться, поскольку подход, который будет рассмотрен в разд. 3.6, кажется более общим.

Два слова относительно соглашения об ограничении величин задержек переходов целыми числами. Оно несущественно в силу возможности замены дробных величин задержек целочисленными путем приведения к общему знаменателю.

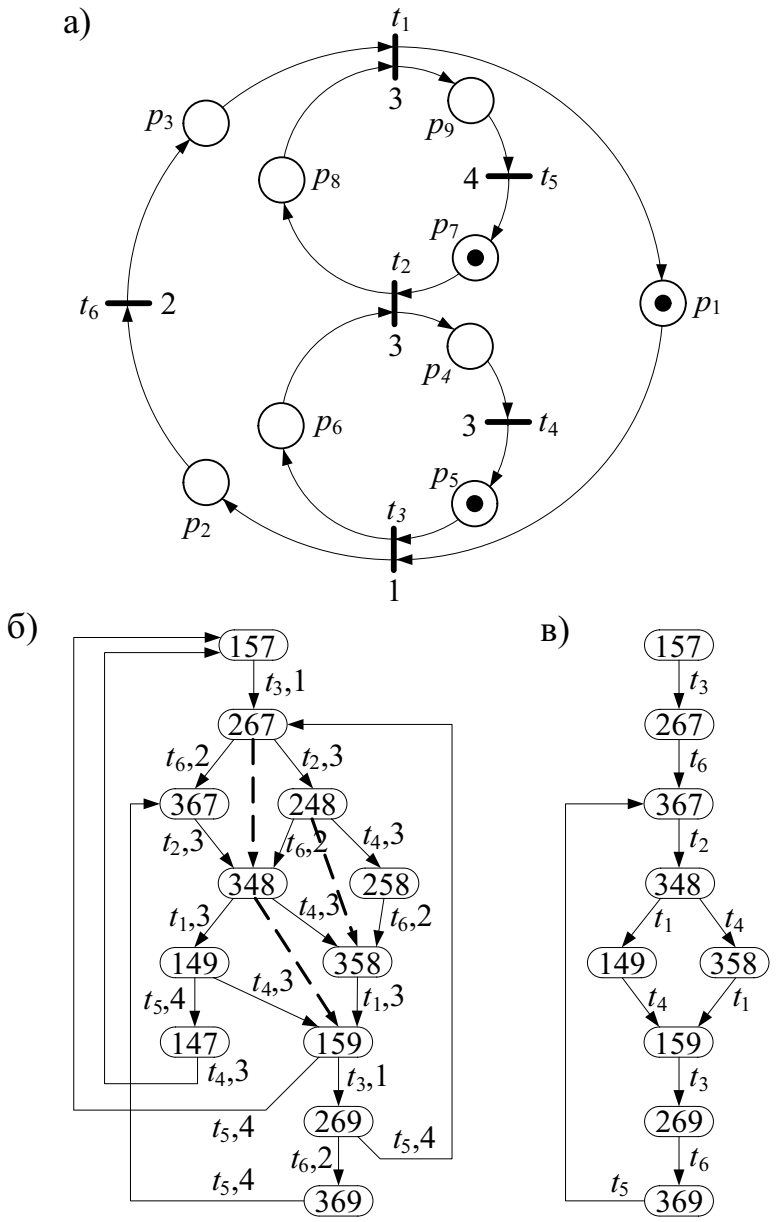


Рис. 3.11. а) Временная сеть Петри, б) диаграмма маркировок соответствующей ординарной сети Петри, в) диаграмма маркировок временной сети

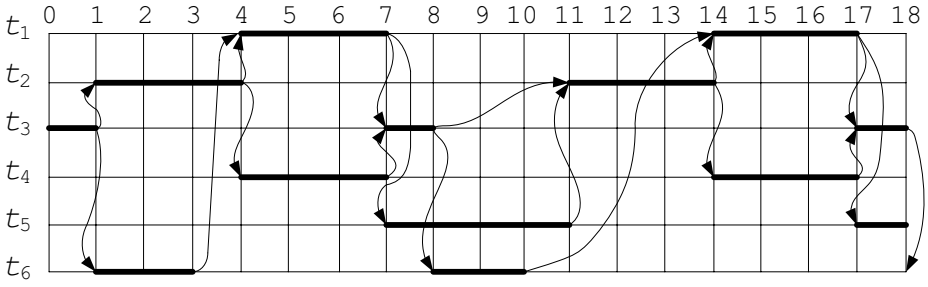


Рис. 3.12. Временная диаграмма временной сети, представленной на рис. 3.11а

Рассмотренная выше модель временных СП не является единственной. **Известны также следующие модификации временных СП:**

1) переходы $t_i \in T$ атрибутируются значениями τ_i, причем считается, что возбужденный переход может сработать в интервале времени от 0 (момент возбуждения) до τ_i;
2) переходы $t_i \in T$ атрибутируются парами неотрицательных целых чисел τ_{1i}, τ_{2i}, $\tau_{2i} \geq \tau_{1i}$, причем принимается, что возбужденный в момент времени 0 переход может сработать в интервале времени от τ_{1i} до τ_{2i}.

Анализ сетей такого рода затруднителен в силу того, что их переходы срабатывают в непрерывном времени и в зависимости от распределения реальных задержек переходов некоторые маркировки и траектории в диаграммах маркировок соответствующих ординарных сетей могут быть недостижимы.

3.5. Синхронные и самомодифицируемые сети

Ординарные сети Петри обладают тем важным свойством, что правила срабатывания возбужденных переходов являются локальными. Информация о возбуждении воспринимается самим переходом вне зависимости от окружения. Это обстоятельство и предопределяет возможность эффективного использования СП для моделирования систем децентрализованного управления. Тем не менее, реализуются попытки обобщения ординарных СП за счет отказа от концепции локальности срабатывания переходов, а следова-

тельно, за счет централизации управления сетью. Одна из моделей такого сорта – *синхронные СП* (ССП).

Синхронные сети Петри **– это сети Петри, в которых вводятся внешние часы, т.е. срабатывания переходов тактируются. В начале каждого такта определяются потенциально возбужденные переходы, из них выбирается некоторое максимально возможное число взаимно неконфликтных переходов. Затем все такие переходы принудительно срабатывают в произвольном порядке, но обязательно в течение того же такта. Таким образом, можно говорить о двух фазах работы синхронных СП –** ***фазе наблюдения*** **и отбора кандидатов на срабатывание и** ***фазе срабатывания*** **отобранных переходов.**

В гл. 2 говорилось о том, что общая задача достижимости (маркировки) сводится к задаче достижимости пустой маркировки, а эта задача, в свою очередь, – к задаче достижимости пустой позиции. Покажем, как эта последняя задача моделируется синхронными СП.

Возвратимся к программному сегменту из разд. 3.3 и представленному на рис. 3.13.

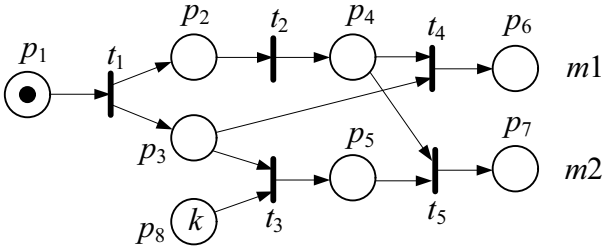


Рис. 3.13. Фрагмент синхронной сети Петри, моделирующей проверку на нуль

На этом рисунке через k обозначено число маркеров в позиции p_8 при начальной маркировке. Пусть $k = 0$. Тогда диаграмма маркировок будет иметь вид:

$$p_1 \xrightarrow{t_1} p_2, p_3 \xrightarrow{t_2} p_3, p_4 \xrightarrow{t_4} p_6,$$

т.е. достижима позиция p_6, соответствующая метке m_1. В этом случае ССП функционирует как обычная СП.

При $k = 1$, например, диаграмма маркировок будет выглядеть иначе:

$$p_1, p_8 \xrightarrow{t_1} p_2, p_3, p_8 \xrightarrow{t_2,t_3} p_4, p_5 \xrightarrow{t_5} p_7,$$

т.е. достижима позиция p_7, соответствующая метке $m2$. Обратим внимание на то, что в маркировке p_2, p_3, p_8 возбуждены одновременно два перехода (t_2 и t_3) и в данном такте они оба должны одновременно сработать.

Следует сказать, что некоторые одновременно возбужденные переходы могут быть взаимно конфликтными. Так, в ССП, представленной на рис. 3.14, при указанной начальной маркировке возбуждены все три перехода, но t_1 конфликтует с t_2, а t_2 – с t_3, потому что они имеют общие ресурсы (по одному маркеру в позициях p_2 и p_3).

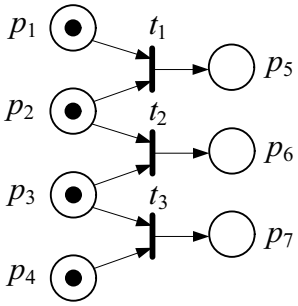


Рис. 3.14. Пример синхронной сети Петри

Естественно, может сработать только один из пары конфликтующих переходов, поэтому в данном случае потенциально могут сработать только следующие комбинации переходов: t_1, t_3 или t_2. Концепция ССП предполагает, что сработать должно максимальное число возбужденных переходов, т.е. пара t_1 и t_3, поэтому результирующая маркировка рассматриваемой ССП может быть только одна – p_5, p_7. Именно поэтому **синхронные СП часто называют *сетями Петри со стратегией срабатывания максимального числа возбужденных переходов***.

Отказ от концепции локальности правил срабатывания переходов в пользу концепции глобальности имеет место и в *самомодифицируемых СП*.

Самомодифицируемые сети Петри **— это класс сетей Петри, у которых дуги являются взвешенными, но для некоторых входящих и выходящих дуг переходов приписаны не фиксированные кратности (константы), а функции кратности (переменные), например, равные текущему числу маркеров в определенных (изначально заданных) позициях сети.**

Типичная картинка для самомодифицируемых сетей Петри представлена на рис. 3.15.

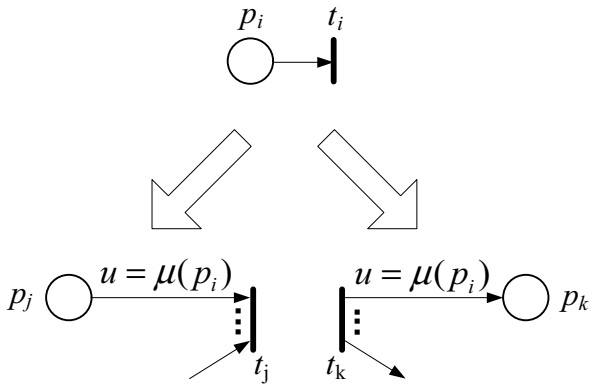


Рис. 3.15. Типичные фрагменты самомодифицируемых сетей Петри

Если в позиции p_i содержится $u = \mu(p_i)$ маркеров, то вес дуг, ведущих из позиции p_j в переход t_j и из перехода t_k в позицию p_k, устанавливается равным u. Таким образом, в этом случае для возбуждения перехода t_j необходимо, чтобы в p_j было, по крайней мере, u маркеров, а при срабатывании t_k в позицию p_k забрасывается u маркеров.

Существенно, что в динамике в p_i может пребывать разное число маркеров, и это в общем случае изменяет число возбужденных переходов, а следовательно, и поведение сети.

Задача проверки нуля решается с помощью самомодифицируемой сети Петри тривиально: на рис. 3.16 дуге, ведущей из p_1 в t_1, приписывается вес $\mu(p_2)+1=k+1$, из-за чего при $k \geq 1$ может сработать лишь переход t_2 (о чем свидетельствует появление маркера в позиции p_4, соответствующей метке $m1$), а при $k = 0$ – переход t_1 (осуществляется переход к метке $m2$).

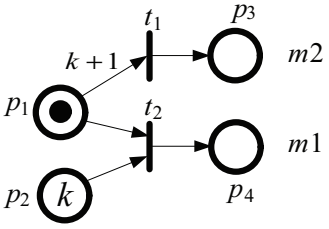


Рис. 3.16. Проверка нуля в самомодифицируемой сети Петри

Читатель, наверное, уже обратил внимание на то, что на рис. 3.16 соответствующая дуга помечена весом $k + 1$, а не $k = \mu(p_2)$. Однако это не противоречит данному выше определению самомодифицируемых сетей: можно считать, что позиция p_1 связана с переходом t_1 двумя дугами, одна из которых имеет вес k, а другая – 1. Тем не менее, соглашение о кратности веса дуги числу маркеров в соответствующей позиции не является обязательным и может быть ослаблено, если считать, что вес модифицируемой входной и (или) выходной дуги может задаваться некоторыми функциями от $\mu(p_i)$.

3.6. Стохастические сети Петри

Сетевые модели, о которых говорилось выше, используются для спецификации дискретных устройств, по которой в результате анализа выявляются характеристические свойства, особенности, свидетельствующие о корректности спецификации или наличии аномальных поведенческих эффектов, выявление которых заставляет модифицировать спецификацию с целью поиска удовлетворительного решения. Обратим внимание на то, что классические задачи анализа по самой своей сути связаны с верификацией спецификации, т.е. ориентированы на получение ответов о наличии или отсутствии некоторых ее качественных характеристик. В то же время, проектировщика аппаратного обеспечения или программиста часто интересуют количественные оценки предложенного решения. Между этими двумя подходами – качественными и количественными – имеется определенное методологическое расхождение, объясняемое, по-видимому, причинами исторического характера: дискретная математика родилась значительно позже классической (непрерывной) и по существу это две различные математики. Тем не менее, интересно наблюдать наличие некоторых “мостов” между указанными двумя подходами. Пример “наведения мостов” дает модель стохастических сетей Петри.

Стохастическая сеть Петри **визуально отличается от ординарной сети тем, что в ней каждый переход помечается либо средней скоростью его срабатывания (время срабатывания возбужденного перехода – случайная величина), либо вероятностью его срабатывания при возбуждении, в то время как в ординарной сети рассматривается лишь возможность срабатывания возбужденного перехода.**

Формально стохастическая сеть Петри (СТСП) представляет собой расширение ординарной СП, т.е. четверки $< P, T, E, \mu^0 >$, за счет введения дополнительного отображения $\lambda : T \to N^+$, где N^+ – множество положительных чисел. Иначе говоря, каждому переходу $t_i \in T$ отображение (функция) λ ставит в соответствие некоторое число λ_i, интерпретируемое как *средняя скорость* срабатывания этого (возбужденного) *перехода*. Эта характеристика носит вероятностный характер.

Обычно полагают, что скорость срабатывания перехода имеет показательное распределение. Имеет место теорема о том, что любая конечная СТСП (т.е. СП, имеющая конечное число позиций и переходов) изоморфна (в данном случае эквивалентна) одномерной дискретной *марковской цепи*. Опуская доказательство этой теоремы во избежание ввода базовых понятий теории марковских цепей (известных в общих чертах читателям), ограничимся рассмотрением примера перехода от конкретной СТСП к соответствующей марковской цепи.

Рассмотрим стохастическую сеть Петри, представленную на рис. 3.17.

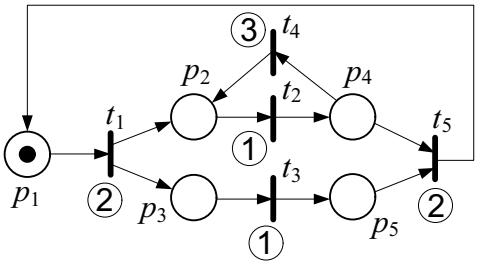


Рис. 3.17. Пример стохастической сети Петри

Средние скорости (интенсивности) λ_i срабатывания переходов t_i, $1 \le i \le 5$, указанны на рисунке (обведены кружками). Не обращая внимания пока на их абсолютные значения (они носят число условный характер), построим для ординарной СП, соответствующей данной СТСП, диаграмму маркировок. Она содержит пять маркировок $\mu^0 - \mu^4$ и показана на рис. 3.18. Эта диаграмма может быть представлена в виде конечной марковской цепи с пятью состояниями, если переходы t_i из одного состояния в другое осуществляются с интенсивностями λ_i. Таким образом, от рис. 3.18 можно перейти к рис. 3.19. Заметим, что такая марковская цепь является эргодической (т.е. финальные вероятности не зависят от начальных условий), так как образующий ее граф переходов является сильно связным.

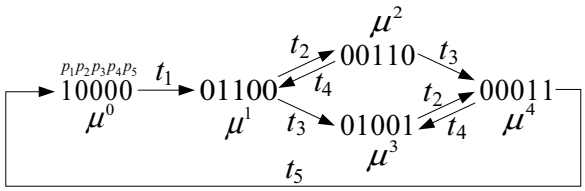


Рис. 3.18. Диаграмма маркировок для СТСП, представленной на рис. 3.17

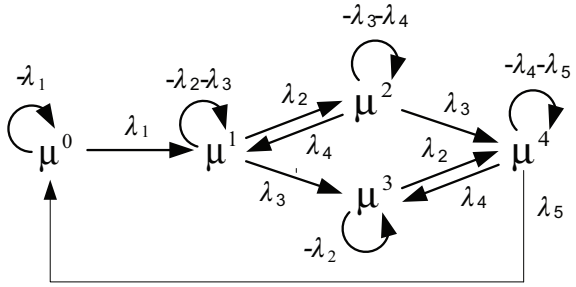


Рис. 3.19. Марковская цепь, изоморфная СТСП, представленной на рис. 3.17

Матрица переходов марковской цепи, построенной для рассматриваемого примера, представлена в табл. 3.1. Заметим, что сумма значений переходных интенсивностей в каждой строке этой матрицы должна быть равно нулю. Непомеченные элементы матрицы являются нулевыми.

Таблица 3.1. Матрица переходов марковской цепи, показанной на рис. 3.19

–	μ^0	μ^1	μ^2	μ^3	μ^4
μ^0	$-\lambda_1$	λ_1			
μ^1		$-\lambda_2-\lambda_3$	λ_2	λ_3	
μ^2		λ_4	$-\lambda_3-\lambda_4$		λ_3
μ^3				$-\lambda_2$	λ_2
μ^4	λ_5			λ_4	$-\lambda_4-\lambda_5$

От матрицы переходов можно перейти к системе уравнений относительно финальных вероятностей ϑ пребывания цепи в данном состоянии (т.е. маркировке). В результате имеем:

$$\begin{aligned}
&-\lambda_1\vartheta_0 + \lambda_5\vartheta_4 = 0,\\
&\lambda_1\vartheta_0 + (-\lambda_2 - \lambda_3)\vartheta_1 + \lambda_4\vartheta_2 = 0,\\
&\lambda_2\vartheta_1 + (-\lambda_3 - \lambda_4)\vartheta_2 = 0,\\
&\lambda_3\vartheta_1 + (-\lambda_2)\vartheta_3 + \lambda_4\vartheta_4 = 0,\\
&\lambda_3\vartheta_2 + \lambda_2\vartheta_3 + (-\lambda_4 - \lambda_5)\vartheta_4 = 0,
\end{aligned}$$

где каждое уравнение соответствует столбцу таблицы. К этим уравнениям следует добавить уравнение нормировки

$$\vartheta_0 + \vartheta_1 + \vartheta_2 + \vartheta_3 + \vartheta_4 = 1,$$

после чего из системы можно исключить одно из уравнений (скажем, последнее), так как только $n - 1$ из n уравнений в общем случае являются линейно независимыми. Далее методом Гаусса (последовательным исключением неизвестных ϑ_i) найти решение системы

$$\begin{cases}
\vartheta_0 + \vartheta_1 + \vartheta_2 + \vartheta_3 + \vartheta_4 = 1\\
-\lambda_1\vartheta_0 + \lambda_5\vartheta_4 = 0\\
\lambda_1\vartheta_0 + (-\lambda_2 - \lambda_3)\vartheta_1 + \lambda_4\vartheta_2 = 0\\
\lambda_2\vartheta_1 + (-\lambda_3 - \lambda_4)\vartheta_2 = 0\\
\lambda_3\vartheta_1 + (-\lambda_2)\vartheta_3 + \lambda_4\vartheta_4 = 0
\end{cases}$$

при $\lambda_1 = \lambda_5 = 2$, $\lambda_2 = \lambda_3 = 1$, $\lambda_4 = 3$:

$$\begin{aligned}
\vartheta_0 &= 0{,}1163\\
\vartheta_1 &= 0{,}1860\\
\vartheta_2 &= 0{,}0465\\
\vartheta_3 &= 0{,}5349\\
\vartheta_4 &= 0{,}1163.
\end{aligned}$$

Финальные вероятности ϑ_i указывают доли времени пребывания системы в данном состоянии (маркировке μ_i).

Легко получить также значения *среднего времени пребывания маркера* в данной позиции p_j. Для этого требуется просуммировать ϑ_i по тем состояниям (маркировкам), в которых $\mu(p_j) \geq 1$, т.е. найти *плотности вероятности* $\delta(p_j)$ по формуле

$$\delta(p_j) = \sum_{i \in \mu(p_j) \geq 1} \vartheta_i .$$

Так, в рассматриваемом примере $\mu(p_1) = 1$ в маркировке μ^0, поэтому $\delta(p_1) = \vartheta_0 = 0{,}1163$;

$\mu(p_2) = 1$ в маркировках μ^1 и μ^3, поэтому

$$\delta(p_2) = \vartheta_1 + \vartheta_3 = 0{,}1860 + 0{,}5349 = 0{,}7209 .$$

Далее получим: $\delta(p_3) = 0{,}2325$, $\delta(p_4) = 0{,}1628$, $\delta(p_5) = 0{,}6521$.

Пользуясь результатами теории марковских цепей, можно определить информацию, полезную для вычислений производительности моделируемой системы. Например, можно вычислить среднее время возврата к данной маркировке. Сделать это можно, исходя из следующих соображений. Ясно, что время срабатывания перехода после его возбуждения – это время нахождения процесса в маркировке, предшествующей этому переходу. Среднее время $\tau(\mu^i)$ нахождения в маркировке μ^i определяется формулой

$$\tau(\mu^i) = 1 / \sum_{T^*} \lambda_i ,$$

где сумма берется по подмножеству $T^* \subset T$ тех переходов, которые возбуждены в данной маркировке μ^i. Тогда с учетом доли времени ϑ_i, которая соответствует пребыванию процесса в маркировке μ^i, среднее время возврата в нее $\tau^*(\mu^i)$ определяется как

$$\tau^*(\mu^i) = \tau(\mu^i) / \vartheta_i - \tau(\mu^i) \text{ (единиц времени).}$$

Результаты подсчета $\tau^*(\mu^i)$ для рассматриваемого примера приведены в табл. 3.2.

В первом столбце этой таблицы перечислены все возможные маркировки (см. рис. 3.19), во втором – подмножества переходов, возбужденных при соответствующих маркировках, в третьем – вычисленные ранее финальные вероятности пребывания в данных маркировках, в четвертом – значения $\tau(\mu^i)$, в пятом – искомые значения $\tau^*(\mu^i)$.

Результаты подсчета среднего времени возврата в маркировки. Таблица 3.2

Маркировка	Возбужденные переходы T^*	ϑ_i	$\tau(\mu^i)$	$\tau^*(\mu^i)$
μ^0	t_1	0,1163	$1/\lambda_1 = 0{,}50$	3,7992
μ^1	t_2, t_3	0,1860	$1/(\lambda_2 + \lambda_3) = 0{,}50$	2,1882
μ^2	t_3, t_4	0,0465	$1/(\lambda_3 + \lambda_4) = 0{,}25$	5,1263
μ^3	t_2	0,5349	$1/\lambda_2 = 1$	1,3346
μ^4	t_4, t_5	0,1163	$1/(\lambda_4 + \lambda_5) = 0{,}20$	1,5197

Недостатком СТСП как модели является в общем случае высокая трудоемкость вычислений: число уравнений в системе в точности равно числу маркировок в диаграмме маркировок, которое растет экспоненциально с ростом числа позиций в СП. Для уменьшения объема вычислений можно воспользоваться редукцией последовательных участков СТСП по числу позиций и переходов. Такой вид редукции осуществляется на основе следующей теоремы.

Теорема. Последовательный фрагмент стохастической сети Петри с двумя позициями $P = \{p_i, p_j\}$ и двумя переходами $T = \{t_i, t_j\}$ такой, что $I(t_i) = \{p_i\}$, $O(i_i) = \{p_j\}$, $I(t_j) = \{p_j\}$ и скорости переходов t_i и t_j равны соответственно λ_i и λ_j, может быть заменен эквивалентным фрагментом СТСП с одной позицией p_{ij} и одним переходом t_{ij} таким, что $I(t_{ij}) = \{p_{ij}\}$, причем

$$\lambda_{ij} = \frac{\lambda_i \lambda_j}{\lambda_i + \lambda_j}.$$

Теорема иллюстрируется рис. 3.20.

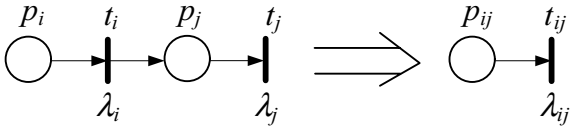


Рис. 3.20. Редукция последовательного фрагмента СТСП

Доказательство теоремы тривиально. Переходы t_i и t_j исходного фрагмента вносят суммарную временную задержку, равную сумме задержек этих переходов, т.е.

$$\frac{1}{\lambda_i} + \frac{1}{\lambda_J} = \frac{\lambda_i + \lambda_j}{\lambda_i \lambda_j}.$$

Очевидно, что финальная вероятность пребывания маркера в позиции p_{ij} в этом случае определяется как сумма финальных вероятностей пребывания маркеров в позициях p_i и p_j, т.е.

$$\delta(p_{ij}) = \delta(p_i) + \delta(p_j).$$

В заключение отметим, что расширение сетей Петри, осуществляемое посредством перехода к стохастическим сетям Петри, позволяет получать дополнительную информацию о производительности моделируемых систем, что является обнадеживающим фактором. Появляется возможность использования развитого аппарата теории марковских цепей, ранее ориентированного в основном на процессы, специфицируемые с помощью вероятностных автоматов (автоматных СТСП), в более общем классе стохастических процессов, специфицируемых на основе ординарных СП.

3.7. Оценочные сети

Так называемые *оценочные сети* или, короче, *Е-сети* (от англ. evaluation nets) представляются двудольными ориентированными помеченными графами, в которых из некоторого множества *ячеек* (аналогов позиций СП) выделяется подмножество так называемых *разрешающих ячеек*, некоторые из ячеек объявляются периферическими (входными или выходными), а переходы являются аналогами переходов СП. Ячейки могут соединяться дугами только с переходами, а переходы – только с ячейками. Обычно задается начальная маркировка Е-сети. Соглашения о правилах смены маркировок определяются дисциплинами, принятыми для пяти примитивов (стандартных фрагментов), композиция которых составляет Е-сеть.

Примитивы оценочных сетей

В примитиве типа T (transition) (см. рис. 3.21а) с одной входной ячейкой *a* и одной выходной ячейкой *b* единственного перехода *t* маркер в *a* возбуждает переход *t* лишь в том случае, когда ячейка *b* пуста (в ней отсутствует маркер). В результате срабатывания возбужденного перехода маркер переместится из *a* в *b*. Время срабатывания не регламентируется, и акт срабатывания считается неделимой операцией, как и в сетях Петри.

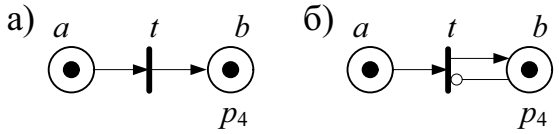


Рис. 3.21. а) Т-примитив и б) эквивалентный ему фрагмент ингибиторной сети Петри

Диаграмма достижимых маркировок для *T*-примитива имеет вид $\overset{ab}{10} \to 01$, а начальные маркировки $\mu(a)$, $\mu(b)$ = {00,01,11} являются тупиковыми, так как при них переход не возбужден. Таким образом, *T*-примитив, показанный на рис. 3.21а, можно представить в виде эквивалентной СП с тормозящими дугами, как показано на рис. 3.21б. Очевидно, что использование при этом тормозящих дуг не является необходимым, так как, воспользовавшись приемом эквивалентного преобразования безопасных СП с тормозящими дугами в ординарную СП, описанным в разд. 3.3 (см. рис. 3.7), можно заменить позицию *b* парой позиций *b* и $\overline{b}$ с взаимно инверсными маркировками. В результате для *T*-примитива получим фрагмент эквивалентной ординарной СП, показанный на рис. 3.22а в правом столбце. Очевидно, что при начальной маркировке $\mu(a) = \mu(b) = 1$, $\mu(\overline{b}) = 0$ переход *t* не возбужден.

В примитивах типов J и F (от англ. *join* и *fork* соответственно), в отличие от примитива типа *T*, имеются соответственно две входные (рис. 3.22б) и две выходные (рис. 3.22в) ячейки, инцидентные переходу *t*. Условием возбуждения является наличие маркеров во всех входных ячейках. В результате срабатывания перехода маркеры (маркер) изымаются из всех входных ячеек и переносятся во все выходные ячейки. По аналогии с *T*-примитивом легко строятся эквивалентные фрагменты соответствующих ординарных СП для *J*- и *F*- примитивов, показанные на рис. 3.22б и рис. 3.22в. Диаграммы маркировок, соответствующие срабатыванию переходов *t*, имеют вид: $\overset{a\ b\ c}{100} \to 011$ для *F*-примитива и $\overset{a\ b\ c}{110} \to 001$ для *J*-примитива.

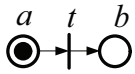
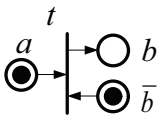
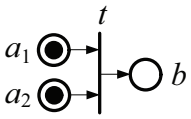
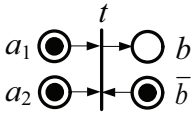
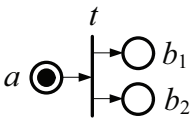
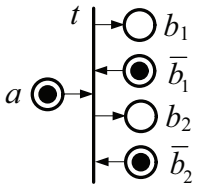
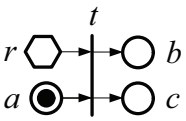
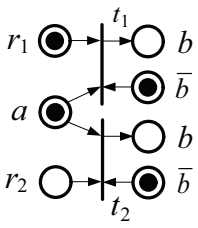
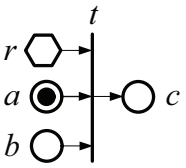
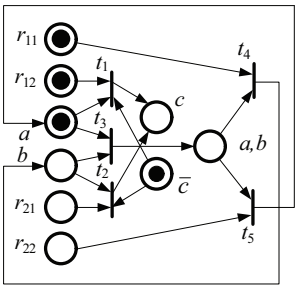
Номер рис.	Название вершины	Изображение примитива	Фрагмент эквивалентной сети Петри
а)	T		
б)	J		
в)	F		
г)	X		
д)	Y		

Рис. 3.22. Примитивы E-сетей и их интерпретация

Четвертый примитив типа X (рис. 3.22г) имеет входные ячейки a и r (ячейка r, изображенная в виде шестиугольника, является разрешающей) и две выходные ячейки b и c. В отличие от обычных ячеек, маркеры в разрешающих ячейках можно трактовать как цветные, скажем, красные и зеленые.

При красном (зеленом) маркере в *r*, наличии маркера в *a* и отсутствии маркеров в *b* и *c* возбужденный переход срабатывает, в результате чего маркер из *a* переносится в ячейку *b* (*c*), а маркер из *r* изымается. Отсутствие маркера в *r* интерпретируется как третье, дополнительное, состояние разрешающей ячейки. При наличии маркеров в *b* и *c* ячейка должна быть пустой. Диаграмма маркировок при разных начальных маркировках для *X*-примитива выглядит следующим образом: $\overset{r\,a\,b\,c}{K10*} \to \varnothing 01*$ или $\overset{r\,a\,b\,c}{З1*0} \to \varnothing 0*1$, где К – красный маркер, З – зеленый маркер, ∅ – пустой маркер, ∗ – 0 или 1.

Если теперь закодировать три цвета маркеров разрешающей ячейки двухразрядным кодом со значениями из множества {10, 01, 00}, которые соответственно представляют красный, зеленый и пустой маркеры, то вместо разрешающей ячейки *r* можно использовать две позиции r_1 и r_2 ординарной СП (они взаимоинверсны, так как комбинация 11 в кодировании цветов маркеров не используется). Если к тому же расщепить переход *t* на два (t_1 и t_2) и ввести инверсные позиции *b* и $\overline{b}$, *c* и $\overline{c}$, то *X*-примитиву будет соответствовать ординарная СП, показанная на рис. 3.22г.

Пятый примитив типа Y (рис. 3.22д) с переходом t имеет три входные ячейки (одну разрешающую *r* и две обычных *a* и *b*) и одну выходную *c*. Если в *a* или *b* (но не в обеих вместе) имеется маркер, то он после срабатывания перехода передается в *c*. Если же маркеры появляются в *a* и *b* одновременно, то, в зависимости от цвета маркера в *r* (также красного или зеленого), в *c* будет перенесен либо маркер из *a*, либо маркер из *b* (но только из одной входной ячейки). Диаграмма маркировок при разных начальных маркировках для Y-примитива выглядит следующим образом: $\overset{r\,a\,b\,c}{K100} \to \varnothing 001$, $\overset{r\,a\,b\,c}{K010} \to \varnothing 001$, $\overset{r\,a\,b\,c}{K110} \to \varnothing 011$, $\overset{r\,a\,b\,c}{З110} \to \varnothing 101$.

Попытки построения эквивалентной ординарной СП, соответствующей *Y*-примитиву, очевидно, не должны привести к успеху. Причина этого состоит в том, что *Y*-примитив моделирует ситуацию, сводящуюся к установлению некоторой системы предпочтений (приоритетов), которая, как известно, не реализуется ординарными сетями Петри. Однако этот примитив можно промоделировать с помощью СП, в которой переходы атрибутируются некоторыми числами, задающими временные интервалы срабатывания переходов после момента их возбуждения. В данном случае (рис. 3.22д, справа) переход *t* примитива моделируется двумя переходами t_1 и t_2 фрагмента временной сети Петри, которая, кроме того, содержит еще три пере-

хода: t_3, t_4 и t_5. Разрешающей ячейке r соответствуют четыре позиции: r_{11}, r_{12}, r_{21}, r_{22}; ячейкам a и b – одноименные позиции, ячейке c – пара инверсных позиций c и $\overline{c}$. Имеется также еще одна позиция, обозначенная как (a,b), являющаяся выходной для перехода t_3 и входной для переходов t_4 и t_5. Функционирование такого фрагмента очевидно: при наличии маркера в a или b (но не в a и b одновременно) срабатывает либо переход t_1, либо переход t_2 (при наличии маркеров в r_{12} или r_{21} соответственно), в результате чего маркер из a или b переводится в позицию c (после срабатывания t_1 или t_2 маркер изымается из инверсной позиции $\overline{c}$).

При наличии маркера в обеих позициях a и b должен сработать переход t_3, переводя маркер в позицию ab, после чего, в зависимости от того, в какой из инверсных позиций r_{11} или r_{22} имеется маркер, срабатывает один из переходов t_4 или t_5. В результате маркер возвращается в одну из позиций b или a в зависимости от того, какая система предпочтений выбрана. Ясно, что для нормального функционирования этой временной СП необходимо, чтобы выполнялись следующие соотношения между максимальными временами срабатывания переходов: $d_3+d_4 < d_1$ и $d_3+d_5 < d_2$, в которых через d_i обозначена задержка, вносимая переходом t_i.

Мотивы появления оценочных сетей

Переходя к обсуждению мотивов, приведших к появлению *E*-сетей как модели для представления параллельных асинхронных процессов, следует сказать следующее.

1. Очевидно, что *E*-сети (точнее композиция примитивов типов *T*, *J*, *F* и *X*) моделируют те же процессы, что и устойчивые, безопасные (относительно всех позиций, кроме входных) сети Петри. Однако их графическое изображение несколько проще, чем у ординарных СП.

2. *E*-сети удобны для представления конвейерных процессов и, в частности, для спецификации буферных устройств типа *FIFO*, работающих с обслуживанием очереди по дисциплине “первый пришел – первый обслужен”. Действительно, *E*-сеть, представленная на рис. 3.23а и составленная из *T*-примитивов, транспортирует маркеры из входной позиции p_1 (ресурс) в последующие по очереди так, что ни один из маркеров не может обогнать предыдущий. Соответствующая диаграмма маркировок (см. рис. 3.23б) указывает на наличие параллелизма. Обратим внимание на то, что сумма числа маркеров во всех маркировках равна трем.

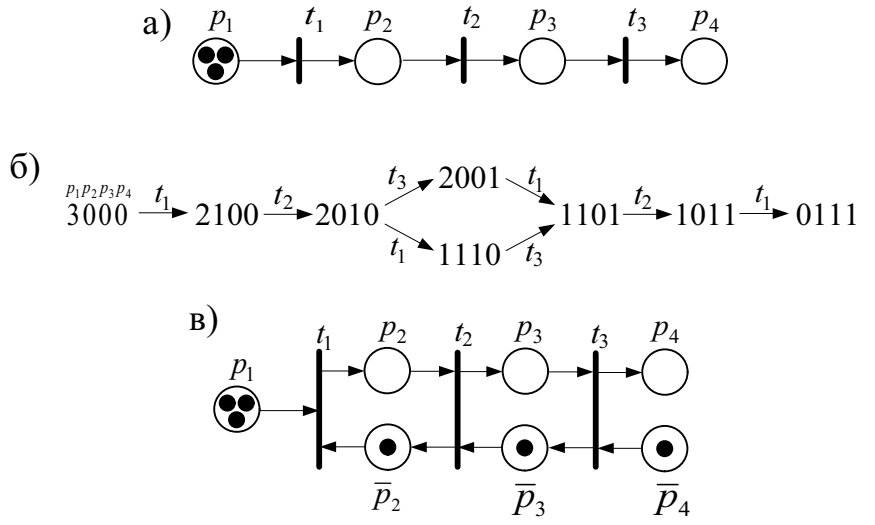


Рис. 3.23. а) Пример E-сети, моделирующей конвейерный процесс; б) диаграмма маркировок этой сети; в) эквивалентная ординарная сеть Петри

Построив диаграмму маркировок для ординарной СП, представленной на рис. 3.23в, и осуществив ее редукцию по $\overline{p}_2$, $\overline{p}_3$ и $\overline{p}_4$, можно получить ту же диаграмму, заданную на подмножестве ее позиций $p_1 - p_4$, что доказывает эквивалентность ординарной СП исходной *E*-сети. Эта сеть явно не является безопасной, но при этом небезопасна только периферийная позиция p_1, моделирующая наличие ресурса. Все остальные позиции сохраняют свойство безопасности.

3. Выше было показано, что примитив типа *Y* моделируется временной сетью Петри. Это указывает на то, что функциональные возможности *E*-сетей, в составе которых используются эти примитивы, могут превышать соответствующие возможности ординарных сетей Петри. Однако во многих работах утверждается, что *E*-сети представляют собой расширение ординарных сетей. Это утверждение ошибочно. Точнее было бы сказать, что *E*-сети – это "расширение сужения" ординарных СП. "Сужение" имеет место в силу принудительного механизма сохранения свойства безопасности примитивов *E*-сетей. Что касается имеющего места расширения, то оно дается трудно: задачи анализа *E*-сетей, содержащих *Y*-примитивы, относятся к числу алгоритмически неразрешимых.

4. Одним из основных побудительных мотивов к созданию модели *E*-сети было, по-видимому, желание приблизить сетевую проблематику к задачам имитационного моделирования процессов функционирования дискретных устройств. При таком подходе имитационная модель, построенная на основе *E*-сети, позволяет воспроизводить в некотором масштабе времени поведение моделируемой системы с той степенью детализации, которая необходима пользователю. Концепция *E*-сетей позволяет осуществлять имитационное моделирование с использованием средств, характерных для распространенного языка моделирования *JPSS*.

Пример. На рис. 3.24 приведен пример E-сети для моделирования процесса прохождения задания в простейшей вычислительной системе, состоящей из центрального процессора и накопителя на жестком диске. Логику этого процесса можно понять, учитывая следующую семантику пребывания маркеров в ячейках $b_1 - b_{13}$ и в разрешающих ячейках $r_1 - r_4$:

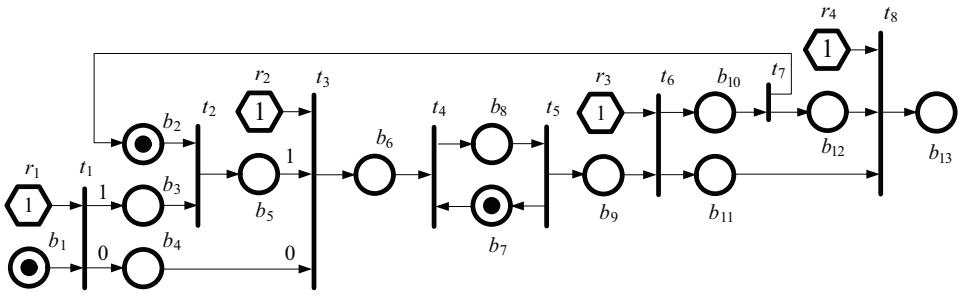


Рис. 3.24. Пример E-сети

b_1 – задание выдано,

b_2 – накопитель на жестком диске (HD) доступен,

b_3 – задание предусматривает использование HD,

b_4 – задание не требует использования HD,

b_5 – задание на HD сгенерировано,

b_6 – задание запрашивает центральный процессор (ЦП),

b_7 – центральный процессор свободен,

b_8 – центральный процессор занят,

b_9 – задание обработано центральным процессором,

b_{10} – HD можно освободить,

b_{11} – задание, не предусматривающее использования HD, выполнено,

b_{12} – задание, предусматривающее использование HD, выполнено,

b_{13} – задание выполнено,

r_1 – выбор между b_3 (1) и b_4 (0),

r_2 – выбор между b_5 (1) и b_4 (0),

r_3 – выбор между b_{10} (1) и b_{11} (0),

r_4 – выбор между b_{12} (1) и b_{11} (0).

Классификация переходов с инцидентными ячейками сети рис. 3.24 по типу примитивов такова: t_1, t_3, t_6 и t_8 принадлежат типу X; t_2, t_4 и t_5 – типу J; t_7 – типу F. Начальная маркировка разрешающих ячеек $r_1 - r_4$ условно обозначена через 1. Это значит, что, например, в результате срабатывания X-примитива с переходом t_1 при $r_1 = 1$ (0) маркер из b_1 будет перенесен в ячейку b_3 (b_4).

3.8. Сети потоков данных

Сети потоков данных были предложены в 1970 г. Дж. Деннисом и его коллегами при выполнении работ по проекту MAC в части создания машин, управляемых потоками данных. Это направление концептуально близко к проблематике проектирования самосинхронных схем, однако особых успехов в области создания самосинхронной схемотехники, насколько выяснилось, достигнуто не было.

Итак сети (схемы) потоков данных – еще одна модель, широко используемая в одном из перспективных направлений развития архитектур высоко параллельных вычислительных машин – ВМ, управляемых потоками данных. Этот класс вычислительных машин, в отличие от машин традиционной фон-неймановской архитектуры, не использует понятий “передача управления”, “счетчик команд” и др., так как команды (операторы) управляются самими данными. Считается, что выполнение команды разрешено, если данные “готовы” во всех ее входных портах и отсутствуют в выходных портах.

Говоря об управлении данными, полезно вернуться к определению параллельных программ в разд. 2.7 (см. рис. 2.23). Рассмотрим тот же пример вычисления выражения

$$y = (a + b) \cdot (c - d) - (e / f).$$

Введем дополнительные переменные

$$z_1 = a + b,$$
$$z_2 = c - d,$$
$$z_3 = e / f,$$
$$z_4 = z_1 \cdot z_2.$$

Тогда параллельную программу, вычисляющую результат y как функцию от входных переменных a, b, c, d, e и f, можно, очевидно, задать в том виде, как она представлена на рис. 3.25.

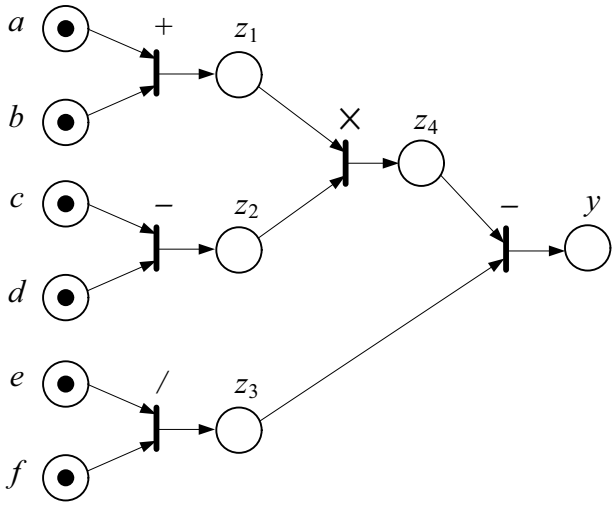


Рис. 3.25. Сеть Петри для вычисления арифметического выражения $y = (a+b)\times(c-d)-(e/f)$ *по принципу управления данными*

Маркеры в позициях, соответствующих входным переменным, означают, что данные готовы. При заданной начальной маркировке параллельно выполняются операции сложения, вычитания и деления, а также операции умножения и деления. По мере их завершения маркеры появляются в позициях, помеченными дополнительными переменными $z_1 - z_4$. По готовности z_1 и z_2 вычисляется z_4, по готовности z_3 и z_4 – окончательный результат y. Таким образом, управление данными весьма естественно реализуется на сетях Петри, если наличие или отсутствие маркера в позиции интерпретируется как предикат, принимающий значение “истина” (“ложь”), если данные готовы (не готовы).

Отличие программы, управляемой потоком данных, от ранее определенной параллельной программы состоит в том, что последняя задает струк-

туру управления программой, в то время как в первой явно представлены позиции, соответствующие данным. Точнее, в соответствии с определением из разд.2.7 оператор из A в функции $\omega: T \to A$ распределения операторов трактуется как оператор от определенных операндов, именующий некоторый переход (например, $a + b$), в то время как при управлении данными задается не сам оператор, а только его тип (+).

Нетрудно заметить, что рис. 3.25 является примером не самой программы, а скорее лишь ее фрагментом, потому что она не приведена к каноническому виду с единственными входной и выходной позициями. Кроме того, в этом примере отсутствуют операторы условного перехода, моделируемые с помощью позиций свободного выбора. Более сложные примеры программ читатель может построить самостоятельно.

Таким образом, другая интерпретация сетей Петри позволяет задавать программы, управляемые потоками данных. Тем не менее, как говорилось в начале этого раздела, распространение получила другая модель – модель сетей потоков данных (СПД). Она скорее более близка к блок-схемам алгоритмов, что, в частности, определяется большим числом типов операторов, но, с другой стороны, родственна модели СП хотя бы наличием маркирования дуг и сходных правил срабатывания операторов.

Минимальный набор операторов сетей потоков данных образуют *актор* и *распознаватель* (аналоги функционального оператора и логического оператора (предиката) соответственно), *склейка* (бифуркатор), а также два типа *клапанов* и *смеситель* (аналоги операторов условного перехода). Условные обозначения операторов, правила их срабатывания и эквивалентные им фрагменты СП приведены на рис. 3.26. Через $a(v_1, ..., v_r)$ обозначен сигнал готовности данных $v_1, ..., v_r$, через $b(v_1, ..., v_r)$ – сигнал готовности результата, вычисляемого оператором. Предикат $p(v_1, ..., v_r)$ является r-местным и принимает два значения: T (истина) и F (ложь).

Таким образом, в операторах распознаватель, клапан и смеситель маркер считается двухцветным (T или F), вследствие чего в эквивалентных фрагментах СП им соответствуют пары взаимоинверсных позиций, обозначаемых T и F.

Обратим внимание на то, что двухцветные управляющие маркеры в СПД сосуществуют с обычными информационными монохроматическими маркерами, интерпретирующими готовность данных. Эти два типа маркеров распространяются вдоль двух типов дуг. Одни дуги заканчиваются обыч-

ными стрелками и появление маркеров на них моделирует перенос данных. Другие дуги заканчиваются треугольниками и используются для моделирования управляющих воздействий.

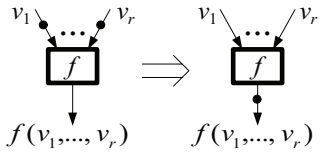
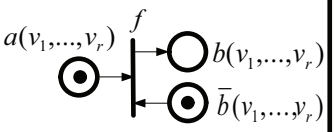
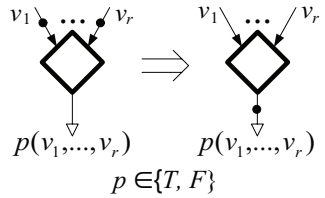
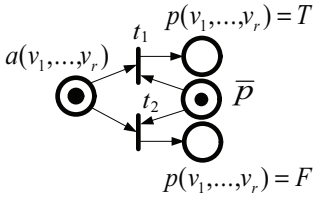
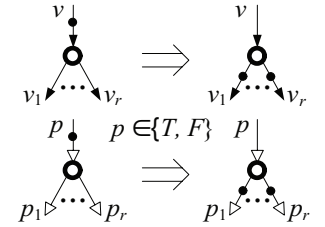
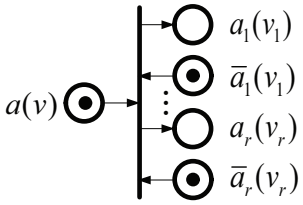
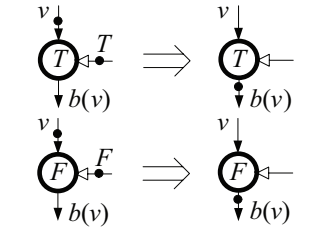
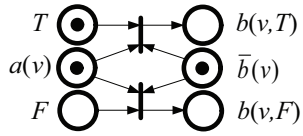
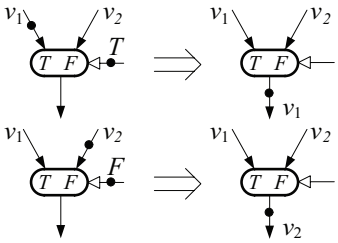
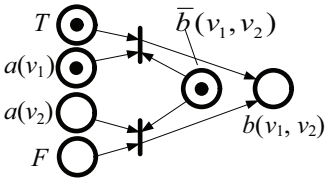
Оператор	Изображение и правило срабатывания	Эквивалентный фрагмент сети Петри
Актор		
Распознаватель		
Склейки (бифуркаторы)		
Клапаны типов *T* и *F*		
Смесители		

Рис. 3.26. Операторы сетей потоков данных и их интерпретация

Наличие пар взаимоинверсных маркеров (a и $\overline{a}$, b и $\overline{b}$) во фрагментах СП, эквивалентных операторам СПД, объясняется необходимостью обеспечения безопасности, как и в случае *E*-сетей.

В качестве иллюстрации рассмотрим следующую итеративную программу:

```
input x, y
n := 0
while x > y do
    x := x - y
    n := n + 1
end
output x, n
```

Очевидно, что она реализует операцию деления $n = x/y$, где x – делимое (остаток), а y – делитель. Соответствующая СПД представлена на рис. 3.27. Она состоит из двух акторов, осуществляющих вычитание $(x - y)$ и прибавление единицы $(n + 1)$, одного распознавателя, осуществляющего вычисление значения предиката $p(x > y)$, трех клапанов типа *T*, двух клапанов типа *F*, трех смесителей и пяти склеек.

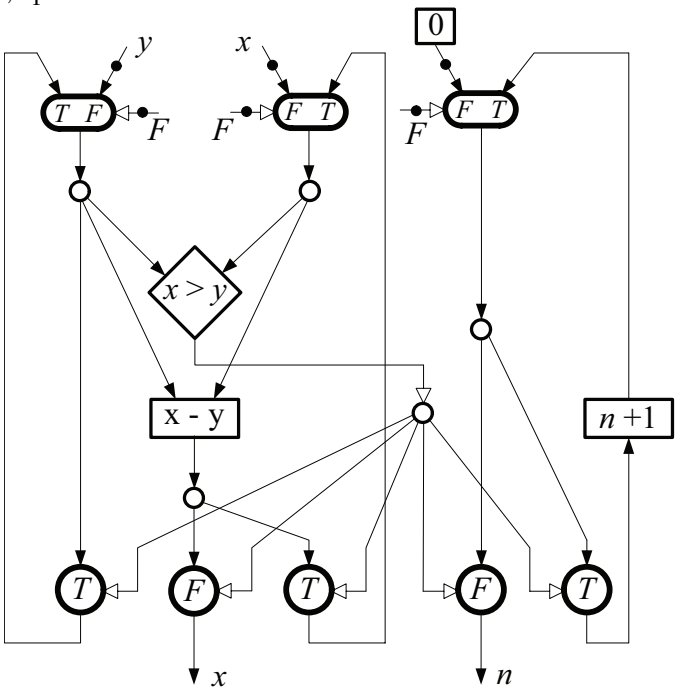


Рис. 3.27. Сеть потоков данных для алгоритма деления

При заданной начальной маркировке дуг возбуждены все три смесителя, срабатывание которых разрешает (после разветвлений склейками) обработку входных данные *x*, *y* и константы 0. После их получения в левом фрагменте схемы параллельно работают актор $x - y$ и распознаватель $x > y$. Если значение этого предиката истинно (*T*), то новое значение *x* и значение *y* через клапаны *T* зацикливаются. Одновременно в правом фрагменте через другой клапан *T* запускается актор $n + 1$, играющий роль счетчика. Этот процесс циклически повторяется до тех пор, пока распознаватель $x > y$ не выдаст маркер *F*. В этом случае в правом и левом фрагментах сработают клапаны *F*, выдавая на выход искомые значения частного *n* и остатка *x*.

Для того, чтобы снова возобновить процесс, необходимо восстановить начальную маркировку. По традиции, сложившейся в литературе по СПД, репозиция констант (в рассматриваемом примере константа 0) не осуществляется (считается, что соответствующий ей маркер стоит вечно). **Эффективность СПД по сравнению с СП сомнительна в силу того, что изобразительные средства СПД (а значит и алгоритмы анализа) явно сложнее, а функциональные возможности значительно ниже, так как они равносильны возможностям безопасных СП свободного выбора. Тем не менее, модель СПД прижилась в проблематике, связанной с работами в области потоковых архитектур вычислительных средств**.

3.9. Разрешимость задач анализа

В теории сетей Петри одним из основных направлений является так называемая общая теория сетей, которая занимается формализаций и изучением различных сетевых моделей, а также связей между ними и моделями из других разделов вычислительной науки и математики. В этой главе уже рассматривались некоторые вопросы общей теории сетей.

Выразительная мощность описанных моделей, под которой понимается возможность компактного задания (спецификации) моделируемых систем, в ряде случаев оказывается выше, чем выразительная мощность ординарных СП. Это обстоятельство, в конечном итоге, является определяющим для практики. Главным мотивом появления сетевых моделей и их изучения являются попытки увеличения моделирующей мощности, т.е. функциональных возможностей модели по сравнению с ординарными СП. Однако, как уже указывалось, эти попытки чреваты потерей разрешающей мощ-

ности моделей, т.е. уменьшением возможностей анализа поведения спецификаций. Часть проблем анализа оказывается при этом алгоритмически неразрешимой.

Уже говорилось, что сети с взвешенными дугами и раскрашенные сети с ограниченным количеством цветов по функциональным возможностям и разрешающей мощности равномощны ординарным СП, но имеют большую выразительную мощность.

Исследования ингибиторных, приоритетных, временных, синхронных и самомодифицируемых сетей показали, что эти классы сетей по моделирующей мощности равномощны машине Тьюринга, т.е. способны моделировать любой вычислительный процесс. Речь идет, конечно, об общем случае этих моделей, допускающем неограниченное накопление маркеров. Идея доказательства этого утверждения базируется на построении специфичных конструкций сетей и демонстрации того факта, что они адекватно моделируют работу так называемых регистровых (или счетчиковых) машин, машин Минского и др. В свою очередь, доказано, что эти машины равномощны машинам Тьюринга. Таким образом, основой доказательства является сводимость одной проблемы к другой. Известно, что регистровая машина эквивалентна машине Тьюринга, если в ней используются несколько бесконечных (по числу разрядов) регистров, способных выполнять следующие команды:

1) увеличить содержимое регистра на единицу,

2) уменьшить содержимое регистра на единицу, если он не пуст,

3) если регистр пуст, перейти к команде с меткой m.

Для спецификации регистровой машины с помощью сетевой модели каждый регистр представляется соответствующей позицией сети, счетчик команд – счетчиком, емкость которого равна числу команд в программе плюс единица. Сами команды реализуются следующим способом. Команды 1) и 2) реализуются ординарными сетями Петри так, как реализуются семафорные s- и p-операции, которые будут описаны в гл. 5. Команды 2) и 3) по существу равносильны программному сегменту, представленному в разд. 3.3, так что центральным моментом доказательства является демонстрация возможности представления операции условного вычитания (проверки на нуль) в рассматриваемой модели. Такие доказательства были приведены в разд. 3.3 – 3.5.

Из вышесказанного непосредственно следует, что **из универсальности указанных сетевых моделей следует ограничение их разрешающей мощности, т.е. невозможность построения эффективных алгоритмов решения некоторых массовых проблем, например, достижимости, живости и др**.

Остается ответить еще на один вопрос, **не является ли факт неразрешимости проблем непреодолимым препятствием для практического использования универсальных сетевых моделей?** Ответ таков: **нет**, не является, потому что **в подавляющем большинстве случаев моделировать реальные дискретные системы удается при конечном числе состояний (маркировок). Такие дискретные системы равномощны конечным автоматам, для которых интересующие нас массовые проблемы разрешимы**.

Что касается стохастических сетей Петри, *E*-сетей и сетей потоков данных, то их проблематика скорее относится не к специальной теории сетей, а к прикладному направлению исследований, когда в центре внимания лежат вопросы приложений сетей к задачам анализа определенного класса дискретных систем.

3.10. Упражнения

1. Постройте граф достижимых маркировок для сети со взвешенными дугами, изображенной на рис. 3.28.

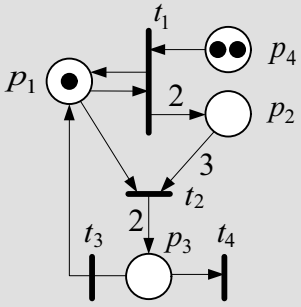


Рис. 3.28. Пример сети с взвешенными дугами

2. Постройте сеть с взвешенными дугами для спецификации химической реакции:

а) $5H_3PO_3 + 2KMnO_4 + 3H_2SO_4 = 5H_3PO_4 + 2MnSO_4 + K_2SO_4 + 3H_2O$,

б) $4HCl + O_2 \Leftrightarrow 2H_2O$ (пар) $+ 2Cl_2$.

3. Постройте сеть Петри, адекватную реакции образования протонов при поглощении α-частиц ядрами азота

$$_2He^4 +_7 N^{14} \rightarrow_9 F^{18} \rightarrow_1 H^1 +_8 O^{17}$$

с учетом нестабильности ядра фтора.

4. Постройте на базе модели СП спецификацию для ядерной реакции, происходящей при облучении бериллия α-частицами с образованием свободного нейтрона $_0n^1$:

$$_4Be^9 +_2 He^4 \rightarrow_6 C^{13} \rightarrow_6 C^{12} +_0 n^1.$$

5. Определим такой подкласс раскрашенных СП, в котором каждый переход пропускает определенный цвет, т.е. если все его входные позиции имеют, скажем, маркеры красного цвета, то он возбуждается и после срабатывания изымает по одному маркеру из каждой входной позиции и помещает по маркеру того же цвета в каждую выходную позицию. Назовем такой подкласс *селективными раскрашенными СП*. Найдите эффективный способ задания функции Ψ, для этого подкласса, приведите простой пример селективной сети и постройте для нее диаграмму маркировок.

6. Определим подкласс *перекрашивающих СП*, в котором каждый переход настроен на восприятие определенного цвета. Точнее, переход возбуждается, если все его входные позиции помечены маркерами того же цвета, и после срабатывания перехода происходит изъятие маркера из каждой входной позиции и добавление в каждую выходную позицию маркера другого, но заранее определенного цвета. Найдите эффективный способ задания функции Ψ для перекрашивающих СП, приведите простой пример такого типа сети и постройте для нее диаграмму маркировок.

7. Представьте фрагмент СП с тормозящими дугами, показанный на рис. 3.29, в виде эквивалентного фрагмента самомодифицируемой СП.

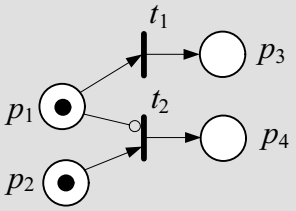


Рис. 3.29. Фрагмент сети Петри с тормозящими дугами

8. Представьте фрагмент приоритетной СП, показанный на рис. 3.30, в виде эквивалентного фрагмента самомодифицируемой СП. Предположите, что приоритет перехода t_1 выше приоритета перехода t_2.

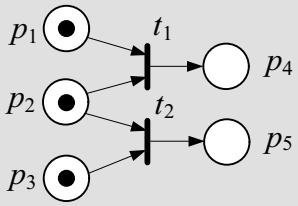


Рис. 3.30. Приоритетная сеть Петри

9. Определите вид условия пребывания одного или более маркеров в позиции p_5 в зависимости от соотношения чисел срабатываний переходов t_1, t_2, t_3 и t_4 для ингибиторной СП, представленной на рис. 3.31.

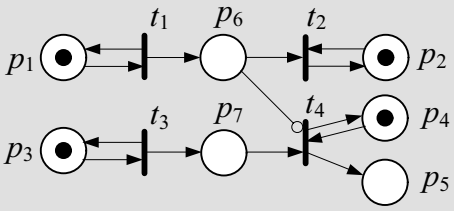


Рис. 3.31. Пример сети Петри с тормозящими дугами, $pr(t_1) \geq pr(t_2)$

10. Представьте фрагмент СП с тормозящими связями, моделирующий неограниченную генерацию маркеров.

11. Предложите формальное описание СП с тормозящими входами и правило срабатывания переходов таких сетей.

12. Постройте ординарную СП, эквивалентную ингибиторной СП, приведенной на рис. 3.6а, и докажите эквивалентность этих сетей.

13. Рассмотрите следующую модификацию модели временных сетей Петри, задаваемую как двойка $< PN, \beta >$, где PN – ординарная СП, а β – функция, задающая собственные задержки позиций сети, $\beta : P \to N^+$, N^+ – множество неотрицательных целых чисел. В этой модели принимается, что 1) маркер в позиции $p_j \in P$ временной сети может находиться в двух статусах: “доступен” и “недоступен”; 2) вре-

мя нахождения маркера в статусе “недоступен” фиксировано функцией β и равно τ_j; 3) переход $t_j \in T$ возбуждается при условии, что во всех его входных позициях содержится, по крайней мере, по одному маркеру в статусе “доступен”; 4) срабатывание возбужденного перехода происходит мгновенно после возбуждения; 5) правило срабатывания возбужденного перехода состоит в удалении из каждой его входной позиции одного маркера и помещении одного маркера в каждую выходную позицию перехода; 6) в результате акта срабатывания перехода маркеры забрасываются в выходные позиции в статусе “недоступен”. Попытайтесь свести эту модификацию к модели временных сетей с функцией задержки, заданной на переходах.

14. Постройте диаграмму маркировок для синхронной СП, представленной на рис. 3.32.

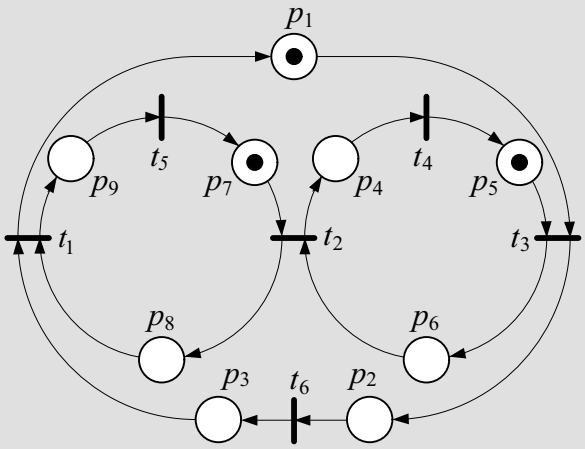


Рис. 3.32. Пример синхронной сети Петри

15. Начните построение диаграммы маркировок для синхронной СП со взвешенными дугами, представленной на рис. 3.33. Постарайтесь выявить закономерности изменения числа маркеров в позициях этой сети.

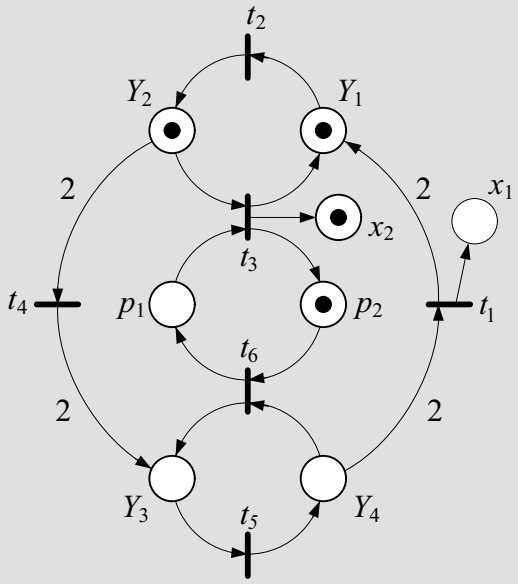


Рис. 3.33. Пример синхронной сети Петри со взвешенными дугами

16. Вычислите финальные вероятности пребывания стохастической СП, изображенной на рис. 3.17, во всех маркировках в предположении, что ее начальная маркировка μ^0 содержит два маркера в позиции p_1.

17. Найдите аналитические выражения для финальных вероятностей пребывания маркера в каждой из позиций p_1 – p_4 СТСП, показанной на рис. 3.34.

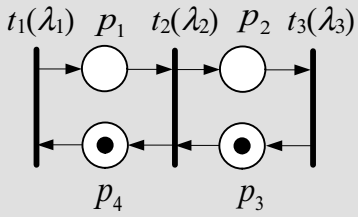


Рис. 3.34. Пример СТСП

18. Рассмотрите следующую элементарную модель экосистемы "Пастбище", которая отражает процесс поведения в ней молекулы фосфора. Эта модель представлена на рис. 3.35. Пребывание маркера в позициях p_1 – p_4 означает следующее:

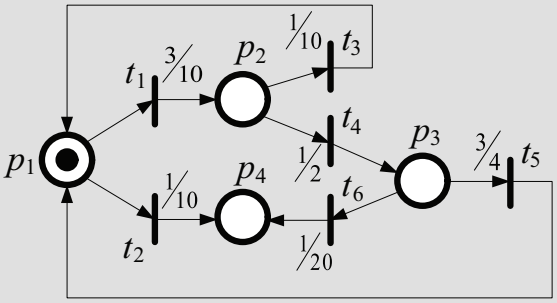


Рис. 3.35. Элементарная модель экосистемы "Пастбище"

p_1 – молекула фосфора находится в почве,

p_2 – молекула абсорбирована каким-либо растением и оказалась в травяном покрове,

p_3 – траву съел скот и она находится в желудке животного,

p_4 – молекула вышла из экосистемы в окружающую среду.

Семантика срабатывания переходов:

t_1 – молекула поглощена растением,

t_2 – эрозия почвы,

t_3 – гибель и разложение растения,

t_4 – поедание растения скотом,

t_5 – молекула возвращена скотом в почву,

t_6 – скот поступил на рынок.

Пусть интенсивности λ_i срабатывания переходов t_i такие, как они указаны на рис. 3.35. Рассчитайте финальную вероятность попадания молекулы фосфора в окружающую среду. Является ли построенная вами марковская цепь эргодической или поглощающей?

19. Анализ хоккейного матча между ЦСКА и столичным "Спартаком" позволил сделать следующие выводы: если шайба находится в центре, то после этого она с вероятностью 0,4 входила на половину поля ЦСКА и с вероятностью 0,6 – на половину поля "Спартака". Со стороны поля ЦСКА шайба посылалась обратно в центр с вероятностью 0,95 или в ворота "Спартака" с вероятностью 0,05 (ЦСКА забивал гол). Со стороны

поля “Спартака” шайба шла в центр с вероятностью 0,95 и забивалась в ворота ЦСКА с вероятностью 0,1. Пусть шайба первоначально находится в центре. Изобразите ССП, соответствующую данной задаче и описывающую ход игры до первого гола. Вычислите вероятность того, что первым гол забьет ЦСКА.

20. Пример генератора маркеров, специфицированного на языке СП, показан на рис. 3.36. Объясните, почему не работоспособна *E*-сеть, представленная тем же рисунком. Как представить конструкцию генератора единиц, пользуясь примитивами *E*-сетей?



Рис. 3.36. Реализация генератора единиц на языке сетей Петри

21. Представьте на языке сетей Петри процесс, специфицированный *E*-сетью, изображенной на рис. 3.24. Оцените сравнительную сложность и наглядность обеих спецификаций.

22. Предложите вариант использования разрешающей ячейки *E*-сетей в качестве выхода примитивов и интерпретацию цветов маркеров в этой ячейке.

23. Изобразите схему потоков данных для следующего фрагмента программы на языке типа PL/1:

```
Z: PROCEDURE (X,Y);
DO I=1 TO X;
   IF Y>1 THEN Y=Y*Y;
          ELSE Y=Y+Y+2;
END;
END;
```

24. Постройте схему потоков данных для вычисления k-го числа Фиббоначи $F(k)$. Рядом Фиббоначи называется ряд 1, 1, 2, 3, 5, 8, 13, 21, ..., в котором i-е число Фиббоначи задается в виде суммы $(i-1)$-го и $(i-2)$-го чисел, т.е. $F(i) = F(i-1) + F(i-2)$, причем $F(1) = F(2) = 1$. Сравните ее с программой, изображаемой сетью Петри.



Глава 4.

ДИНАМИЧЕСКАЯ ЛОГИКА

“Но трудно разобраться
Іде ясно, где туман,
В потоке информации
С поправкой на обман”.
Андрей Макаревич “Нас еще не согнули годы”.

Несколько замечаний, являющихся необходимой преамбулой, поясняющей предмет рассмотрения этой главы.

Термин “динамическая система” обычно употребляют для систем различной природы, изучаемой в различных областях науки и техники – в математике, информатике, искусственном интеллекте, технике, гуманитарных науках (например, философии и истории) и т.д. Необходимым свойством динамических систем является то обстоятельство, что процессы в них развиваются во времени.

Название этой главы – *динамическая логика* – может немедленно вызвать глобальные ассоциации, от которых авторы просят читателей дистанцироваться, а сами авторы решительно открещиваются от расширительного толкования названия. Дело в том, что в научной литературе можно встретить далеко не «пустое» множество терминов, эксплуатирующих название «динамическая логика» или близкие к нему для разных предметных областей. Приведем некоторые из них: логика динамических систем, динамическая логика знания, динамическая логика игр и т.п. В математике есть раздел, называемый *динамическими логиками* (во множественном числе), в котором рассматриваются различные формализмы. Часто динамические логики объединяются единым понятием *модальная логика*, в которой некоторые детально разработанные формализмы заключает стандартная фраза: “интерпретация в настоящий момент не найдена”. Предложены языки динамических логик, существуют также так называемые *программные логики*. Языки динамических логик предоставляют богатые выразительные средства, благодаря чему находят приложения в различных областях знаний (в первую очередь, связанных с описанием вычислений). Часть этих языков практически не может быть использована в силу того, что они не являются полиномиально разрешимыми.

В этой главе предметом рассмотрения будет модель, предназначенная главным образом для изучения процессов в асинхронных схемах. Эта модель, содержащая средства как статического, так и динамического описания схем и процессов их функционирования, имеет много общего с концепцией асин-

хронных процессов и с моделью сетей Петри. **Совокупность методов анализа и синтеза асинхронных схем определенного класса – так называемых самосинхронных схем (рассматриваемых в главе 6), поведение которых инвариантно к величинам задержек элементов, – авторы по ряду причин были вынуждены назвать динамической логикой, не претендуя на расширительное толкование этого термина.** По-видимому, можно было бы придумать и более адекватное название, скажем – логика самосинхронных схем, но аппарат, изложенный далее, не ограничивается применением только в схемотехнике. Одним из существенных достоинств подходов, рассмотренных в этой главе, является полиномиальная, а не экспоненциальная сложность алгоритмов.

К моделям, объединенных названием динамическая логика, будем относить следующие модели: диаграммы переходов, схемы и диаграммы Маллера, сигнальные графы и диаграммы изменений. Как уже говорилось, эти модели развивались применительно к анализу и синтезу самосинхронных схем. Тем не менее они могут оказаться полезными и в программировании при разработке управления программными средствами, поскольку хорошо описывают координацию и синхронизацию процессов на уровне их причинно следственных отношений. Поэтому даже читателям, которых интересуют только вопросы программирования и не интересуют проблемы, связанные со схемотехникой, рекомендуется прочитать нижеследующие параграфы, опуская вопросы, относящиеся к схемотехнике.

4.1. Диаграммы переходов и их классификация

Диаграмма переходов **(ДП) представляет собой ориентированный граф** $< S, F >$, **где** S – **конечное множество вершин, или *состояний*, а** $F \subseteq S \times S$ – **конечное множество дуг, или *переходов*. Каждое состояние кодируется набором символов фиксированной длины** n, **а символами (компонентами) этого набора являются значения переменных** x_i, $1 \le i \le n, x_i \in X$, где X – **конечное множество дискретных переменных**. Значения переменных могут быть заданы на конечном множестве неотрицательных целых чисел N^+. **Если** $N^+ = \{0,1\}$, **то ДП называется двоичной или бинарной**[1] (см. рис. 4.1).

[1] Иными словами, бинарная ДП состоит из наборов (векторов) значений двоичных переменных.

Если $s_u F s_v$, $u \neq v$, то дуга ведет из s_u в s_v, и значения тех компонент (переменных) x_i, которые в этих состояниях имеют разные значения, помечаются звездочкой в состоянии s_u. Компонента, значение которой помечено звездочкой, называется *возбужденной*, и для нее возможно выполнение некоторого действия, связанного с изменением ее значения. Компоненты, значения которых не помечены звездочками, называются *устойчивыми*. *Срабатыванием* переменной x_i будем называть действие, связанное с фактом смены состояния s_u, в котором x_i была возбуждена, состоянием s_v, в котором значение x_i изменилось. Обычно из диаграмм переходов удаляются все дуги, связанные с одновременным переключением двух и более возбужденных переменных, так как понятие одновременности не определено.

Таким образом, диаграмма переходов описывает все возможные последовательности изменения значений переменных, упорядоченных отношением F, причем время каждого перехода (изменения значения переменной) считается произвольным, но конечным.

Если в диаграмме переходов выделяется исходное состояние, то такая диаграмма переходов называется *инициальной*.

Пример диаграммы переходов представлен на рис. 4.1. **При сравнении моделей асинхронного процесса (см. главу 1) и диаграммы переходов легко видеть, что ДП является модельной интерпретацией АП**, причем:

а) состояния бинарной диаграммы переходов суть структурированные ситуации асинхронного процесса, состоящие из n компонент, каждая из которых принимает значения из множества {0, 0*, 1, 1*};

б) отношение $F \subseteq S \times S$ в диаграмме переходов – это отношение непосредственного следования ситуаций в асинхронном процессе;

в) инициальное состояние в диаграмме переходов соответствует единственному инициатору в асинхронном процессе, если только диаграмма переходов не является автономной;

г) тупиковые состояния неавтономной диаграммы переходов могут быть объявлены результантами.

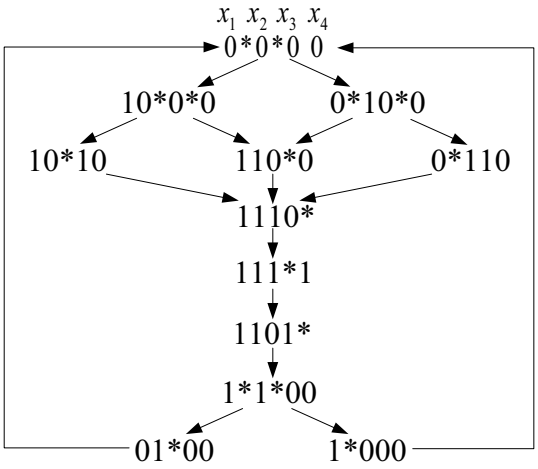


Рис. 4.1. Пример диаграммы переходов

Далее с целью дать классификацию диаграмм переходов приведены некоторые определения, причем только для бинарных ДП, так как обобщения на случай ДП произвольной значности могут быть получены достаточно просто.

Состояния бинарной диаграммы переходов

Состояние бинарной диаграммы переходов называется:

- ***кратным***, если в ДП найдется, по крайней мере, еще одно такое состояние, которое закодировано тем же набором значений переменных, что и данное состояние (без учета звездочек, например, 10*1 и 1*01); если в ДП встречаются кратные состояния, то это означает потерю информации о месте пребывания процесса в диаграмме в некоторый момент его развития (т.е. наличие недетерминизма);
- ***бифуркантным***[2], если в нем возбуждено более одной переменной;
- ***детонантным по*** x_i, если оно бифуркантно, переменная x_i в нем устойчива, но в двух или более непосредственно достижимых из него состояниях она становится возбужденной, т.е. при переходе из детонантного состояния обязательно происходит ветвление и непо-

[2] Термин “бифуркантное состояние” происходит от англ. *biforked* – раздвоенный; в русской терминологии обозначает состояние ветвления не обязательно на два, но, возможно, и на большее число ветвей

средственной причиной возбуждения переменной x_i является переключение любой из двух или более других переменных;

– ***конфликтным по*** x_i, если переменная x_i в нем возбуждена и найдется такое непосредственно достижимое из него состояние, в котором x_i имеет то же значение, но является устойчивой, т.е. для x_i возможны переходы вида $1^* \to 1$ или $0^* \to 0$.

Бифуркантное состояние

Отдельно рассмотрим бифуркатное состояние. Пусть в бифуркантном состоянии s возбуждены $k \geq 2$ компонент.

Подкубом[3] бифуркантного состояния назовем множество из 2^k состояний, образованных из s всеми возможными фиксациями значений возбужденных компонент (нулями и единицами). Если все состояния подкуба непосредственно достижимы из s, то такой фрагмент назовем *гамаком*. *Подгамаком* назовем гамак, в котором некоторые состояния подкуба либо не заданы, либо отсутствуют некоторые дуги, имеющиеся в гамаке.

Так, фрагмент ДП, изображенный на рис. 4.2,а, является гамаком, так как он содержит все 4 (2^2) состояния и полный набор связывающих их дуг, в отличие от подгамака на рис. 4.2,б (в нем отсутствует переход $0^*1 \to 11$) или на рис. 4.2,в (в нем отсутствует состояние 0^*1).

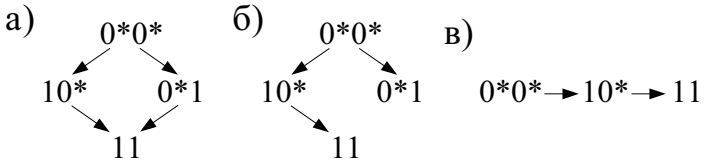


Рис. 4.2. Примеры а) гамака, б) и в) подгамаков

Диаграмму переходов будем называть *правильной*, если для любого бифуркантного состояния s выполняется одно из следующих условий:

[3] При использовании геометрической интерпретации диаграммы переходов в виде гиперкуба, по аналогии, как это делается при геометрической интерпретации булевых функций, состояния диаграммы (векторы длины n) располагаются в вершинах n-мерного гиперкуба. Тогда конфигурация из вершин, число которых равно k-ой степени двойки ($k < n$), может образовывать подкуб данного гиперкуба размерности k.

1) все состояния его подкуба образуют гамак;

2) некоторые состояния его подкуба образуют подгамак, а остальные состояния подкуба вне гамака в ДП не встречаются.

Пример правильной диаграммы представлен на рис. 4.1, пример неправильной диаграммы – на рис. 4.3.

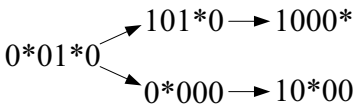


Рис. 4.3. Пример неправильной диаграммы переходов

Классы диаграмм переходов

Бинарная диаграмма переходов называется:

- ***противоречивой***, если она содержит хотя бы одну пару кратных состояний;
- ***полумодулярной***, если она непротиворечива и не содержит конфликтных состояний;
- ***дистрибутивной***, если она полумодулярна и не содержит детонантных состояний;
- ***последовательной***, если она дистрибутивна и не содержит бифуркантных состояний.

Поясним происхождение этой классификации. Любой бинарной ДП можно поставить в соответствие так называемую кумулятивную диаграмму (см. разд. 4.4), которая представляет собой алгебраическую структуру, или решетку. Указанная классификация принята в математике для алгебраических структур. Оказалось разумным использовать эту же классификацию и для диаграмм переходов. Опуская формальные определения классов алгебраических структур, будем указывать свойства диаграмм переходов, относящие их к тому или иному классу, при рассмотрении соответствующих примеров.

В качестве иллюстрации введенных понятий рассмотрим бинарную ДП, показанную на рис. 4.1. В ней состояния 110*0 и 1*1*00 являются кратными, или противоречивыми, поскольку они представлены по сути одним и тем же двоичным вектором, но с разными возбужденными элементами. Состоя-

ние 0*0*00 детонантно по переменной x_3, так как x_3 возбуждается либо при переключении переменной x_1 (10*0*0), либо при переключении переменной x_2 (0*10*0), т.е. в соответствии с правилом ИЛИ, что является нарушением дистрибутивности. Состояния 0*0*00, 10*0*0, 0*10*0 и 1*1*00 бифуркантны. Конфликтные состояния, т.е. состояния в которых переключение одного из возбужденных элементов вызывает снятие возбуждения другого элемента, в данной ДП отсутствуют. Таким образом, она является противоречивой и, следовательно, неполумодулярной.

На рис. 4.4 приведены примеры классов диаграмм переходов. Все диаграммы являются непротиворечивыми. В диаграмме переходов на рис. 4.4,а состояние 0*0*00 является детонантным по переменной x_3. Эта диаграмма не содержит конфликтных состояний и, следовательно, является полумодулярной. Обозначим класс полумодулярных диаграмм через *U*.

В диаграмме переходов, изображенной на рис. 4.4,б, нет детонантных состояний, и она относится к классу *D* дистрибутивных диаграмм. Диаграмма переходов на рис. 4.4,в относится к классу *K* последовательных диаграмм.

Согласно введенным определениям легко заметить, что для классов *K*, *D* и *U* диаграмм справедливо следующее отношение $K \subset D \subset U$, характеризующее вложенность классов.

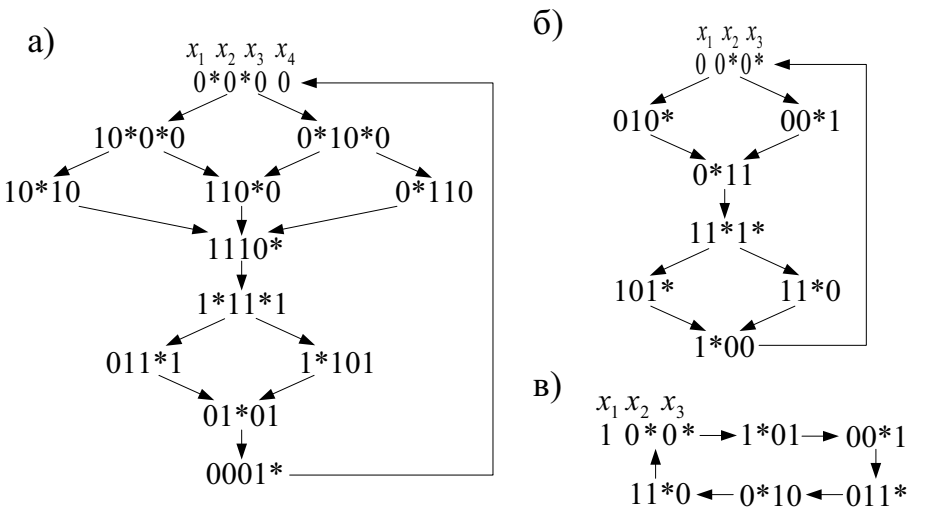


Рис. 4.4. Примеры диаграмм переходов: а) полумодулярной недистрибутивной, б) дистрибутивной и в) последовательной

4.2. Схемы Маллера

Ниже будет показано, что бинарной[4] полумодулярной диаграмме переходов может быть поставлена в соответствие схема Маллера, представляющая собой систему из булевых уравнений. Эта система может быть реализована как схемотехническими, так и программными средствами. В этом смысле схема Маллера может оказаться полезной для специалистов обоих профилей.

Схемой Маллера **будем называть модель, заданную системой булевых уравнений вида** $x_i = f_i(X) = f_i(x_1,...,x_i,...,x_n)$. В инициальной модели задается также исходное состояние, представленное фиксированными значениями булевых переменных x_1, x_2,..., x_n, т.е. набором $s = \{0,1\}^n$.

Поясним происхождение названия “схема Маллера”. Системе из n булевых функций может быть однозначно поставлена в соответствие асинхронная логическая схема, состоящая их n логических элементов с выходами $x_1,...,x_i,...,x_n$. Собственной функцией элемента с выходом x_i является функция f_i системы, аргументы которой принадлежат некоторому подмножеству выходов всех элементов схемы. Так, системе

$$\begin{cases} x_1 = x_2 x_3, \\ x_2 = \overline{x}_1, \\ x_3 = \overline{x}_1 \end{cases}$$

может быть поставлена в соответствие изображенная на рис. 4.5 схема, состоящая из трех элементов: элемента типа И с выходом x_1 и со входами x_2, x_3 и двух инверторов с выходами x_2, x_3 и со входом x_1. Описание в виде системы уравнений задает не только типы элементов, но и связи (соединения) между ними.

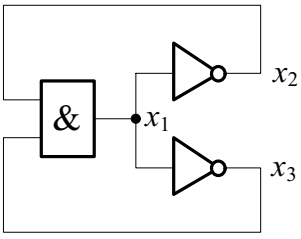


Рис. 4.5. Пример схемы Маллера

[4] Вершинами бинарной ДП являются двоичные векторы одинаковой длины.

Такой способ спецификации асинхронных схем в виде системы уравнений, был предложен Д.Е. Маллером (США).

В последнем примере элементы схемы Маллера не содержат петель обратной связи от выхода элемента к его входам.

Те переменные x_i, для которых в уравнении $x_i = f_i(X)$ его собственная функция f_i несущественно зависит от x_i (т.е. имеет место $f_i(x_i = 0) = f_i(x_i = 1)$), будем называть *несамозависимыми*. Напротив, **если собственная функция переменной x_i существенно зависит от x_i, то такую переменную будем называть *самозависимой*.** Эти определения понадобятся в дальнейшем.

Переменная x_i в схеме называется *возбужденной* в состоянии s, если $s_i \neq f_i(s)$, где s_i – значение переменной x_i, а f_i – значение собственной функции на наборе s. В противном случае, когда $s_i = f_i(s)$, переменная x_i называется *устойчивой*.

Возбужденные переменные, стремясь стать устойчивыми, порождают динамическое поведение схемы, состоящее в переходе из начального состояния в другие состояния, которые определяются в соответствии с системой уравнений. В свою очередь, срабатывание переменных, возбужденных в этих состояниях, приводит к новым состояниям и т.д.

Гипотезы о задержках в логических элементах и проводах

Говоря о динамике поведения логических схем, следует сначала принять гипотезу о характере задержек элементов асинхронной схемы и соединительных проводов. Далее предполагается, что асинхронные схемы должны удовлетворять следующей гипотезе:

1) задержка элемента сосредоточена на его выходе, иными словами модель элемента схемы представляется в виде последовательного соединения безынерционного элемента, реализующего собственную логическую функцию, и элемента задержки;

2) величина задержек элементов заранее не известна, т.е. может быть произвольной, но конечной;

3) задержки элементов являются *инерциальными*, т.е. если элемент имеет задержку D, то изменения значений сигналов на его входах, длительность которых меньше D, не пропускаются на его выход (фильтруются), а сигналы с длительностью более D могут появиться на выходе элемента со сдвигом на время D;

4) задержка в проводе до разветвления может быть любой, но конечной, так как она приводится к задержке элемента, к выходу которого подключен этот провод (т.е. если задержка элемента равна D, а задержка провода равна d, то в результате приведения принимается, что задержка элемента равна $D + d$, а задержка провода равна нулю); задержки в проводах после разветвления, напротив, приводятся к входам элементов, с которыми они соединены, и разброс задержек в этих проводах должен быть меньше минимальной задержки элементов схемы.

В рамках модели Маллера каждый элемент схемы работает следующим образом. Будучи возбужден, элемент через некоторое время переходит в устойчивое состояние, изменив значение своего выхода. Если же в возбужденном состоянии элемента набор значений сигналов на его входах изменится так, что условие возбуждения будет снято, то элемент оказывается в устойчивом состоянии и не изменяет значение выхода. Интуитивно ясно, что переход из возбужденного состояния в устойчивое посредством снятия возбуждения неприемлем в силу того , что в зависимости от быстродействия элемента он может либо успеть переключиться, либо не успеть и его возбуждение будет снято. В этом случае независимо от инерциальных свойств элементов могут возникнуть аномальные явления типа всплесков и дребезга сигналов, называемые обычно *логическими состязаниями*.

Самосинхронные схемы

В дальнейшем нас будут интересовать только схемы, поведение которых не зависит от величин задержек элементов. Такие схемы принято называть *самосинхронными*. На языке модели Маллера к ним как раз относятся *полумодулярные схемы*[5], т.е. схемы, диаграммы переходов которых полумодулярны. Для самосинхронных схем требование инерциальности задержек элементов может быть ослаблено: для них достаточно, чтобы задержки элементов были *чистыми*, т.е. не обладающими фильтрующими свойствами.

Диаграммы Маллера

Очевидно, что функционирование схемы Маллера может быть описано посредством бинарных диаграмм переходов, но так как такие диаграммы обладают определенной спецификой, в дальнейшем будем называть ДП, порождаемые схемами Маллера, *диаграммами Маллера* (ДМ).

[5] Принятую для диаграмм переходов классификацию удобно распространить также и на самосинхронные схемы (полумодулярные, дистрибутивные, недистрибутивные, последовательные).

Относительно диаграмм Маллера известно следующее. Во-первых, не всякая диаграмма Маллера является полумодулярной. Во-вторых, диаграмма переходов принадлежит классу диаграмм Маллера тогда и только тогда, когда она бинарная, непротиворечивая и правильная.

Справедливость первого утверждения иллюстрируется рис. 4.6, на котором показана ДМ, порождаемая схемой Маллера, приведенной на рис. 4.5 с инициальным состоянием $x_1 = x_2 = x_3 = 0$. Исходное состояние схемы – 00*0*. Легко видеть, что состояния 1*01* и 1*1*0 являются конфликтными, так как из них достижимы состояния 00*1 и 01*0, в которых компоненты x_3 и x_2 соответственно перестают быть возбужденными, не изменяя при этом своего значения. Следовательно, ДМ не является полумодулярной.

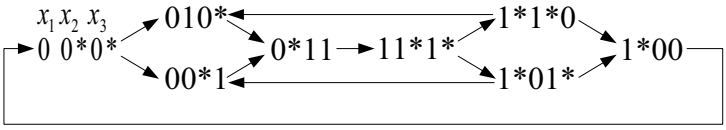


Рис. 4.6. Пример неполумодулярной диаграммы переходов

Справедливость второго утверждения может быть продемонстрирована сведением ДП, обладающей названными свойствами, к схеме Маллера, порождающей ДМ, совпадающую с заданной ДП. Правила такого перехода сводятся к составлению таблицы истинности в следующем порядке:

1) в левой части таблицы выписываются все наборы, встречающиеся в ДП;

2) против каждого набора левой части таблицы в правую ее часть выписываются наборы, в которых помеченные звездочками значения компонент набора левой части, изменяются на противоположные (т.е. 0* заменяется на 1, а 1* на 0), а остальные компоненты сохраняют свои значения.

3) не встречающиеся в левой части таблицы наборы могут быть доопределены произвольно в правой части таблицы.

Рассмотрим ДП, показанную на рис. 4.4,б. Она является бинарной, непротиворечивой и правильной (следовательно, является ДМ) и, кроме того, дистрибутивной. Выпишем для этой ДМ таблицу истинности, в которой в данном случае определены все $2^3 = 8$ наборов.

Таблица истинности Таблица 4.1

$x_1\ x_2\ x_3$	$x_1\ x_2\ x_3$
0 0*0*	0 1 1
1*0 0	0 0 0
0 1 0*	0 1 1
1 1*0	1 0 0
0 0*1	0 1 1
1 0 1*	1 0 0
0*1 1	1 1 1
1 1*1*	1 0 0

Рассматривая компоненты наборов правой части таблицы как функции, а наборы левой части таблицы как область задания этих функций (не обращая внимания на звездочки), нетрудно получить следующую систему булевых уравнений:

$$\begin{aligned} x_1 &= x_2x_3 \vee x_1(x_2 \vee x_3), \\ x_2 &= \overline{x}_1, \\ x_3 &= \overline{x}_1. \end{aligned} \qquad (4.1)$$

В общем случае одной диаграмме Маллера, в которой задано менее 2^n наборов длины n, соответствует некоторое множество систем уравнений (схем Маллера). И наоборот, системе уравнений с заданным начальным состоянием можно поставить в соответствие единственную ДМ.

Выше уже говорилось, что схема Маллера порождает диаграмму переходов. Вспомним, что описание сети Петри в виде орграфа с начальной маркировкой являлось структурной спецификацией, отражающей статическую компоненту модели. Ее динамической компонентой, порождаемой статической, являлась диаграмма маркировок. **Существование пары "статическая часть – динамическая часть" имеет место и в динамической логике. Статику задает схема Маллера (система уравнений), а динамику – соответствующая диаграмма переходов.** При этом статика задается весьма компактно: n уравнений для схемы из n элементов. Динамика, как всегда, требует экспоненциального роста сложности описания – до 2^n состояний диаграммы переходов.

Понятие динамической логики

Совокупность методов анализа и синтеза асинхронных схем в рамках модели Маллера естественно назвать динамической логикой. Некоторые вопросы, связанные с применением аппарата динамической логики, будут рассмотрены в данном разделе, дальнейшее изучение возможностей применения этого аппарата будет продолжено в гл. 6.

Алгоритм построения диаграммы переходов для заданной схемы Маллера основан на построении множества рабочих состояний схемы или так называемого *рабочего цикла схемы*. При этом ***рабочими состояниями*** называют состояния, достижимые из заданного исходного состояния. Зачастую мощность множества рабочих состояний существенно меньше 2^n. Учет этого обстоятельства позволяет упрощать анализ рабочего цикла.

Построение и анализ множества рабочих состояний схемы с начальным состоянием могут быть основаны на итеративных методах решения задачи прямой достижимости. Часто при анализе свойств схемы, в частности, принадлежности схемы тому или иному классу (полумодулярных, дистрибутивных или последовательных схем), целесообразно чередовать шаги решения задачи достижимости с проверкой наличия в полученном множестве достижимых состояний соответственно конфликтных, детонантных и бифуркантных состояний.

Задача построения рабочего цикла

Задача построения рабочего цикла может решаться, например, “лобовым” путем – итерационным вычислением значений правых частей системы уравнений для каждого из состояний очередного яруса и сравнением их со значениями переменных в этих состояниях. После выявления возбужденных переменных эти переменные могут быть “переключены” и в соответствии с их новыми значениями могут строиться состояния следующего яруса и т.д.

С целью повышения эффективности для решения задачи построения рабочего цикла используют метод, основанный на характеристических булевых функциях множеств состояний. Рассмотрим этот метод[6].

[6] Предполагается, что читатель знаком с некоторыми элементарными понятиями теории множеств и теории переключательных функций.

Состоянию s схемы можно поставить в соответствие конституенту единицы $\omega(s)$, которая определяется как конъюнкция ранга n

$$\omega(s) = \&_{i=1}^{n} x_i^{s_i},$$

где $x_i^{s_i} = x_i$, если $s_i = 1$, и $x_i^{s_i} = \bar{x}_i$, если $s_i = 0$.

В общем случае для некоторого множества состояний A характеристической функцией этого множества будет

$$\varphi_A(X) = \bigvee_{s \in A} \omega(s).$$

Например, для множества состояний A = {000, 010, 100, 110}, Заданных наборами значений переменных ир X ={x_1, x_2, x_3}, характеристическая функция имеет вид

$$\varphi_A(X) = \bar{x}_1\bar{x}_2\bar{x}_3 \vee \bar{x}_1 x_2 \bar{x}_3 \vee x_1\bar{x}_2\bar{x}_3 \vee x_1 x_2 \bar{x}_3 = \bar{x}_3.$$

Обозначим через W множество рабочих состояний. Для нахождения W можно использовать формулы вычисления характеристических функций состояний, достижимых на очередном шаге вычислений из некоторого множества уже найденных состояний. (Состояние s_v, достижимое из s_u за один шаг, образуется путем изменения значения одной и только одной из переменных x_i, возбужденных в s_u.) Примером такой формулы может служить следующее выражение

$$R^1(\varphi_A(X)) = \bigvee_{i=1}^{n} \varphi_A(x_i = \bar{x}_i)(\bar{x}_i \oplus f_i(x_i = \bar{x}_i)), \qquad (4.2)$$

где $R^1(\varphi_A(X))$ – множество состояний, достижимое из некоторого множества A за один шаг, x_i – i-я компонента состояний, а f_i – собственная функция i-й компоненты. Смысл формулы (4.2) состоит в следующем.

Во-первых, легко видеть, что выражение $\varphi_G(X) = \varphi_A(X)(x_i \oplus f_i(X))$ задает подмножество G состояний таких, что в них возбуждена переменная x_i. Действительно, функция $\varphi_A(X)$ истинна, если состояние принадлежит множеству A, а функция $x_i \oplus f_i(X)$ истинна, если в этом состоянии возбуждена компонента x_i, т.е. $x_i \neq f_i(X)$.

Во-вторых, искомое множество состояний $R^1(\varphi_A(X))$, достижимых из множества A за один шаг, состоит из состояний, отличающихся от состоя-

ний множества G значением компоненты x_i, возбужденной G. Такие состояния могут быть представлены в виде

$$\varphi_G(x_i = \bar{x}_i) = \varphi_A(x_i = \bar{x}_i)(\bar{x}_i \oplus f_i(x_i = \bar{x}_i)).$$

Здесь запись вида $F(x_i = \bar{x}_i)$ эквивалентна записи

$$F(x_1, x_2, ..., x_{i-1}, \bar{x}_i, x_{i+1}, ..., x_n),$$

т.е. в выражении для $F(X)$ переменная x_i заменяется на $\bar{x}_i$.

Используя пошагово формулу (4.2), можно, исходя из начального состояния, последовательно пополнять множество рабочих состояний схемы до тех пор, пока не будет построено все множество W. Для завершения этого процесса необходимо убедиться, что на очередном шаге не получены новые состояния, не фигурирующие на предыдущих шагах. Не останавливаясь подробно на способах эффективного манипулирования с вычисляемыми подмножествами рабочих состояний, отметим лишь, что все множество рабочих состояний разбивается на слои, множество которых упорядочено. Результативность алгоритмов работы со слоями состояний достигается тем, что в силу конечного числа состояний схемы ($|W| \le 2^n$), начиная с некоторого шага, слои состояний начнут повторяться.

В качестве примера рассмотрим построение рабочего цикла для схемы Маллера (4.1). Пусть начальным состоянием схемы является состояние 000, т.е. $A = \{000\}$. Обозначим нулевой слой $L_0 = \varphi_A(X) = \bar{x}_1\bar{x}_2\bar{x}_3$. Слой L_1 определяется по формуле (4.2) примененной к $\varphi_A(X)$,

$$L_1 = R^1(\bar{x}_1\bar{x}_2\bar{x}_3) = (x_1\bar{x}_2\bar{x}_3(\bar{x}_1 \oplus (x_2x_3 \vee \bar{x}_1(x_2 \vee x_3)))) \vee$$
$$\vee (\bar{x}_1x_2\bar{x}_3(\bar{x}_2 \oplus \bar{x}_1)) \vee (\bar{x}_1\bar{x}_2x_3(\bar{x}_3 \oplus \bar{x}_1)) = \bar{x}_1(x_2\bar{x}_3 \vee \bar{x}_2x_3).$$

Это выражение задает состояния 010 и 001. Аналогично могут быть получены следующие слои: $L_2 = \bar{x}_1x_2x_3$, $L_3 = x_1x_2x_3$, $L_4 = x_1(x_2\bar{x}_3 \vee \bar{x}_2x_3)$, $L_5 = x_1\bar{x}_2\bar{x}_3$, $L_6 = \bar{x}_1\bar{x}_2\bar{x}_3$. Легко видеть, что, так как $L_0 = L_6$, построение рабочего цикла схемы этим и завершается. В итоге $W = \bigcup_{i=0}^{5} L_1$.

4.3. Маркированные графы

Напомним, что *маркированные графы* - это такие сети Петри, в которых ни одна позиция не содержит более одного входного и одного выходного перехода. Это позволяет изображать их в виде однодольных орграфов, полученных заменой позиций дугами, на которые переносятся маркеры из исключенной позиции, и заменой переходов вершинами. Правила срабатывания возбужденных вершин маркированного графа состоит в изъятии одного маркера из каждой ее входной дуги и добавления одного маркера в каждую ее выходную дугу.

Отвлечемся на время от темы, относящейся к динамической логике, и вернемся к вопросам, связанным с изучением введенного в разд. 2.3 структурного подкласса сетей Петри, – маркированных графов. Это отвлечение является тактическим. В разд. 4.4 будет продолжена основная тема данной главы.

Маркированные графы эффективно моделируют параллельные процессы с ограниченным параллелизмом, что определяет необходимость их использования как базовой модели, на основе которой будет сформулирована модель сигнальных графов, адекватно описывающая поведение представительного класса асинхронных схем.

В силу своих структурных особенностей маркированные графы (МГ) могут быть изображены в виде монохроматических ориентированных графов с маркированными дугами.

Маркированный граф определяется как пара $< G, \mu^0 >$, **где**
$G = < V, E >$ – **орграф, в котором** V – **конечное множество вершин,**
$E \in V \times V$ – **множество дуг, а** μ^0 – **начальная маркировка,**
$\mu^0 : E \to N^+$, N^+ – **множество неотрицательных целых чисел.**

Пример маркированнного графа показан на рис. 4.7. Его начальная маркировка представлена размещением маркеров на дугах так, что дуга $e = (\nu_i, \nu_j)$ содержит $\mu(e)$ точек. Динамика работы маркированного графа задается правилом смены маркировок. Для орграфа $G = < V, E >$ вершина $\nu \in V$ возбуждена при маркировке μ, если $\mu(e) > 0$ для всех дуг $e \in I(\nu)$, где $I(\nu)$ – множество всех входящих в ν дуг. Обозначим через $O(\nu)$ множество дуг, исходящих из ν.

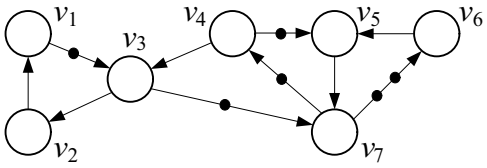


Рис. 4.7. Пример маркированного графа

После срабатывания вершины ν образуется новая маркировка μ', определяемая следующим образом. Для каждой дуги

$$\mu'(e) = \begin{cases} \mu(e) - 1, \text{ если } e \in I(v) \setminus O(v); \\ \mu(e) + 1, \text{ если } e \in O(v) \setminus I(v); \\ \mu(e), \text{ если } e \in O(v) \cap I(v). \end{cases}$$

Следовательно, срабатывание вершины (неделимая операция) вызывает изъятие одного маркера с каждой дуги из $I(\nu)$ и добавление одного маркера на каждую дугу из $O(\nu)$. На рис. 4.7 при заданной маркировке возбуждены вершины ν_3, ν_4 и ν_6.

Поскольку модель маркированного графа имеет очевидную специфику по сравнению с "материнской" моделью сетей Петри, целесообразно дать новую интерпретацию понятий и свойств МГ, несколько отличающуюся от интерпретации аналогичных понятий и свойств для СП, рассмотренных в главе 2.

В маркированном графе $< V, E, \mu^0 >$ маркировка μ^* называется достижимой из μ^0, если существует последовательность срабатывания вершин $\nu_1, \nu_2, ..., \nu_n$ $(n > 0)$ такая, что ей соответствует последовательность маркировок $\mu^1, \mu^2, ..., \mu^n$, для которых справедливо $\mu^1 = \mu^0$, $\mu^n = \mu^*$ и переход $\mu^{i-1} \to \mu^i$ сопровождается срабатыванием вершины ν_i, $1 \le i \le n$.

Проблема достижимости для маркированных графов является алгоритмически разрешимой. Вершина ν в МГ называется потенциально срабатываемой при заданной маркировке μ, если эта вершина ν принадлежит хотя бы одной последовательности срабатываний. В маркированном графе на рис. 4.7 вершина ν_7 потенциально срабатываемая, так как она принадлежит последовательности срабатываний $\nu_6\nu_5\nu_7$, выполняемой при заданной начальной маркировке.

Исходная маркировка считается активной, если при ней все вершины МГ потенциально срабатываемые. Понятие активной маркировки имеет смысл

для МГ, граф G которого не является сильно связным. Практически это свойство относится к описанию неавтономных процессов. Вершина $\nu \in V$ в МГ называется живой, если для любой маркировки μ', достижимой из μ^0, существует такая маркировка μ'', при которой ν срабатывает. Маркированный граф называется живым, если все его вершины живые. Исследование условий живости МГ показали, что если в МГ каждая из вершин принадлежит какому-либо циклу (элементарному пути, или контуру, из ν_0 в ν_n, такому, что $\nu_0 = \nu_n$, а все остальные вершины попарно различны) и каждый цикл содержит, по крайней мере, один маркер на своих дугах, то граф является живым.

Наличие свойства живости, как и в сетях Петри, гарантирует отсутствие тупиковых ситуаций в моделируемом процессе. Для автономных процессов естественно потребовать наличие свойства живости.

Рис. 4.7 показывает пример живого МГ, поскольку все его циклы содержат не менее одного маркера. Действительно, список циклов этого графа содержит четыре контура: $\nu_1\nu_3\nu_2$, $\nu_4\nu_3\nu_7$, $\nu_4\nu_5\nu_7$ и $\nu_6\nu_5\nu_7$, причем дуги, включенные в эти циклы, соответственно содержат 1, 2, 2 и 2 маркеров.

Маркированный граф безопасен (на дуге не более одной точки) (k-ограничен), если для любой достижимой маркировки μ' и любой дуги $e \in E$ выполняется $\mu'(e) \le 1$ ($\mu'(e) \le k$).

Известно, что живой МГ является безопасным тогда и только тогда, когда каждая дуга графа содержит в цикле $\pi = e_1, e_2, ..., e_k$ таком, что $\mu(\pi) = \sum_{i=1}^{k} \mu(e_i) = 1$. **Циклы, обладающие таким свойством, называются синхронизирующими, или просто синхроциклами.**

Для маркированного графа, изображенного на рис. 4.7, лишь один цикл $\nu_1\nu_3\nu_2$ является синхроциклом.

Свойство безопасности (или, по крайней мере, k-ограниченности), так же, как и для сетей Петри, для МГ является чрезвычайно важным. Так, например, при моделировании обмена сообщениями оно позволяет отражать ограничения емкости буферной памяти в каналах связи. Маркированные графы, представляющие поведение логических схем, обязаны быть безопасными.

Ввиду того, что при срабатывании вершин маркированного графа число маркеров $\mu(\pi)$ в каждом цикле постоянно, понятие сихроцикла мож-

но обобщить на случай k-ограниченного МГ. Необходимым и достаточным условием k-ограниченности является $\mu(\pi) \le k$ для всех циклов. Маркированный граф на рис. 4.7 является небезопасным, но 2-ограниченным.

Понятия живости и безопасности маркированного графа очевидным образом распространяются на его маркировки. Известен следующий факт.

Для любого конечного сильно связного орграфа всегда существует живая и безопасная маркировка.

В процессе анализа описаний, заданных в форме ориентированных или маркированных графов, могут возникать следующие вопросы анализа:

- сильной связности орграфа,
- живости МГ,
- безопасности МГ.

Для получения ответов на эти вопросы могут применяться различные теоретико-графовые алгоритмы.[7]

Заметим, что фактом отсутствия живости может служить наличие в маркированном графе хотя бы одного цикла π, не имеющего маркеров, т.е. $\mu(\pi) = 0$. Поиск такого цикла можно осуществить в орграфе, полученном исключением из исходного маркированного графа всех маркированных дуг. Для нахождения цикла можно применить алгоритм поиска “в глубину” со сложностью $O(n + m)$, где m – число дуг в орграфе. Указанный способ анализа живости МГ “от противного” с точки зрения алгоритмической сложности, несомненно, более привлекателен, чем поиск полного множества циклов и проверка для каждого из них условия $\mu(\pi) = 0$. Этот способ может быть применен лишь тогда, когда заранее известно, что МГ является сильно связным, поскольку в противном случае, как показывает рис. 4.8, отсутствие безмаркерного цикла не гарантирует живости МГ, поскольку вершина ν_4 не принадлежит ни одному из циклов.

Для проверки безопасности МГ можно осуществить поиск множества синхроциклов и проверку принадлежности дуг хотя бы одному из них. Однако это требует применения какого-либо из известных алгоритмов вычис-

[7] В частности, для установления сильной связности одним из простых и наглядных методов является известный алгоритм Уоршелла нахождения транзитивного замыкания отношения смежности вершин. Сложность такого алгоритма имеет порядок $O(n^3)$, где n – число вершин графа.

ления полного множества циклов, имеющих экспоненциальную сложность. Например, возможно применение алгоритма нахождения определителя $\|R \cup I\|$, где R – матрица смежности, а I – единичная матрица, с последующим перебором его слагаемых со сложностью $O(n!(n-1))$; либо алгоритма поиска в глубину со сложностью $O(n^2 e)$, где $e = O(\sum_{i=2}^{n} C_n^i (i-1)!)$. Можно предложить и более эффективные алгоритмы проверки безопасности маркированных графов, имеющие полиномиальную сложность.

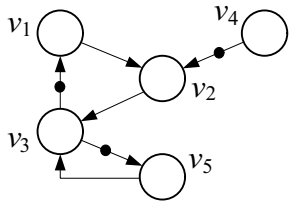


Рис. 4.8. Пример неживого маркированного графа

Существенной спецификой задач анализа маркированных графов является использование теоретико-графовых алгоритмов, не требующих построения диаграмм маркировок, порождаемых маркированными графами.

4.4. Сигнальные графы

Применение диаграмм переходов для анализа и синтеза самосинхронных схем (или программ управления) связано со значительными трудностями. Дело в том, что диаграмма переходов строится на полных состояниях проектируемого устройства, представленных векторами. Очевидно, что рост сложности устройства связан с увеличением длины вектора состояний, что, в свою очередь, приводит к так называемому “взрыву состояний”, т.е. к экспоненциальному росту числа состояний. Та же проблема возникает при использовании графа маркировок (состояний) сети Петри. По этой причине **алгоритмы, основанные на работе с состояниями, как правило, не эффективны.** С другой стороны в каждом состоянии диаграммы переходов может быть возбуждено лишь ограниченное число переменных. Возникает естественный вопрос: **можно ли построить такую диаграмму, в которой бы указывались только изменения возбужденных переменных (вместо самих состояний)**? Ответ на этот вопрос привел к появлению моделей сигнальных графов.

Для установления связи между моделями дискретных систем, задаваемыми различными подклассами сетей Петри и диаграммами переходов (и тем самым схемами Маллера), следует рассмотреть такую интерпретацию сетей Петри, при которой переходам сети Петри поставлены в соответствие изменения значений переменных, являющихся компонентами состояний описываемой системы. Пусть $X = \{x_1, x_2, ..., x_n\}$ – множество дискретных переменных. Каждая переменная x_i имеет свое конечное множество значений (состояний) $\{\sigma_1^i, ..., \sigma_{k_i}^i\}$, закодированных, скажем, неотрицательными целыми числами. Для каждой переменной x_i определим множество $D(x_i)$ допустимых изменений ее значений. Например, если x_i моделирует двоичный сигнал, то $D(x_i) = \{x_i^{0-1}, x_i^{1-0}\}$. Для краткости обозначим x_i^{0-1} через $+x_i$, а x_i^{1-0} – через $-x_i$. Если x_i – троичный сигнал, заданный на множестве {0, 1, 2}, где значения 0, 1 и 2 соответствуют, например, низкому, среднему и высокому уровням потенциала на линии, то

$$D(x_i) = \{x_i^{0-1}, x_i^{1-2}, x_i^{2-1}, x_i^{1-0}\}.$$

Если x_i моделирует пучок параллельных линий связи (шину), то два отдельных класса значений сигналов на шине – класс информационных кодов и класс спейсеров (промежуточных наборов) – могут быть заданы соответственно как 1 и 0.

Указанная выше интерпретация переходов сети Петри может быть формально представлена функцией разметки $\varepsilon : T \to D$, где T – множество переходов сети, а $D = \bigcup_{i=1}^{n} D(x_i)$. **Таким образом, тройка** $< PN, D, \varepsilon >$ **является интепретированной сетью Петри** *PN*, **которую будем называть в этом случае *сигнальной сетью Петри*.**

Иногда полезно использовать интерпретацию модели маркированного графа - *сигнальный граф*. Для его задания требуется ввести множество булевых переменных (сигналов), множество их изменений типа $+a$ и $-a$, которые означают соответственно переходы 0-1 и 1-0 сигналов, и функцию разметки, ставящую в соответствие вершинам маркированного графа изменения значений сигналов. Определим понятие сигнального графа (СГ) более формально.

Сигнальный граф — это маркированный граф $MG = <V, E, \mu^0>$, **на который "навешена" функция** ε **наименования (разметки) вершин,** $\varepsilon : T \to D$. **Таким образом, сигнальные графы оказываются подклассом сигнальных сетей Петри.**

Примеры сигнальной сети Петри и сигнального графа показаны на рис. 4.9.

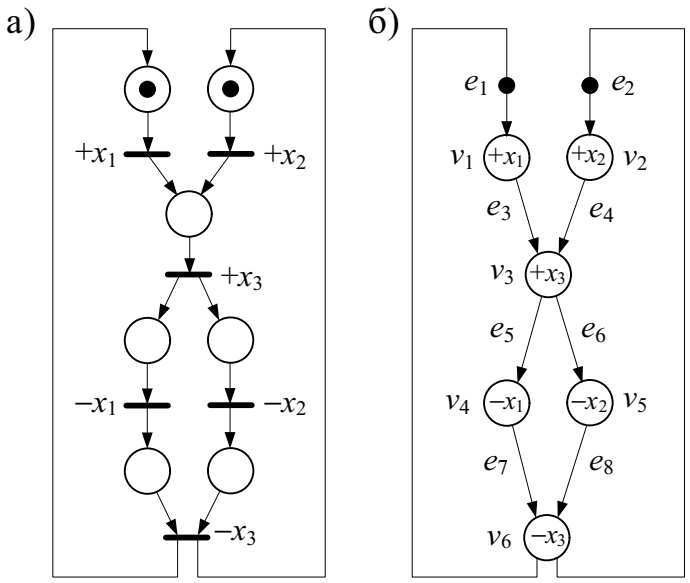


Рис. 4.9. Примеры а) сигнальной сети Петри и б) сигнального графа

Алгоритмы анализа сигнальных графов, заданных на автономных, живых и безопасных маркированных графах, имеют полиномиальную сложность порядка куба от числа вершин графа. Это заставляет считать соответствующий аппарат более предпочтительным по сравнению с анализом по диаграммам Маллера, имеющим экспоненциальную сложность. Цель такого анализа - установление наличия некоторых специфических свойств сигнальных графов, позволяющих использовать их как спецификацию для последующего синтеза самосинхронных схем.

Остановимся подробнее на изучении свойств сигнальных графов и той связи, которая существует между ними и диаграммами переходов.

Прежде всего, отметим, что все свойства, относящиеся к маркированным графам – связность, живость, безопасность, ограниченность – могут быть перенесены на сигнальные графы. Помимо этих общих свойств, сигнальные графы могут обладать определенными специфическими свойствами, зависящими от вида функции разметки вершин графа изменениями значений дискретных переменных.

Проницательный читатель, вероятно, заметит, что такая разметка позволяет сигнальному графу (разумеется, как и сигнальной сети Петри) порождать не только диаграмму маркировок, но и диаграмму переходов, в которой ситуации соответствуют маркировкам, но представляются наборами значений переменных, изменениями которых размечен сигнальный граф. Однако оказывается, что далеко не каждый сигнальный граф порождает диаграмму переходов в том виде, как она была определена в предыдущем разделе.

Построение кумулятивной диаграммы по маркированному графу

Для установления связи сигнальных графов с диаграммами переходов следует проанализировать связь маркированных графов с тем классом диаграмм маркировок, который они порождают в своем динамическом поведении. Пусть некоторый живой маркированный граф порождает диаграмму маркировок, в которой дуги могут быть помечены символами срабатывающих вершин. Для такой диаграммы можно построить относительно начальной маркировки μ^0 еще одну диаграмму, называемую *кумулятивной диаграммой* срабатывания вершин.

Вершинами этой новой диаграммы являются значения вектора чисел срабатывания вершин маркированного графа. Размерность вектора равна мощности $|V|$ множества V. Начальным значением этого вектора, соответствующим маркировке μ^0, является набор a^0, состояний из всех нулей. Для перехода $\mu^0 \xrightarrow{v_i} \mu^1$ построим вектор a^1, который отличается от a^0 только наличием единицы в i-й компоненте. Это означает, что вершина v_i сработала к этому моменту один раз. Двигаясь по цепочке срабатываний в диаграмме маркировок, легко отобразить встречающиеся маркировки в наборы $a^0, a^1, ..., a^r$. Значение каждой компоненты вектора равно числу срабатываний соответствующей вершины маркированного графа на данной цепочке.

Таким образом, очевидно, что кумулятивная диаграмма представляет собой “историю работы” маркированного графа. Например, для сигнального графа, изображенного на рис. 4.9,б, диаграмма маркировок и фрагмент соответствующей кумулятивной диаграммы приведены на рис. 4.10.

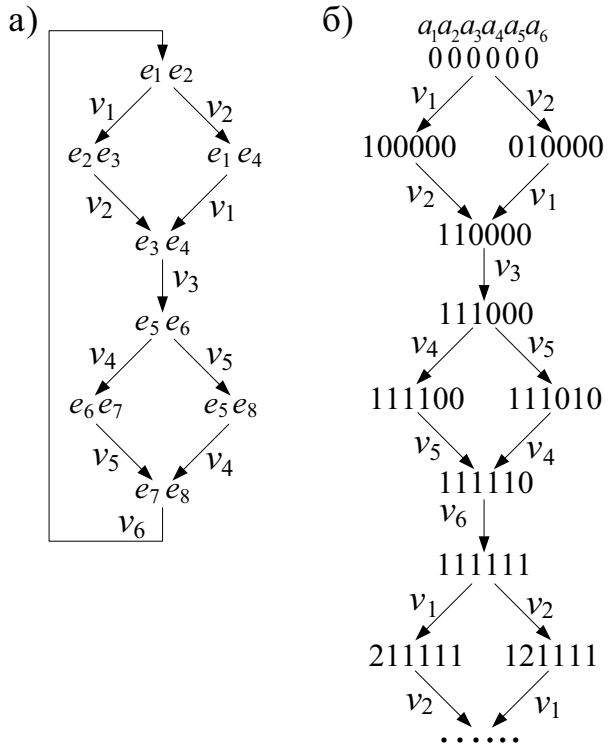


Рис. 4.10. а) Диаграмма маркировок для СГ рис. 4.9,б и б) ее кумулятивная диаграмма

Более строго кумулятивная диаграмма может быть определена как множество (бесконечное для живого маркированного графа) неотрицательных значений вектора размерности $|V|$ с заданным на этом множестве отношением частичного порядка: $a \le b$, если $a_j \le b_j$ для всех $j = 1, 2,..., |V|$. Такая диаграмма в теории частично упорядоченных множеств называется ***диаграммой Хассе***.

Легко убедиться, что подобная диаграмма может быть построена для любой сети Петри. Однако для маркированного графа можно доказать, что его кумулятивная диаграмма является ***дистрибутивной решеткой[8] с нулевым элементом***.

[8] Данные свойства кумулятивных диаграмм могут иметь интерес (и особый смысл) для читателя, знакомого с основами теории решеток (см., например, Гретцер Г., *Общая теория решеток*, М.,Мир, 1982, 452 с.)

Свойства кумулЯтивных диаграмм

Говоря о кумулятивных диаграммах сетей Петри, можно отметить их следующие два свойства:

- **кумулятивная диаграмма устойчивой сети Петри является полумодулярной решеткой с нулем;**
- **кумулятивная диаграмма устойчивой и безопасной сети Петри является дистрибутивной решеткой с нулем.**

Указанные свойства кумулятивных диаграмм естественно перенести на связанные с ними диаграммы маркировок. **Важной особенностью маркированного графа является свойство порождать дистрибутивную диаграмму маркировок.**

Связь сигнальных графов с диаграммами переходов

Вернемся вновь к сигнальным графам и рассмотрим свойства функции разметки. Ограничимся рассмотрением двоичных сигнальных графов, в которых каждая переменная x_i задана на множестве значений {0, 1}, т.е. $D(x_i) = \{+x_i, -x_i\}$. Такое ограничение не носит принципиального характера, и последующие свойства достаточно просто распространить на общий случай.

Очевидно, что если разметка, определяемая функцией ε, выполнена произвольно, то при функционировании некоторые маркировки будут приводить к одновременному возбуждению (и срабатыванию) вершин, помеченных одной и той же переменной. Для схем такая ситуация недопустима. Поэтому необходима формальная проверка исходного задания на наличие такого свойства у выполненной разметки.

Будем называть разметку ε *бесконфликтной*, если в любой достижимой маркировке не может быть возбуждено более одной вершины, соответствующей переменной x_i. Сигнальный граф, для которого задана бесконфликтная разметка, назовем *когерентным*.

Свойство когерентности, однако, не является достаточным для того, чтобы описание можно было считать корректным. Дело в том, что даже при упорядоченности изменений dx_i для каждой переменной x_i возможно отсутствие согласованности этих изменений. Иначе говоря, за $+x_i$ не должно следовать $+x_i$, т.е. должно быть выполнено условие корректности по переключениям переменных.

Условимся считать разметку ε знакосогласованной, если в любой допустимой относительно начальной маркировки последовательности изменений в СГ между любыми двумя изменениями $+x_i$ ($-x_i$) найдется изменение $-x_i$ ($+x_i$). СГ со знакосогласованной разметкой будем называть *регулярным*.

Итак, класс регулярных сигнальных графов является подклассом когерентных сигнальных графов.

Учитывая, что свойство регулярности обеспечивает корректность СГ по всем изменениям значений сигналов (корректность по переключениям), можно сформулировать утверждение о том, что **регулярный (двоичный) сигнальный граф порождает бинарную диаграмму переходов.**

Не будем приводить строгое описание алгоритмов перехода от регулярных сигнальных графов к диаграммам переходов и обратно, а лишь отметим, что при построении диаграммы переходов по сигнальному графу следует использовать обычную процедуру построения диаграммы маркировок, например, алгоритмом поиска в глубину, в котором каждой маркировке ставится в соответствие ситуация ДП. Возбуждение вершины вида dx_i, приводящее к переключению переменной x_i из состояния s_i^1 в состояние s_i^2 и соответствующее смене маркировок, отмечается в ДП звездочкой над значением s_i^1.

Легко заметить, что регулярный сигнальный граф может порождать диаграмму переходов с кратными ситуациями, т.е. такая ДП будет противоречивой. Содержательно противоречивость означает, что в описываемой системе существуют неэквивалентные ситуации (приводящие к различным историям процесса, поскольку они соответствуют различным маркировкам), помеченные одинаковыми наборами значений двоичных переменных из X.

Нормальность сигнального графа

Сигнальный граф называется *нормальным*, если для него не существует подмножества переменных $X' \subset X$ с полным циклом изменений своих состояний на некоторой последовательности маркировок при фиксированных значениях остальных переменных $X \setminus X'$. Иначе говоря, нормальный сигнальный граф порождает непротиворечивую ДП, которая изоморфна его диаграмме маркировок, и потому является дистрибутивной.

Противоречивость ДП свидетельствует о неполноте задания (“нехватке памяти”) в проектируемой системе, Устранение такого вида некорректности требует введения дополнительных сигналов (вершин) в сигнальный граф с

целью его приведения к нормальному виду. Существуют алгоритмы приведения СГ к нормальному виду, реализованные, например, в системе автоматизации проектирования асинхронных схем PETRIFY. Однако эти методы работают с представлениями ДП, а, следовательно, подразумевают оценки сложности экспоненциального характера относительно размера СГ.

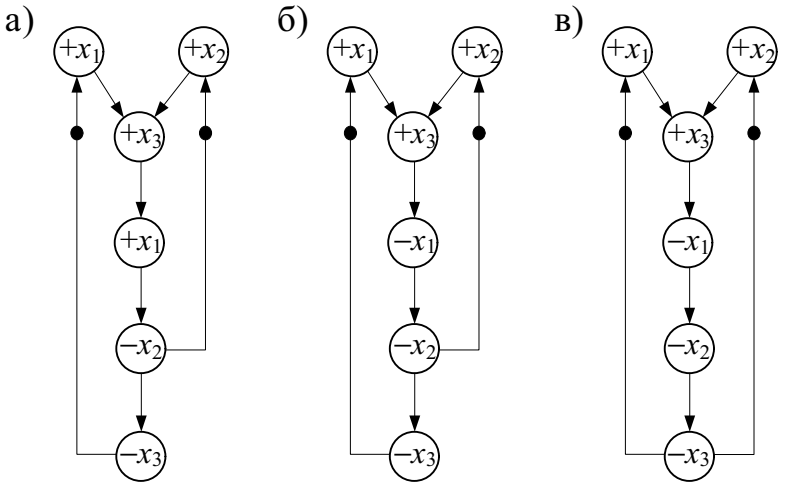


Рис. 4.11. Примеры сигнальных графов: а) когерентного, б) регулярного и в) нормального

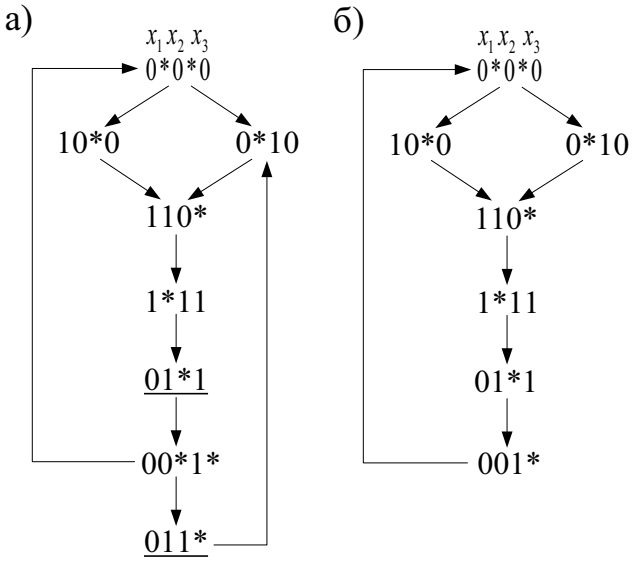


Рис. 4.12. а) Противоречивая диаграмма переходов для графа на рис. 4.11,б, б) непротиворечивая диаграмма переходов для графа на рис. 4.11,в

Примеры когерентного (но не регулярного), регулярного (но не нормального) и нормального СГ приведены на рис. 4.11.

Диаграммы переходов для сигнальных графов, изображенных на рис. 4.11,б и рис. 4.11,в показаны на рис. 4.12.

Понятия одновременности и предшествования сигналов

Анализ свойств регулярности и нормальности СГ удобно проводить, используя отношения *одновременности* и *предшествования* (или следования) на множестве изменений графа. Чтобы формализовать эти понятия применяется тот же методологический прием, что и при переходе от диаграммы маркировок к кумулятивной диаграмме, т.е. циклический объект заменяется бесконечным ациклическим (в данном случае называемым *разверткой* СГ).

Каждая реализация события циклического СГ порождает в развертке уникальную вершину с тем же именем, верхний индекс которого указывает на порядковый номер реализации. Правила порождения вершин развертки определяются механизмом смены маркировок. Множество вершин с i-ым индексом образует i-й *период развертки*. Пример развертки приведен на рис. 4.13.

Нетрудно понять, что i-я и j-я реализации события dx_l и j-я реализация события dx_k находятся в отношении предшествования $dx_l^i \Rightarrow dx_k^j$, если в развертке соответствующие им вершины соединены путем от dx_l^i к dx_k^j,

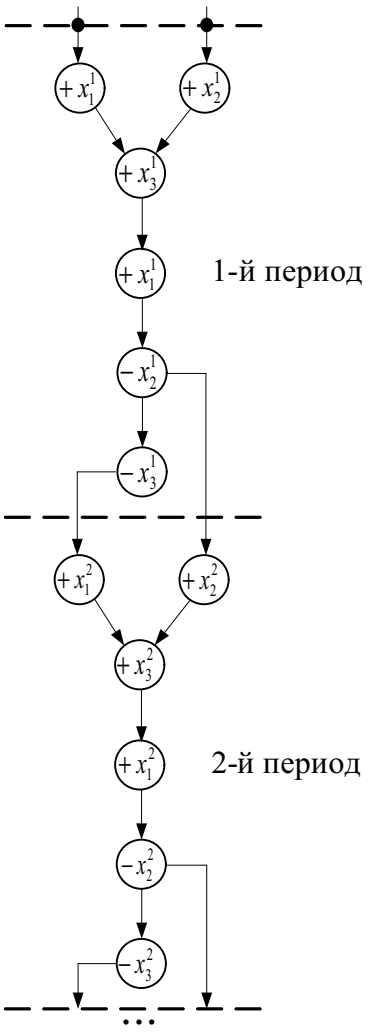


Рис. 4.13. Развертка сигнального графа, показанного на рис. 4.11,а

если же такой путь отсутствует ($\neg(dx_l^i \Rightarrow dx_k^j)\&\neg(dx_k^j \Rightarrow dx_l^i)$) то реализации dx_l^i и dx_k^j находятся в отношении одновременности, т.е. $dx_l^i \| dx_k^j$.

При рассмотрении циклических событий отношение одновременности между событиями dx_l и dx_k определяется как наличие одновременности между какими-либо их реализациями, а отношение предшествования между dx_l и dx_k – как выполнение условия $dx_l^i \Rightarrow dx_k^j \Rightarrow dx_l^{i+1}$ для любого i.

Можно показать, что вследствие повторяемости периодов развертки отношения $\Rightarrow$ и $\|$ для изменений СГ могут быть установлены по первым двум периодам его развертки. Алгоритмически определение отношений $\Rightarrow$ и $\|$ сводится к транзитивному замыканию графа, представляющего два первых периода развертки, и поэтому имеет сложность $O(n^3)$, где n – число вершин в исходном СГ.

Анализ сигнального графа на свойства регулЯрности и нормальности

Возвращаясь к вопросам анализа СГ, заметим, что в терминах введенных отношений бесконфликтность разметки ε означает для каждой переменной x_i отсутствие одновременности между всеми ее изменениями dx_i и $\bar{d}x_i$. Если разметка ε бесконфликтна, то все изменения dx_i переменной x_i *упорядочены* по отношению предшествования, поэтому каждому изменению dx_i можно присвоить индекс t так, что $dx_{i,t} \Rightarrow dx_{i,t+1}$. Тем самым анализ знакосогласованности ε сводится к проверке по каждой переменной x_i выполнения условия $dx_{i,t} \Rightarrow \bar{d}x_{i,t} \Rightarrow dx_{i,t+1}$.

При анализе на нормальность исходного СГ будем рассматривать переменные $x_{i,t}$ как разноименные (бесповторные). Это ограничение не является принципиальным и служит лишь для упрощения рассматриваемых конструкций.

Пусть задан регулярный СГ. **Переменные x_i и x_j из X будем называть:**

– ***непосредственно сильно связными*, если в сигнальном графе существует упорядочение вида**

$$dx_i \Rightarrow \ldots \Rightarrow dx_j \Rightarrow \ldots \Rightarrow \bar{d}x_i \Rightarrow \ldots \Rightarrow \bar{d}x_j$$

относительно заданной начальной маркировки, где dx_i обозначает $+x_i$ ($-x_i$), а $\bar{d}x_i$ – соответственно $-x_i$ ($+x_i$); отношение непосредственной сильной связности будем обозначать через CON и писать $(x_i, x_j) \in CON$;

– ***сильно связными*, если $(x_i, x_j) \in CON^*$, где CON^* – транзитивное, рефлексивное, и симметричное замыкание отношения CON;**

Прокомментируем вышеприведенные определения.

Непосредственная сильная связность двух переменных имеет место, если их переключения взаимно чередуются. Например, $+x_1,...,+x_2,...,-x_1,...,-x_2,...$. Легко понять, что при этом, одна из переменных не может переключиться из 0 в 1 и обратно из 1 в 0, без хотя бы одного переключения своей напарницы – другой переменной.

Сильная связность подразумевает, что переменные либо чередуются в своих переключениях непосредственно, либо через посредника, третью переменную, с которой заданные переменные уже сильно связаны.

Будучи отношением эквивалентности (транзитивное, рефлексивное и симметричное замыкание), отношение сильной связности CON^* задает разбиение всего множества переменных X на непересекающиеся классы сильной связности, т.е. $C_1 \cup C_2 \cup ... \cup C_r = X$.

Достаточно легко показать, что для переменных, принадлежащих одному классу сильной связанности C_i, справедливо следующее утверждение. **Срабатывание какого-либо подмножества $Y \subset C_i$ переменных по полному циклу не может произойти без хотя бы одного переключения переменной из его дополнения до всего класса C_i.**

Из этого утверждения тривиально следует следующее очень важное утверждение. Если все переменные регулярного (двоичного) сигнального графа принадлежат одному классу сильной связности, то сигнальный граф является нормальным.

Интересно, что обратное утверждение о том, что из нормальности следует принадлежность всех переменных одному классу сильной связности, не справедливо. Соответствующий контрпример показан на рис. 4.14.

Здесь переменная x_1 образует отдельный класс сильной связности от всех остальных переменных. Тем не менее, сигнальный граф нормален, о чем свидетельствует непротиворечивая ДП, показанная на рис. 4.15 .

Для того, чтобы “приблизить” достаточное условие нормальности к его необходимости, можно ввести еще более тонкую классификацию отношения связности переменных, в которой в вышеуказанном примере переменная x_1 окажется связанной с остальными переменными.

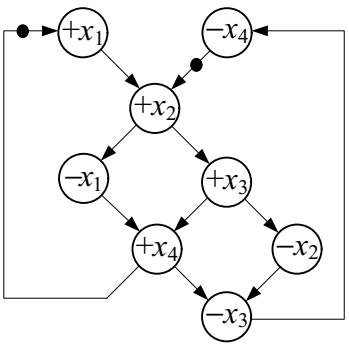


Рис. 4.14. Пример сигнального графа

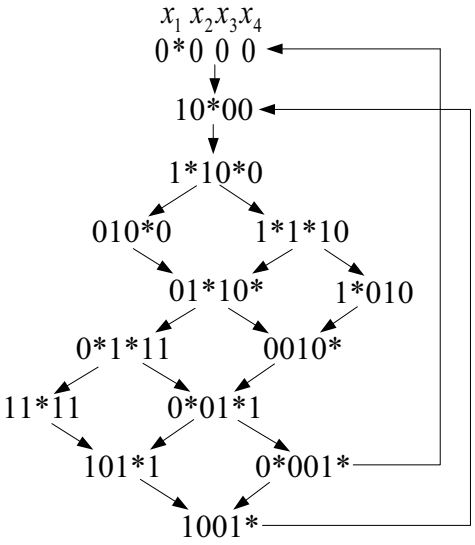


Рис. 4.15. Диаграмма переходов, порождаемая сигнальным графом, изображенном на рис. 4.14

Такое обобщение отношения связности требует нетривиального определения связности, основанного на рекурсии, и мы его не будем приводить здесь (оставив его как упражнение для читателя с повышенным интересом к математическим конструкциям). Неформально оно сводится к следующему

алгоритму вычисления отношения связности.

Сначала вычисляется отношение сильной связности (используя вышеуказанные определения и методы вычисления транзитивного замыкания отношений). Оно разбивает исходное множество сигналов X на начальные клас-

сы связности (сильная связность). Далее, итерационно, делается следующее. Для каждой пары классов проверяется условие наличия «чередующегося» упорядочения:

$$dx_i \Rightarrow ... \Rightarrow dx_j \Rightarrow ... \Rightarrow dx_k \Rightarrow ... \Rightarrow dx_l$$

где x_i, x_k принадлежат классу C_1, а x_j, x_l - классу C_2. Если это условие выполнено, то классы C_1 и C_2 могут быть объединены в один класс связности. И так далее. Справедливость правомочности объединения основана на факте чередования изменения состояния переменных из разных классов. А именно, легко понять, что при полном цикле переключений переменных из класса C_1 должно произойти хотя бы одно переключение переменной из класса C_2.

Вышеприведенный алгоритм находит разбиение на максимальные классы связности. Таким образом, он ведет к обобщенному отношению связности, частным случаем которого является сильная связность.

Итак, можно утверждать, что **регулярный (двоичный) сигнальный граф является нормальным, если все его переменные принадлежат одному классу связности.**

Вернувшись к контрпримеру на рис. 4.14, заметим, что классы C_1 и C_2 могут быть объединены, так как существует упорядочение вида:

$$+x_1 \Rightarrow ... \Rightarrow +x_2 \Rightarrow ... \Rightarrow -x_1 \Rightarrow ... \Rightarrow +x_4$$

Таким образом, все переменные этого сигнального графа связаны, и, следовательно, он нормален.

Нетрудно понять, что каждый итеративный шаг построения отношения связности имеет сложность, не превышающую $O(n^3)$, где n – число вершин исходного сигнального графа. Это так, поскольку сложность вычисления отношений упорядочения в развертке сигнального графа имеет оценку $O(n^3)$, и отношения упорядочения могут быть сохранены в соответствующей таблице для работы с отношениями связности. Поэтому дальнейшие вычисления и проверки связаны с парными отношениями на множестве переменных и построением классов связности итерационным путем. Поскольку число итераций объединения классов связности не превышает n, общая сложность анализа на нормальность не превышает $O(n^4)$.

В заключение хотелось бы отметить, что **метод проверки на нормальность на основе отношений связности, хотя и имеет несомненную привлекательность, поскольку не требует построения диаграммы переходов, тем не менее, так и не исследован до конца, как в теории, так и на практике.** А именно, остаются открытыми некоторые проблемы. Например, не найдено такого отношения связности переменных, для которого удалось бы доказать утверждение о его необходимости (а не достаточности) для нормальности СГ. Кроме того, не существует удобного программного средства проверки на связность для достаточно широкого класса сетей Петри с сигнальной интерпретацией. Напомним, что вышеприведенное определение сигнального графа было ограничено маркированным графом, а далее рассматривались только бесповторные СГ.

В настоящее время, единственным практическим средством проверки нормальности СГ (или непротиворечивости для ДП), а также приведения СГ к нормальному виду, остается система PETRIFY, в которой реализован другой подход, связанный с переходом от СГ к ДП и обратно. Тем не менее, на практике, при использовании системы PETRIFY для проектирования самосинхронных схем управления, часто целесообразно делать преобразования спецификации на СГ с использованием подхода связности “на интуитивном уровне”.

4.5. Упражнения

1. Для диаграммы переходов, изображенной на рис. 4.1, получите систему булевых уравнений модели Маллера.

2. Для полумодулярной диаграммы переходов может быть построена соответствующая ей сигнальная сеть Петри (устойчивая и, возможно, небезопасная). Построить такую сеть Петри для ДП, показанной на рис. 4.4, а. Указать, в какой из позиций нарушается условие безопасности.

3. Построить диаграмму переходов для модели Маллера, заданной системой:

$$\begin{cases} x_1 = \overline{x_2 \vee x_3} \\ x_2 = x_1 \\ x_3 = x_1 \end{cases},$$

используя аппарат характеристических функций.

4. Проверить условия полумодулярности схемы, заданной в упр. 3.

5. Известно, что сеть Петри свободного выбора может быть представлена как совокупность маркированных графов (теорема Хака), сохраняющих порядок срабатываний переходов и имеющих общие позиции. Нарисовать пример сети Петри свободного выбора и предложить вариант ее разбиения на подсети, являющиеся маркированными графами.

6. Почему наличие нулевого столбца в матрице смежности МГ свидетельствует об отсутствии его живости?

7. Почему наличие нулевой строки в матрице смежности живого МГ свидетельствует о его небезопасности (не k-ограниченности)?

8. Предложить полиномиальный алгоритм проверки безопасности живого МГ. (Подсказка: перебор вести по маркированным ребрам).

9. Предложить полиномиальный алгоритм поиска живой и безопасной маркировки для сильно-связного орграфа. (Существование такой маркировки подтверждается теоремой.)

10. Для маркированного графа, показанного на рис. 4.7, проверить живость и ограниченность матричными методами.

11. Приведите примеры сигнальных графов, заданных на целочисленных множествах значений дискретных переменных. Выделите среди них графы с бесконфликтной и знакосогласованной разметкой.

12. Проанализируйте на нормальность сигнальный граф, показанный на рис. 4.16.

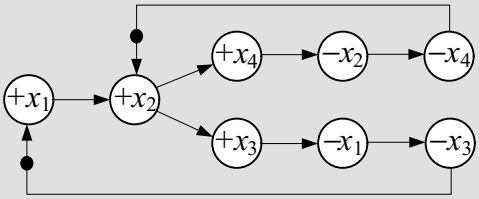
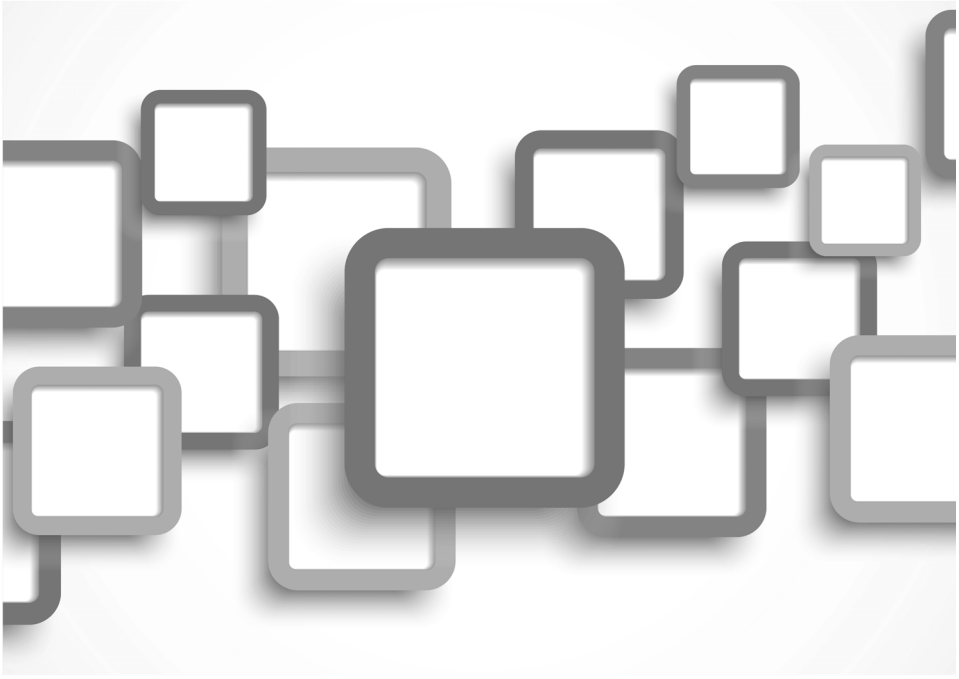


Рис. 4.16. Пример сигнального графа

Путем введения дополнительных переменных попытайтесь нормализовать этот граф.

13. Предложите пути обобщения метода вычисления отношений связности на множестве двоичных переменных $X = \{x_1, x_2, ..., x_n\}$, среди которых могут быть *небесповторные* переменные (такие переменные, для которых период развертки содержит более одной пары переключений вида $+x_i$ и $-x_i$).

14. Предложите конкретные варианты алгоритмов проверки когерентности, регулярности и нормальности заданного сигнального графа на основе метода анализа периодов его развертки.



Глава 5.

ПАРАЛЛЕЛЬНОЕ ПРОГРАММИРОВАНИЕ. ЗАДАЧИ КООРДИНАЦИИ И СИНХРОНИЗАЦИИ ВЗАИМОДЕЙСТВУЮЩИХ ПРОЦЕССОВ

Системный программист – это тот, который всех остальных программистов называет пользователями.
(Из программистского фольклора)

Бурное развитие микроэлектроники продолжает следовать по пути наращивания числа переключательных элементов на единицу площади кристалла согласно закону Гордона Мура. В современной формулировке закон Мура гласит, что каждые два года количество транзисторов, размещаемых на кристалле интегральной схемы, удваивается. Такое развитие, казалось бы, должно приводить к удваиванию производительности процессоров и систем в целом. Однако, с недавних пор, на пути прямого наращивания производительности возникли проблемы, связанные с неукротимым ростом затрат энергии от источника питания (как правило, батарейки) и потерями в резистивных цепях (в виде тепла), что требует сложных методов отвода тепловой энергии от кристаллов интегральных схем. С точки зрения обеспечения вычислительных устройств питанием эти проблемы, прежде всего, обусловлены отставанием в темпах роста эффективности батареек от темпов (потенциального) роста производительности микросхем. Другим тормозом роста производительности является проблема передачи информации в пределах кристалла, где задержки линий передачи растут несмотря на ускорение работы транзисторов и вентилей. Все эти аспекты затормаживания действия закона Мура приводят к тому, что **у создателей вычислительных систем нет другой альтернативы, как переходить от масштабирования роста систем по числу транзисторов к масштабированию по числу процессорных ядер**. Таким образом, **системы на кристалле начинают себя все более вести как распределенные системы, для которых более адекватными являются модели параллельного поведения.**

Кроме того, такой подход оправдывается и другой фундаментальной проблемой в росте сложности микросхем, а именно **проблемой повышения эффективности проектирования систем на кристалле или систем в упаковке** (например, с использованием 3D-технологий, т.е. стека из кристаллов). Эта проблема находит свое решение в активном использовании принципа “повторного использования” (re-use), согласно которому инженеры избегают перепроектирования новых процессорных решений с нуля. Вместо этого они переносят “оптом” уже известные реализации процессоров (так на-

зываемые IP-ядра) в новые системные архитектуры, все более становящиеся сложными, мультиядерными вычислительными системами. Возникает, со всей неизбежностью, проблема активного "озадачивания" массы процессоров, зачастую разных по своим функциональным возможностям (т.е. гетерогенных), путем оснащения их соответствующими программными средствами. В результате во многих применениях, например, задачах реального времени, уровень параллельности процессов становится чрезвычайно высоким. **В современных условиях парадигма программирования массы процессоров становится все более сродни парадигме "программирования" массы логических вентилей в самосинхронных схемах (см. предыдущую главу о динамической логике)**.

В вышеописанных условиях проектирования задача программирования систем выходит за пределы одного процессора или даже небольшой группы процессоров, где уровень параллельности и независимости протекания процессов во времени является умеренным и может охватываться программистом непосредственно в терминах существующих языков и конструкций программирования. Таким образом, ставится "под сомнение возникшее чувство удовлетворения преемственностью архитектуры машин прошлого и настоящего", что было отмечено Г. Майерсом еще пару десятков лет назад. Сейчас же это замечание Майерса переходит из разряда наблюдений в разряд требований, предъявляемых к подходам к программированию вообще.

Это неудовлетворение программными подходами и острая нехватка в понимании того, как же следует программировать многоядерные вычислительные системы, ставит ряд проблем, связанных, прежде всего, с теорией программирования, особенно *параллельного программирования*. Из этих проблем **здесь будут затронуты только те, которые связаны с типичными задачами и механизмами синхронизации (координации) взаимодействующих процессов**. В отличие от традиционного программирования последовательных систем, в системах с параллельными процессами (например, multi-thread programs) именно эта проблематика находится в центре сферы недостатка понимания основных принципов и модельного представления параллельных программ.

В этой главе авторами преследуется цель помочь читателю увидеть, как при помощи сетей Петри можно элегантно и компактно представить задачи координации взаимодействующих процессов в параллельных программах на чисто модельном уровне. Причем в условиях максимального

использования аппарата ординарных сетей Петри (избегая, где возможно расширений сетей Петри, приводящих к неразрешимым проблемам анализа) станут понятными пути эффективной проверки самих механизмов координации, а следовательно, и способы проектирования корректных параллельных программ.

В главе 2 (разд. 2.8) было дано определение параллельной программы посредством соответствующей интерпретации сетей Петри. В этом определении параллельная программа представляет собой спецификацию асинхронного процесса, который порождает глобальное описание функционирования программы в динамике. Поэтому в нем непосредственно не выделяются компонентные процессы (подпроцессы), которые функционируют, взаимодействуя друг с другом, потому что сама модель АП не содержит средств декомпозиции. Между тем, вполне разумен альтернативный **подход, связанный с представлением асинхронного процесса как композиции взаимодействующих процессов.**

Этот подход иллюстрируется рис. 5.1, в котором P_1 и P_2 – это процессы, взаимодействующие друг с другом через третий, *согласующий* процесс P_c. Роль процесса P_c состоит в координации (или иначе, синхронизации) событий, реализуемых подпроцессами P_1 и P_2. В параллельном программировании эти подпроцессы принято называть процессами, причем обычно полагают, что они являются по своей природе *последовательными* (циклическими), т.е. осуществляют последовательную парадигму вычислений.

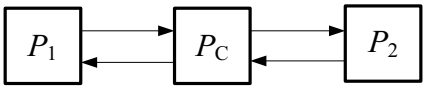


Рис. 5.1. Взаимодействие процессов P_1 и P_2 через согласующий процесс P_C

Разночтения в понимании понятия “процесс” не должны смущать. По существу совместное функционирование тройки P_1, P_2 и P_c порождает некоторый общий асинхронный процесс, реализующий обмен информацией между процессами P_1 и P_2, организованный по задаваемым правилам синхронизации. Эти правила могут выражаться с помощью вербальных конструкций типа “процесс P_1 должен быть выполнен раньше процесса P_2” или “процесс P_1 имеет приоритет над P_2” и соответствующих механизмов синхронизации. Часто на согласующий процесс P_c накладывается также требование *распределения разделяемых ресурсов* между процессами.

В параллельном программировании ставится задача синхронизации, т.е. задача координации совместного выполнения взаимодействующих процессов. Число различных задач синхронизации не ограничено, но имеется некоторый набор типовых (классических) задач, в которых используется набор механизмов (средств) синхронизации, достаточный для решения многих реально встречающихся задач. **К классическим относятся такие задачи, как задача об обедающих философах, задачи "производитель – потребитель", "читатели – писатели" и др.** Они будут рассмотрены в этой главе, причем соответствующие механизмы синхронизации будут описываться по мере изложения задач. В отличие от многих работ, в которых изучение возникающих при синхронизации эффектов иллюстрируются чисто программными нотациями, здесь предпринимается попытка использования сетевых представлений программ и внимание акцентируется на вопросах организации управления. Такой подход имеет преимущества перед традиционным, потому что он не зависит от языков программирования и более естественен в контексте данной книги.

5.1. Аномалии динамики

Типовым представлением синхронизируемого процесса в классической задаче параллельного программирования является фрагмент сети Петри, показанный на рис. 5.2.

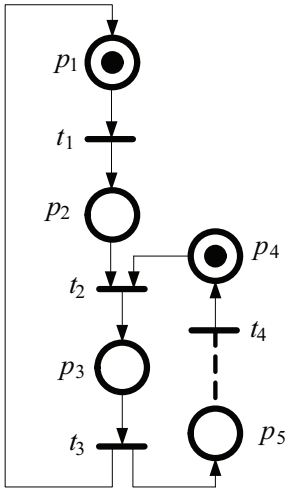


Рис. 5.2. Типичное представление синхронизируемого процесса

Интерпретация событий, соответствующих срабатываниям переходов, такова:

t_1 – запрос на решение задачи,

t_2 – начало выполнения задачи,

t_3 – конец выполнения задачи.

Интерпретация условий, соответствующих пребыванию маркера в позициях:

p_1 – пассивная фаза задачи,

p_2 – фаза ожидания,

p_3 – активная фаза задачи.

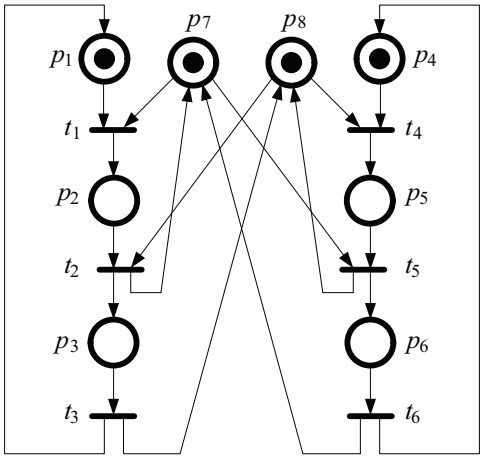


Рис. 5.3. Пример СП, моделирующий динамические аномалии

Цикл, включающий позиции p_4, p_5 и переход t_4, обычно называют *синхронизационным*. Его назначение – репозиция процесса. Синхронизационный цикл обязательно содержит хотя бы один маркер, моделирующий некоторые ресурсные ограничения для решаемой задачи, и появление маркера в позиции p_4 переводит процесс из фазы ожидания в активную фазу, выход из которой в пассивную фазу после окончания решения сопровождается сообщением о том, что задача решена (появление маркера в позиции p_5 – начало синхронизационного цикла). Сеть Петри на рис. 5.2 является безопас-

ной. Однако условие безопасности в общем случае не является гарантией отсутствия в моделирующей сети Петри системы *аномальных эффектов*. В качестве примера рассмотрим сеть Петри, представленную на рис. 5.3.

На этом рисунке изображены два подпроцесса, похожих на процесс, показанный на рис.5.2. Первый подпроцесс образован позициями p_1, p_2, p_3 и переходами t_1, t_2, t_3, а второй – позициями p_4, p_5, p_6 и переходами t_4, t_5, t_6. Два возобновляемых ресурса моделируются позициями p_7 и p_8. Сеть безопасна при заданной начальной маркировке. Однако нет необходимости строить всю диаграмму маркировок, чтобы убедиться в том, что при одновременном срабатывании переходов t_1 и t_4 осуществляется переход из начальной маркировки к маркировке p_2, p_5 (остальные позиции пусты), при которой переходы t_2 и t_5 не возбуждены и, следовательно, сработать не могут. Эта маркировка является тупиковой, следовательно, моделируемая сетью Петри, представленной на рис. 5.3, система входит в состояние, которое обычно называют ***дедлоком*** (*взаимоблокировкой*, *клинчем*, *отталкиванием*, *смертельным объятием* и др.). Иначе говоря, в поведении системы наблюдается нежелательная аномалия.

Следует отметить, что эти эффекты связаны с нарушением живости сети Петри в том смысле, что если во множестве достижимых ситуаций описываемого процесса имеет место ситуация–дедлок, то этот процесс не может быть живым.

Поведенческий и структурный дедлоки

Приведенный выше пример демонстрирует поведение, при котором попадание системы в дедлок не является в полном смысле неизбежным, поскольку поведение зависит от сочетания задержек в выполнении отдельных действий. Действительно, при благоприятном сочетании задержек переходов сеть Петри может функционировать внешне нормально, например, при чередовании срабатываний переходов одного и другого процессов $t_1t_2t_3t_4t_5t_6\ldots$.

Интересно, что понятие дедлока иногда носит не поведенческий характер, как было показано выше, при возникновении тупиковой маркировки в диаграмме маркировок СП, а структурный характер. Пример ***структурного дедлока***, определение которого было дано в гл. 2, показан на рис. 5.4. В этом примере дедлок образован позициями p_1, p_2, p_3.

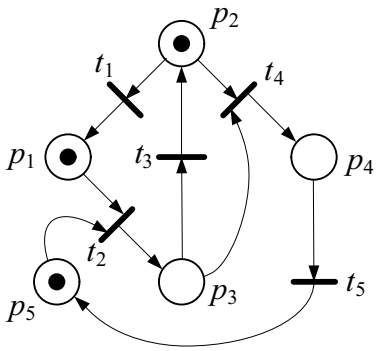


Рис. 5.4. Пример сети Петри, содержащей дедлок

Обратим внимание на то, что понятия поведенческого дедлока и структурного дедлока связаны через понятие живости сети Петри. Приведенный пример показывает, что система, имеющая структурный дедлок, в процессе своей работы может попасть в тупиковое состояние (маркировки p_1, p_3 или p_5). **В общем случае наличие структурного дедлока не является достаточным условием для существования дедлока–маркировки**. Действительно, понятие тупиковой маркировки носит характер глобальной утраты живости в системе, в то время как структурный дедлок означает возможность лишь частичной потери ее "трудоспособности". Пример соответствующей аномалии с "отмиранием" подсистемы иллюстрируется рис. 5.5.

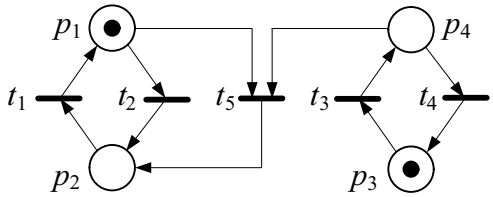


Рис. 5.5. Пример эффекта каннибализма

Оба процесса, один из которых образован позициями p_1, p_2 и переходами t_1, t_2, а второй – позициями p_3, p_4 и переходами t_3, t_4, взаимодействуют через переход t_5. Переходы t_2, t_5 и t_4, t_5 при маркировке p_1, p_4 находятся в состоянии конфликта. Если конфликт разрешается в пользу переходов t_2 и t_4, то процессы циклически функционируют, не мешая друг другу. Если же первым срабатывает переход t_5, то в дальнейшем продолжает работать только пер-

вый процесс, а второй навечно остается "мертвым". Таким образом, система является *несправедливой* по отношению ко второму процессу.

Эффект "каннибализма"

При неблагоприятной ситуации один процесс (имеющий больший приоритет) продолжает "жить за счет другого". Можно привести биологическую аналогию. Первый и второй процессы моделируют жизнедеятельность самки и самца богомолов, после совокупления которых (переход t_5 на рис. 5.5) самка убивает самца. Поэтому эффект, соответствующий данному типу аномалии, можно назвать эффектом каннибализма.

Эффект "голодной смерти"

Несколько иными является **эффект так называемой "голодной смерти", или, чуть мягче, "продолжительного голодания", когда из-за наличия конфликта в системе один из процессов или переходов может простаивать при неблагоприятном сочетании задержек, хотя и оставаться потенциально живым.** Рассмотрим пример, иллюстрируемый СП на рис. 5.6, в котором имеет место эффект "голодной смерти" на глобальном уровне, т.е. на уровне процессов. Здесь пара процессов пользуется общим ресурсом, наличие которого моделируется маркером в позиции p_1. Представим себе, что первый процесс (левый цикл) обычно более "расторопен" при выполнении захвата ресурса, т.е. переход t_1 срабатывает быстрее, чем переход t_2. При такой организации системы один из процессов может при желании постоянно владеть ресурсом, обрекая второй процесс на "голодание".

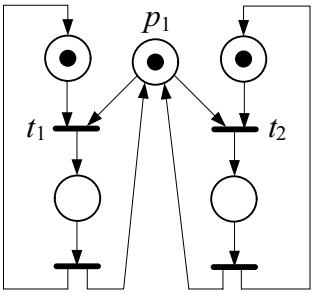


Рис. 5.6. Иллюстрация возможности "голодной смерти" одного из процессов

Рассмотрим еще один пример, демонстрирующий средствами сетей Петри эффект "голодания". Обратимся к рис. 5.7.

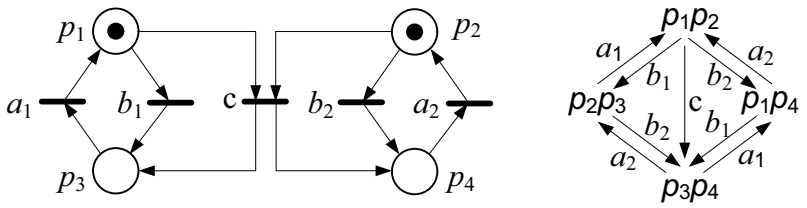


Рис. 5.7. Иллюстрация эффекта "голодания" в системе на локальном уровне

Данный пример иллюстрирует специфику свойства типа "голодная смерть" на уровне локальных отношений между элементами процессов в отличие от ранее рассмотренного глобального отношения между процессами. Локальный характер такого свойства также проявляется при наличии конфликтной ситуации в процессе. В данном примере каждый из двух взаимодействующих через переход *c* процессов в своем независимом функционировании имеет конфликт между переходами b_i и *c*, связанный с тем, что циклическая работа процесса потенциально может происходить бесконечно с выполнением какого-либо одного перехода: либо b_i, либо *c*. Аналогичный конфликт сохраняется и при совместной работе процессов, однако этот конфликт приобретает специфическую окраску.

Во-первых, в системе из пары процессов каждый из них может работать независимо друг от друга через свои переходы b_1 и b_2. Такое движение системы по частному циклу называется неэффективной работой.

Во-вторых, локальные конфликты между b_i и *c* начинают носить ориентированный характер. Имеется в виду то, что из возбуждения *c* всегда следует возбуждение b_i, но из возбуждения какого-либо b_i еще не следует возбуждения *c*. В этом смысле единственной причиной "голода" b_i является "несправедливое отношение" к b_i со стороны *c*; если же от "голода" страдает *c*, то ввиду возможной рассинхронизации процессов (возможность маркировок p_2, p_3 и p_1, p_4), в этом нельзя винить ни b_1, ни b_2.

Свойства "справедливости" и "голодания"

Современные исследования в области семантики параллелизма породили весьма обширную "картотеку" различных видов свойств "справедливости" и "голодания" в организации процессов. Насколько то или иное свойство является аномалией, зависит от предметной семантики, т.е. от той содержательной интерпретации, которая "навешена" на переходы и позиции сете-

вой модели. Обратимся в связи с этим к набору задач, называемых задачами согласования процессов, анализ которых попутно познакомит с рядом распространенных механизмов синхронизации.

5.2. Задача об обедающих философах

Это одна из классических задач параллельного программирования, которая демонстрирует многие основные эффекты взаимодействия процессов.

Пять философов проводят время на вилле, созерцая природу и размышляя по поводу устройства мира, и время от времени, проголодавшись, заходят в столовую, в которой за круглым столом расставлены пять стульев. Каждое место закреплено за определенным философом. Против каждого места на сервированном столе поставлена тарелка, а между тарелками лежат вилки (их тоже пять). В центре стола лежит блюдо со спагетти (рис. 5.8).

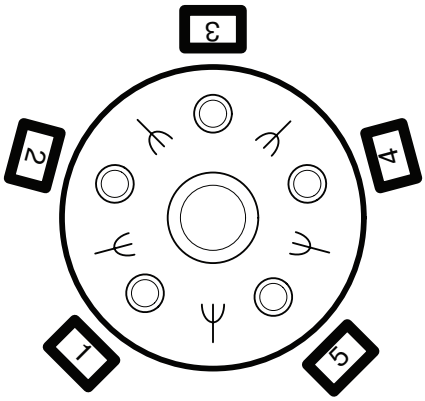


Рис. 5.8. Сцена для задачи об обедающих философах (вид на столовую в плане)

Проголодавшийся философ занимает свое место и по правилам этикета берет вилку слева от себя. Но спагетти в блюде нельзя подцепить одной вилкой, и если вилка справа от философа свободна, он завладевает ею, подцепляет обеими вилками спагетти, переносит его в свою тарелку и начинает трапезу. Закончив ее, философ кладет вилки на свои места и выходит из столовой. Собственно задача состоит в том, чтобы проанализировать возможные неприятные эффекты, возникающие в процессе "кормления" философов.

Для выявления таких эффектов воспользуемся интерпретацией данной задачи на языке сетей Петри.

Процесс, специфицирующий действия *i*-го философа, показан на рис. 5.9.

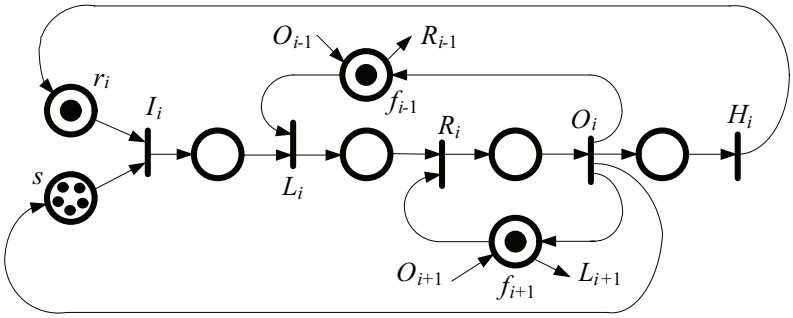


Рис. 5.9. Фрагмент спецификации задачи об обедающих философах

Семантика событий, соответствующих срабатыванию переходов этой СП, такова:

I_i – *i*-й философ вошел в столовую,

L_i – *i*-й философ взял левую от себя вилку,

R_i – *i*-й философ взял правую от себя вилку,

O_i – *i*-й философ вышел из столовой,

H_i – *i*-й философ проголодался, $1 \le i \le 5$.

Семантика некоторых условий, соответствующих наличию маркеров в позициях:

s – наличие спагетти в блюде,

r_i – *i*-й философ направляется в столовую,

f_{i-1} – наличие вилки слева от *i*-го философа,

f_{i+1} – наличие вилки справа от *i*-го философа (сложение и вычитание осуществляется по модулю 4). Позиции f_{i-1} и f_{i+1} являются позициями свободного выбора.

Склеивание пяти фрагментов, представленных на рис. 5.9, дает полную спецификацию задачи об обедающих философах. Диаграмма маркировок, построенная по полной спецификации, содержит исчерпывающую информацию для анализа возникающих эффектов. В данном случае диаграмма весьма громоздка, и нет смысла приводить ее здесь. Но некоторые нежелательные эффекты могут быть выявлены и из анализа приведенного фрагмента на живость для различных вариантов маркировки.

Если позиции f_{i-1} или f_{i+1} пусты (не содержат маркеров) то переходы L_i или R_i и следующие за ними переходы вплоть до H_i неживые. Понятно, что если все пять философов входят в столовую и одновременно берут по левой от себя вилке, то никто из них не сможет начать есть спагетти. Происходит типичная блокировка в системе (эффект голодания или "голодной смерти"). Возможен заговор против *i*-го философа двух его соседей: видя, что он направляется к столу, они опережают его и поочередно забирают вилку то слава, то справа от него, не давая ему поесть. Эффект блокировки сохраняется и при уменьшении числа маркеров в позиции *s*. Только в случае, когда сохраняется безопасность этой позиции, т.е. в каждый данный момент в столовой находится один философ, от этого нежелательного эффекта можно избавиться. Но любое более полное использование ресурсов столовой может привести к блокировке, так что при данной постановке этой задачи не существует ее корректного решения.

5.3. Задачи взаимного исключения и семафоры

Суть задачи взаимного исключения – создание такого механизма управления программой, который обеспечивает условие, что в любой момент времени только один процесс может иметь доступ к разделяемому ресурсу (например, к памяти).

Постановка этой задачи предусматривает, что каждый процесс (программа) содержит некоторую область (критический интервал), работающую с разделяемым ресурсом. От ее решения (нахождения механизма синхронизации) требуется следующее:

а) в каждый момент времени лишь один процесс может находиться внутри критического интервала,

б) обеспечивается справедливость по отношению к каждому из процессов,

в) не допускаются поведенческие аномалии,

г) остановка любого процесса вне критического интервала не должна блокировать другие процессы.

Семафоры Дейкстры

Решение задачи взаимного исключения при сетевой интерпретации практически не встречает никаких затруднений. Оно приведено на рис. 5.10.

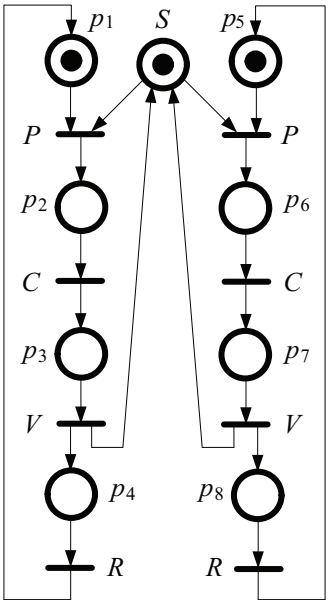


Рис. 5.10. Реализация задачи взаимного исключения с использованием семафорных синхропримитивов

Прежде, чем прокомментировать это решение, введем понятия семафоров и операций над ними, предложенные Э. Дейкстрой.

Семафором называют неотрицательную целочисленную переменную. Операции над семафором *S* определяются следующим образом.

1. Операция *V(S)* – переменная *S* увеличивается на единицу (инкремент) одним неделимым действием. Во время ее выполнения в одном подпроцессе к *S* невозможен доступ из других подпроцессов.

2. Операция *P(S)* – переменная *S* уменьшается на единицу (декремент), если $S \geq 1$. Если $S = 0$, то подпроцесс, инициирующий *P*-операцию, ждет, пока будет возможно ее выполнение. Эта операция также является неделимой.

Обозначения этих операций происходят от датских слов *passeren* (контролер) и *vrijgeven* (турникет). Читатель, видимо, уже обратил внимание, что понятие *P*-операции эквивалентно понятию "условное вычитание" (см. разд. 3.3).

Принято, что два или более взаимодействующих процессов не могут обращаться к *P*- и *V*-операциям одновременно. Считается, что неделимость этих операций программно реализуется благодаря тому, что три ее этапа – 1) считывание значения семафора из памяти, 2) увеличение или уменьшение значения семафора, 3) запись нового значения в память – выполняются при условии доступа к *S* только от одного из процессов и запрещения доступа от других процессов.

Модель *P*- и *V*-операций на языке сетей Петри представлена на рис. 5.11. По запросу *P*-операции, т.е. появлению маркера в позиции p_1 из некоторого буфера, она осуществляется, если в семафоре содержится $\mu(S) \geq 1$ маркеров, при этом в p_2 переносится маркер. При $\mu(p_2) = 1$ возможно выполнение *V*-операции, в результате которой по одному маркеру заносится в позиции p_3 и семафорную позицию *S*. Неделимость семафорных операций гарантируется неделимостью операции срабатывания перехода СП.

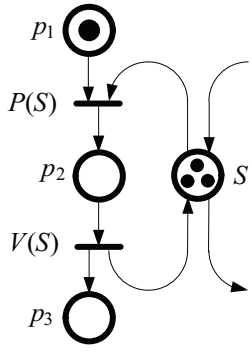


Рис. 5.11. P- и V-операции над семафором S

Вновь обратимся к рис. 5.10 и поясним, как решается задача взаимного исключения. Два процесса, один из которых образован позициями p_1–p_4, а другой – позициями p_5–p_8 с соответствующими переходами, взаимодействуют с общим разделяемым ресурсом, представленным семафорной перемен-

ной *S*. Позиция *S* и вся сеть являются безопасными. Переходы, помеченные буквами *P* и *V*, реализуют соответствующие семафорные операции. После срабатывания перехода *P* одного процесса (например, первого) маркер будет изъят из p_1 и *S* и перемещен в p_2. При этом невозможно, во-первых, обращение ко второму процессу, поскольку $\mu(S) = 0$, и, во-вторых, обращение к первому процессу, находящемуся в критическом интервале. До срабатывания перехода *C* (перемещения маркера из p_2 в p_3) первый процесс находится в критическом интервале (иногда говорят в "критической секции"). После срабатывания перехода *C* может сработать переход *V*, реализующий *V*-операцию, при этом маркер изымается из p_3 и помещается в p_4 и *S*. Появление маркера в семафоре *S* разрешает обращение к любому из процессов, не находящихся в своих критических интервалах. Переходы *R* моделируют репозицию процессов.

В данном случае семафор S может иметь только значения 0 и 1, вследствие чего он называется *двоичным семафором*.

Понятие семафора Дейкстры подвергалось многочисленным переформулировкам и расширениям из-за того, что текущее значение семафора несет только информацию о числе возможных обращений к разделяемому ресурсу.

Семафоры Цикритзиса и Бринч-Хансена

Семафоры Цикритзиса включают в область значений отрицательные числа, что позволяет считать число задержанных процессов. *Семафоры Бринч-Хансена* представляют собой структуру данных, состоящую из трех переменных: начального значения семафора при инициации задачи, числа выполнений операций *SIGNAL* (*CONTINUE*) – аналог *V*-операции – и *WAIT* (*DELAY*) – аналог *P*-операции. Функционально все эти типы семафоров одинаковы.

Семафоры Агервалы и Вентилборга

Семафоры Агервалы и Вентилборга допускают задание приоритета в *P*-операции и (или) содержат механизмы, препятствующие возникновению тупиковых ситуаций при работе с несколькими семафорами, каждый из которых управляет своим разделяемым ресурсом, а также реализует возможность одновременного обслуживания нескольких процессов. Усложненность семантики и реализации послужила причиной ограниченного практического применения последних двух типов семафоров.

Использование семафоров в задаче взаимного исключения предполагает последовательное выполнение обеих операций P и V в каждом из процессов, при этом важным является неделимость (атомарность) этих операций. Интересны и другие применения семафоров, в частности, для решения задачи синхронизации процессов. В последнем случае операции P и V над одним и тем же семафором распределяются между синхронизируемыми парами процессов. Примером типичной задачи синхронизации является *задача* "*производители – потребители*".

5.4. Задача "производители – потребители"

Пусть имеется несколько параллельных процессов. Часть из них, называемая ***производителями****, служат для подготовки (производства) некоторой информации (продукции), предназначенной для использования другой частью процессов, называемых* ***потребителями****. Проблема организации взаимодействия между этими процессами носит название задачи "производители – потребители".*

Рассмотрим случай двух параллельных процессов, один из которых является производителем, а другой – потребителем. На основе рис. 5.2 легко строится сетевая модель задачи "производитель – потребитель", представленная на рис. 5.12. Здесь подпроцесс, образованный позициями p_1, p_2, p_3 и переходами t_1, t_2, t_3, моделирует производителя, а подпроцесс, образованный позициями p_6, p_7, p_8 и переходами t_4, t_5, t_6, – потребителя.

Синхронизирующий цикл для производителя замыкается через потребителя ($p_5 t_5 p_8 t_6 p_4$) и наоборот для потребителя замыкается через производителя ($p_4 t_2 p_3 t_3 p_5$). Число маркеров в позиции p_4 характеризует число заказов в портфеле производителя (при данной начальной маркировке, равной 3), позиция p_5 моделирует склад готовой продукции (буфер ограниченной емкости).

Очевидно, что число маркеров в p_5 не может превышать числа маркеров в начальной маркировке позиции p_4. Суммарное число маркеров в позициях p_4 и p_5 является константой, что гарантирует "сбалансированный выпуск продукции". При этом темпы производства и потребления могут изменять-

ся в процессе совместного функционирования, но при их "разбегании" возможны простои, связанные с нахождением подпроцессов в фазах ожидания. Возможности такой рассинхронизации ограничены общим числом n точек в позициях p_4 и p_5. Степень рассинхронизации может оцениваться максимальным *эксцессом* между переходами t_1 и t_4 (число непрерывных появлений t_1 без появления t_4) во всех возможных последовательностях срабатываний.

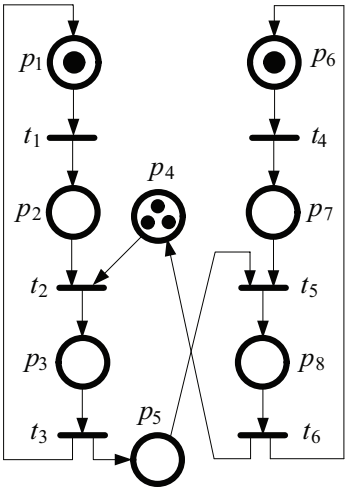


Рис. 5.12. Задача "производитель – потребитель"

"Семафорная" интерпретация p_4 и p_5 наталкивает на мысль об интерпретации пар переходов $t_2 t_3$ и $t_3 t_5$ операциями над семафорами $P(S_1)$, $V(S_1)$ и $P(S_2)$, $V(S_2)$, где S_1 – семафор наличия заказов в портфеле производителя, а S_2 – семафор наличия товара на складе готовой продукции. Роль семафоров в синхронизации рассматриваемой пары процессов состоит в том, что первый процесс, выполняя переход t_1, может опередить второй процесс не более чем на n+1, $n=S_1+S_2$, шагов по отношению к t_4.

Легко построить обобщения рассматриваемой задачи для случая нескольких производителей и (или) потребителей. Вариант для случая двух производителей и одного потребителя представлен на рис. 5.13.

Можно убедиться в наличии конкуренции между производителями вследствие того, что ресурс заказов является общим, и при существенной разнице в производительности отстающий производитель может оказаться в фазе ожидания, и заказы чаще будут выполняться передовым произво-

дителем. Естественно считать, что производители выпускают одинаковую продукцию. В такой форме задача демонстрирует обобщенное использование семафоров S_1 и S_2 не только для синхронизации, но и для взаимного исключения, поскольку семафор S_1 является разделяемым между процессами производителей.

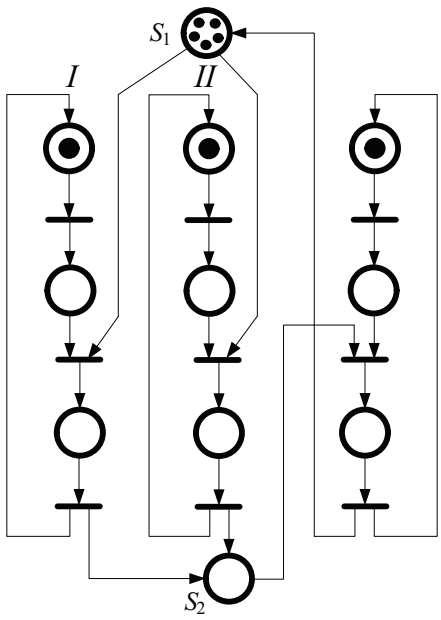


Рис. 5.13. Два производителя и один потребитель

Спецификация решения задачи "производитель – потребитель" непосредственно на сетях Петри представляет собой пример модельно чистого подхода, имеющего несомненное превосходство в наглядности по сравнению с обычной "программистской" нотацией, за "синтаксическими шторами" которой могут ускользнуть существенные эффекты. В качестве не совсем удачного примера решения данной задачи можно привести запись на псевдокоде фрагмента спецификации, сделанную Г. Дейтелом.

Фрагмент спецификации Г. Дейтела

Этот фрагмент изображен на рис. 5.14. Предполагается, что семафоры имеют следующие исходные значения: ИСКЛЮЧЕНИЕ = 1, ГОТОВО = 0.

```
Process   Процесс-производитель;
begin while  истина  do
        begin
                Производство товара;
                P (ИСКЛЮЧЕНИЕ );
                СКЛАД:= ТОВАР1;
                V (ИСКЛЮЧЕНИЕ );
                V (ГОТОВО);
        end
end;
Process   Процесс-потребитель;
begin while  истина  do
        begin
                Производство товара;
                P (ГОТОВО);
                P (ИСКЛЮЧЕНИЕ );
                ТОВАР2:= СКЛАД
                V (ИСКЛЮЧЕНИЕ );
                Использование товара;
        end
end;
```

Рис. 5.14. Фрагмент решения задачи "производитель – потребитель", приведенный Г. Дейтелом

Сеть Петри, соответствующая этой программе, приведена на рис. 5.15. Анализ сети показывает, что в данном решении помимо синхронизации процессов производителя и потребителя, которые работают с одноместным складом-буфером, необходимо также ограничивать возможности обращения к буферу (подъезда к складу) таким образом, чтобы его загрузка и разгрузка не производилась одновременно. Иначе говоря, переходы, помеченные операторами СКЛАД := ТОВАР1 и ТОВАР2 := СКЛАД, не должны выполняться одновременно.

Синхронизация обеспечивается семафором ГОТОВО, а взаимное исключение – семафором ИСКЛЮЧЕНИЕ. Казалось бы, обе требуемые задачи решены. Однако анализируя сеть Петри более внимательно, легко заметить, что возможна ситуация, когда потребитель еще не успел забрать первую партию

товара со склада (наличие маркера в позициях p_1 и p_2), а производитель, после производства второй партии товара, вновь входит первым в критический участок и загружает вторую партию на склад. При условии, что склад одноместный, это может рассматриваться как аномалия поведения системы.

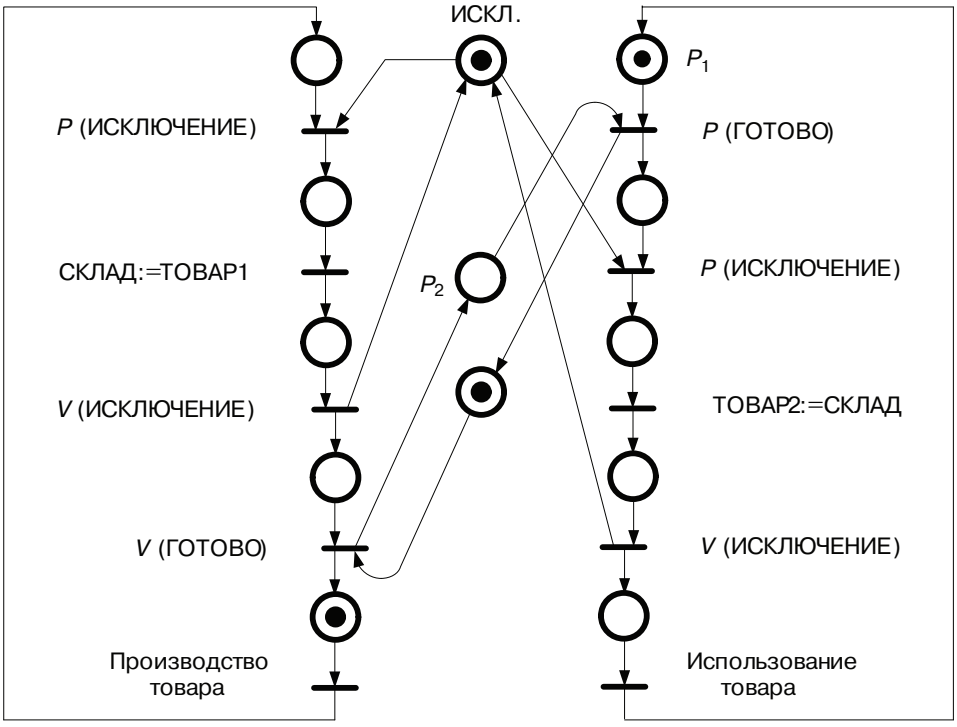


Рис. 5.15. Сеть Петри, соответствующая программе на рис. 5.14

Программа Д. Цикритзиса и Ф. Бернстайна

Другим возможным вариантом решения задачи "производители-потребители" может быть программа Д. Цикритзиса и Ф. Бернстайна с использованием трех семафоров вместо двух. Из них два семафора ПУСТОЙ и ПОЛНЫЙ обеспечивают безопасную синхронизацию, а третий ИСКЛЮЧЕНИЕ по-прежнему реализует взаимное исключение процессов при обращении к буферу-складу. Сеть Петри для этого случая показана на рис. 5.16.

Легко видеть, что производитель не сможет пройти в критический участок вторично без ответного прохода через него потребителя: это предотвращается благодаря использованию семафора ПУСТОЙ.

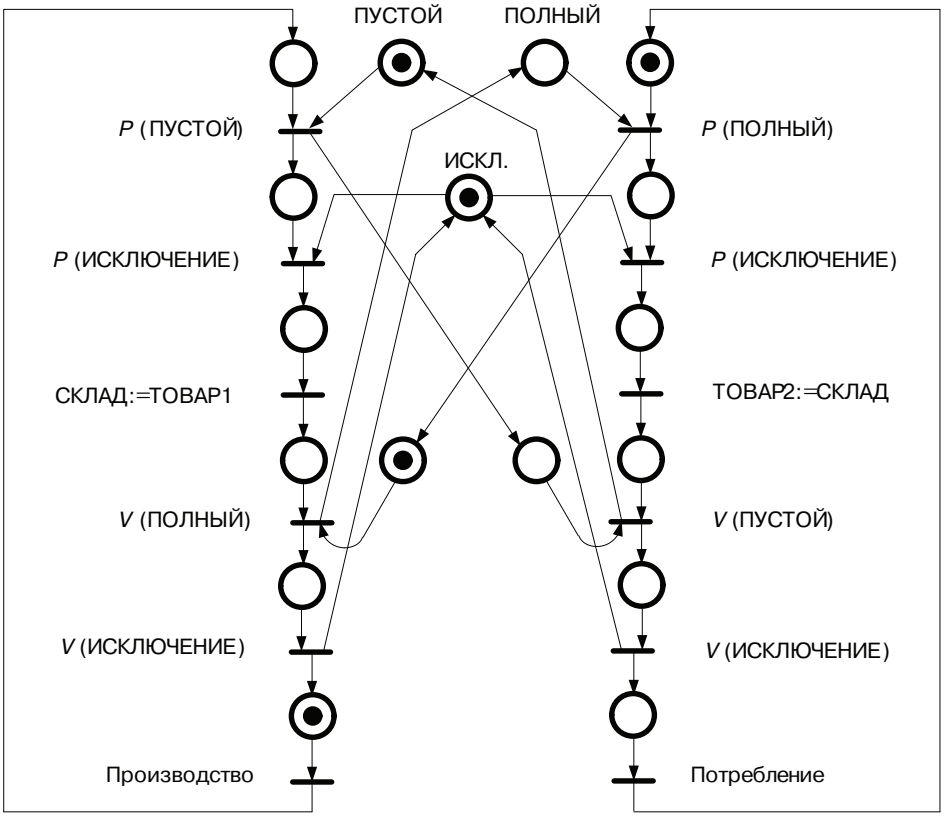


Рис. 5.16. Вариант корректного решения задачи "производитель – потребитель"

Существует ряд задач, в которых требуется обеспечить определенную дифференциацию между процессами при выполнении функций взаимного исключения и синхронизации. Примером задачи такого типа является задача "читатели – писатели".

5.5. Задача "читатели – писатели"

Задача о читателях и писателях *состоит в следующем. Имеется разделяемый ресурс, например, область оперативной памяти или файл, и два типа связанных с ним процессов. Процессы первого типа, называемые "читателями", считывают информацию из ресурса и могут иметь к нему одновременный доступ. Процессы второго типа, называемые "писателями", вносят изменения в ресурс путем записи но-*

вой информации в него и поэтому имеют к нему исключительный доступ. Следовательно, писатели и читатели не могут иметь одновременного доступа к ресурсу, кроме того, писатели взаимно исключают друг друга. Существуют различные постановки этой задачи в зависимости от требований, связанных с приоритетностью доступа писателей по отношению к читателям. Например, можно различать следующие варианты:

а) читатели получают доступ к ресурсу с минимальной задержкой; при этом, если хотя бы один читатель будет постоянно пользоваться ресурсом, то последний будет постоянно недоступен писателям, которые ждут, когда ресурс освободится;

б) читатели в отсутствие писателей получают доступ к ресурсу с минимальной задержкой; писатель, желающий получить доступ, не должен ждать, пока все читатели освободят ресурс; иначе говоря, при появлении запроса от писателя необходимо закрывать ресурс для запросов от читателей, пришедших после него.

В литературе по параллельному программированию задача "читатели – писатели" является, пожалуй, самой популярной. Это объясняется многообразием различных постановок, а также благодаря ее чрезвычайно широкой практической применимостью в системах информационного доступа. Существуют попытки ее решения с помощью таких общеизвестных изобразительных средств, как семафоры, мониторы, рандеву, траекторные выражения, язык *CSP* Хоара и т.д.

Сравнивая варианты (а) и (б), легко заметить, что вариант (а) уступает варианту (б) из-за того, что в нем может возникнуть эффект "голодания" которому будут подвержены писатели в случае постоянного "читательского" спроса. В то же время решение задачи для варианта (б) существенно сложнее, поскольку требует механизма внеочередного допуска к ресурсу писателя с блокировкой всех читателей, появляющихся после этого писателя.

Не будем останавливаться на всех механизмах, используемых для решения указанной задачи во всех ее постановках. Тем более что с точки зрения операционной (т.е. поведенческой) семантики, связанной с порядком выполнения отдельных операций записи в ресурс и чтения из ресурса, различие между всеми этими механизмами весьма условное. Остановимся лишь на следующем специфическом свойстве этой задачи.

В отличие от задачи “производители – потребители”, в данной задаче, например, для варианта (а) процессы имеют дифференцированный доступ к ресурсу: читатель получает доступ, даже если с ресурсом работает другой читатель, а писатель не может работать с ресурсом одновременно с кем бы то ни было. Такое специфическое поведение можно представить достаточно просто в виде сети Петри, как это показано на рис. 5.17. Эта сеть соответствует модели системы, состоящей из двух читателей и одного писателя. Позиция P^* фактически связана с некоторой числовой переменной ЧИТАТЕЛИ, задающее текущее число читающих читателей, т.е. тех, кто находится в критическом (в данном случае для писателя) участке.

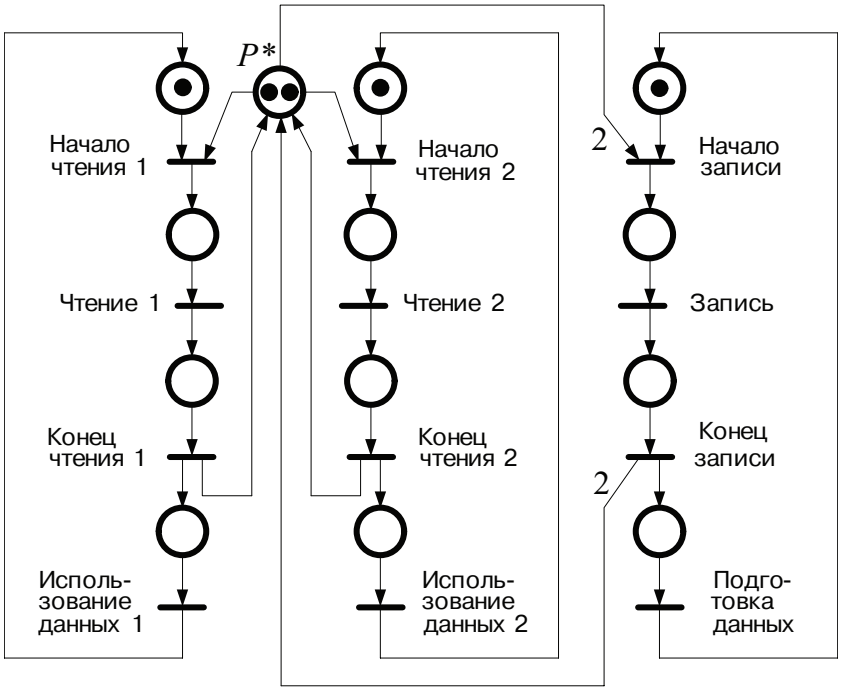


Рис. 5.17. Пример взаимодействия с ресурсом в задаче “читатели – писатели”

Наличие двух маркеров в P^* свидетельствует о том, что ресурс свободен. В этом случае писатель может войти в критический участок для записи. При этом существенное значение имеет метка 2 на дугах, ведущих из P^* и в P^* со стороны писателя, означающая, что для срабатывания перехода “начало записи” требуется два маркера в P^*, а после срабатывания перехода “конец записи” в P^* вновь появляется два маркера.

Реализация механизма добавления и вычитания единицы из переменной ЧИТАТЕЛИ (соответственно переходы "начало чтения *i*" и "конец чтения *i*") требует взаимного исключения процессов-читателей на этих операциях. В систему необходимо добавить отдельные семафоры на входы указанных выше переходов.

Полученное таким образом описание будет громоздким. К аналогичному результату можно придти, используя концепцию монитора Хоара. Монитор – это совокупность (пассивный синтаксический объект) процедур, осуществляющих функции синхронизации и взаимного исключения между процессами, вызывающими эти процедуры.

С точки зрения семантики сетей Петри представление процедур монитора, как уже говорилось, практически не отличается от спецификации операций, осуществляемых над семафорами. Существенно, что все процедуры монитора постулируются как взаимоисключаемые. С учетом высказанных выше замечаний по поводу изменения значений переменной ЧИТАТЕЛИ построим описание той же системы на основе вызовов мониторных процедур. Рис. 5.18 задает работу процессов типа читатель и писатель. Переходы сетей Петри, отмеченные вызовом процедур вида "Монитор. *", связаны с выполнением соответствующих подпроцессов, которые показаны на рис. 5.19.

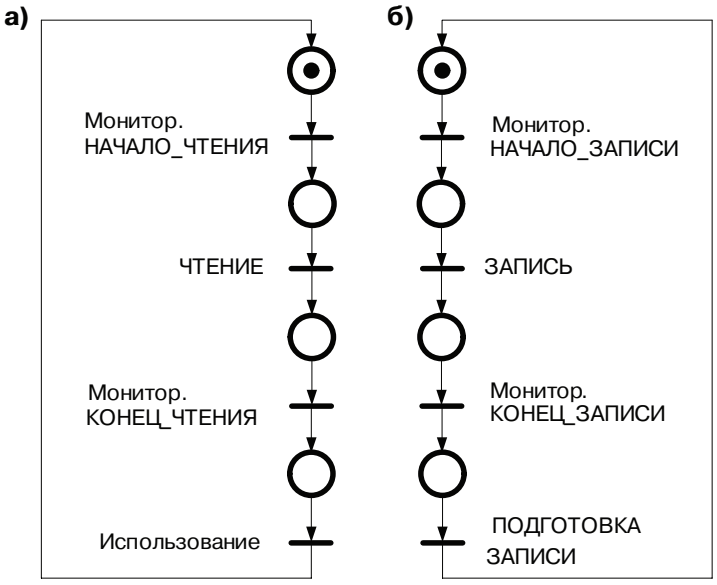


Рис. 5.18. Описание процессов типа а) "читатель" и б) "писатель", взаимодействующих через монитор

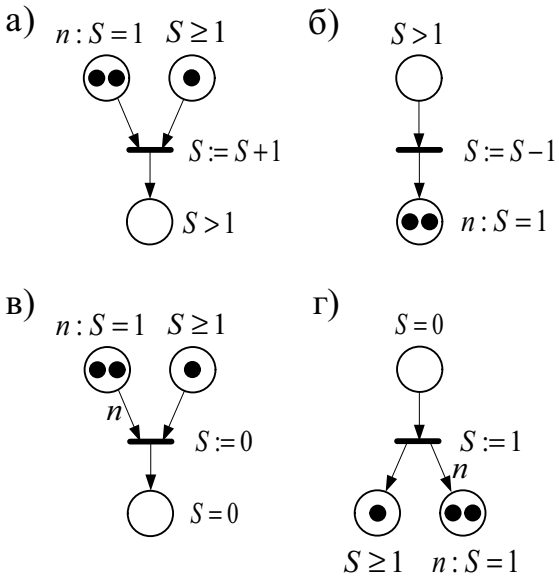


Рис. 5.19. Описание процедур монитора в задаче "читатели – писатели": а) начало чтения, б) конец чтения, в) начало записи, г) конец записи

На этом рисунке позиции помечены предикатами, отражающими семантику наличия маркера в этих позициях. Специфическая запись $n : S = 1$ означает, что если данная позиция содержит n точек (n – максимальное число читателей, которые могут использовать ресурс), то предикат $S = 1$ истинен. В свою очередь переменная S имеет следующий смысл (для $n = 2$):

$S = 0$ – один писатель использует ресурс;

$S = 1$ – ресурс свободен;

$S = 2$ – один читатель использует ресурс;

$S = 3$ – два читателя используют ресурс.

Для анализа системы из двух читателей и одного писателя достаточно изобразить два процесса как на рис. 5.18,а, один процесс, показанный на рис. 5.18,б, и вместо переходов, обозначающих вызовы процедур монитора, подставить соответствующие подпроцессы, изображенные на рис. 5.19. При этом одноименные позиции должны быть соответствующим образом совмещены, а начальная маркировка позиций мониторных процедур должна соответствовать состоянию $S = 1$ (ресурс свободен).

5.6. Задача о курильщиках

Сеть Петри на рис. 5.10 создает основу для более изощренных подходов к решению взаимного исключения, некоторые из которых предлагаются читателю в качестве упражнений.

Интересно заметить, что семафорные синхропримитивы не являются универсальным средством для решения всех задач синхронизации (координации) параллельных процессов. Оказалось, например, что этот механизм не позволяет решить *задачу о курильщиках*, которая формулируется следующим образом.

Курильщик, если у него есть табак, бумага и спички, скручивает самокрутку и закуривает ее. Но каждый курильщик имеет лишь один необходимый ингредиент: один табак, второй бумагу, третий спички. Поэтому они должны обращаться к агенту (владельцу табачной лавки), который имеет неограниченные запасы всех трех ингредиентов. Агент выдает курильщикам лишь два из трех ингредиентов. Курильщик, имеющий третий ингредиент, забирает эти два, скручивает самокрутку и закуривает, после чего уведомляет об этом агента. Затем агент предлагает два других ингредиента, и цикл повторяется.

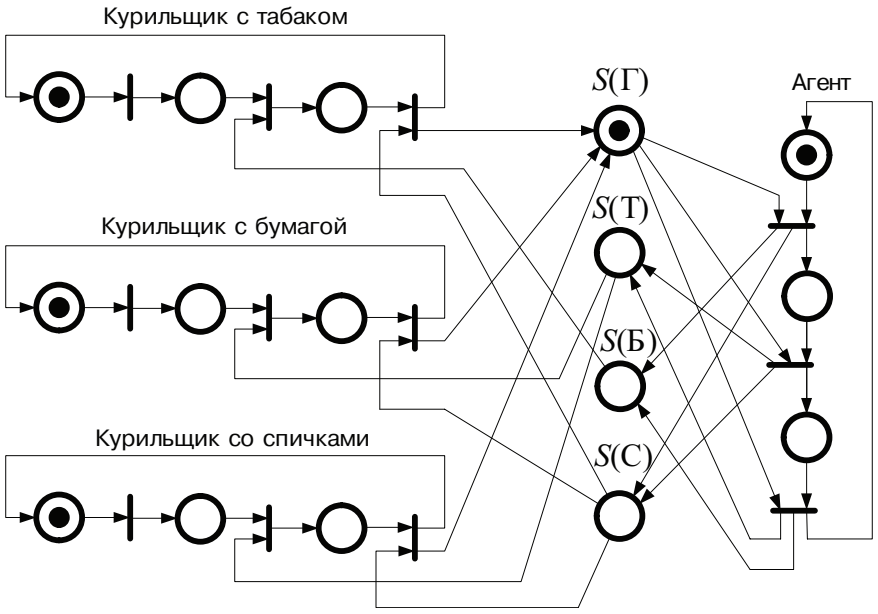


Рис. 5.20. Задача о курильщиках

Вариант спецификации задачи о курильщиках с помощью семафоров показан на рис. 5.20. Из рисунка видно, что в предлагаемом варианте используются четыре семафора: $S(Г)$, моделирующий уведомления от курильщиков о завершении очередного цикла, $S(Т)$, $S(Б)$ и $S(С)$, моделирующие наличие соответствующего ресурса: табака, бумаги и спичек.

Очевидно, что предложенный вариант решения некорректен.

Задача о курильщиках, несмотря на "обиходную" постановку, находит практическое применение в операционных системах. Например, возможна такая интерпретация этой задачи. Три процесса (1, 2 и 3) пользуются общим разделяемым ресурсом, состоящим из устройства ввода, терминала и печатающего устройства. При этом процессу 1 одновременно требуется устройство ввода и терминал, процессу 2 – устройство ввода и печатающее устройство, а процессу 3 – терминал и печатающее устройство.

5.7. Механизм рандеву

Рассмотрим механизм взаимодействия параллельных процессов, названный привлекательным французским словом "рандеву". Термин "рандеву" используется в различных моделях и его семантика в этих моделях различается незначительно (с нашей точки зрения). Остановимся лишь на двух основных видах рандеву:

а) рандеву при выполнении команд ввода-вывода процессами в языке *CSP* (теория взаимодействующих процессов Хоара);

б) рандеву при вызове одной задачи другой задачей в языке Ада.

Рассмотрим первый тип рандеву. Синхронизация и взаимодействие процессов в языке *CSP* осуществляется с помощью так называемых команд ввода-вывода, задаваемых в парах взаимодействующих процессов. Эти процессы взаимно ссылаются друг на друга, а передаваемые от одного к другому процессу данные должны быть совместимыми по типам. Взаимодействие по принципу рандеву состоит в передаче данных от процесса-источника к процессу приемнику и происходит в тот момент, когда оба процесса выполняют соответствующие команды ввода и вывода. Такое общение процессов аналогично общению людей, сущность которого отражена в лозунге: "ты мне – я тебе". Иначе говоря, обмен сообщениями между процессами являет-

ся асинхронным и согласованным, т.е. осуществляется при взаимной готовности обоих процессов к акту взаимодействия. Действительно, если процесс выполняет команду ввода (вывода), а соответствующая команда вывода (ввода) в процессе-партере еще не выполнена, то этот процесс задерживается до момента выполнения команды процессом–партнером.

Простейшая программа на языке *CSP* для решения задачи о взаимодействии процессов производителя и потребителя через одноместный буфер может быть описана следующим образом:

ПРОИЗВОДИТЕЛЬ ::

* [TRUE → БУФЕР ! ТОВАР ;]

ПОТРЕБИТЕЛЬ ::

* [TRUE → БУФЕР ? ТОВАР ;]

БУФЕР ::

* [TRUE → ПРОИЗВОДИТЕЛЬ ? ТОВАР

→ ПОТРЕБИТЕЛЬ ! ТОВАР ;]

Ограничимся лишь этим примером, нотация которого является вполне прозрачной. Семантика рандеву обмена сообщениями ясно прослеживается в сетевой модели этой программы, представленной на рис. 5.21.

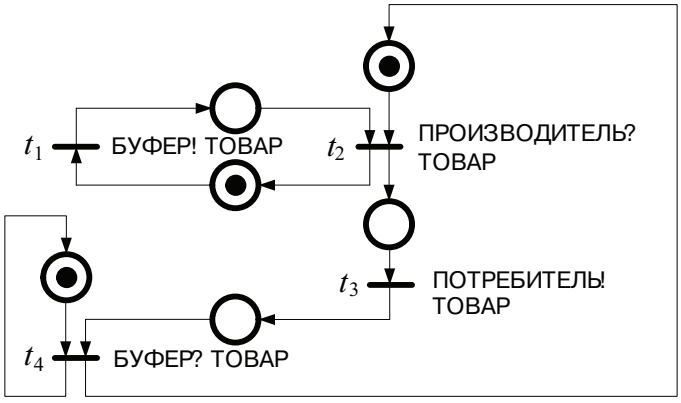


Рис. 5.21. Сеть Петри, иллюстрирующая механизм рандеву на основе обмена сообщениями

Остановимся теперь кратко на механизме рандеву, применяемом для межзадачных взаимодействий в языке Ада. Принцип рандеву в языке Ада состоит в следующем. Для того, чтобы между задачами состоялось рандеву, необходимо наличие двух задач: вызывающей и вызываемой (обслуживающей). Вызывающая задача обращается к определенному входу обслуживающей задачи, которая выдает команду ACCEPT, если она готова это сделать. Если обслуживающая задача не выдала этой команды (команды приема), то вызывающей задаче приходится ждать. Если обслуживающая задача выдает команду ACCEPT для входа, к которому вызывающая задача пока не обратилась, то обслуживающая задача ждет обращения вызывающей задачи к данному входу.

Рандеву происходит в тот момент, когда вызов принимается. Вызывающая задача передает обслуживающей данные как параметры вызова. Эти данные обрабатываются при помощи операторов, входящих в тело команды ACCEPT (между операторными скобками do ... end). Когда эти действия завершаться, параметры-результаты передаются вызывающей задаче. Рандеву заканчивается, после чего вызывающая и вызываемая задачи продолжают свою работу.

Отличительная особенность механизма рандеву в Аде заключается в том, что вызывающая задача должна знать о существовании обслуживающей задачи и иметь информацию о множестве ее входов. В то же время, обслуживающая задача принимает вызов от любой вызывающей задачи, причем много вызывающих задач могут пытаться обратиться к одной обслуживающей. В этом смысле механизм рандеву является несимметричным.

В каждый момент времени возможность рандеву с обслуживающей задачей предоставляется только одной вызывающей задаче, т.е. в реализацию механизма рандеву встраивается механизм взаимного исключения. Остальные вызывающие задачи, одновременно пытающиеся получить рандеву, переводятся в режим ожидания.

Моделью обычного двухстороннего Ада-рандеву для задачи производитель-буфер-потребитель может служить та же сеть Петри (рис. 5.21), в которой переходы $t_1 - t_4$ именуются соответственно как: БУФЕР. ПОМЕСТИТЬ ТОВАР; ACCEPT. ПОМЕСТИТЬ ТОВАР; ПОТРЕБИТЕЛЬ. ПОМЕСТИТЬ ТОВАР; ACCEPT. ПОМЕСТИТЬ ТОВАР. Возможна несколько иная орга-

низация поведения в том случае, когда буфер всегда является обслуживающей задачей. Соответствующая сеть Петри приведена на рис. 5.22.

Пример взаимодействия задач с множественным (случай $n = 2$) обращением к входу обслуживающей задачи (стек) представлен сетью Петри на рис. 5.23. Встроенный механизм взаимного исключения задается частью сети, обведенной пунктиром.

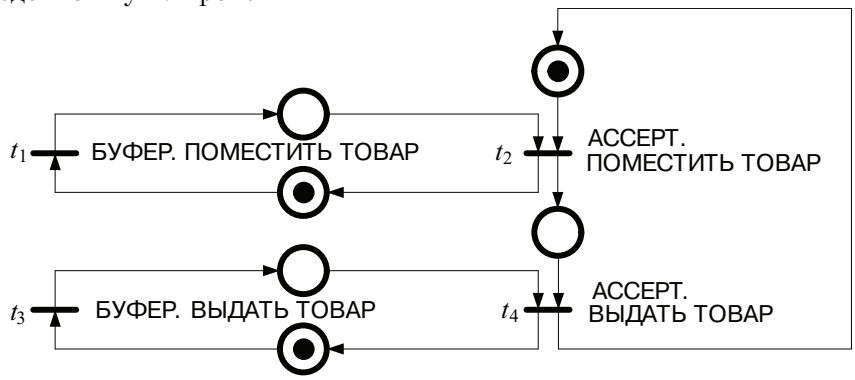


Рис. 5.22. Иллюстрация альтернативного подхода к описанию рандеву типа межзадачного вызова

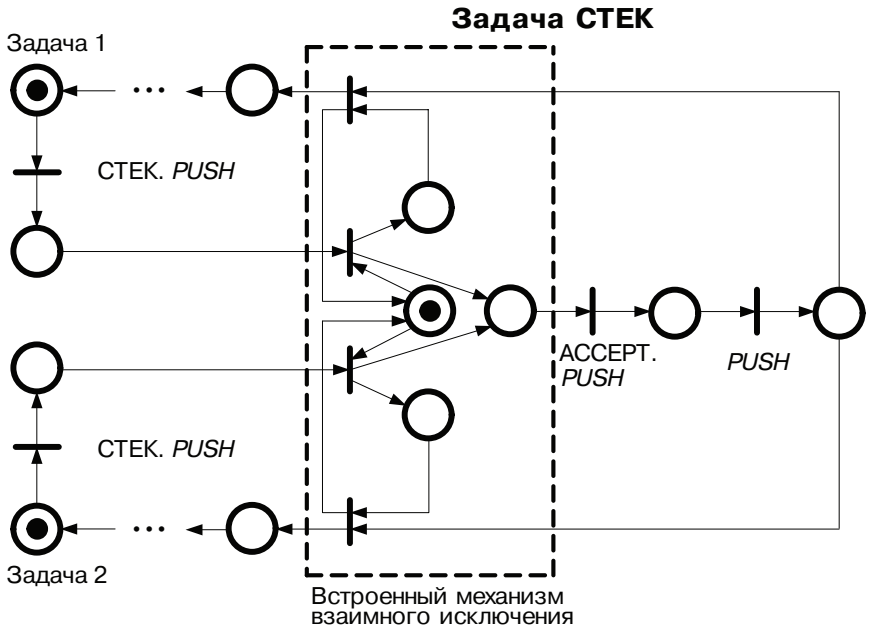


Рис. 5.23. Иллюстрация модели рандеву со встроенным механизмом взаимного исключения

5.8. Семантика взаимодействующих процессов

Рассмотрим проблему спецификации параллельных систем (устройств или программ) несколько шире, чем это делалось раньше. Общий подход к описанию алгоритмических систем предполагает во всякой формальной нотации сочетание синтаксиса и семантики. До сих пор обращалось внимание на семантику (смысл) таких описаний лишь с точки зрения интерпретации синтаксических элементов выбранных моделей. Например, вершинам АП (ситуациям) ставились в соответствие двоичные векторы, а вершинам маркированного графа – изменения значений дискретных переменных и т.п. Однако такой вид семантики ("содержательная интерпретация") является лишь отображением предметной области на некоторый объект в иерархии моделей, адекватный проектируемому аппаратному или программному модулю или системе.

Существует более фундаментальный характер семантики, или угол зрения, относительно которого можно производить сравнительный анализ различных синтаксических построений. Что же в данном случае понимается под семантикой? Это, прежде всего, некоторая формальная структура, которая отражает специфику описываемого объекта и позволяет говорить об эквивалентности различных синтаксических описаний, относящихся к одному и тому же объекту.

Описываемый реальный объект в действительности зачастую не обладает какой-либо специфической структурой. Структура, которая "накладывается" на объект наблюдателем, позволяет ему легче представить себе объект и производить над ним необходимые для анализа и синтеза манипуляции. Например, прямоугольный массив чисел сам по себе не обязательно содержит такие структуры, как строка и столбец. Таким мы его представляем себе для того, чтобы записать его в линейной форме.

В этом разделе рассматривается семантика параллельного поведения. Что позволяет нам судить о том, что некоторая параллельная программа, записанная, скажем, на языке сетей Петри, эквивалентна программе, представленной в нотации Милнера? Во-первых, для ответа на этот вопрос следует оговорить, что такое сравнение в данном контексте проводится именно с точки зрения семантики этих программ, а не тех вычислительных функций, которые они реализуют в предметной семантике. Во-вторых, суще-

ственным фактором является вопрос о наблюдении за параллельными действиями. Здесь следует сделать ряд предположений, носящих принципиальный характер. В системе наблюдения за поведением параллельной системы отсутствует тот идеальный часовой механизм, который может абсолютно точно зафиксировать, какое из двух действий, между которыми нет причинной связи, произошло в системе раньше.

Таким образом, **единственным критерием того, что действие А произошло раньше, чем действие Б, может быть знание того, что А есть необходимая причина для Б.** При наличии такого знания отпадает необходимость ставить нескольких наблюдателей, фиксирующих по своим часам время возникновения А и Б. Если между этими действиями нет причинной связи, то такие наблюдения могут давать разные ответы на вопрос: "Что произошло раньше А или Б?" **Если отсутствует информация о причинности между действиями, то все же приходится прибегать к наблюдателям и делать вывод об упорядочении действий А и Б, исходя из следующего критерия. Если во всех последовательностях наблюдений А происходит раньше Б, то А предшествует Б; если же существует хотя бы одна последовательность, в которой Б замечено раньше, чем А, то А и В неупорядочены и могут считаться параллельными.**

Эти соображения приводят к выводу о том, что **семантика параллельного поведения может базироваться либо на данных о последовательностях наблюдений за системой, действия которой видимы извне, либо на знании о причинных связях между действиями.**

Рассмотрим простую параллельную программу, заданную четырьмя различными способами: сетями Петри, маршрутными выражениями, нотацией Милнера, на языке ОККАМ (нотация Хоара).

- **Сеть Петри** для данного примера показана на рис. 5.24.

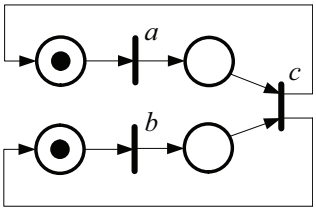


Рис. 5.24. Спецификация параллельной программы на языке сетей Петри

- **Маршрутное выражение** записывается следующим образом:

  path *a*; *c* end

  path *b*; *c* end

Здесь первая траектория (маршрут) задает чередование действий *a* и *c*, начинающихся с *a*. Вторая траектория задает аналогичное чередование действий *b* и *c*. Совместное функционирование последовательных процессов (траекторий) дает параллельную систему (программу), в которой действия *a* и *b* не упорядочены, а активизация *c* чередуется с параллельной активацией *a* и *b*.

- **На языке CCS Милнера** подобная программа будет выглядеть как $p \Leftarrow (a \mid b).c.p$.

- **Наконец, язык ОККАМ**, концептуально близкий теории взаимодействующих последовательных процессов Хоара, задает эту программу в виде

```
while true
    seq
      par
        a
        b
      c
```

Приведенные четыре фрагмента описывают одну и ту же систему, но их семантика различается в значительной степени.

Сеть Петри не акцентирует внимания на том, сколько процессов взаимодействуют друг с другом и каково это взаимодействие, зато она явно указывает, что причинами *c* являются *a* и *b*. В то же время траекторные выражения явно показывают, что существуют два параллельно функционирующих процесса, совместно выполняющих действие *c*.

И тем не менее, между этими моделями есть нечто общее, что позволяет использовать методы анализа и верификации, адекватные одной модели, для другой модели. Так, например, известная из литературы теорема Генриха-Коммонера-Холта о необходимых и достаточных условиях дедлока может быть применена к сетевому представлению и позволяет установить отсутствие дедлока в соответствующей программе. С другой стороны, нотация

CCS дает возможность проводить алгебраические манипуляции и доказательства в таких системах. Язык ОККАМ является исполнимым языком программирования и для него существуют компиляторы для ряда вычислительных систем, что позволяет реализовать описанные процессы.

Что касается общего средства представления поведения этих систем в виде модели АП, то на рис. 5.25 показан соответствующий приведенным моделям автономный АП, в котором переходы отмечены действиями *a*, *b* и *c*. Оказывается, что такой АП не полностью удовлетворяет требованиям, предъявляемым к описанию семантики программы, включающей действия, которые могут считаться параллельными в силу отсутствия причинной связи. Действительно, АП на рис. 5.25 описывает поведение, в котором действия *a* и *b* могут происходить в любом порядке, но строго последовательно, если рассматривать АП как множество альтернативных траекторий.

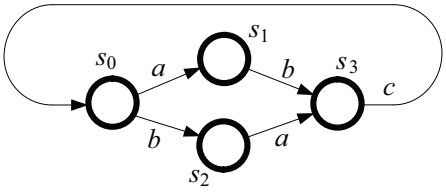


Рис. 5.25. Автономный АП, задающий альтернативные траектории в алфавите {a,b,c}

В нотации *CCS* эта семантика явно выражена, так как согласно принятым в ней правилам, запись $p \Leftarrow (a \mid b).c.p$ эквивалентна выражению

$$\begin{aligned} p' &\Leftarrow a.b.q' + b.a.q' \\ q' &\Leftarrow c.p' \end{aligned}.$$

В то же время и в сетях Петри, и в маршрутных выражениях такое представление не является вполне адекватным, поскольку в них явно имеется существенная разница между перестановочной семантикой (“в любом порядке, но последовательно”) и семантикой причинности (“порядок отсутствует ввиду параллельности”).

Сказанное подтверждает сеть Петри, представленная на рис. 5.26. Она не эквивалентна сети на рис. 5.24, хотя ей может быть поставлен в соответствие тот же АП, представленный на рис. 5.25, поскольку множество последовательностей срабатывания, т.е. перестановочная семантика, является одинаковым.

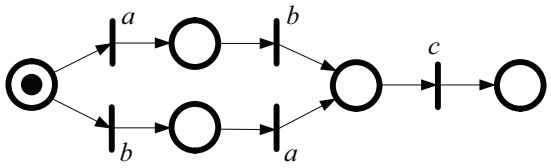


Рис. 5.26. Сетевое представление перестановочной семантики

Это позволяет дифференцировать локальную форму недетерминизма (см. гл.1) на параллельность и перестановочность, невидимые в терминах неструктурированного АП, поскольку эта дифференциация возможна лишь в терминах более детальной модельной интерпретации, которая достигается в сетях Петри.

Для более полного освещения различий между этими двумя фундаментальными семантическими формами (предмет активных научных дискуссий и исследований недавнего прошлого) рассмотрим еще ряд примеров, показывающих большую модельную силу семантики причинности, или семантики частичного порядка.

Пошаговая детализация действий программ

Продемонстрируем, что семантика причинности в отличие от перестановочной семантики способна сохраняться в адекватном виде при декомпозиции (детализации) действий.

Пусть в системе параллельно происходят действия a и b. В рамках перестановочной семантики записи $\alpha = a \parallel b$ и $\beta = a;b + b;a$ являются эквивалентными моделями такой системы. На языке сетей Петри α и β представлены соответственно на рис. 5.27,а и б.

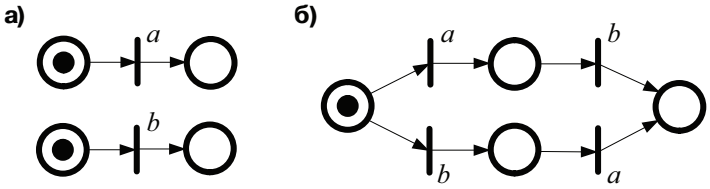


Рис. 5.27. Сетевое представление а) семантики причинности и б) перестановочной семантики

Пусть действие a может быть представлено двумя действиями a_1 и a_2. Внесем эту детализацию в обе модели путем замены a на $(a_1; a_2)$. Получим $\alpha' = (a_1; a_2) \mid b$ и $\beta' = (a_1; a_2); b + b; (a_1; a_2)$. Соответствующие сети Петри показаны на рис. 5.28а и рис. 5.28б. Анализ последовательностей срабатываний в этих моделях показывает, что последовательность $a_1 b a_2$ имеет место в α', но отсутствует в β'. Из этого следует, что эквивалентность моделей в терминах перестановочной семантики не сохраняется при детализации действий программы.

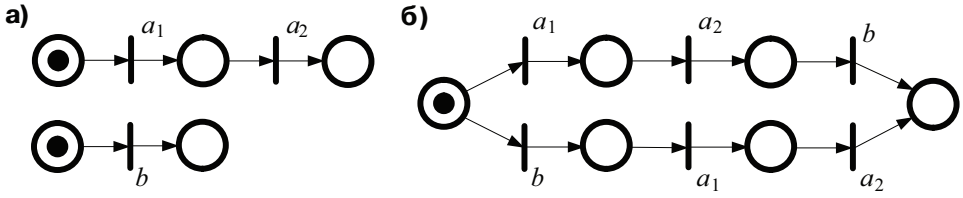


Рис. 5.28. Результат детализации в семантиках, представленных на рис. 5.27

Напротив, в терминах семантики причинности модели α и β являются различными и, более того, можно показать, что семантическая эквивалентность сохраняется после детализации. Действительно, в α действия a и b причинно не упорядочены, а в β они упорядочены. Сохранность семантической эквивалентности при детализации действий следует из рассмотрения множества $Co(a)=\{b\}$ параллельных (не связанных причинно с a) действий. Легко видеть, что после детализации указанным выше образом получим $Co(a_1) = Co(a_2) = Co(a)$. Это говорит о том, что отношения параллельности между элементами действий наследуются в неизменном виде после детализации действий. Следовательно, две эквивалентные по частичному порядку системы остаются эквивалентными и после детализации соответствующих действий в этих системах. Такое свойство причинной семантики можно назвать *детализируемостью*.

Взаимные переходы между двумя формами семантик

Обсудим следующую проблему. **Пусть задано множество** *PO*, **состоящее из списка отношений причинности (множества частично упорядоченных исполнений программы). Пусть также** *IL* **задает множество перестановочных последовательностей исполнения программы. Естественны вопросы: можно ли вывести** *PO* **из** *IL* **и можно ли вывести** *IL* **из** *PO*? **Покажем на конкретном примере, что ответ на первый вопрос в общем случае отрицателен, а на второй – положителен.**

Пусть $PO_0 = \{p_1, p_2\}$, где p_i – частично упорядоченные исполнения $p_1 = a \xrightarrow{b} c$ и $p_2 = b \xrightarrow{a} c$. Запись вида $a \xrightarrow{b} c$ означает, что $a < c$ и b не упорядочено ни с a, ни с c. Данному PO_0 соответствует сеть Петри, показанная на рис. 5.29.

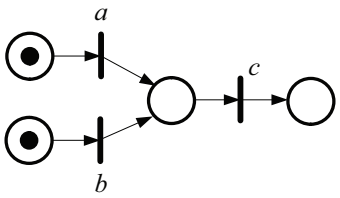


Рис. 5.29. Иллюстрация частичного упорядочения PO_0

Тривиальным является факт однозначности получения из такой сети Петри множества $IL_0 = \{abc, bac, acb, bca\}$, состоящего из перестановочных последовательностей срабатываний переходов. Однако оказывается, что знание лишь IL_0 является недостаточным для восстановления PO_0, поскольку существующий способ установления соответствия между *PO* и *IL*, основанный на так называемой лемме о линеаризации, в данном случае не может быть применен по следующей причине. Согласно этой лемме каждый частичный порядок однозначно определяется множеством его линеаризаций, т.е. строгих порядков, не нарушающих заданный частичный порядок. Знание IL_0 не дает полной информации о том, какие подмножества $IL_0' \subseteq IL_0$, составляющие IL_0 ($\bigcup IL_0' = IL_0$), следует взять для восстановления PO_0. Например, если взять {*acb*, *bac*, *abc*} и {*bac*, *abc*, *bca*}, то получим $\{p_1, p_2\}$. Но если взять {*abc*, *bac*}, { *abc*, *acb*}, { *bac*, *bca*}, то результатом перехода к частичным порядкам будет $\{p_1, p_2, p_3\}$, где $p_1 = \{a \to c, b \to c\}$, $p_2 = \{a \to b, a \to c\}$, $p_3 = \{b \to a, b \to c\}$. Сеть Петри, соответствующая таким частичным порядкам, показана на рис. 5.30. Она отображает причинно-следственные связи другой системы.

Интересным фактом, касающимся возможности восстановления *PO* по *IL*, является то, что такая возможность существует для дистрибутивных систем, т.е. систем, поведение которых представляется дистрибутивной кумулятивной диаграммой векторов срабатываний (см. разд. 4.4). Так для системы, заданной сетью Петри на рис. 5.29, соответствующая кумулятивная диаграмма показана на рис. 5.31.

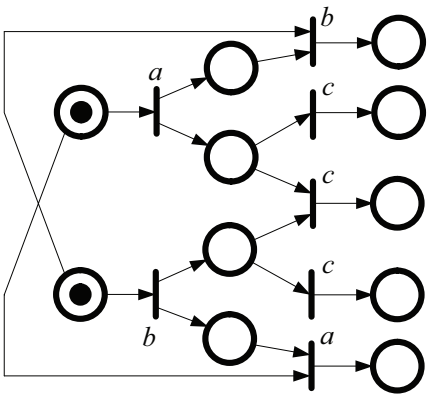


Рис. 5.30. Альтернативный вариант семантики частичного порядка с той же перестановочной семантикой, что и для PO_0

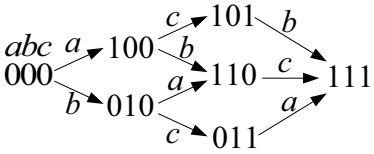


Рис. 5.31. Недистрибутивная кумулятивная диаграмма для СП на рис. 5.29

Она полумодулярна, но не дистрибутивна, поскольку в ней отсутствует набор 001, который мог бы служить наибольшей нижней гранью для пары наборов 101 и 011 (покомпонентный $\min_i(101,011)$ равен 001).

Проверка условий справедливого решения конфликта

Данный пример иллюстрирует возможности для адекватного отображения важного семантического свойства – справедливого исполнения (fairness) – в терминах семантики частичного порядка, а не перестановочных последовательностей.

Сравним две представленные сетями Петри системы, показанные на рис. 5.32.

По перестановочной семантике они неразличимы: существует три возможных последовательности исполнения {*ab*, *ba*, *ac*}. С другой стороны,

семантика причинности дает возможность сказать, что во второй системе (рис. 5.32,б) условие возникновения *c* зависит от того, в каком порядке будут выполнены *a* и *b*, причем порядок выполнения *a* и *b* фиксирован в том смысле, что при наличии двух возможностей (*ab* и *ba*), отсутствует возможность их одновременного срабатывания. Однако в первой системе (рис. 5.32,а) допустимо одновременное срабатывание *a* и *b*. Следовательно, в первой системе вообще бессмысленно связывать возникновение *c* с порядком срабатывания *a* и *b*.

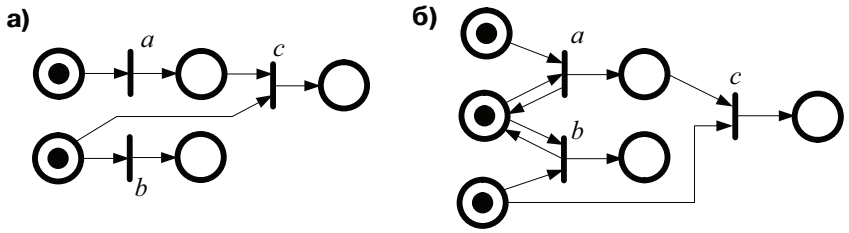


Рис. 5.32. Иллюстрация различий свойства справедливого исполнения в семантике частичного порядка

Каким же образом можно рассуждать о наличии справедливости в отношении действия *c*? В семантике перестановок обе системы могут быть квалифицированы как одинаково справедливые по отношению к *b* и *c*. О справедливости можно говорить только тогда, когда два конфликтующих (одновременно возбужденных) перехода находятся “в равных условиях”, т.е. имеет место свободный выбор. В данном случае семантика частичного порядка позволяет заметить, что возникновение конфликта между *b* и *c* после срабатывания *a* создает такие равные условия для них только во второй схеме, в которой ни один из этих переходов не получает предпочтения при выборе. В первой же схеме переход *b* имеет предпочтение перед *c*, так как он возбуждается раньше *c* и его срабатывание делает невозможным срабатывание *c*.

Выводы

Рассмотренные примеры позволяют сделать вывод о том, что для адекватного представления различных свойств параллельных программ перестановочная семантика является недостаточной. В этой семантике отодвигаются на второй план такие отношения меду действиями, как параллелизм (одновременность действий), а также конфликты и способы их разрешения.

5.9. Заключение

В заключение главы хотелось бы отметить, что проблематика семантики параллельных вычислений, точнее ее фундаментальные вопросы, является чрезвычайно важной для практики синтеза и анализа параллельных программ, поскольку без определения отправных точек для представления поведения таких систем нельзя строго охарактеризовать такие их особенности, как конфликтность, справедливость, одновременность событий и причинно-следственные отношения между ними.

5.10. Упражнения

1. Рассмотрите следующие программные сегменты, задающие две стратегии поведения философа в задаче об обедающих философах:

а)
```
procedure типичный_философ;
begin
    while истина do
        begin
            мыслить_некоторое_время;
            взять_обе_вилки_сразу;
            есть_некоторое_время;
            положить_обе_вилки_сразу;
        end
    end
```

б)
```
procedure типичный_философ;
begin
    while истина do
        begin
            мыслить_некоторое_время;
            repeat
                взять_левую_вилку;
                if правой_вилки_нет
                    then положить_левую_вилку
                        else взять_правую_вилку
            until в_руках_обе_вилки;
            есть_некоторое_время;
            положить_левую_вилку;
            положить_правую_вилку
        end
    end
```

Предложите варианты использования различных модификаций сетей Петри для описания этих стратегий. Проанализируйте предложенные описания.

2. Представьте решение задачи производитель–потребитель для случая, когда склад (буфер) имеет ограниченную емкость, меньшую, чем число заказов.

3. Напишите спецификацию для задачи производитель–потребитель для случая двух производителей разных товаров и одного потребителя, пользуясь цветными сетями Петри.

4. Представьте интерпретацию задачи производитель–потребитель для случая одного производителя и двух потребителей, обладающих разными потребностями. Воспользуйтесь моделью сети Петри с взвешенными дугами.

5. Используя в качестве заготовок рис. 5.18 и рис. 5.19, представьте спецификацию для решения задачи о трех читателях и двух писателях.

6. Представьте спецификацию решения следующей задачи.

Три менеджера базы данных взаимодействуют друг с другом. Каждый из них может контролировать собственную базу данных. В то же время он должен послать всем другим менеджерам сообщение, информируя их о корректировке. Послав эти сообщения, менеджер ждет, пока другие менеджеры получат это сообщение, обработают его и пошлют соответствующие подтверждения (квитанции) об этом. Когда все квитанции будут получены, пославший сообщение менеджер переходит в пассивное состояние. После этого, но не ранее, другой менеджер имеет право корректировки данных и посылки сообщений.

Каждый менеджер может пребывать в одном из трех состояний: “пассивное”, “ожидание” (квитанций) и “рабочее” (корректировка данных по запросу другого менеджера).

Буфер каждого сообщения может находиться в одном из четырех состояний: “не используется”, “послано”, “получено” “квитировано”.

7. Представьте решение задачи “*о спящем парикмахере*”, которое формулируется следующим образом.

Парикмахерская состоит из комнаты ожидания *W* с конечным числом кресел *n* и рабочего зала B. Комнаты *W* и *B* соединены дверью *D*, а выход на улицу имеет лишь комната *W*. Если все *n* кресел в комнате *W* заняты, то пришедший клиент идет в другую парикмахерскую. Если парикмахер заходит в комнату *W* и не находит там ни одного клиента, то он идет спать. Если клиент входит в парикмахерскую и находит парикмахера спящим, то он должен его разбудить. Представьте поведение парикмахера и клиентов как процессы и опишите их работу с помощью сетей Петри. Модифицируйте описание для случая двух парикмахеров.

8. В языке Ада существует несколько способов согласования вызывающей (A) и вызываемой (B) задач, зависящих от выбранных стратегий взаимодействия по принципу рандеву. Со стороны вызывающей задачи возможны следующие три типа вызова:

а) безусловный вызов

```
B. ЗАПРОС ,
```

б) условный вызов

```
select
B. ЗАПРОС ;
else
    АЛЬТЕРНАТИВНОЕ_ДЕЙСТВИЕ;
end select;
```

в) временной вызов

```
select
B. ЗАПРОС ;
or
delay T;
АЛЬТЕРНАТИВНОЕ_ДЕЙСТВИЕ;
end select;
```

где ЗАПРОС является наименованием входа в задачу B, а АЛЬТЕРНАТИВНОЕ_ ДЕЙСТВИЕ соответствует выполнению другой ветви, например, внутренней для вызывающей задачи процедуры.

Со стороны вызываемой задачи потенциальные стратегии определяются также возможностью работы с несколькими, в данном примере двумя, вызывающими программами, которые вызывают задачу по входам X и Y, а именно:

а) фиксированный порядок обслуживания вызовов

```
accept X do ... end;
ЗАВЕРШЕНИЕ_ОБРАБОТКИ_ПО_ВХОДУ_X;
accept Y do ... end;
ЗАВЕРШЕНИЕ_ОБРАБОТКИ_ПО_ВХОДУ_Y;
```

б) обслуживание в порядке поступления запросов (с взаимоисключением)

```
select
        accept X do ... end; – критический интервал
        ЗАВЕРШЕНИЕ_ОБРАБОТКИ_ПО_ВХОДУ_X;
or
        accept Y do ... end; – критический интервал
        ЗАВЕРШЕНИЕ_ОБРАБОТКИ_ПО_ВХОДУ_Y;
end select;
```

в) обслуживание в порядке поступления запросов с дополнительными логическими условиями (V и W)

```
select
        when V ⇒
        accept X do ... end; – критический интервал
        ЗАВЕРШЕНИЕ_ОБРАБОТКИ_ПО_ВХОДУ_X;
or
        when W ⇒
        accept Y do ... end; – критический интервал
        ЗАВЕРШЕНИЕ_ОБРАБОТКИ_ПО_ВХОДУ_Y;
end select;
```

г) то же, что и в пункте б), но с выделением лимита времени ожидания рандеву

```
select
        accept X do ... end; – критический интервал
        ЗАВЕРШЕНИЕ_ОБРАБОТКИ_ПО_ВХОДУ_X;
or
        accept Y do ... end; – критический интервал
        ЗАВЕРШЕНИЕ_ОБРАБОТКИ_ПО_ВХОДУ_Y;
or
        delay T;
        ОБРАБОТКА_ПО_ТАЙМ-АУТУ;
end select;
```

Приведите адекватные сетевые представления (ординарные, временные, приоритетные и прочие СП) для задания каждой из стратегий. Рассмотрите варианты комбинирования стратегий в системе из двух вызывающих и одной вызываемой задачи.



Глава 6.

ПРИНЦИПЫ АППАРАТНОЙ РЕАЛИЗАЦИИ

– Вот об этом я бы поговорил с удовольствием, но вы ничего не поймете.
– Почему же это я ничего не пойму? Я все-таки инженер.
А. Стругацкий, Б. Стругацкий. "Пикник на обочине".

В динамических моделях, рассмотренных в предыдущих главах, акцентировалось внимание на одной их важной особенности – ориентации на описание процессов, инвариантных к временным интервалам между системными событиями. Естественно потребовать, чтобы адекватные средства аппаратной поддержки также обеспечивали независимость функционирования реализующих спецификации устройств от временных параметров компонент, из которых они построены. В настоящее время господствуют два подхода к построению аппаратных средств: синхронный и асинхронный. Первый нашел применение в вычислительной технике, а второй, разработанный в значительно меньшей степени, – в системах логического управления.

Общим недостатком этих подходов является необходимость использования единого временного механизма (часов). В синхронных схемах он реализуется явно путем применения тактовых (стробирующих) сигналов, в асинхронных схемах этот механизм присутствует неявно: повторную инициацию схемы нельзя производить, пока в ней не закончились переходные процессы, вызванные предыдущей инициацией. Это значит, что в асинхронных схемах интервал времени между сменой значений входных сигналов не может быть меньше, чем суммарная временная задержка элементов, участвующих в переходных процессах. В этом смысле такие схемы не имеют никаких достоинств перед синхронными. В задачу данной книги не входит сравнение преимуществ и недостатков обеих традиционных подходов. Важнее понять, как строить схемы, которые работают без тактовых генераторов и одновременно являются нечувствительными к вариациям временных параметров элементов. Соответствующий класс схем получил название самосинхронных схем.

6.1. Особенности самосинхронной схемотехники

Самосинхронные схемы адекватны базовым динамическим моделям. Для подтверждения этого факта требуется показать возможность стандартным способом аппаратно реализовать типовые фрагменты базовых моделей. Это и будет сделано ниже. Однако прежде следует разобраться в вопросе о том, как композировать типовые фрагменты. **Любое устройство можно представить состоящим из операционного и управляющего блоков. Это традиционное представление используется и в самосинхронном подходе**.

Будем считать, что самосинхронные схемы управляются посредством двух специальных сигналов: сигнала запроса (его часто называют фазовым сигналом), **который инициирует работу схемы, и сигнала подтверждения** (завершения, ответа), **фиксирующего окончание переходного процесса в схеме.** Сигнал подтверждения может инициировать работу другой схемы, так что одна схема будет управлять работой другой. Для повторной инициации схемы требуется снятие сигнала запуска (что аналогично паузе в синхронном случае), на который после окончания переходных процессов схема должна ответить снятием сигнала подтверждения.

Таким образом, работа самосинхронной схемы относительно пары управляющих сигналов (запуска и подтверждения) моделируется *элементом задержки*. Величина этой задержки конечна и заранее неизвестна, она зависит от величин задержек компонентов, т.е. элементов схемы. Координация совместной работы схем осуществляется только за счет их взаимодействия, определяемого способами соединения входов и выходов управляющих сигналов, а также задержками, вносимыми этими схемами.

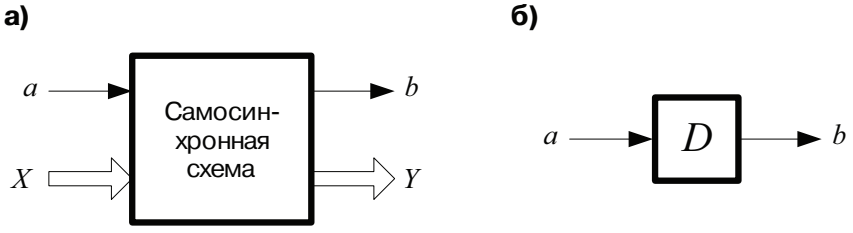


Рис. 6.1. а) Самосинхронная схема и б) ее эквивалент относительно управляющих сигналов

Типовая самосинхронная схема может быть представлена так, как показано на рис. 6.1,а, на котором через X и Y обозначены соответственно информационные входы и выходы, а через a и b – управляющие сигналы запроса и ответа (подтверждения). На самом деле наличие пары управляющих сигналов в самосинхронной схеме не является необходимым условием. **Роль управляющих сигналов совместно с информационными сигналами могут играть специальные кодовые наборы, образующие так называемые самосинхронные коды.** Множество наборов самосинхронного кода разбивается на два непересекающихся подмножества так, что наборы одного подмножества воспринимаются схемой как сигнал запуска (подтверждения) и готовности входной (выходной) информации, а наборы второго подмножества – как сигнал снятия запуска (подтверждения) и, возможно, гашения информации.

Из сказанного ясно, что **работа самосинхронной схемы осуществляется в две фазы: одна фаза, называемая *рабочей*, соответствует переходу схемы из одного состояния в некоторое другое финальное состояние, а вторая фаза (*фаза гашения*) подготавливает схему к повторному использованию.** *Двухфазная работа* является отличительной характеристикой большей части самосинхронных схем, но не является обязательной: существуют самосинхронные схемы, работающие в однофазном режиме, который будем называть режимом работы “в изменениях”.

В двухфазной самосинхронной схеме сигнал $\pm b$ (сигнал ответа) должен вырабатываться последним, чтобы он мог служить сигналом окончания переходных процессов в схеме, вызываемых изменениями сигнала $\pm a$ (сигнала запроса). Это накладывает определенные требования на конструкции самосинхронных схем, которые существенно отличаются от конструкций схем традиционной схемотехники. В рамках данного пособия невозможно подробно осветить эти требования, при желании читатель может самостоятельно познакомиться с соответствующей литературой. Важно лишь отметить, что относительно пары управляющих сигналов моделью самосинхронной схемы является элемент чистой задержки произвольной (заранее неизвестной) величины, включенный между сигналом запроса a и сигналом ответа b (см. рис. 6.1,б). Эта задержка не обязательно обладает фильтрующими свойствами, т.е. может не быть инерциальной.

Такой же запрос–ответный режим функционирования, что и для схемы (режим рукопожатия), должен выполняться и для любого ее элемента. Точнее: все комбинации входных наборов, встречающиеся при функционировании элемента, могут быть разделены на два непересекающихся класса. Одни на-

боры вызывают возбуждение элемента, находящегося, например, в состоянии 0, который через некоторый конечный промежуток времени переключается в состояние 1, а другие вызывают возбуждение этого элемента в состоянии 1 и его переход в состояние 0. Таким образом, каждый элемент самосинхронной схемы также функционирует как бы в режиме *хендшейка (режиме рукопожатия)*, переключаясь в цикле работы дважды (из 0 в 1 и из 1 в 0), если он задействован в этом цикле.

Режим рукопожатия является типовым и для операционных устройств, работающих в условиях самосинхронизации. Операционный блок тоже может моделироваться задержкой, как показано на рис. 6.1,б. Поэтому если в управляющей схеме последовательно с некоторым логическим элементом E с выходом x_i включить операционный блок A_i в соответствии с рис. 6.2,а, то этот блок будет вносить дополнительную задержку в сигнал x_i, а поскольку задержка элемента может быть произвольной (но конечной), то задержка блока A_i может быть приведена к выходу x_i элемента E. Таким образом, если имеется композиция операционного и управляющего устройств, то операционные блоки можно исключить из рассмотрения и иметь дело лишь с управляющей схемой, которая реализует заданное спецификацией упорядочение событий, состоящих в срабатываний элементов. Иными словами, так как управляющая схема моделирует совместное функционирование с операционными блоками, то ее можно синтезировать отдельно от операционных блоков. Композиция моделирующей схемы с операционными блоками осуществляется их включением своими хендшейк сигналами в разрывы выходных проводов элементов моделирующей схемы.

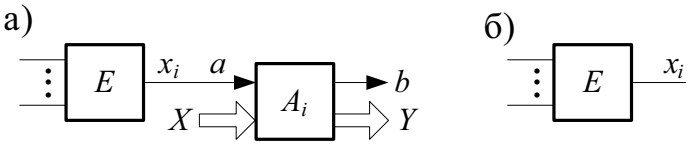


Рис. 6.2. а) Способ включения операционного блока и б) эквивалентное представление

Возможность выделения отдельного этапа проектирования управляющей схемы, осуществляющей координацию работы операционных блоков, является специфичной для самосинхронного подхода и делает его предпочтительным по сравнению с традиционным подходом, когда нельзя проектировать управляющую схему без учета характеристик операционных устройств и при изменении этих характеристик требуется перепроектировать управление.

Рассмотрение методов построения управляющих схем удобно начать с модели Маллера.

6.2. Синтез по модели Маллера

В динамической логике поведение асинхронной схемы задается системой уравнений. Если соответствующая этой системе диаграмма переходов полумодулярна относительно начального состояния, а в имеющемся базисе найдутся элементы, собственные функции которых покрывают все функции системы, то непосредственный переход от системы уравнений к схеме, ее реализующей, тривиален. Но если хотя бы одна из функций системы не соответствует возможностям базисных элементов, то такую функцию требуется представить в виде суперпозиции из элементов базиса. В результате получается схема, содержащая большее число элементов. Такой схеме соответствует другая диаграмма переходов, в которой свойство полумодулярности может быть нарушено.

Для корректного решения задачи синтеза необходимо, чтобы полученная диаграмма была расширением исходной, т.е. исходная диаграмма должны быть проекцией полученной диаграммы на множество переменных исходной диаграммы. Естественно потребовать, чтобы расширенная диаграмма была полумодулярной, ибо только такой класс диаграмм гарантирует независимость поведения схемы от задержек элементов.

Сначала рассмотрим операции над диаграммами, которые понадобятся в дальнейшем.

Редукция диаграммы переходов

Редукцией диаграммы переходов **по некоторой переменной x_i назовем ДП, полученную из исходной применением следующих операций:**

- **исключение из кодов состояний этой переменной;**
- **склеивание полученных кодов таким образом, что если некоторые коды отличаются разными возбужденными переменными (отмеченными звездочками), то в ДП оставляются лишь те из кодов, которые имеют максимальное число возбуждений;**
- **исключение лишних дуг в процессе склеивания.**

Например, редукция ДП, приведенной на рис. 6.3,а, по переменной x_2 сначала дает ДП, показанную на рис. 6.3,б, а после склеивания кодов и исключения дуг – последовательную ДП, показанную на рис. 6.3,в.

Операция редукции может быть произведена непосредственно по системе уравнений без использования ДП. Имеет место следующий факт. Если задана система уравнений вида $x_j = f_j(X) = f_j(x_1, x_2, ..., x_n), j = 1,2,...,n$, и известно, что некоторая переменная $x_i \in X$ *несамозависима*, т.е. $x_i = f_i(X \setminus x_i)$, то результат редукции получается путем исключения уравнения для x_i и замены в правых частях уравнений $x_j, j \neq i$, переменных x_i на f_i, т.е. путем выполнения свертки вида $x_j = f_j(x_1,...,x_{i-1}, f_i, x_{i+1},...,x_n)$, которую далее будем обозначать как $x_j = f_j(x_i = f_i)$. Например, диаграмме на рис. 6.3,а соответствует система уравнений:

$$x_1 = x_2x_3 \vee x_1(x_2 \vee x_3),\ x_2 = \overline{x}_1, x_3 = \overline{x}_1.$$

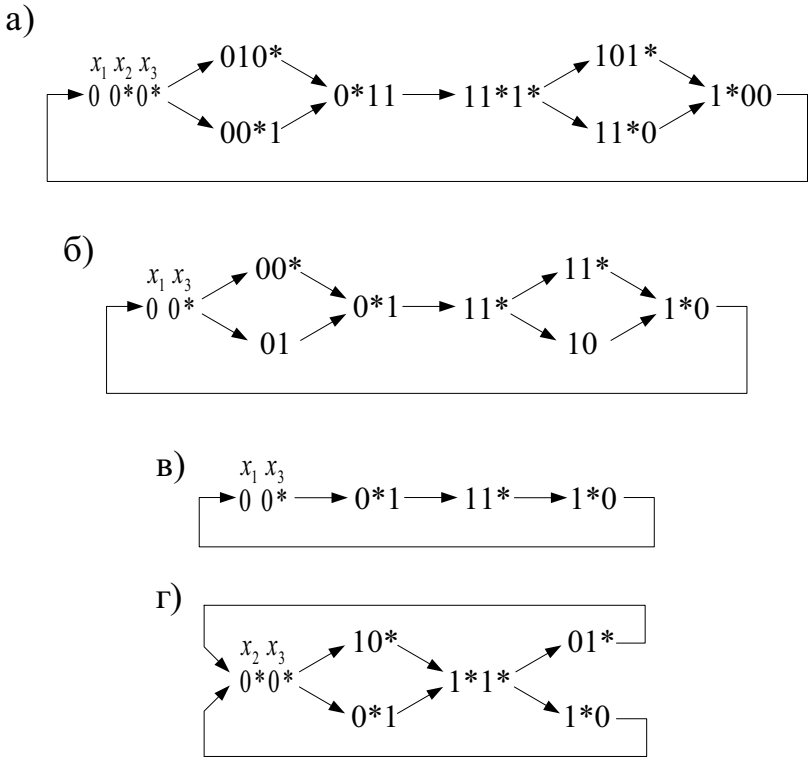


Рис. 6.3. а) Исходная диаграмма переходов, б) промежуточная редукция по x_2, в) окончательная редукция по x_2, г) результат редукции по x_1

Редукция по x_2 дает

$$x_1 = \bar{x}_1 x_3 \vee x_1(\bar{x}_1 \vee x_3) = x_3,\ x_3 = \bar{x}_1,$$

что совпадает с диаграммой переходов на рис. 6.3,в.

Теперь проведем редукцию диаграммы переходов на рис.6.3,а по *самозависимой* переменной x_1. Получится диаграмма переходов, показанная на рис. 6.3,г, которая содержит противоречивые состояния (1*0 и 10*, 0*1 и 01*) и поэтому не является полумодулярной.

Приведенные примеры свидетельствуют о следующем. **Если задана схема Маллера, т.е. схема, которой соответствует полумодулярная диаграмма переходов, то редукция по несамозависимой переменной также является схемой Маллера. Если же редукция производится по самозависимой переменной, то полученная в результате диаграмма обязательно содержит кратные состояния и является противоречивой.** В этом случае редукции нельзя поставить в соответствие никакую схему, поскольку соответствующая диаграмме таблица истинности не является булевой.

Кратные состояния

Множество кратных состояний, возникающее при исключении самозависимой переменной, можно определить аналитически. Если представить самозависимую переменную уравнением вида $x_i = S_i \vee x_i \overline{R}_i$, где S_i и R_i – функции переключения x_i из 0 в 1 и из 1 в 0 соответственно, то характеристическую функцию множества кратных состояний можно задать в виде $\overline{S}_i \overline{R}_i$. Так, для рассматриваемого примера $S_1 = x_2 x_3$, $\overline{R}_1 = x_2 \vee x_3$, $\overline{S}_1 \overline{R}_1 = \overline{x_2 x_3}(x_2 \vee x_3) = x_2 \bar{x}_3 \vee \bar{x}_2 x_3$. Поэтому кратными состояниями являются 10 и 01, что соответствует диаграмме переходов на рис. 6.3,г.

Проверка спецификации на полумодулярность

Смысл изложенных выше результатов состоит в следующем. Этапу синтеза самосинхронной схемы должна предшествовать проверка спецификации (системы уравнений) на полумодулярность. Эта процедура имеет экспоненциальную сложность по числу переменных, поэтому снижение размерности задачи может быть произведено редукцией по несамозависимым переменным, гарантирующей сохранение свойства полумодулярности, если это свойство имело место в исходной спецификации.

Проекции диаграмм переходов

Рассмотрим две диаграммы переходов. Первая $D(X)$ задана на множестве переменных X, вторая $D(Y)$ – на множестве переменных $Y = X \bigcup Z, Z \neq \varnothing$. Будем говорить, что диаграмма $D(X)$ является проекцией диаграммы $D(Y)$ на множество переменных X, если

1) каждой переменной $x_i \in X$ поставлена в соответствие одна и только одна переменная $y_j \in Y$,

2) упорядочение подмножеств состояний диаграммы $D(Y)$ в точности повторяет упорядочение соответствующих состояний диаграммы $D(X)$; $D(Y)$ будем называть *расширением диаграммы* $D(X)$.

Из определения проекции ясно, что каждому состоянию из $D(X)$ соответствует некоторое подмножество состояний из $D(Y)$. Например, проекции диаграммы, приведенной на рис. 6.4,а на подмножества переменных $X\backslash x_3$, $X\backslash x_2$, $X\backslash x_1$ и $X\backslash x_4$ изображены на рис. 6.4,б, в, г и д соответственно.

Обратим внимание на два обстоятельства:

- **Первое обстоятельство.** Исходная диаграмма является полумодулярной (диаграммой Маллера), а переменные x_1, x_2 и x_3 – несамозависимыми. Поэтому диаграммы на рис. 6.4,б–г являются диаграммами Маллера. Напротив, переменная x_4 самозависима и поэтому редукция на рис. 6.4,д не является диаграммой Маллера. Она содержит кратные состояния и поэтому противоречива.

- **Второе обстоятельство.** Диаграмма на рис. 6.4,а порождает таблицу истинности переменных, после доопределения которой получим систему: $x_1 = \overline{x}_4$, $x_2 = x_1 \vee x_3 \vee \overline{x}_4$, $x_3 = x_1\overline{x}_4 \vee x_2\overline{x}_4$, $x_4 = x_1x_2x_3 \vee x_3x_4$. Редукция системы по x_2 приводит к системе: $x_1 = \overline{x}_4$, $x_3 = x_1\overline{x}_4$, $x_4 = x_1(x_1 \vee x_3 \vee \overline{x}_4)x_3 \vee x_3x_4 = x_1x_3 \vee x_3x_4$. В то же время, если выписать уравнение для x_4 непосредственно по диаграмме рис. 6.4,в, то после доопределения таблицы истинности получим другое уравнение: $x_4 = x_3 \vee x_1x_4$. Ошибки здесь нет. Дело в том, что ДП рис. 6.4,а содержит 12 состояний рабочего цикла из $2^4 = 16$ возможных, так что возможны $2^{16-12} = 16$ различных доопределений. Приведенная выше система фиксирует только одно из них и поэтому игнорируются доопределения, возможные при работе с редуцированной по x_2 диаграммой. Лишь одно из доопределений редукции должно совпадать с выражением, полученным алгебраическими манипуляциями.

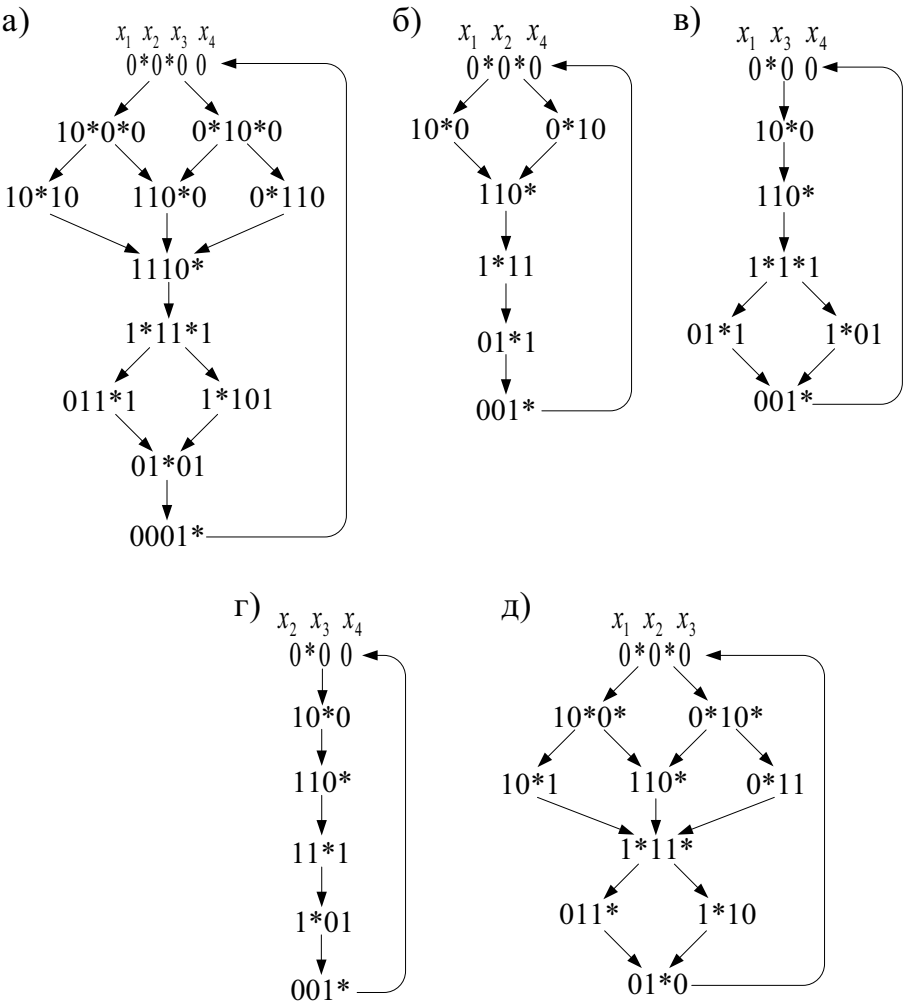


Рис. 6.4. Проекции а) исходной диаграммы переходов на подмножества переменных: б) {x_1, x_2, x_4}, в) {x_1, x_3, x_4}, г) {x_2, x_3, x_4}, д) {x_1, x_2, x_3}

Устранение противоречивости диаграмм переходов

Диаграмма на рис.6.4,а является расширением диаграмм на рис. 6.4,б–д. Однако, если исходные спецификации были бы заданы в виде диаграмм на рис. 6.4,б–г, то не было бы необходимости в их расширении. Другое дело в случае, если исходная спецификация была бы задана диаграммой на рис. 6.4,д. Эта диаграмма противоречива, поэтому исходным этапом синтеза самосин-

хронной схемы, ее реализующей (моделирующей), должно быть расширение диаграммы до непротиворечивой, которое в данном случае осуществляется с помощью одной (дополнительной) переменной x_4. Более того, в общем случае трудно ответить на вопрос, сколько дополнительных переменных следует ввести, чтобы получить искомое решение, и как вводить. Разработаны достаточно сложные методы введения минимального числа дополнительных переменных, реализованные в системе автоматизации проектирования асинхронных схем PETRIFY. Здесь эти методы рассматриваться не будут. Заметим лишь, что введение минимального числа дополнительных переменных не гарантирует минимальной сложности полученного решения. Часто лучшие решения могут быть получены, если процедуру расширения противоречивой диаграммы выполнять с учетом семантики задачи, что весьма трудно поддается формализации.

Свойства логических переменных полумодулярных диаграмм переходов

Обратим внимание на следующий факт. Для полумодулярной диаграммы переходов любая переменная $x_j = f_j(X)$ является либо изотонной[1] по x_j, т.е. может быть представлена в виде $x_j = \alpha_j \vee \beta_j x_j$, где α_j и β_j – функции, существенно не зависящие от x_j, либо несамозависимой, т.е. функция $f_j(X)$ существенно не зависит от переменной x_j. Действительно, разложение Шеннона по переменной x_j для функции $f_j(X)$ имеет вид: $x_j = \bar{x}_j f_j(x_j = 0) \vee x_j f_j(x_j = 1)$ или $x_j = \bar{x}_j \alpha_j \vee x_j \beta_j$.

- Если $\alpha_j = 0$, $\beta_j = 0$, то $x_j = 0$ и не зависит от себя.
- Если $\alpha_j = 1$, $\beta_j = 1$, то $x_j = 1$ и от себя не зависит.
- Если $\alpha_j = 0$, $\beta_j = 1$, то $x_j = x_j$, т.е. является изотонной относительно себя.
- Если $\alpha_j = 1$, $\beta_j = 0$, то $x_j = \bar{x}_j$, т.е. x_j не имеет невозбужденного состояния, что противоречит свойству полумодулярности.

Стандартные реализации самозависимых переменных

Нетрудно заметить сходство функции $x_j = \alpha_j \vee \beta_j x_j$ с известными уравнениями триггера с раздельными входами $x_j = S_j \vee \overline{R}_j x_j$, $S_j R_j = 0$. Это

[1] Понятия “изотонная” и “антитонная” по отношению к булевой функции синонимичны общепринятым терминам “неубывающая” и “невозрастающая” соответственно.

сходство указывает на то, что полумодулярная схема может быть реализована на триггерах.

Из общих соображений ясно, что в функцию самозависимой переменной $x_k = f_k(X)$, которой ставится в соответствие триггер на элементах И-НЕ или ИЛИ-НЕ (см. рис. 6.5,а), некоторая переменная x_j может входить как в прямом виде, так и в инверсном ($\overline{x}_j$). Казалось бы, что инверсное или прямое значение переменной x_j можно получить с помощью инвертора, как это показано на рис. 6.5,б. Однако это заблуждение. Диаграммы переходов для пары сигналов x_j, $\overline{x}_j$, один из которых реализован с помощью инвертора, могут принадлежать одному из двух типов, показанных на рис. 6.5,в. Из этих диаграмм видно, что возможны транзитные состояния двух видов 00 и 11. Это недопустимо, так как такие состояния могут явиться причиной того, что функции возбуждения S и R триггера, соответствующего переменной x_k, перестанет удовлетворять условию $SR = 0$ и триггер как элемент памяти перестанет существовать.

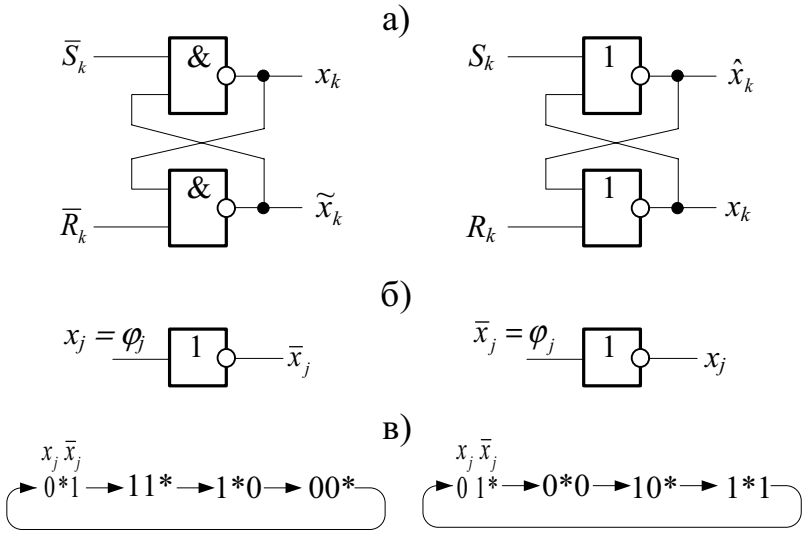


Рис. 6.5. а) Триггер на элементах И-НЕ и ИЛИ-НЕ, б) использование инверторов для инвертирования переменных, в) диаграммы переходов для пар взаимоинверсных переменных

Триггеры на элементах И-НЕ и ИЛИ-НЕ имеют разные транзитные состояния переключения соответственно 11 и 00 (см. рис. 6.5,а). Инверсное плечо триггера обозначается переменной, над которой стоит знак “тильда”, если триггер переключается через транзитное состояние 11, и знак “крыш-

ка", если он переключается через транзитное состояние 00. Очевидно, что в стандартных (канонических) реализациях имеет смысл пользоваться триггерами одного типа. Про такие реализации говорят, что они являются монотранзитными.

В *монотранзитных реализациях* транзитные состояния 00 или 11 одного триггера не воспринимаются другими триггерами до тех пор, пока этот триггер не перейдет в устойчивое состояние 10 или 01.

Метод совершенной реализации схемы Маллера

В общем случае функции S_j и R_j не могут быть реализованы непосредственно на простых элементах, образующих триггер, поскольку они представлены нормальными формами, имеющими более одного конъюнктивного или дизъюнктивного члена. Однако эти функции могут быть реализованы, если триггер образуют сложные элементы И-ИЛИ-НЕ с неограниченным числом входов и произвольной нагрузочной способностью.

Однако и здесь требуется применять специальные меры, препятствующие возможности возникновения состязаний.

Рассмотри пример. Пусть схема задана системой уравнений:

$$z_1 = z_2\bar{z}_4 \vee \bar{z}_2\bar{z}_3,\ z_2 = \bar{z}_3\bar{z}_4 \vee z_3z_4 \vee z_2(\bar{z}_1z_3 \vee z_1\bar{z}_3),$$
$$z_3 = z_2\bar{z}_4 \vee \bar{z}_2z_3,\ z_4 = z_2z_4 \vee \bar{z}_2z_3.$$

Система уравнений правильного расширения этой схемы имеет вид:

$$z_1 = \overline{\text{£}_1 \vee z_2z_4 \vee \text{£}_2z_3 \vee z_3z_4},\quad \text{£}_1 = \overline{z_1 \vee z_2\text{£}_4 \vee \text{£}_2\text{£}_3 \vee \text{£}_3\text{£}_4},$$
$$z_2 = \overline{\text{£}_2 \vee z_1z_3\text{£}_4 \vee \text{£}_1\text{£}_3z_4},\qquad \text{£}_2 = \overline{z_2 \vee \text{£}_3\text{£}_4 \vee z_3z_4},$$
$$z_3 = \overline{\text{£}_3 \vee z_3z_4},\ \text{£}_3 = \overline{z_3 \vee z_2\text{£}_4},\ z_4 = \overline{\text{£}_4 \vee \text{£}_2\text{£}_3},\ \text{£}_4 = \overline{z_4 \vee \text{£}_2z_3}.$$

Обратим внимание на избыточность реализации, скажем, триггера с выходами z_1 и £_1. Минимальными формами для S_1 и R_1 являются

$$S_1 = z_2\text{£}_4 \vee \text{£}_2\text{£}_3,\quad R_1 = z_2z_4 \vee \text{£}_2z_3.$$

Однако, когда триггер, соответствующий переменной z_2, проходит транзитное состояние $z_2 = \text{£}_2 = 0$, как можно убедиться, построив диаграмму переходов, возникают состязания, связанные со снятием возбуждения элемента с выходом z_1.

Для борьбы с такими состязаниями может быть использован стандартный прием, состоящий в замене минимальной д.н.ф. каждой функции возбуждения ее сокращенной формой представления, что позволяет осуществить так называемый перехват термов за счет введения в д.н.ф. дополнительных импликант. Такой способ построения схем называют совершенной реализацией.

Он позволяет построить в неограниченном базисе И-ИЛИ-НЕ самосинхронную схему, диаграмма переходов которой является расширением исходной полумодулярной диаграммы путем, в общем случае, удвоения числа переменных.

Базовая конструкция для совершенной реализации на элементах И-ИЛИ-НЕ представлена на рис. 6.6. Функции возбуждения S_i и R_i представляются сокращенными д.н.ф.

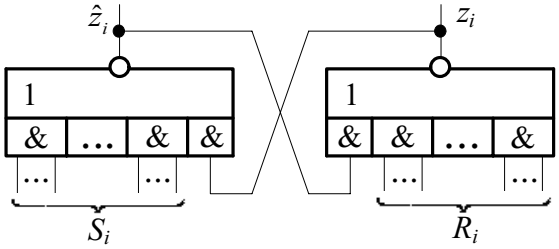


Рис. 6.6. Базовая конструкция для совершенной реализации

Пример. По полумодулярной ДП, представленной на рис. 6.7,а, получим систему уравнений:

$$x_1 = \bar{x}_4,\ x_2 = \bar{x}_4,$$
$$x_3 = (x_1 \vee x_2)\bar{x}_4 \vee (x_1 x_2 \vee \bar{x}_4)x_3,$$
$$x_4 = x_1 x_2 x_3 \vee (x_1 \vee x_2 \vee x_3)x_4.$$

Приведем их к виду, соответствующему сокращенным д.н.ф. функций возбуждения:

$$x_1 = \bar{x}_4 \vee \bar{x}_4 x_1,$$
$$x_2 = \bar{x}_4 \vee \bar{x}_4 x_2,$$
$$x_3 = x_1\bar{x}_4 \vee x_2\bar{x}_4 \vee \overline{\bar{x}_1 x_4 \vee \bar{x}_2 x_4} \cdot x_3,$$
$$x_4 = x_1 x_2 x_3 \vee \overline{\bar{x}_1\bar{x}_2\bar{x}_3} \cdot x_4.$$

Совершенная реализация этой системы на триггерах типа рис. 6.6 приведена на рис. 6.7,б.

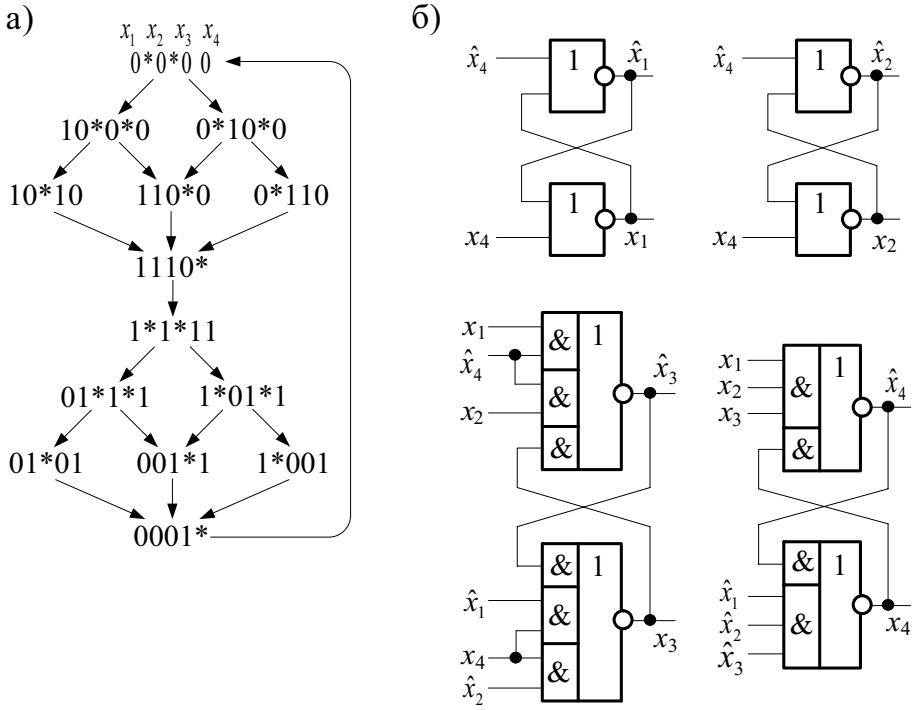


Рис. 6.7. а) Диаграмма переходов и б) ее совершенная реализация

Как всякий универсальный подход, совершенная реализация имеет завышенную сложность. Оценка сложности от числа n переменных исходной полумодулярной диаграммы составляет $2n$, в то время как, по-видимому, должны существовать полумодулярные схемы с оценкой сложности по числу элементов порядка $n + \log_2 n$. Однако метод синтеза самосинхронных схем с такой оценкой сложности неизвестен.

Эти оценки не учитывают ограничений по числу входов и нагрузочной способности, накладываемых на систему элементов технологиями изготовления реальных микросхем. Поэтому возникает вопрос о минимальном базисе, позволяющем реализовать произвольные полумодулярные схемы. Ответ на этот вопрос получен и состоит в следующем.

Минимальный базис реализации полумодулярных схем

Любую полумодулярную схему можно реализовать в базисе, состоящем из двух элементов {2И-НЕ2, 2ИЛИ-НЕ2}, а любую дистрибутивную схему – в базисе 2И-НЕ2 или в двойственном базисе 2ИЛИ-НЕ2.

Доказательства этих положений достаточно сложны и опираются на громоздкие конструкции, но факт их существования дает разработчику уверенность в результативности процесса синтеза. Несмотря на то, что минимальный базис существует, строить самосинхронные схемы в этом базисе не имеет особого практического смысла ввиду чрезвычайно высокой сложности получающихся схем. С увеличением функциональных возможностей базиса значительно снижается сложность самосинхронных схем. В современной КМОП технологии допустимая сложность однокаскадных элементов не превышает 4И-4ИЛИ-НЕ. Именно такие элементы наиболее целесообразно использовать при проектировании самосинхронных схем. И тем не менее их сложность в 3 – 4 раза может превышать сложность синхронных реализаций.

6.3. Стандартные реализации фрагментов сетей Петри

Подавляющая часть работ по сетям Петри "обслуживает" проблематику общей теории систем и параллельного программирования, но часть вопросов теории сетей Петри, связанных с выделением подкласса живых и безопасных сетей и их исследованием, близка к концепции самосинхронизации, причем несомненным достоинством сетевого подхода является возможность перехода от сети из указанного подкласса к самосинхронной схеме в обход процедуры кодирования.

Говоря о реализации сетей Петри, соблазнительно иметь **правила прямой трансляции их типовых фрагментов в соответствующие схемные модули.** Такие правила существуют.

Сначала рассмотрим их для случая неинтерпретированных СП, когда объектом реализации являются фрагменты устойчивых и безопасных СП, показанные на рис. 6.8.

Фрагменты, изображенные на рис. 6.8,а–г, особых пояснений не требуют. Напомним лишь, что в силу безопасности исходных СП в позициях не может быть более одного маркера.

Фрагмент на рис. 6.8,д осуществляет *функцию синхронизации* входных сигналов. Если обозначить входные сигналы через x_1 и x_2, а выходной – через y, то этому фрагменту должна соответствовать собственная функция $y = x_1 x_2 \vee (x_1 \vee x_2) y$. Выход y принимает значение 1 при $x_1 = x_2 = 1$ и значение 0 при $x_1 = x_2 = 0$. Во всех остальных случаях, когда $x_1 \neq x_2$, выход y сохраняет предыдущее значение. Триггер такого типа обычно называют

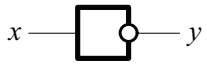
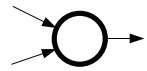
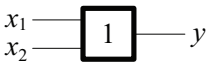
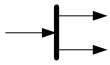
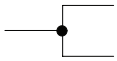
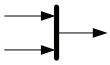
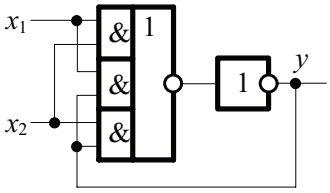
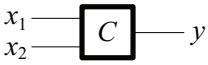
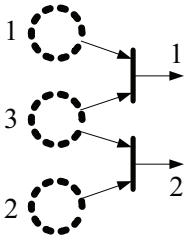
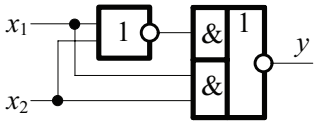
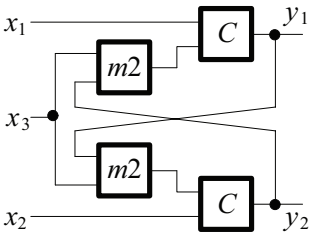
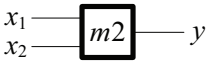
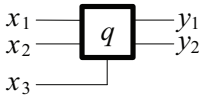
№ п.п.	Фрагмент сети Петри	Модуль	Условное обозначение
а)		провод	—
б)			—
в)			—
г)			—
д)			
е)		 	 

Рис. 6.8. Реализация "малых" фрагментов сети Петри

С-элементом Маллера. Его условное обозначение и реализация на элементе 2И-3ИЛИ-НЕ и инверторе представлены на рис. 6.8,д. Здесь требуется пояснить, почему переход такого типа моделируется *С*-элементом. Дело в том, что срабатыванию перехода ставится в соответствие событие переключения

триггера, а не значение, которое принимает его выход. При этом предполагается, что при срабатывании перехода изымаются маркеры из его входных условий. Этот факт не виден на модельном уровне, так как переменные x_1 и x_2 сохраняют свои значения. Повторное появление маркеров во всех его входных условиях соответствует изменению значений переменных x_1 и x_2 на противоположные, что должно вызывать переключение триггера в противоположное состояние. Очевидно, что такой триггер представляет собой *C*-элемент Маллера.

Во фрагменте СП на рис. 6.8,е инцидентные ему входные позиции не относятся собственно к самому фрагменту и потому обозначены пунктиром. Кроме того в позициях 1 и 2 не разрешается одновременное нахождение маркеров. Реализация фрагмента опирается на две схемы: схему *C*-элемента и схему сумматора по модулю 2 (обозначенную через *m*2). Если входами сумматора являются x_1 и x_2, а выходом y, то реализуется функция $y = \overline{\overline{x_1 \vee x_2} \vee x_1 x_2} = x_1 \oplus x_2$. Общая схема модуля, представляющего данный фрагмент СП, реализует две самозависимые функции y_1 и y_2. Функции y_1 и y_2 симметричны друг другу и имеют соответственно вид $y_1 = x_1(x_3 \oplus y_2) \vee (x_1 \vee x_3 \oplus y_2)y_1$ и $y_2 = x_2(x_3 \oplus y_1) \vee (x_2 \vee x_3 \oplus y_1)y_2$.

Пусть при начальных значениях входных сигналов $x_1 = x_2 = x_3 = 0$ выходы модуля имеют значения $y_1 = y_2 = 0$. При появлении маркера в позиции 3 вход x_3 принимает значение 1 и срабатывают оба сумматора (их выходы становится равным 1), но сигналы на выходах y_1 и y_2 не изменяют своих значений. И только при комбинации $x_1 = x_3 = 1$ ($x_2 = x_3 = 1$) срабатывает (переключается в 1) выход y_1 (y_2). Одновременное изменение x_1 и x_2 недопустимо. Допустим, что переключился выход y_1 ($y_1 = 1$, а $y_2 = 0$). Следующими комбинациями, вызывающими переключение одного из выходов схемы, могут быть либо $x_1 = x_3 = 0$, либо $x_2 = 1, x_3 = 0$. Первая из них приводит к появлению 0 на выходе y_1, а вторая – к появлению 1 на выходе y_2. В дальнейшем изменение значений каждой пары сигналов x_1, x_3 или x_2, x_3 вызывает изменение значения одного из выходов y_1 или y_2 соответственно.

Итак, безопасную и устойчивую СП можно реализовать, поставив в соответствие ее фрагментам модули, представленные на рис. 6.8, и соединив эти модули так, как это предписывается структурой СП с учетом начальной маркировки. Однако требуется решить вопрос о том, как быть в тех случаях, когда позиции и переходы имеют большее число входов и выходов, чем указано в типовых фрагментах. В частности, это относится к *C*-элементу.

Реализация многовходовых С-элементов Маллера

С-элемент с *n* входами может быть реализован как *пирамидальная схема* из двухвходовых *С*-элементов или, что более эффективно, посредством так называемой *схемы параллельного сжатия*, показанной на рис. 6.9, в которой используется двухвходовый *С*-элемент и два n-входовых элемента И и ИЛИ. В свою очередь, каждый из этих элементов при ограничениях на число входов может быть представлен пирамидальной схемой из элементов И и ИЛИ с заданным числом входов.

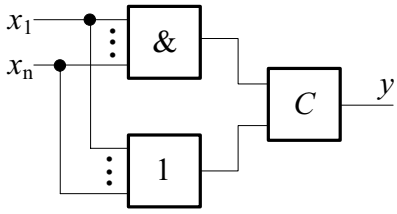


Рис. 6.9. Схема параллельного сжатия

Теперь обратимся к методам реализации СП, основанные на моделировании продвижения маркеров в сети.

Моделирование продвижения маркера в сети Петри элементом Давида

В сети Петри срабатывание перехода считается неделимой операцией: маркеры изымаются из входных позиций этого перехода и помещаются в его выходные позиции строго одновременно. В схемной реализации это требование непосредственно не может быть выполнено, так как в физической природе не существует понятия одновременности и возможно говорить лишь об одновременности с некоторой точностью. Если отказаться от свойства неделимости операции срабатывания перехода, то последовательность продвижения маркера в простейшем устойчивом и безопасном фрагменте сети Петри (рис. 6.10,а) можно представить так, как это показано на рис. 6.10,б. Наличие маркера во входной позиции возбуждает переход и после его срабатывания в моделирующей схеме сначала должна возникнуть транзитная ситуация, связанная с появлением маркера в выходной позиции p_2, после чего может быть изъят маркер из входной позиции p_1. Далее вместо неделимости операции вводится критический интервал с помощью дополнительного ограничения: следующий маркер во входной позиции может появиться только в случае изъятия маркера из выходной позиции. Ди-

аграмма маркировок, соответствующая измененным правилам срабатывания перехода, представлена на рис. 6.10,в.

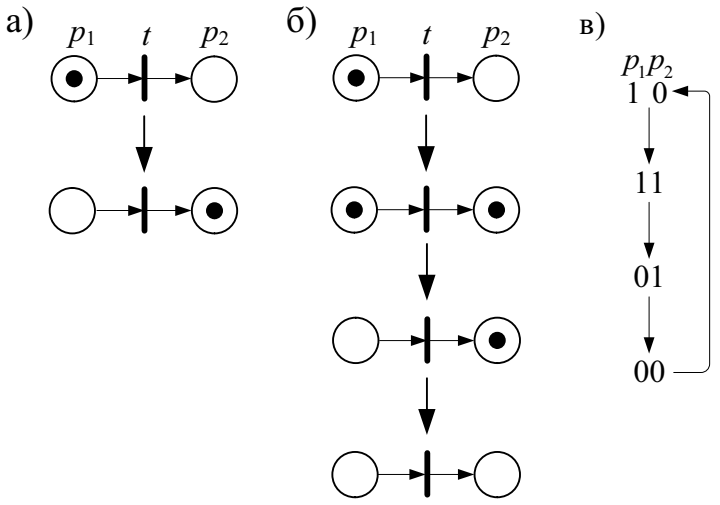


Рис. 6.10. Правило срабатывания перехода для элементарного фрагмента сети Петри: а) удовлетворяющее свойству неделимости , б) интерпретация при отсутствии этого свойства, в) соответствующая диаграмма маркировок

Рассмотрим простейшую сеть Петри, представленную на рис. 6.11,а. Выделим в ней фрагмент, очерченный пунктиром и состоящий из одной позиции и одного перехода. Воспользуемся следующей интерпретацией позиции и перехода этого фрагмента, при которой им поставлены в соответствие изменения значения некоторого сигнала (переменной) моделирующей схемы, а именно:

а) появлению маркера в позициях p_1 и p_2 моделируется соответственно сигналами $+p_1$ и $+p_2$;

б) изъятию маркера из позиций p_1 и p_2 соответствует появление сигналов $-p_1$ и $-p_{2;}$

в) возбужденным переходам t_1 и t_2 соответствуют $t_1 = 0^*$ и $t_2 = 0^*$;

г) срабатывание возбужденного перехода t_1 или t_2 моделируется соответствующими сигналами $+t_1$ или $+t_2$.

В таком случае фрагменту СП на рис. 6.11,а будет соответствовать диаграмма переходов для указанных сигналов, представленная на рис. 6.11,в. Из та-

блицы истинности, построенной по этой диаграмме, с учетом возможных доопределений нетрудно найти минимальные выражения для переменных p_1 и t_2: $p_1 = t_1\overline{p}_2 \vee p_1\overline{p}_2$, $t_2 = p_1\overline{t}_1$.

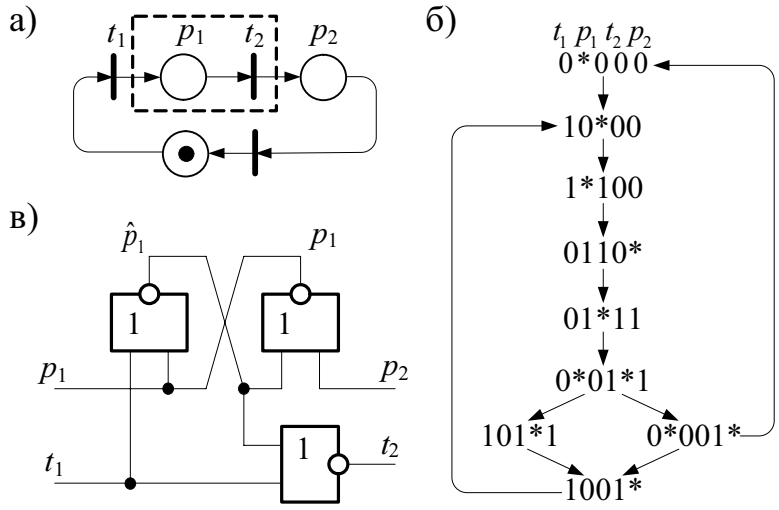


Рис. 6.11. а) Фрагмент сети Петри, б) его диаграмма переходов и в) элемент Давида, моделирующий этот фрагмент

Переменная p_1 является самозависимой, следовательно, она может быть реализована с помощью простого триггера, функции плеч которого имеют вид: $p_1 = \overline{\hat{p}_1 \vee p_2}$, $\hat{p}_1 = \overline{p_1 \vee t_1}$. Действительно, если подставить выражение для второго плеча в выражение для первого плеча, то при использовании преобразований де Моргана нетрудно получить исходное выражение для переменной p_1. Переменная t_2 может быть реализована как $t_2 = \overline{\hat{p}_1 \vee t_1}$. В результате получим схему (рис. 6.11,в), моделирующую продвижение маркера и срабатывание перехода для простейшего фрагмента сети Петри. Эта схема получила название *ячейки Давида* в честь французского ученого, который предложил несколько более сложную схему, содержащую 4 вентиля. В дальнейшем схема Давида была упрощена до трех вентилей.

Итак, **в схеме Давида условие моделируется состоянием триггера, а переход – состоянием выхода вентиля.** Как нетрудно видеть, ячейка Давида взаимодействует с соседними ячейками, которые могут быть соединены с ней слева и справа, по принципу хендшейка (запрос – ответ). Началом критического интервала является появление маркера в выходной позиции

перехода (переключение триггера соседней справа ячейки). Нетрудно видеть, что пока этот триггер не вернется в исходное состояние (маркер исчезает в выходном условии) в триггер ячейки Давида не может быть записана единица (не может появиться маркер во входной позиции). При попытке записи единицы в этот триггер оба его плеча будут находиться в состоянии 00 до тех пор, пока не переключится в состояние 10 триггер соседней справа ячейки (конец критического интервала), и только после этого в данный триггер будет записана единица (состояние 01).

Чаще всего используется схема, двойственная схеме, изображенной на рис 6.11.в. Такая схема построена из элементов 2И-НЕ (см. рис. 6.12,а) и использует отрицательную логику, в которой активным значением сигнала является 0. Причина предпочтения элементов И-НЕ элементам ИЛИ-НЕ заключается в том, что они занимают меньшую площадь кристалла и работают быстрее.

Цепочка (или кольцо) из ячеек Давида по сути дела моделирует последовательный процесс в регистре сдвига, в котором на фоне "пустой" информации перемещается порция данных, например, единица на фоне нулей.

Прямая трансляция сетей Петри в логические схемы

На основе ячейки Давида можно построить схемы, моделирующие работу типовых фрагментов устойчивых и безопасных сетей Петри, как это показано на рис. 6.12. Схемы, представленные на этом рисунке, соответствуют частям фрагментов, обведенным пунктирными линиями.

На рис. 6.12,б представлен фрагмент синхронизатора и моделирующая его схема. Она состоит из двух ячеек Давида и *С*-элемента, выход которого представляет соответствующей переход фрагмента. Фрагмент, который будем называть разветвителем, изображен на рис. 6.12,в. Его схема также содержит два элемента Давида и С-элемент, формирующий сигнал ответа на срабатывание перехода в предыдущей схеме. Приведенные реализации фрагментов не являются минимальными. Они ориентированы на легкое расширение при увеличении числа дуг, инцидентных моделируемым вершинам. Это расширение заключается в увеличении числа элементов Давида и соответственно числа входов С-элементов. Если число инцидентных дуг не превышает двух, то эти схемы могут быть упрощены.

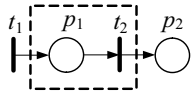
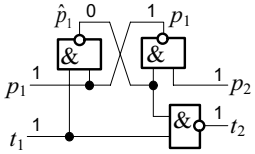
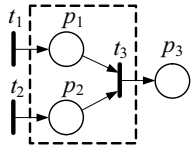
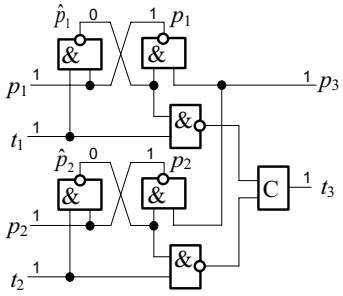
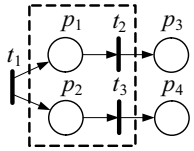
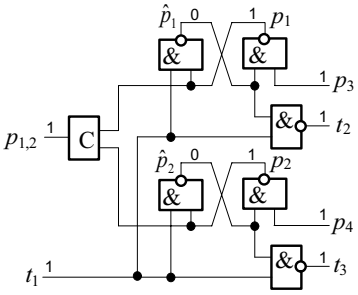
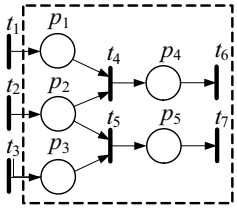
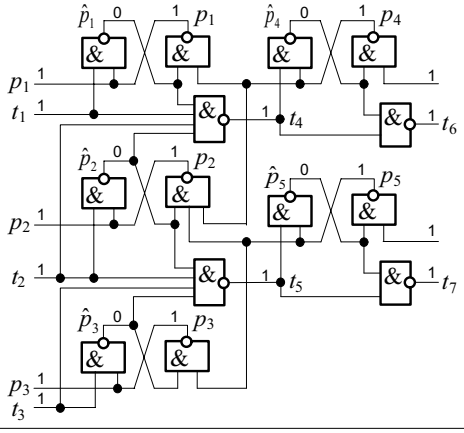
№ п.п.	Фрагмент сети Петри	Реализация
а)		
б)		
в)		
г)		

Рис. 6.12. Реализация фрагментов сети Петри на элементах Давида: а) элемент Давида, б) синхронизатор, в) разветвитель, г) альтернативный разветвитель (условный переход)

Фрагмент, представленный на рис. 6.12,г, будем называть альтернативным разветвителем, или условным переходом. Исходная маркировка в этом фрагменте должна сохранять устойчивость, т.е. входные позиции 1 и 3 должны быть взаимоинверсны. Схема, реализующая этот фрагмент, содержит два совмещенных по одному из условий синхронизатора и иллюстрирует возможность их упрощения. Оба синхронизатора содержат два триггера p_1, p_2 и p_2, p_3 (триггер p_2 является совмещенным) и синхронизируют события t_1, t_2 и t_2, t_3 на выходах t_4 и t_5 элементов 4И-НЕ соответственно. *С*-элементы для таких синхронизаторов не нужны, так как их функции выполняются элементами с выходами t_4 и t_5. Кроме указанных синхронизаторов в состав схемы фрагмента входят два выходных элемента Давида.

Схема двухвходового разветвителя (рис. 6.12,в) тоже может быть упрощена путем удаления С-элемента и заменой правого плеча триггера предыдущей ячейки Давида элементом 2И-2ИЛИ-НЕ, один вход каждого элемента И которого, используется для триггерной связи, а на вторые входы этих элементов поступают сигналы ответа от триггеров разветвителя.

На основе представленных реализаций нетрудно построить схемы и других возможных фрагментов, например, схему моделирующую поведение фрагмента, представляющего собой совмещение синхронизатора и разветвителя.

Легко понять, что произвольная устойчивая и безопасная сеть Петри строится как параллельно-последовательная композиция схем, реализующих выделенные фрагменты. Однако при такой композиции могут возникнуть определенные трудности, связанные с заменой неделимости операции срабатывания перехода критическим интервалом. Например, сеть Петри не должна содержать циклы, содержащие менее трех позиций и трех переходов, так как два последовательно соединенных элемента Давида при замыкании их в цикл попадают в устойчивое состояние, т.е. тупик. Это нетрудно установить, рассмотрев соответствующую схему. Для устранения этого явления необходимо введение фиктивных позиций и переходов с таким расчетом, чтобы в реализуемых сетях Петри не встречались циклы длины два.

Идея распараллеливания процессов осуществляется путем перехода от исходных представлений сетей Петри к их конвейерным аналогам. Однако схемное моделирование сетей Петри на основе ячеек Давида допускает конвейеризацию процессов на основе исходного представления сетей Петри без перехода к их конвейерным аналогам, поскольку ячейка Давида содержит критический интервал, разрешающий конвейерную работу, и поэ-

тому моделирующую схему, можно рассматривать как конвейерную. Заметим, что на этом пути следует ожидать много трудностей, которые требуют глубокого анализа.

Минимальный базис реализации сетей Петри методом прямой трансляции

Рассмотренный подход требует в общем случае неограниченного схемного базиса с неограниченной нагрузочной способностью. Весьма важным является вопрос о минимальном схемном базисе, поскольку он связан с возможностью реализации самосинхронных схем в разных интегральных технологиях. По-видимому, не требуется доказательства, что **любая устойчивая и безопасная сеть Петри может быть расширена введением новых позиций и переходов до вида, в котором степени всех вершин сети не превышают двойки**. Очевидно также, что типовые фрагменты такой сети, представленные на рис. 6.12, могут быть реализованы в базисе из элементов 2И-НЕ и двухвходового *С*-элемента. Таким образом, чтобы ответить на вопрос о минимальном базисе, необходимо рассмотреть возможность реализации двухвходового *С*-элемента из простых логических элементов.

Если выход двухвходового С-элемента замкнуть на его входы через два инвертора и построить диаграмму переходов для такой автономной схемы, то диаграмма окажется дистрибутивной. Следовательно, возможно реализация этой диаграммы в базисе 2И-НЕ (или 2ИЛИ-НЕ), однако в такой реализации входные сигналы имеют парафазное представление. **Реализация разомкнутой схемы, когда входные сигналы представлены однофазно, в этом базисе оказывается невозможной.**

Самосинхронная схема двухвходового *С*-элемента, построенная из элементов 2И-НЕ и 2ИЛИ-НЕ, приведена на рис. 6.13,а. Входно-выходное поведение этой схемы можно описать сигнальным графом, представленным на рис. 6.13,б. На этом рисунке пунктирными стрелками изображены связи, реализуемые во внешней среде. Из рис. 6.13,а видно, что необходимо присутствие хотя бы одного элемента 2ИЛИ-НЕ. Следовательно, минимальный базис для схемного моделирования устойчивых и безопасных сетей Петри состоит из двух элементов: 2И-НЕ и 2ИЛИ-НЕ.

Интерпретации сетей Петри, в которых игнорируются требование неделимости процесса срабатывания ее переходов, возможны, по-видимому, только для класса устойчивых и безопасных сетей.

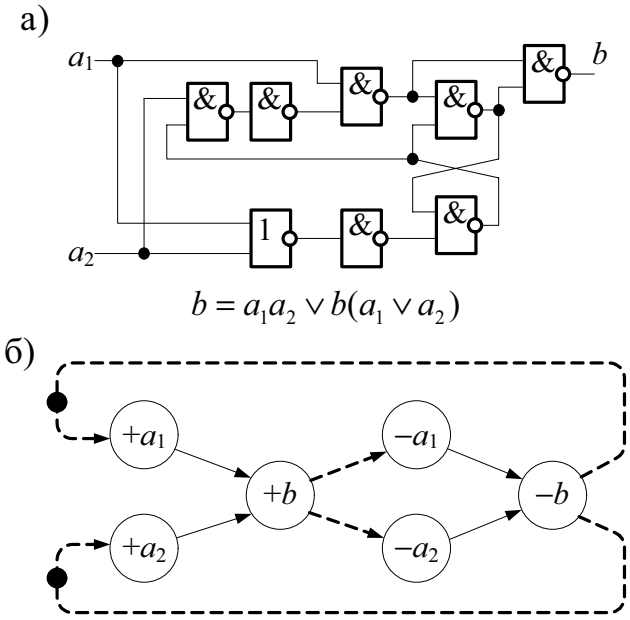


Рис. 6.13. Реализация из простых элементов двухвходового С-элемента

6.4. Принципы композиции

В реализациях типовых фрагментов СП на ячейках Давида рассматривался случай неинтерпретированных сетей. Имеется в виду, что на маркеры “не навешивалась” никакая сигнальная интерпретация.

Ранее говорилось, что самосинхронный операционный блок моделируется элементом задержки, включенным между проводами запроса и ответа. Поэтому в классе самосинхронных реализаций оказывается возможной нетрадиционная композиция управляющей (полумодулярной!) схемы и операционных блоков, в которой эти блоки включаются в разрывы проводов элементов управляющей схемы.

Тривиальный пример, иллюстрирующий стандартный подход к композиции самосинхронных блоков и управляющей схемы, представлен на рис. 6.14. Самосинхронные блоки A_1 и A_2 с сигналами запроса a_1, a_2 и сигналами ответа b_1, b_2 включены в разрывы проводов, соединяющих выходы инверто-

ров с входами *C*-элемента. Возможность такого включения следует из высказанных выше соображений и полумодулярности диаграммы переходов, соответствующей схеме, состоящей из *C*-элемента и двух инверторов.

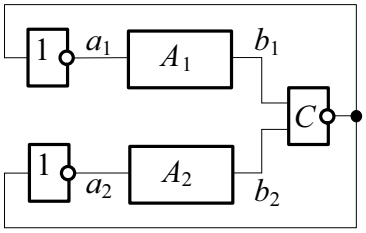


Рис. 6.14. Композиция управляющей схемы и операционных блоков

6.4.1. Дополнительные модули

При композиции самосинхронных схем управления и операционных блоков возникают две основные проблемы. Первая из них связана с необходимостью запуска одного и того же операционного блока из различных мест схемы управления, а вторая – с необходимостью организации циклической работы некоторых операционных блоков. Эти проблемы могут быть решены путем введения дополнительных модулей.

Модуль преобразования фазовой дисциплины

Введем сначала *модуль*, осуществляющий *преобразование фазовой дисциплины* работы блока. На рис. 6.15,а дано общее обозначение такого модуля (*МПФ*), работающего с операционным блоком *A*. Модуль связан со схемой

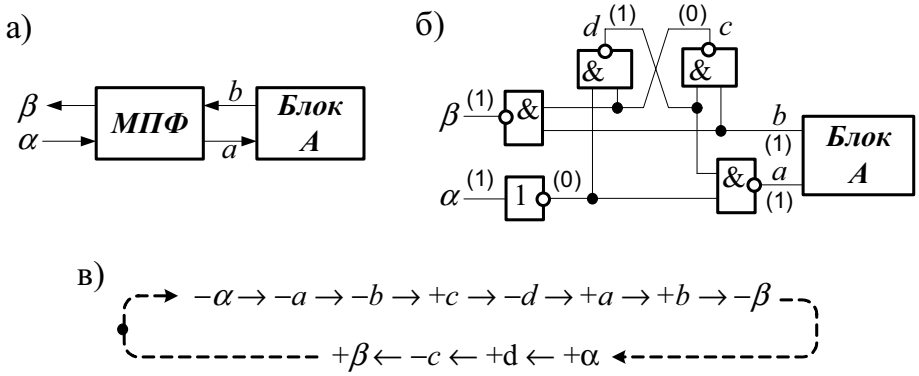


Рис. 6.15. Модуль для преобразования фазовой дисциплины: а) условное обозначение, б) схема модуля и в) сигнальный граф ее работы

управления парой сигналов α, β и блоком A парой сигналов a, b. При получении сигнала $\alpha = 0$ сигналы a, b должны дважды изменить свои значения, обеспечивая полный цикл работы блока, после чего сигнал β принимает значение 0. При получении модулем сигнала $\alpha = 1$ модуль не инициирует работу блока, а просто отвечает схеме управления сигналом $\beta = 1$. Таким образом, данный модуль используется для преобразования дисциплины "в изменениях" входного сигнала в стандартную дисциплину двухфазной работы самосинхронного блока. Схема модуля приведена на рис. 6.15,б. Его работа описывается сигнальным графом, представленным на рис. 6.15,в.

Модуль многократного вхождения оператора

Из модулей преобразования фазовой дисциплины может быть построен *модуль* (*автомат*) *МВО многократного вхождения оператора* (*операционного блока*). На рис. 6.16,а дано общее обозначение такого модуля, работающего с блоком, реализующим оператор A_i. Пара сигналов $(\alpha_{ij}, \beta_{ij})$ обеспечивает j-е включение блока A_i. Схема модуля приведена на рис. 6.16,б. Предполагается недопустимым одновременный запуск модуля из нескольких мест схемы управления.

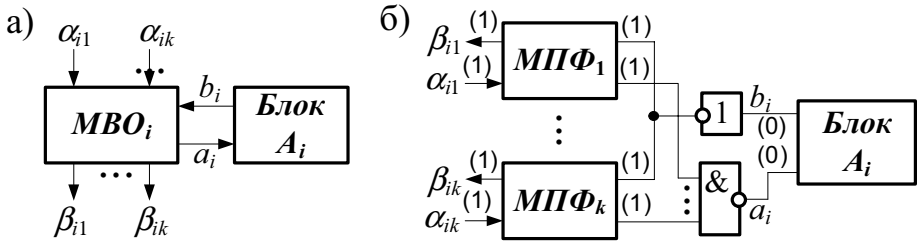


Рис. 6.16. Модуль многократного вхождения операционного блока: а) условное обозначение, б) реализация

Модуль управления циклом

В случае, когда операционный блок после однократного запуска должен выполнять циклически повторяемые действия с выходом из цикла при выполнении некоторого условия, необходимо между таким операционным блоком и схемой управления включать специальный *модуль управления циклом* (*МУЦ*), как это показано на рис. 6.17. Сигналами α и β модуль связан со схемой управления, запускающей циклические вычисления, а сигналами a и b – с операционным блоком, выполняющим тело цикла. Операционный

блок, будучи запущенным, вырабатывает условие окончание циклического процесса (сигнал $q = 1$). Переменная q представлена парафазно с произвольным транзитным состоянием 00 или 11, поэтому на рис. 6.17,а сигнал, представляющий инверсное значение переменной, обозначен как $\overline{q}$ или $\tilde{q}$. Схема модуля *МУЦ* приведена на рис. 6.17,б. Работа этой схемы в режимах входа в цикл, повторения тела цикла, выхода из цикла и возвращения модуля в исходное состояние описывается сигнальными графами на рис. 6.17,в. Следует заметить, что схема будет правильно работать только при выполнении операционным блоком следующего требования: при $b = 1$ сигналы $q, \overline{q}(\tilde{q})$ сохраняют свои значения и $q \neq \overline{q}$ ($q \neq \tilde{q}$), при $b = 0$ эти сигналы могут изменять свои значения через произвольное транзитное состояние.

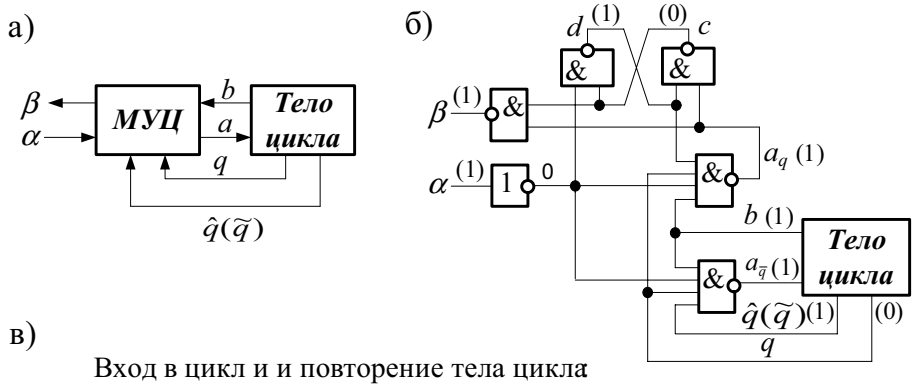


Рис. 6.17. Модуль управления циклом: а) условное обозначение, б) схема модуля, в) сигнальные графы, описывающие порядок срабатывания ее элементов

На основе модуля управления циклом можно построить схему *модуля многократного использования цикла* точно так же, как была построена схема многократного вхождения оператора. Оставляем читателю возможность самому построить такой модуль.

Кроме описанных выше модулей могут быть использованы также уже ранее рассмотренные схемы, например, синхронизатор (*С*-элемент), сборка, разветвитель, альтернативный разветвитель и др. Применение таких схем при композиции схемы управления и операционных блоков может упростить схему управления.

До сих пор рассматривалась возможность схемного моделирования устойчивых и безопасных сетей Петри. Однако некоторые сети Петри, в которых нарушено свойство безопасности, тоже могут иметь схемную реализацию. К таким сетям относятся интерпретированные СП, представимые полумодулярными недистрибутивными диаграммами переходов. Рассмотренные выше методы схемного моделирования применимы также и к таким сетям.

6.4.2. Реализация неустойчивых сетей Петри

Реализация сетей Петри с вершинами свободного выбора

В инженерной практике используются также СП, не обладающие свойством устойчивости. К таким сетям относятся сети свободного выбора, т.е. сети, содержащие вершины свободного выбора. *Свободный выбор* (см. рис. 2.9,б-1), к сожалению, не может быть реализован средствами двоичной схемотехники и требует применения специального элемента – арбитра, содержащего аналоговые цепи.

Явление электронного арбитража

Электронным арбитром называют устройство, принимающее решение о предоставлении разделяемого ресурса.

Схемотехнически решение проблемы разделения ресурса наталкивается на определенные трудности, которые можно проиллюстрировать на примере поведения приемника, на вход которого подается асинхронный сигнал (рис. 6.18,а). Приемник состоит из D-триггера T, прямой и инверсный выходы которого обозначены q и $\overline{q}$, и элемента 2И на его входе. На входы этого элемента подается внешняя асинхронная последовательность x и последовательность тактовых сигналов c.

Возможное поведение схемы, изображенной на рис. 6.18,а, приведено на рис. 6.18,б. На этом рисунке: c – синхротакты с периодом T; x – асинхронный сигнал; d – сигнал на выходе элемента И; $q, \overline{q}$ – выходные сигналы D-триггера; W – *окно арбитража* – интервал времени, при попадании в который фронта асинхронного сигнала со 100% вероятностью поведение схемы становится *аномальным*. Из рисунка видно, что на выходе вентиля 2И могут появиться короткие импульсы, энергии которых достаточно для инициации переходного процесса в триггере приемника, но недостаточно для того, чтобы триггер полностью переключился. В результате возникающего внутри триггера арбитража, триггер может оказаться в состоянии не-

устойчивого равновесия, (метастабильном состоянии), при котором значения сигналов на выходах совпадают и не представляют ни логического 0, ни логической 1. Такое поведение триггера является аномальным и называется *арбитражем*. Очевидно, что вероятность попадания схемы в аномальное состояние $p = W / T$. Длительность аномального поведения распределена по Пуассону и может значительно превышать период тактирования. Подобные устройства, принимающие асинхронно приходящие на их входы сигналов по тактовым сигналам, получили название *синхронизаторов*.

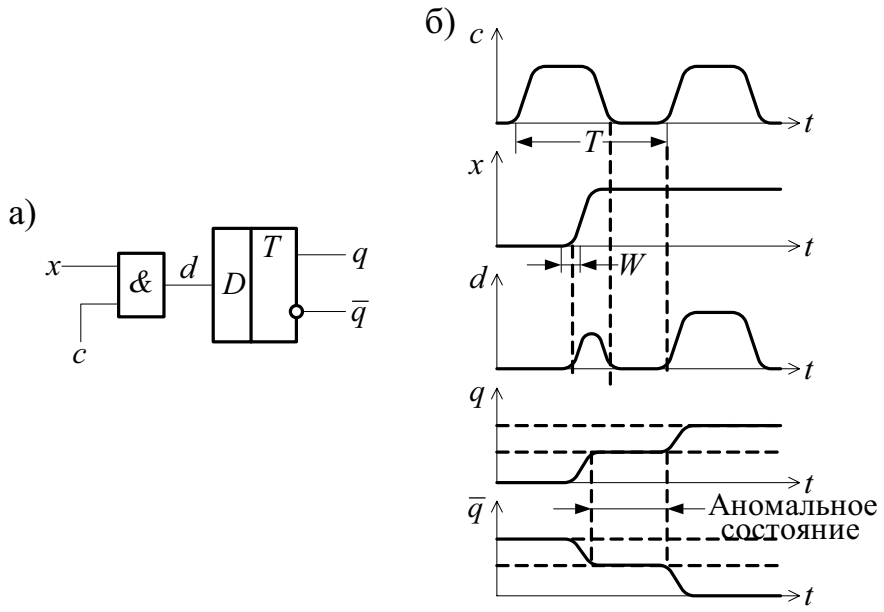


Рис. 6.18. Поведение синхронной схемы при поступлении асинхронного сигнала: а) схема привязки асинхронного сигнала к фронту сихросигнала, б) осциллограмма выделенных сигналов схемы

Электронные арбитры

Аномальное поведение характерно и для арбитров – устройств, служащих для предоставления общего ресурса только одному из процессов, запрашивающих этот ресурс. *Проблема арбитража* не зависит от природы процессов, запрашивающих ресурс, и общего ресурса. Даже если процессы являются программами, проблема, в конечном счете, сводится к построению работоспособного аппаратного арбитра. Общепринятая структура внешних соединений k-входового арбитра приведена на рис. 6.19. Она содержит: a_1,..., a_k – сигналы запроса у арбитра общего ресурса от процессов P_1,..., P_k;

b_1,..., b_k – сигналы ответа от арбитра о предоставлении ресурса выигравшему процессу; $a_г$ – сигнал запроса от арбитра на работу ресурса; $b_г$ – сигнал ответа о готовности ресурса на запрос арбитра.

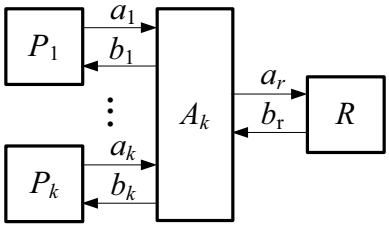


Рис. 6.19. Арбитр с k входами

В поведении логических схем, реализующих арбитры и синхронизаторы, известны четыре вида аномалий:

- уже упомянутая *метастабильная аномалия* (рис. 6.20,а);
- *колебательная аномалия* (рис. 6.20,б), представляющая собой синфазные колебания сигналов на обоих выходах, при которых значения сигналов на этих выходах совпадают в каждый момент времени;
- *нестабильный переходной процесс* (рис. 6.20,в);
- *затягивание одного из фронтов* в переходном процессе (рис. 6.20,г). Аномальное поведение может возникнуть при подача короткого импульса, амплитуда которого незначительно выше порога срабатывания логического элемента, на один из входов арбитра или синхронизатора, или в случае, когда запросы на общий ресурс приходят от разных процессов с интервалом, меньшим, чем время завершения переходных процессов в арбитре.

Возможность возникновения аномалий двух последних типов зависит от структуры устройства и технологии его изготовления. С ними можно бороться достаточно простыми средствами (правильным проектированием). *Аномалии первых двух типов носят фундаментальный характер* и избавление от них представляет наибольшие трудности.

Экспериментальные данные показывают, что аномальное поведение арбитров и синхронизаторов, реализованных на двоичных логических элементах, является заметным источником сбоев вычислительных и управляющих систем.

Не вдаваясь в теоретические вопросы, отметим, что для построения работоспособных арбитров можно воспользоваться элементами, поведение которых не может быть адекватно выражено средствами двоичной логики.

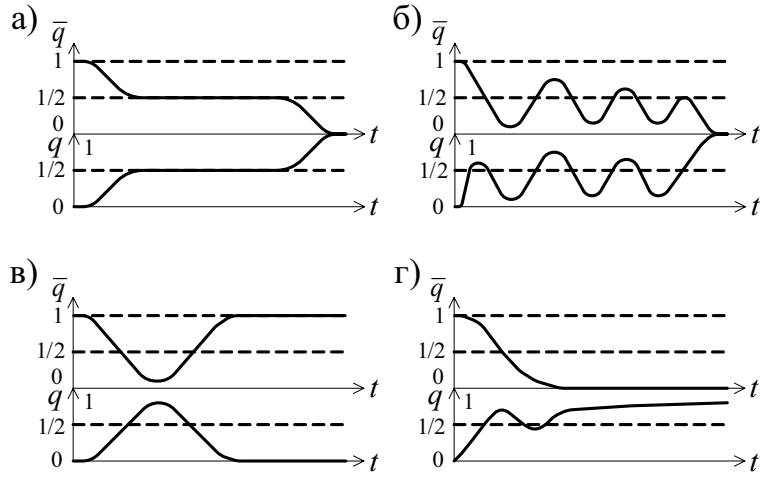


Рис. 6.20. Виды аномалий в схемах, реализующих арбитры: а) метастабильная аномалия, б) колебательная аномалия, в) нестабильный переходный процесс, г) затягивание фронта

Существуют два типа арбитров: *ограниченные арбитры* и *неограниченные арбитры*. Арбитры первого типа имеют ограниченное время ответа на запросы процессов вне зависимости от того, приходят ли они на входы арбитра в одиночестве или одновременно с другими запросами.

Ограниченные арбитры

Ограниченные арбитры — это арбитры, которые имеют синхронную реализацию, т.е. поступающие запросы синхронизируются тактовыми сигналами.

Частным случаем такого арбитра является синхронизатор, так как его можно рассматривать в качестве арбитра асинхронного сигнала и сигнала синхронизации. Таким образом, в состав ограниченных арбитров входит синхронизатор и, следовательно, для таких арбитров всегда существует некоторая вероятность сбоя. (Л.Р. Марино доказал теорему, утверждающую, что попытка синхронизации любых асинхронных сигналов всегда приводит к явлению арбитража.) Для уменьшения вероятности сбоя увеличивают время, отводимое на арбитраж (время аномального поведения). Обычно это два или три периода синхронизации. Таким образом снижают вероятность ошибки до чрезвычайно низкого уровня, но не до нуля.

Неограниченные арбитры

Неограниченные арбитры – это арбитры, которые имеют неопределенное время ответа на одновременные или близкие по времени запросы, т.е. время ответа такого арбитра не определяется величинами задержек элементов.

Суть подхода к построению такого арбитра состоит в фиксации момента окончания аномального поведения и формировании специального сигнала, блокирующего выходы арбитра до окончания его аномального поведения.

Один из способов фиксации момента выхода триггера из метастабильного состояния иллюстрируется схемой, представленной на рис. 6.21,а. В этой схеме *K* – компаратор с пороговым элементом на выходе. Когда плечи триггера разойдутся на некоторую величину, можно считать, что триггер вышел из метастабильного состояния. Заметим, что эта же идея работает и в случае колебательной аномалии, когда напряжения на обоих плечах триггера изменяется синфазно.

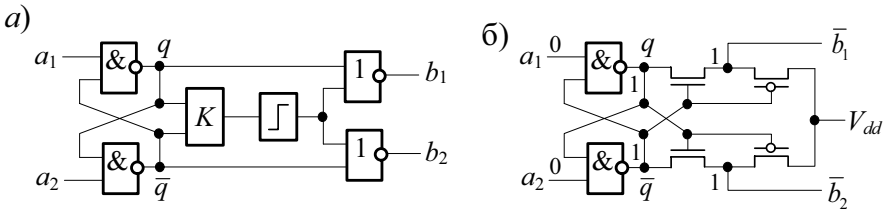


Рис. 6.21. Двухвходовый элемент взаимного исключения (мутекс): а) структурная схема, б) реализация

Реализация на транзистрах компаратора с порогом показана на рисунке рис. 6.21,б. Триггер с таким компаратором часто называют *элементом взаимного исключения*, или *мутексом* (от англ. MUTEX – MUTually EXclusive element). Выходные сигналы схемы сохраняют значение 11 до тех пор, пока плечи триггера не разойдутся на величину порога транзистора (т.е.выйдут из метастабильной зоны). После этого одно из плеч (в чью пользу разрешился арбитраж) начнет переключаться в состояние 0. Таким образом, мутекс маскирует матастабильную и синфазную колебательную аномалии и не пропускает на выход неопределенных сигналов.

На основе мутакса могут быть построены неограниченные арбитры асинхронных запросов, разрешающие доступ к распределяемому ресурсу. Рассмотрим двухвходовый неограниченный арбитр, представленный на рис. 6.22.

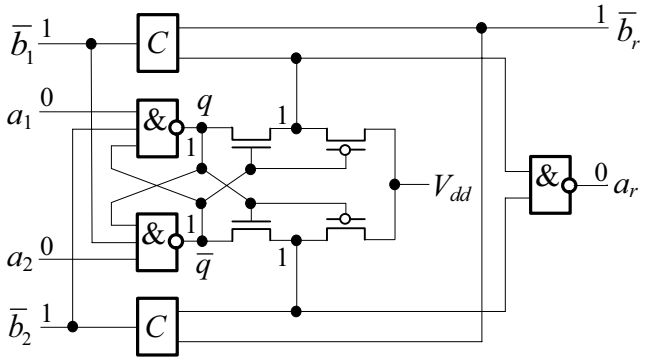


Рис. 6.22. Двухвходовый неограниченный арбитр

Он содержит мутекс, два двухвходовых *C*-элемента и двухвходовый элемент И-НЕ. Арбитр связан сигналами "запрос – ответ" (хендшейком) с каждым процессом, запрашивающим ресерс, и с ресурсом. Как только один из выходов мутакса станет равным "0", вырабатывается сигнал запроса ресурса $a_r = 1$. При появлении сигнала ответа от ресурса $\overline{b}_r = 0$ срабатывает тот *C*-элемент, который связан с процессом, выигрывшим арбитраж, и блокирует вход на арбитр от другого процесса до тех пор, пока выигравший процесс не снимет запрос на ресурс. Только после ответа ресурса на снятие запроса ресурс может быть предоставлен другому процессу.

Многовходовый арбитр может быть построен из двухвходовых в виде пирамидальной схемы в случае одинаковых приоритетов запросов от процессов на некоторый ресурс. При наличии статических приоритетов может быть построена схема из последовательных и пирамидальных участков. Динамические приоритеты требуют реконфигурации структуры многовходового арбитра, что трудно реализуемо.

Реализация неустойчивых ограниченных сетей Петри

Использование неограниченных арбитров позволяет реализовать неустойчивые безопасные сети Петри. Что касается реализации небезопасных СП и СГ, то в этом вопросе много неясностей, вытекающих из необходимости адекватной интерпретации наличия двух или более маркеров в некоторой позиции. Но некоторые соображения по этому поводу можно привести. Об-

ратимся к рис. 6.23, на котором изображена небезопасная 2-ограниченная СП. Действительно, после срабатывания t_1 и t_2 в позиции p_3 находятся два маркера. Однако наличие пары взаимоинверсных позиций p_4 и p_5 гарантирует такую последовательность срабатываний, что сначала может сработать переход t_3, забирая из p_3 один маркер, и только затем переход t_4, забирая из p_3 второй маркер. Если же при начальной маркировке сработал только один из переходов t_1 или t_2, то переход t_4 не может быть возбужден до тех пор, пока не сработают переходы t_3 и второй из пары переходов t_1, t_2. Таким образом, фрагмент, включающий переходы t_1, t_2, t_3 с инцидентными им позициями работает как элемент ИЛИ, но переход t_4 ожидает прихода второго "застрявшего" маркера, чтобы не потерять его в данном цикле функционирования. Приведенный пример наталкивает на мысль о возможности реализации некоторого класса ограниченных сетей Петри полумодулярными, но недистрибутивными схемами.

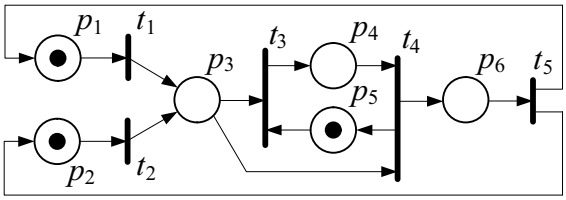


Рис. 6.23. 2-ограниченная сеть Петри

6.5. Упражнения

1. Для схемы Маллера задана система уравнений

$$\begin{cases} x_1 = x_2x_3 \vee x_1(x_2 \vee x_3) \\ x_2 = \overline{x}_1 \\ x_3 = \overline{x}_4 \\ x_4 = x_1x_3 \vee x_4(x_1 \vee x_3) \end{cases}.$$

Проведите редукцию по x_2 и x_4 и проанализируйте результаты на принадлежность классу схем Маллера.

2. Для системы, заданной в упр.1, постройте совершенную реализацию.

3. Для системы, заданной в упр.1, постройте схему в базисе двухвходовых С-элементов и инверторов.

4. Постройте сигнальный граф, соответствующий системе, заданной в упр.1, и проанализируйте ее на нормальность.

5. Продемонстрируйте эффекты редукции по x_2 и x_4 в упр.1 путем исследования нормальности соответствующих сигнальных графов.

6. Для сигнального графа, показанного на рис. 6.24

а) проверьте условия нормальности;

б) постройте соответствующую ДП;

в) найдите систему уравнений путем доопределения таблицы истинности, соответствующей ДП;

г) один из возможных способов доопределения функции x_2 дает уравнение

$$x_2 = x_1\bar{x}_3 \vee x_2(x_1 \vee \bar{x}_3);$$

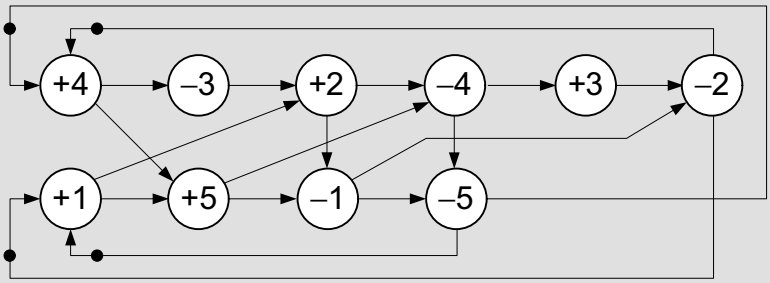


Рис. 6.24. Пример сигнального графа

Проведите редукцию исходной ДП по x_2, попытайтесь объяснить, почему при редукции по самозависимой переменной в результате все же получилась схема Маллера; попытайтесь провести редукцию по x_2 на СГ и проверьте сохранение свойства нормальности.

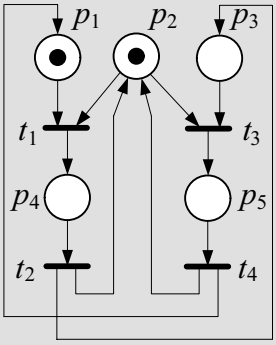


Рис. 6.25. Пример сети Петри

7. Установите устойчивость и безопасность сети Петри, изображенной на рис. 6.25, приведите и исследуйте ее стандартную реализацию в соответствии с рис. 6.8.

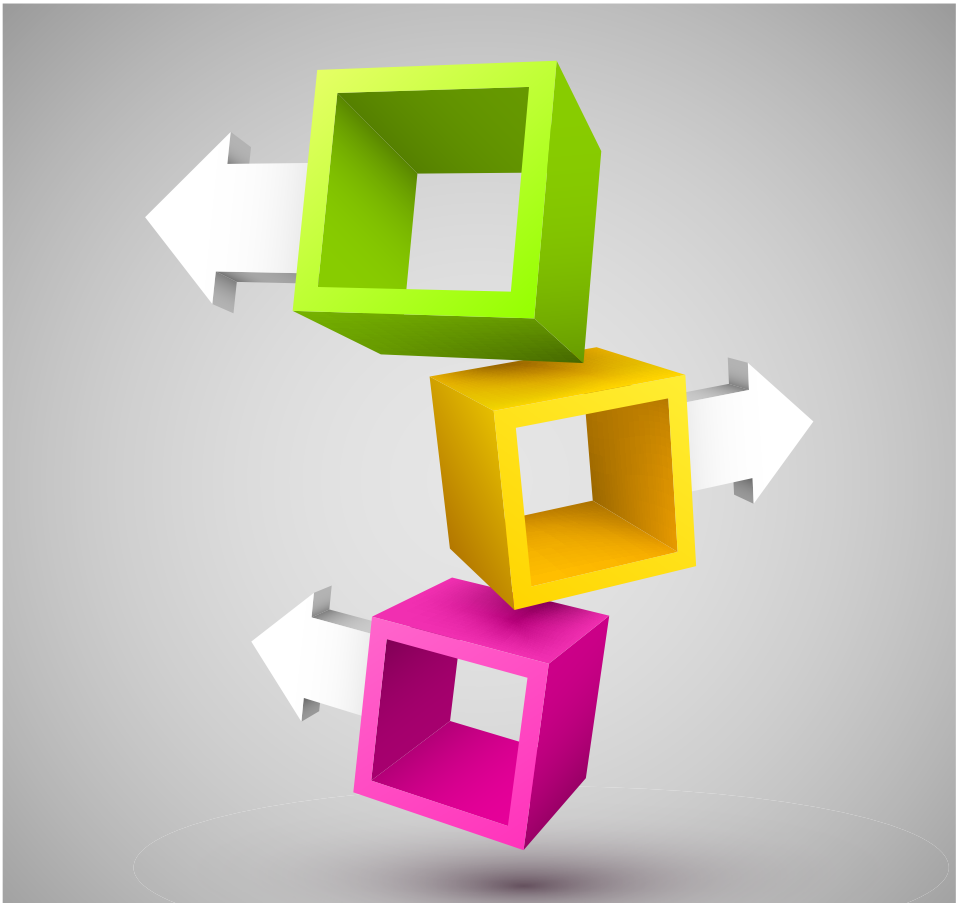
8. Проверьте на работоспособность схему, представляющую собой кольцо из двух ячеек Давида.

9. Изобразите СГ для схемы, представляющей собой кольцо из трех ячеек Давида.

10. Изобразите схему на ячейках Давида для сети Петри, показанной на рис. 6.25.

Указание: учесть необходимость введения дополнительных переходов и позиций с “пустой” семантикой.

11. Приведите схемную реализацию задачи взаимного исключения (рис. 6.10) с использованием арбитра.



Глава 7.

ПРОТОКОЛЫ ИНФОРМАЦИОННОГО ОБМЕНА

– Здорово, кума!
– На рынке была.
– Аль ты глуха?
– Купила петуха.
(Пример диалога)

Уже говорилось, что современные вычислительные машины и системы, а также их программное обеспечение строятся на основе модульного (агрегатного) принципа: аппаратные средства, равно как и программное обеспечение, выполняются в виде отдельных функциональных и структурных единиц (*модулей*). Последние могут объединяться (комплексироваться), образуя сложные системы. Организация модулей в систему предполагает взаимодействие между модулями и требует значительных временных, аппаратурных и программных ресурсов. Фундаментальным понятием, отражающим аспект взаимодействия модулей в системе, является понятие интерфейса.

В широком смысле под интерфейсом (или сопряжением) понимается совокупность правил, обеспечивающих совместное функционирование аппаратных и программных средств. Иногда интерфейсом называют и сами технические средства, созданные в соответствии с этими правилами.

Сопряжение (в отличие от устройств сопряжения) не является частью аппаратных средств, а представляет собой границу, или "сечение", между модулями системы, на котором строго определены сигналы и процедуры обмена. Сопряжение точно указывает, какие сигналы (сообщения) могут пересекать это сечение и каков их смысл. Оно должно обеспечивать единый метод передачи данных независимо от внутренних особенностей взаимодействующих модулей и относится исключительно к обмену сигналами между устройствами. Правильно определенное сопряжение является необходимым условием успешной разработки интерфейса. Подчас нелегко установить его точно и детально; по мере накопления опыта разработки и использования интерфейса его содержание совершенствуется.

Совокупность правил и средств, составляющих структуру сопряжения, или интерфейса, принято делить на три уровня: механический, электрический и логический.

- **Механический (конструктивный) уровень** определяет совокупность требований, которым должны удовлетворять узлы интерфейсной аппара-

туры: кабели, линии связи, разъемные соединения, интерфейсные карты (адаптеры) и т.п.

- **Электрический уровень** сопряжения оговаривает допуски на параметры входных и выходных электрических сигналов и допустимые токовые нагрузки на их источники. Допуски зависят от принятой элементной базы и общего уровня развития электрических средств.

- **Логический уровень** сопряжения обеспечивает координацию протекания информационных потоков между устройствами. Он определяет способ представления сигналов, символов, сообщений, обозначения и классификацию сигналов и сообщений. В зависимости от вида передаваемой информации и требований достоверности передачи могут применяться различные способы кодирования. Логический уровень – наиболее сложная часть правил сопряжения. Логическое согласование объединенных в систему устройств предусматривает функционирование сопрягаемых модулей по установленным алгоритмам.

При проектировании вычислительных машин и систем стремятся к стандартизации интерфейсов на всех уровнях сопряжения: форматов передаваемой информации, схем шин интерфейса, конструктивных параметров, алгоритмов функционирования. Стандартизация обеспечивает совместимость разрабатываемых и изготовляемых на разных предприятиях устройств вычислительной системы. Кроме того, она позволяет модернизировать вычислительные системы путем замены устаревшей аппаратуры вновь созданной (принцип “открытости” системы).

При наличии стандартного сопряжения вычислительная система может быть представлена как совокупность модулей, выполняющих функцию источников (задатчиков), которые передают информацию модулям-приемникам (исполнителям), и контроллеров. Последние обычно реализуют, кроме функций источника и приемника, управление обменом информации, т.е. устанавливают, какой из источников в данный операции обмена должен выдавать информацию. Обычно устройства выполняют несколько функций по передаче информации: функции приема, передачи и управления. Например, в интерфейсе “Общая шина” в качестве задатчиков могут выступать контроллеры периферийных устройств, в качестве исполнителей – модули оперативной памяти, а функции контроллера шины может выполнять процессор со встроенным арбитром.

Основными характеристиками интерфейса следует считать время, затрачиваемое на передачу единицы информации, достоверность передачи данных, помехоустойчивость, возможность и простоту наращивания числа устройств, простоту схемотехнической и конструктивной поддержки.

Соединение устройств осуществляется *проводами*, часто называемыми *линиями*. Группы линий, объединяемых по некоторому функциональному признаку, именуются *шинами* (например, шина данных, шина адреса).

Если же по одной шине передается разнородная информация (данные, адреса, команды), то выделяется группа линий для передачи признаков вида информации либо предусматривается хранение этих признаков в самих модулях (во внутренних регистрах соответствующих блоков сопряжения).

7.1. Классификация интерфейсов

На классификацию интерфейсов влияют многие аспекты, например: структурные особенности, состав шин и сигналов, способы передачи данных, способы синхронизации передаваемой информации, набор интерфейсных функций, технические характеристики интерфейсов, назначение и т.д. Остановимся лишь на некоторых из перечисленных аспектов, имеющих прямое отношение к тематике данного пособия.

Выделим следующие классификационные признаки интерфейсов:

- **способ соединения компонент** (радиальный, магистральный, кольцеобразный, комбинированный);
- **способ передачи информации** (параллельный, последовательный, параллельно-последовательный);
- **принцип реализации обмена информацией** (синхронный, асинхронный, самосинхронный);
- **способ обмена информаций:** односторонняя передача, дуплекс (одновременная двухсторонняя передача), полудуплекс (поочередная двухсторонняя передача).

7.1.1. Классификация по топологии

По структурному признаку различают интерфейсы с кольцевой, радиальной, магистральной и смешанной структурой (топологией).

Интерфейсы с кольцевой структурой

В системе с *кольцевой структурой* интерфейса все участники обмена соединяются последовательно интерфейсными линиями в кольцо и являются равноправными. Центральное устройство, как правило, отсутствует. Интерфейсы такого типа иногда используются в бортовых системах управления.

Интерфейсы с радиальной структурой

В системе с *радиальной структурой* имеется центральное устройство, являющееся контроллером. С ним связано каждое из подчиненных устройств (источников и приемников) посредством индивидуальной группы линий. Группы, как правило, идентичны по номенклатуре линий. Обмен данными происходит непосредственно между устройством (абонентом) и контроллером под управлением контроллера. Интерфейс с радиальной структурой применяется обычно в системах, обеспечивающих связь с удаленными периферийными устройствами и устройствами промышленной автоматики. Линии являются однонаправленными и, как правило, не мультиплексируются.

Интерфейсы с магистральной структурой

В интерфейсах с *магистральной структурой* имеется набор общих линий, к которым подсоединяются все устройства и контроллер, содержащий арбитр. Сигналы, передаваемые по линиям, в принципе доступны всем подключенным к интерфейсу устройствам. Однако логический уровень интерфейса должен обеспечивать возможность монополизации магистрали только одним задатчиком в каждый момент времени.

Связь между устройствами может быть установлена по инициативе, как самих устройств, так и контроллера. В первом случае устройство, желающее захватить магистраль с целью обмена данными, посылает контроллеру запрос. При освобождении магистрали от предыдущего захвата контроллер разрешает захват магистрали тому запрашивающему устройству, которое имеет наибольший приоритет. После захвата магистрали устройство реализует операцию обмена данными с любым другим устройством по его адресу. Таким образом, в отличие от радиального интерфейса, информация может поступать от источника в приемник непосредственно, минуя контроллер.

Системам с магистральной структурой свойственна гибкость и легкость реконфигурации.

Интерфейсы со смешанной структурой

В системах со *смешанной структурой* интерфейса возможна комбинация обоих способов, например, выделение для шин адреса устройств индивидуальных линий, а для шин данных – магистрали.

7.1.2. Классификация по способу передачи данных

По способу передачи данных интерфейсы можно разделить *на последовательные параллельные*, и *параллельно-последовательные*. При параллельной передаче данных *n*-разрядное слово передается по *n* линиям, при последовательной – по одной линии (хотя общее число линий связи может быть и больше, например, может быть выделена линия для тактовых сигналов и т.п.). Для интерфейса последовательного типа характерны наименьшая скорость передачи данных и минимальная стоимость кабельной сети. Такой способ оправдан для систем с удаленными друг от друга устройствами, когда не требуется высокое быстродействие, а средства связи дешевы.

7.1.3. Классификация по принципу реализации

По *способу синхронизации передачи информации* интерфейсы можно разделить на синхронные и асинхронные, среди которых особое место занимают самосинхронные.

Синхронные интерфейсы

При *синхронном способе* передачи источник задает темп выдачи и приема информации и синхронизирует все процессы, связанные с передачей данных приемнику. Для правильной работы источник информации должен обеспечивать некоторый заданный интервал в выдаче последовательных порций информации, который учитывал бы как максимальную задержку линий связи, так и максимальное время, затрачиваемое приемником на использование очередной порции данных.

Асинхронные интерфейсы

***Асинхронный принцип передачи*, называемый также принципом "запрос – ответ", предполагает, что прием каждой порции информации (сообще-**

ния), посылаемой от источника к приемнику, подтверждается (квитируется) специальным сигналом ответа. Таким образом, интервал времени между выдачей источником последовательных сообщений является переменным и зависит от параметров схем источника, приемника и линий связи. Благодаря применению асинхронного принципа, можно сопрягать устройства с разным быстродействием. Этот способ является более подходящим для систем сопряжения с магистральной структурой, в то время как синхронный способ передачи – для систем с радиальной структурой.

Самосинхронные интерфейсы

Самосинхронный принцип передачи данных **является частным случаем асинхронного принципа и использует сомосинхронизирующиеся коды**, что позволяет осуществлять правильную передачу данных при любых разбросах задержек в интерфейсных соединительных линиях, а также в передающих и приемных устройствах . Применяется в специальных разработках и широкого распространения еще не получил.

7.1.4. Классификация интерфейсов по назначению

В качестве критерия классификации интерфейсов по назначению можно использовать степень распределенности компонент вычислительной системы или сети, характеризующуюся максимально возможным расстоянием между взаимодействующими модулями. По этому признаку различают три группы интерфейсов: *локальные, локально распределенные* и *территориально распределенные*.

Кроме того, интерфейсы можно разделить на *внутрисистемные* и *сетевые*. Как правило, локальные интерфейсы являются внутрисистемными структурами и имеют магистральную организацию. В локально распределенных структурах сопряжения используются ретрансляторы и расширители интерфейса, а иногда и специальные драйверы (усилители) параллельной передачи сигналов, которые обеспечивают удаление периферийной аппаратуры на расстояние до 300 метров. Реализация территориально распределенных структур связи большой протяженности с использованием системных параллельных интерфейсов широкого применения не получила. Подобные сопряжения создаются в основном в рамках разработок физического уровня архитектуры вычислительных сетей. (Примером могут служить так называемые гигабитные сети.) Так, использование сетевого интерфейса ИРПР (радиальный, параллельный) ограничено расстоянием в 2 км.

7.2. Задание протоколов

Как уже отмечалось, наиболее сложной частью интерфейсных правил является логический уровень сопряжения устройств. На логическом уровне все сопрягаемые устройства называются объектами взаимодействия. **Основное понятие, отражающее координированное функционирование объектов логического уровня интерфейса, - это понятие протокола информационного обмена.** Известно, что понятие протокола вычислительных сетей является ключевым при описании логической и программной структуры распределенной системы. В качестве примера такой структуры может рассматриваться семиуровневая модель распределенных открытых систем, предложенная международной организацией по стандартизации (ISO) в 1978 году. С каждым уровнем этой модели связаны свои объекты взаимодействия и свои протоколы обмена.

В широком смысле под *протоколом информационного обмена* понимается совокупность правил, которым следуют объекты, рассматриваемые на некотором уровне структуры распределенной системы, при взаимодействии друг с другом путем обмена сообщениями.

Интерфейсные правила, определяющие структуру сопряжения устройств и способ передачи данных между ними по линиям связи, относятся к нижнему (физическому) уровню указанной выше модели.

Применительно к интерфейсам физического уровня под протоколом будем понимать совокупность правил, определяющих порядок установки и снятия логических сигналов на интерфейсных линиях, соединяющих взаимодействующие объекты. Действительно, протоколы физического уровня структуры распределенной системы (далее будем называть такие протоколы *интерфейсными*) характеризуются тем, что в качестве сообщений выступают изменения логических состояний линий связи. Следует отметить, что в практике проектирования интерфейсов нельзя выделить единого способа описания интерфейсных протоколов. Часто для этих целей применяются неформальные или полуформальные средства: естественный язык, временные диаграммы, циклограммы, блок-схемы алгоритмов.

7.2.1. Протокол чтения данных в интерфейсе «Общая шина»

Рассмотрим в качестве примера неформального задания протокола *описание операции чтения данных ведущим устройством* (задатчиком) из ведо-

мого устройства (исполнителя) для интерфейса “Общая шина”. Структура связи показана на рис. 7.1.

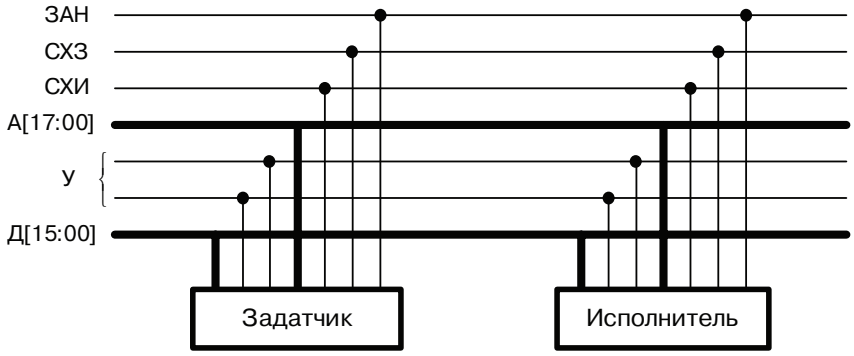


Рис. 7.1. Структура связей интерфейса “Общая шина” для протокола чтения данных

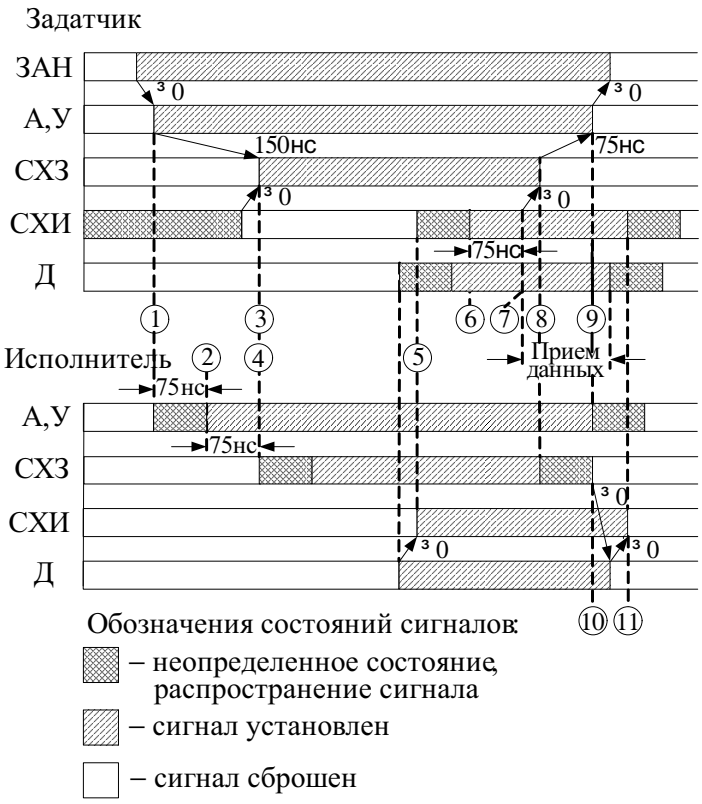


Рис. 7.2. Временная диаграмма работы задатчика и исполнителя для протокола чтения данных в интерфейсе “Общая шина”

Нотация линий ясна из описания процедур обмена. Временные диаграммы работы задатчика и исполнителя показаны на рис.7.2. Соответствующее этой диаграмме словесное описание протокола (номера шагов в словесном описании суть номера переходов временной диаграммы) состоит в следующем.

1. Задатчик, захватив магистраль (установлен сигнал “занято” – ЗАН), выставляет адрес и биты управления на соответствующие линии общей шины.

2. По истечении времени, равного задержке распространения сигналов по каналу связи, каждое устройство, подключенное к магистрали, принимает адрес и биты управления и декодирует их.

3. Задатчик ждет минимум 150 нс после выставления адреса на шине А и битов управления на шине У (компенсация перекоса переднего фронта), затем, если сигнал “синхронизация исполнителя” СХИ сброшен, устанавливает сигнал “синхронизация задатчика” СХЗ. Задатчик не должен устанавливать СХЗ на входе драйвера (магистрального усилителя) этой линии до тех пор, пока не истекут 150 нс с момента установки адреса, битов управления и сигнала стробирования входов драйверов шин А и У.

   Замечание. Компенсация перекоса переднего фронта состоит из задержки величиной 75 нс для компенсации чистого перекоса сигналов на шинах А и У в исполнителе и задержки в 75 нс для компенсации времени декодирования исполнителем адреса и битов управления.

4. По истечении времени, равного задержке распространения сигнала по каналам связи, каждое устройство принимает установку СХЗ. То устройство, в котором декодированный адрес совпадает с присвоенным ему адресом, становится исполнителем в данном акте обмена.

5. Спустя некоторое время после приема сигнала установки СХЗ исполнитель выдает требуемые данные на шину Д и устанавливает СХИ. Важно, что исполнитель не должен устанавливать СХИ до момента установки данных и сигнала стробирования данных на входы драйверов шины Д.

   Замечание. СХИ не должен быть установлен до того, как выставлены данные на шине Д. Это необходимо для того, чтобы задатчик мог компенсировать перекос сигналов на линиях данных относительно СХИ и затем стробировать их правильное значение.

6. По истечении времени, равного задержке распространения сигналов по каналу связи, задатчик воспринимает установку СХИ.

   Замечание. Если установка СХИ не принята задатчиком в течение заданного времени после установки СХЗ (таймаут), может быть выполнен шаг 7 и должны быть выполнены шаги 8 и 9. Может быть установлен бит ошибки. Таймауты обычно выбирают равными 10 – 20 мкс.

7. Спустя 75 нс после приема установки СХИ (компенсация *перекоса данных*) задатчик считывает данные с выходов своих чувствительных элементов (магистральных приемников).

   Замечание. Компенсация перекоса сигналов шины Д осуществляется задатчиком.

8. Задатчик сбрасывает СХЗ.

9. После минимальной задержки, называемой компенсацией перекоса заднего фронта, задатчик снимает адрес и биты управления с шин А и У. Если этот акт обмена при проведенном захвате шины последний, то задатчик далее сбрасывает ЗАН.

   Замечание. Компенсация перекоса заднего фронта гарантирует, что линии шины Ф не изменяют своего состояния в любом из устройств, пока в устройство поступает установленный сигнал СХЗ. Это предотвращает ошибочную выборку устройства из-за возможного изменения состояний линий адреса на то время, пока установлен СХЗ.

10. По истечении времени, равного задержке распространения сигналов по каналу связи, исполнитель принимает сброс СХЗ.

11. Исполнитель снимает данные с шины Д, а затем сбрасывает СХИ.

    Замечание. Сигнал СХИ не должен быть сброшен до того, как будут сняты данные с шины Д. Это гарантирует, что перед сбросом СХИ биты данных сняты с шины Д. Таким образом, сброс СХИ фиксирует снятие данных.

Очевидно, что приведенное словесное описание протокола довольно громоздко. Создавая лишь иллюзию ясности толкования и возможности высокой степени детализации протокола, неформальное задание в то же время затрудняет анализ протокола в целом. Сложно установить, является ли протокол логически корректным, т.е. непротиворечивым и полным с точки зрения достижимости заданных ситуаций. Обычно проверка неформального задания протокола значительно усложняется, если протокол допускает параллельное выполнение некоторых действий объектами взаимодействия,

либо имеет много альтернатив поведения. В таких случаях без применения формальных средств практически не обойтись. В качестве таких средств далее используются (среди прочих) и те модели динамического типа, которые рассматривались в предыдущих главах. Будет также показано, что приведенный выше протокол для интерфейса “Общая шина” может быть задан существенно более компактно. Однако, прежде всего, рассмотрим задачи, возникающие при проектировании интерфейсов, и остановимся на особенностях асинхронных интерфейсов.

7.2.2. Примерная процедура проектирования протокола

Как правило, процедура проектирования протокола состоит в следующем.

1. **На первом этапе** строится основной сценарий взаимодействия, который включает определение перечня основных сигналов или сообщений, а также базовой последовательности этих сигналов. На этом этапе создания протокола строится ***первичная модель***, которая не обязательно должна учитывать влияние нижнего уровня взаимодействия (канала связи) и свойства восстанавливаемости после ошибок.

2. **На втором этапе** (построение ***вторичной модели***) исследуются альтернативные ситуации, связанные с усложнением структурной модели взаимодействия. Обычно это моделирование неидеального канала связи, обладающего конечной, возможно, неограниченной каким-то конкретным значением задержкой, а также способностью вносить искажения или утерю сообщений. Однако на этом этапе может потребоваться введение новых сигналов и последовательностей. Таким образом, вторичная модель обеспечивает не только заданный сервис верхнего уровня, но также учитывает влияние нижнего уровня, возможность возникновения ошибок и средства восстановления правильного функционирования. Учет влияния нижнего уровня действительно необходим, поскольку он является “почтальоном” для данного уровня взаимодействия.

3. **Третий этап** (построение ***окончательной модели***) обычно связан с преобразованием модели управляемого взаимодействия с целью последующей реализации протокола. При этом устанавливается набор сигналов и порядок обмена с верхним уровнем. Таким образом, разрабатываются модели участников обмена, которые потом реализуются блоками сопряжения модулей или устройств, включаемых в систему.

Проектирование аппаратных и программных средств сопряжения – проблема нелегкая. В общем случае ее решение требует учета большого числа

различных по своему содержанию вопросов, касающихся выбора структуры интерфейса, способа передачи информации (параллельного или последовательного), принципа синхронизации, оценки производительности, надежности и экономичности средств связи, проведения самого трудного этапа – разработки описания протокола, его анализа, реализации и, наконец, оценки сложности его аппаратной и программной поддержки. Остановимся на этом наиболее сложном этапе.

7.2.3. Разработка описания протокола, его анализа и реализации. Оценка сложности аппаратной и программной поддержки протокола

Сервисный уровень протокола

Прежде всего, необходимо определить, каким образом сформулировать цель и функции разрабатываемого интерфейсного протокола, который, будучи средством взаимодействия объектов нижнего (физического) уровня распределенной системы, является основой функционирования всей (возможно, многоуровневой) системы протоколов обмена. В связи с этим важным является понятие *сервиса* (сервисных функций), предоставляемого данным протокольным уровнем верхнему протокольному уровню, который выступает в роли пользователя данного интерфейсного протокола. С точки зрения пользователя данный протокольный уровень представляется "черным ящиком", допускающим определенный набор действий (см. рис. 7.3).

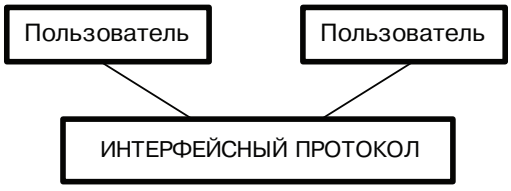


Рис. 7.3. Интерфейсный протокол как средство связи объектов верхнего уровня

Протокол-пользователь должен четко представлять, что обеспечивает используемый им интерфейс более низкого уровня, но вовсе не обязан понимать, как он это делает. Например, сервисной функцией интерфейса "Общая шина" служит обеспечение программных операций, связанных с обращением к элементам адресного пространства (256 кбайт при 18-разрядном адресе), а именно: операций с адресацией ячеек памяти или регистров периферийных устройств при выполнении процессорной программы; операций прямого доступа к памяти, выполняемых периферийными устройствами; операций прерывания процессорной программы.

Таким образом, **первая задача логического проектирования интерфейса состоит в описании (спецификации) сервиса, предоставляемого интерфейсным протоколом пользователю.** Строго говоря, задание входно-выходного поведения протокольного уровня и составляет описание сервиса протокола.

Задав некоторым образом сервис, предоставляемый данным уровнем, далее необходимо определить, какой сервис обеспечивается нижним уровнем. Ранее было установлено, что логический уровень интерфейса имеет в качестве своих нижних уровней электрический и механический уровни сопряжения. Следовательно, сервис нижнего уровня – это описание структуры магистрали, основных функций линий связи, значений логических уровней (активных и пассивных состояний) сигналов, характеристик работы магистральных усилителей и чувствительных элементов.

Построение структурной модели интерфейсного протокола

Следующей задачей является построение структурной модели протокольного уровня. Внутренняя структура логического уровня интерфейса задается в терминах объектов взаимодействия и отношений между ними (см. рис. 7.4).

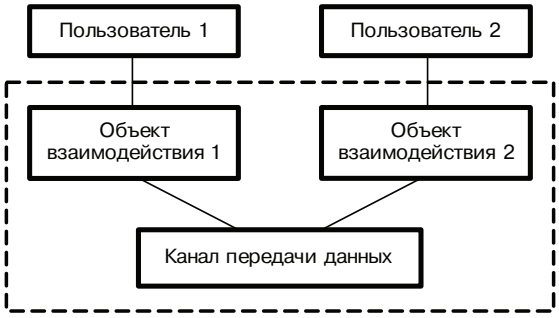


Рис. 7.4. Структура интерфейсного протокола

Например, возможны структуры: источник – шина данных – приемник; задатчик – шины адреса, управления, данных, линии синхронизации – исполнитель; арбитр – линии запросов, разрешения, подтверждения выбора, занятости – потенциальные задатчики. Заметим, что структура уровня не обязательно одинакова для всех функций, исполняемых протоколом, поэтому возможно создание множества структур, удовлетворяющих заданным интерфейсным функциям. Так для общей шины существуют свои структуры взаимодействия для функции обмена данными, функции захвата маги-

страли, функции передачи процессору вектора прерывания. После решения указанных задач возможен переход к этапу собственно описания интерфейсных протоколов.

Формализация описания интерфейсного протокола

Описание протокола обмена данными между объектами состоит в выполнении следующих шагов:

1. общее неформальное описание работы объектов для выделенной структуры взаимодействия;
2. выбор формальных средств задания протокола;
3. спецификация протокола, включающая список типов и форматов сообщений или сигналов, которыми обмениваются объекты, а также правила реакции каждого объекта на команды верхнего уровня, сообщения других объектов и внутренние события;
4. детализация фрагментов описания, например, характеристик эффективности, предложений по выбору средств реализации или описания, максимально приближенного к реализации.

Существующие методы формального описания протоколов, а точнее, модели, положенные в их основу, можно разбить на три основные группы: *модели с конечным числом состояний*, к которым относятся конечные автоматы, сети Петри, диаграммы переходов, сигнальные графы, формальные грамматики, логические матрицы и т.п.; *алгоритмы на языках программирования высокого уровня*; *комбинации моделей* первой и второй групп.

Для протоколов верхних уровней распределенных систем наиболее перспективными являются комбинированные модели, в которых управляющие функции (синхронизация, инициация, установление связи, разъединение) представлены моделью из первой группы, а функции обработки данных (нумерация и счет сообщений) описаны на алгоритмическом языке. Для протоколов нижнего уровня обычно достаточно использовать только модели с конечным числом состояний (модели переходов), от которых возможен переход к аппаратной реализации сопряжения.

Синтез средств сопряжения

В процессе синтеза средств сопряжения необходим поэтапный анализ полученных на каждом шаге представлений протокола. **Сущность задачи анализа, называемого иногда верификацией, состоит в выяснении**

того, удовлетворяет ли поведение системы взаимодействующих объектов своему описанию с точностью до заданных допустимых состояний системы, а также проверке наличия определенных свойств (эффектов взаимодействия). Здесь особый интерес представляют эффекты, присущие асинхронному взаимодействию объектов, такие, как отсутствие дедлоков и блокировок, живость, возможность самовосстановления после дефектов, самосинхронизация, отсутствие влияния перекосов задержек, индицируемость, безопасность (см. гл.2).

Очевидно, что основная цель взаимодействия устройств – передача информации от объекта-источника к объекту-приемнику в неискаженном виде. Определять достоверность передачи информации программным путем зачастую весьма сложно, в связи с чем, одним из важнейших вопросов организации интерфейса является использование специальных кодов и аппаратуры, обеспечивающих решение указанной задачи. Оно связано с проблемой синхронизации передачи информации по параллельным линиям связи. Использование одного синхросигнала при передаче информации от источника к приемнику отвергает возможность подтверждения факта (квитирования) правильного приема.

Использование традиционных способов асинхронной передачи с подтверждением приема (например, в “Общей шине” или приборных интерфейсах МЭК) требует учета разброса задержек сигналов в линиях (явление перекоса) путем дополнительной искусственной задержки в выдаче сигнала сопровождения информации по шине. Это приводит к значительному уменьшению скорости и надежности передачи информации. Так, например, в “Общей шине” величина компенсирующей задержки принята равной 75 нс, что приблизительно равно времени переключения двадцати вентилей малой степени интеграции. Тот же самый перекос приводит к расползанию синхроимпульсов и, как следствие необходимость *арбитража* (установления очередности появления фронтов независимых сигналов) во входных цепях блоков, либо компенсирующего увеличения длительности синхротакта. Заметим, что эта проблема возникает не только при достаточно удаленных модулях, но и на малых расстояниях внутри больших интегральных схем.

Использование согласованной самосинхронной передачи информации позволяет обеспечить синхронизацию в процессе обмена независимо от задержек, локализовать неисправность канала на схемном уровне и в некоторых случаях повысить устойчивость передачи к помехам извне за счет позиционной избыточности самосинхронизирующихся кодов либо за счет организации порязрядного квитирования, например при использовании трехста-

бильных линий связи. В трехстабильной линии гашение передаваемого сигнала приемником может быть использовано как сигнал квитанции, посылаемый по тому же проводу.

Реализация протоколов

Наконец, **одной из важных задач, возникающих при синтезе интерфейсных средств, является задача реализации протоколов (создание программно-аппаратной поддержки)**. Аппаратура сопряжения, разрабатываемая по заданному протоколу, должна полностью соответствовать всем требованиям, установленным данным интерфейсом.

Если интерфейс является асинхронным и требует применения принципа самосинхронизации при передаче информации, то реализация блоков сопряжения выбирается самосинхронной: она не должна зависеть от задержек элементов и линий связи. Для построения аппаратуры поддержки в этом случае могут использоваться методы синтеза самосинхронных схем, основанные на применении тех же формальных моделей, которые служат основой для задания протоколов. Это позволяет говорить о возможности построения сквозных процедур перехода от исходного задания протокола асинхронного интерфейса к асинхронно функционирующей аппаратуре сопряжения.

7.2.4. Формальная модель протокола информационного обмена физического уровня

Понятие протокола информационного обмена рассматривалось выше преимущественно применительно к нижнему, физическому, уровню структуры распределенных вычислительных систем. Поэтому основным объектом описания являлись интерфейсные протоколы, координирующие упорядоченный обмен сигналами по линиям,связывающим сопрягаемые устройства. Ясно, что **для однозначности трактовки и точности отображения протокола при реализации необходимо его формальное представление.**

Развитие методов формального описания протоколов ни в коей мере не ограничено применением их лишь к интерфейсным протоколам. Например, существующие модели охватывают преимущественно верхние уровни системной структуры и ориентированы на программную реализацию. Учитывая несомненный интерес к проблеме описания протоколов вообще, внимание в дальнейшем не будет акцентироваться на определенном уровне взаимодействия, чтобы не потерять общности. Однако предметной

иллюстрацией моделей все же будут служить именно интерфейсные протоколы. Что же будет пониматься под формальной моделью протокола обмена? Выше говорилось, что этапу описания протокола обязательно предшествует этап задания структуры взаимодействия объектов. Такая структурная модель должна быть инвариантна к языку описания и средствам реализации протокола.

Простейшей структурой взаимодействия может служить модель непосредственного взаимодействия пары объектов (см. рис.7.5). В ней отсутствует какое-либо описание согласующего объекта (процесса) – адаптера, канала или другого средства передачи информации на нижнем уровне. Кроме того, не выделяется верхний уровень, т.е. запросы на сервис такого взаимодействия и ответы на них. Предполагается, что этот уровень относится к внутренним событиям объектов O_1 и O_2. В этой модели будем интересоваться только теми событиями, которые разворачиваются в прямоугольнике, обозначенном пунктиром.

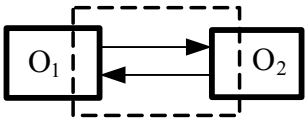


Рис. 7.5. Простейшая структура взаимодействия пары объектов

Иначе представляется модель опосредованного взаимодействия пары объектов, представленная на рис. 7.6,а. Она предполагает наличие третьего объекта – согласующего звена, моделирующего средство нижнего уровня для передачи данных, например, канал связи (интерфейсную магистраль). Такая структура позволяет моделировать неидеальный канал, в котором возможны перекосы сигналов, искажения или потеря сообщений.

Еще более сложной является модель взаимодействия, управляемого с верхнего уровня, изображенная на рис. 7.6,б. Здесь предполагается, что каждый из взаимодействующих объектов разбит на два звена: участник обмена и пользователь (управляющее звено). Такая модель позволяет задавать взаимодействие на некотором выбранном уровне и вместе с тем отражать сервис, предоставляемый верхнему уровню. Кроме того, она наиболее приближена к этапу реализации протокола. Действительно, по описанию участника обмена модно прямо синтезировать блок сопряжения периферийного устройства с интерфейсной магистралью.

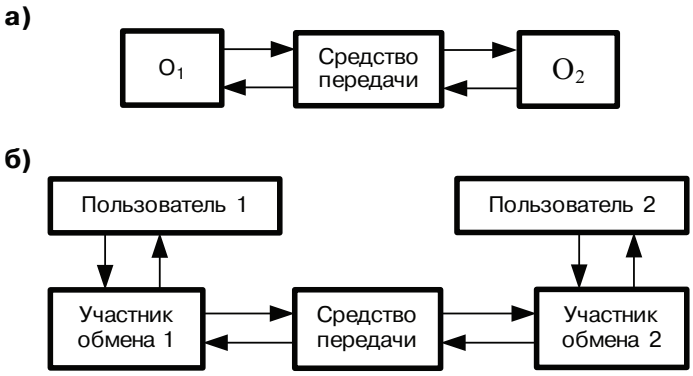


Рис. 7.6. Структура взаимодействия а) через канал связи, б) с управлением от верхнего уровня

Выше рассматривались структурные модели взаимодействия пары объектов, но методы описания протоколов не должны ограничиваться моделями парного взаимодействия. Структуры взаимодействия в общем случае охватывают несколько устройств или процессов. Это касается, например, сетевых протоколов маршрутизации сообщений. При организации межмодульных интерфейсов возникают проблемы, связанные с совместным функционированием одного источника и нескольких приемников через общую магистраль, с описанием функции захвата общего ресурса – шины обмена данными – путем арбитража запросов, и с описанием взаимодействия вида "один – всем". Оказывается, что эти модели не всегда могут быть сведены к суперпозиции моделей парного взаимодействия, в частности, проверка работоспособности сложной по топологии системы объектов не всегда обеспечивается анализом всех парных протоколов.

Итак, **формальная модель протокола включает не только модель структуры взаимодействия (структурную модель), заданную множеством объектов (участников взаимодействия) и отношением взаимной связи между ними, но и модели поведения каждого объекта структуры (функциональные модели), заданные правилами реакции данного объекта на сигналы (сообщения), поступающие от других объектов.**

Структурная модель **задается графом, вершинами которого являются объекты, а дугами – взаимные связи.** ***Функциональная модель*** **задается с помощью какого-либо формального аппарата, отражающего динамику объекта в процессе его взаимодействия с другими объектами.** Здесь могут использоваться конечные автоматы, сети Петри, сигнальные графы, диаграммы переходов, а также другие изобразительные средства.

Поскольку сети Петри представляют собой более мощный аппарат, чем конечные автоматы, здесь не будем рассматривать конечноавтоматные способы задания протоколов, а сразу перейдем к примеру использования сети Петри для спецификации некоторого протокола.

7.2.5. Пример использования сети Петри для задания протокола обмена

В качестве примера протокола обмена рассмотрим систему, состоящую из канального процессора КП, адаптера Ад и двух абонентов Аб1 и Аб2 и представленную на рис. 7.7,а. КП связан с абонентами общей магистралью данных МД, при помощи которой все эти устройства осуществляют прием и передачу информации.

Система работает по принципу ведущий–ведомый. Роль ведущего играет КП, ведомых – абоненты. В исходном состоянии $a = a_1 = a_2 = b_1 = b_2 = b = 0$. КП устанавливает адрес A абонента и затем выставляет сигнал запроса $a = 1$. По адресу и этому сигналу адаптер вырабатывает либо $a_1 = 1$, либо $a_2 = 1$, инициирующие прием или передачу информации. После окончания работы с информацией соответствующий абонент отвечает установкой $b_i = 1$. После приема этой квитанции адаптер устанавливает $b = 1$. Затем начинается процесс отключения абонента: КП устанавливает $a = 0$ (при этом он может менять адрес), по которому адаптер сбрасывает в 0 тот сигнал a_i, который был равен 1. В ответ на $a_i = 0$ i-й абонент устанавливает $b_i = 0$, после чего адаптер выставляет $b = 0$ и система оказывается в исходном состоянии.

На рис. 7.7,б изображена сеть Петри, задающая взаимодействие указанных устройств на уровне установления и разрыва связи. Эта сеть состоит из четырех фрагментов, относящихся к каждому из взаимодействующих устройств. Принято следующее соответствие наличия маркеров в некоторых позициях значениям сигналов обмена: 4, 5 – $a = 1$, адрес Аб1; 4, 6 – $a = 1$, адрес Аб2; 2, 4 – $a = 0$; 7 – $b = 1$ или $b = 0$; 9 – $a_1 = 1$ или $a_1 = 0$; 11 – $a_2 = 1$ или $a_2 = 0$; 12 – $b_1 = 1$ или $b_1 = 0$; 13 – $b_2 = 1$ или $b_2 = 0$; 14 – адрес Аб1; 15 – адрес Аб2. Одновременное появление маркеров в позициях 14 и 15 исключается.

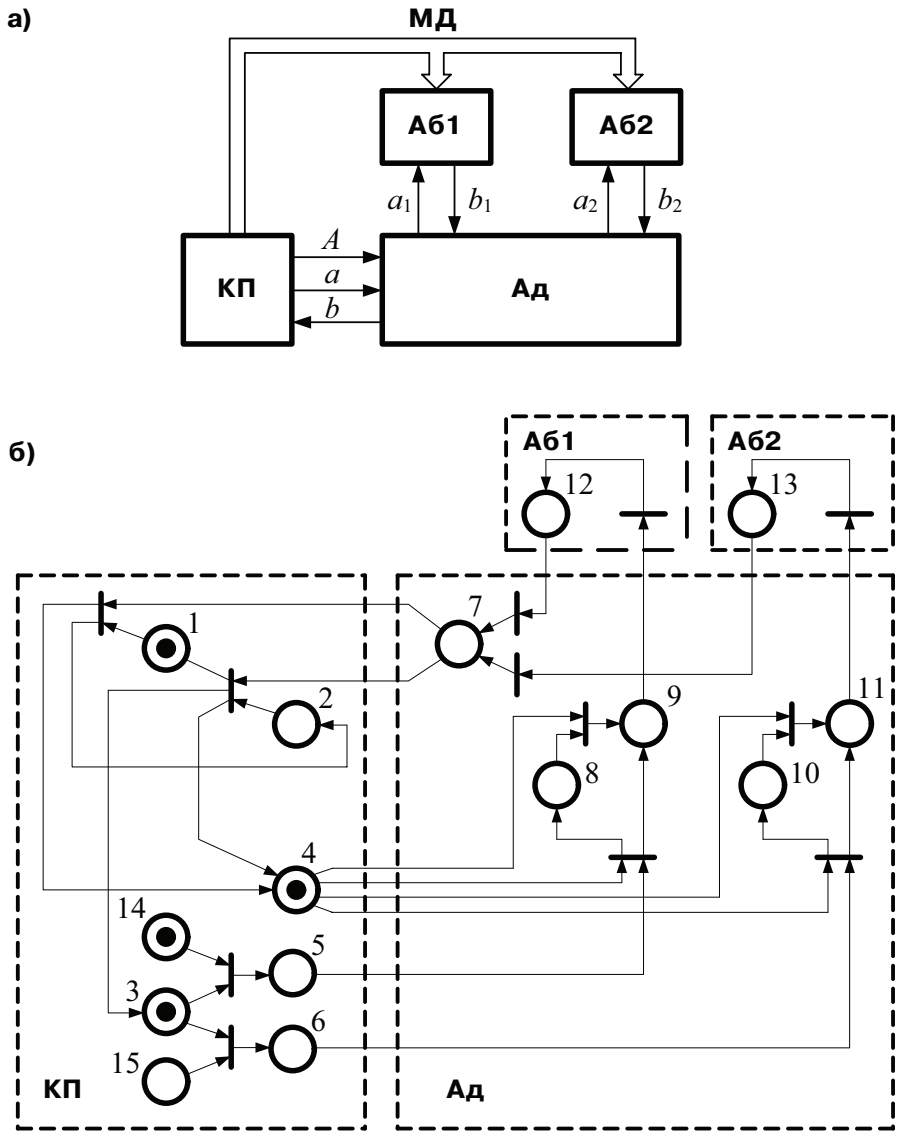


Рис. 7.7. а) Структура системы информационного обмена и б) соответствующий ей протокол

Сеть на рис. 7.7,б относится к классу устойчивых и безопасных СП, что позволяет синтезировать управляющий автомат адаптера методами, изложенными в главе 6.

7.3. Верификация протоколов

В процессе синтеза протокола необходимо осуществлять *поэтапный анализ* (*верификацию*) полученной модели, который состоит в установлении полноты и непротиворечивости описания, а также в выявлении необходимых свойств поведения объектов взаимодействия. **Можно выделить две основные задачи анализа:**

1. **Первая состоит в проверке заданных разработчиком свойств, называемых эффектами взаимодействия, путем исследования всех возможных вариантов взаимодействия объектов и необходимых ситуаций.** Эта задача решается на первых двух этапах синтеза протокола (см. подраздел "*Примерная процедура проектирования протокола*" раздела 3.2).

2. **Вторая задача заключается в проверке реализации каждого участника взаимодействия на соответствие своему исходному описанию (третий этап синтеза протокола).** Для асинхронных интерфейсов решение второй задачи может быть связано с анализом аппаратурной реализации на независимость от задержек элементов и проводных соединений в схемах сопряжения. Основное внимание далее будем уделять решению первой задачи.

7.3.1. Верификация эффектов взаимодействия

Ключевые вопросы анализа протоколов – это выбор способа анализа и выбор совокупности желаемых эффектов. И то, и другое существенно зависит от принятого разработчиком формального аппарата описания протокола.

Выбор способа анализа протокола

В большинстве случаев при использовании динамических моделей, интерпретируемых в терминах асинхронных процессов, процедура анализа основана на исследовании достижимости на множестве ситуаций. Исследование достижимости сводится к перебору всех возможных последовательностей взаимодействия двух или более объектов данного уровня. Полная, или глобальная, ситуация процесса взаимодействия определяется как комбинация (кортеж) состояний взаимодействующих объектов и средств передачи. Вначале задается некоторое исходное состояние и генерируется множество достижимых ситуаций путем выполнения всех допустимых в каждой из ситуаций действий. Такая процедура повторяется для каждой вновь об-

разованной ситуации до тех пор, пока не перестанут встречаться новые ситуации. Для проведения такой процедуры, естественно, необходимо быть уверенным в эффективности (гарантии возможности завершения) процедуры. Для моделей с конечным числом состояний эффективность гарантирована. Однако, например, для сетей Петри требуется ограничить класс сетей, используемых для описания протокола, поскольку вопрос о разрешимости проблемы достижимости некоторой ситуации из исходной ситуации для сетей Петри общего вида пока остается открытым. В некоторых случаях (в частности, при анализе сигнальных графов) могут быть использованы другие способы анализа, отличные от исследования достижимости.

Выбор совокупности желаемых эффектов

Какие же эффекты взаимодействия могут быть установлены в ходе анализа? Выделяются две категории эффектов (свойств) протоколов: общие и частные. *Общие эффекты* присущи всем протоколам. Наиболее примечательными являются следующие:

- ***полнота***, т.е. обеспечение описания объекта при всех возможных входных воздействиях;
- ***отсутствие дедлоков*** – ситуаций (классов ситуаций), из которых недостижима никакая ситуация глобальной модели;
- ***живость*** или ***связность***, означающая, что либо из каждой ситуации достижима любая другая ситуация, либо для каждой достижимой ситуации и любого события достижима ситуация, при которой это событие произойдет;
- ***отсутствие блокировки темпа***, т.е. отсутствие неэффективных циклов – замкнутых траекторий, проходящих через исходную ситуацию, но не проходящих через некоторое подмножество заданных семантикой ситуаций;
- ***самоконтролируемость***–автоматическийвозвратпротоколапослеошибки к нормальному циклу за конечное число действий;
- ***самосинхронизация***, т.е. функционирование независимо от временных характеристик взаимодействующих объектов.

Общие эффекты взаимодействия определяются непосредственно по графу достижимости. Например, ситуации, имеющие только входящие дуги, интерпретируются как дедлоки.

Специфические (частные) *эффекты* требуют проверки обеспечения протоколов определенного сервиса – гарантии достижимости определенной цели протокола. Интерфейсные протоколы, как правило, должны обладать следующими свойствами. Для протокола передачи данных необходима правильная доставка данных от источника к приемнику независимо от задержек модулей и линий связи. Для протокола арбитража запросов основная задача – обеспечение 1) отсутствия "голодной смерти" любого из запрашивающих объектов и 2) монополизации шины одним объектом в любой момент времени. Отсутствие "голодной смерти" означает, что если несколько процессов требуют бесконечно многократного использования ресурса, то нет такого процесса, который никогда не смог бы получить этот ресурс; иногда это свойство называют еще "честной диспетчеризацией".

Наибольший интерес для анализа протоколов представляет использование сетей Петри, поскольку многое эффекты асинхронного взаимодействия легко интерпретируются в терминах свойств этих сетей. Кроме того, в рамках сетей Петри можно выделить подклассы, для которых теоретически доказана алгоритмическая разрешимость многих вопросов анализа, а также установлены оценки сложности процедур анализа.

7.3.2. Выявление свойств протоколов

Рассмотрим, каким образом выявляются свойства протоколов при использовании формальных модельных средств. В качестве примера, иллюстрирующего процесс анализа, фигурирует асинхронный протокол, получивший название "чередующегося бита".

Анализ асинхронного протокола «Чередующегося бита»

Сущность этого протокола состоит в следующем. Взаимодействуют два объекта – источник и приемник, соединенные каналом связи. Источник посылает приемнику сообщение, приемник в случае правильного приема отвечает подтверждением источнику. Каждое сообщение содержит один бит управления. Значение бите (0 или 1) изменяется (чередуется) в последовательных соседних сообщениях. Каждое подтверждение также содержит бит,

равный по значению биту управления подтверждаемого сообщения. Рассмотрим этапы анализа описания *протокола чередующегося бита*.

Модель с идеальным каналом

Источник и приемник связаны через идеальный канал связи. Соответствующая этому уровню сеть Петри показана на рис. 7.8.

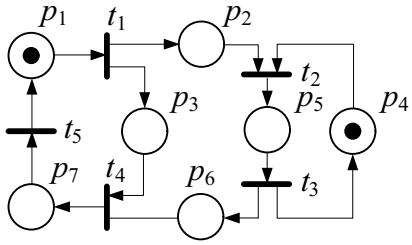


Рис. 7.8. Протокол чередующегося бита при идеальном канале связи

Интерпретация переходов: t_1 – передача сообщения; t_2 – прием сообщения; t_3 – обработка сообщения, связанная с изменением значения двоичной переменной r ($r(i+1) := \bar{r}(i)$), которая определяет текущее значение бита управления в приемнике, причем источнику передается подтверждение с битом управления, равным новому значению r; t_4 – прием подтверждения; t_5 – выработка нового сообщения и изменение значения двоичной переменной s ($s(i+1) := \bar{s}(i)$), которая задает текущее значение бита управления очередного сообщения в источнике. Граф маркировок, изображенный на рис. 7.9, показывает, что протокол задается живой, безопасной и устойчивой сетью Петри.

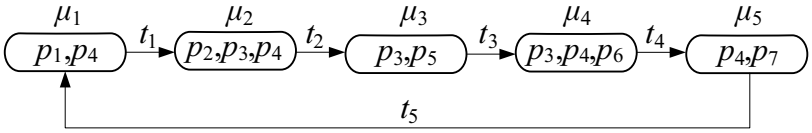


Рис. 7.9. Диаграмма маркировок сети Петри, представленной на рис. 7.8

На этом рисунке запись p_i, p_j означает такую маркировку вершины, при которой $\mu(p_i) = \mu(p_j) = 1$, в то время как остальные позиции не имеют маркеров.

Легко видеть, что поскольку диаграмма маркировок последовательная, протокол можно было бы описать более компактным образом. Однако позиции

p_3 и p_4 понадобятся в дальнейшем, поэтому они сохранены в первичной модели. Для изображения протокола в виде асинхронного процесса каждую ситуацию необходимо представить в виде кортежа (μ_i, s, r). В результате получим АП, показанный на рис. 7.10.

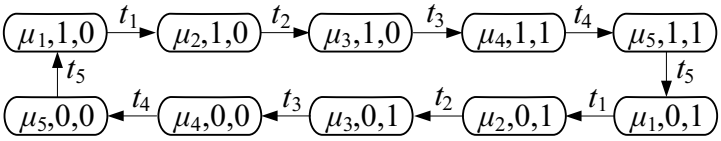


Рис. 7.10. Асинхронный процесс для протокола рис. 7.8

Далее рассмотрим некоторые модификации протокола чередующегося бита, позволяющие учесть реально возникающие эффекты асинхронного взаимодействия. Будет построена некоторая система таких спецификаций, носящая характер пошаговой детализации.

Модель с искажением сообщения в канале

Пусть в канале передачи возможно искажение данных или бита управления. Условимся, что в этом случае при установлении ошибки в сообщении (подтверждении) источник (приемник) повторяет в точности предыдущую посылку, оставляя неизменным также и бит управления.

Соответствующая этому протоколу сеть Петри показана на рис. 7.11.

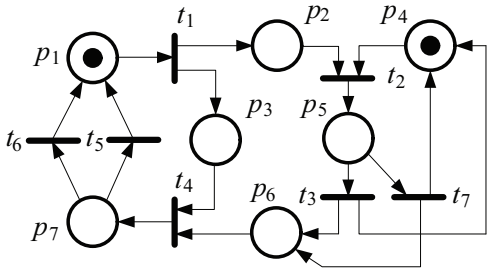


Рис. 7.11. Протокол чередующегося бита с учетом искажений в канале связи

Событие t_6 означает, что сохраняется значение переменной s и принятое подтверждение игнорируется, а событие t_7, – что исключается принятое сообщение и вновь посылается подтверждение с битом, равным значению переменной r. Такая сеть Петри является неустойчивой, та как события t_5 и t_6 (t_3 и t_7) находятся в конфликте при маркировке $\mu_5 = p_4, p_7$ ($\mu_3 = p_3, p_5$). Диаграмма маркировок показана на рис. 7.12.

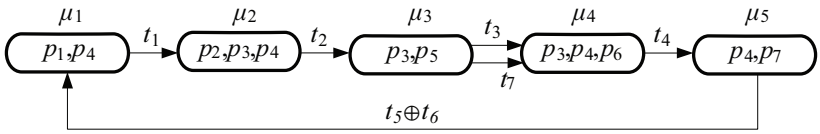


Рис. 7.12. Диаграмма маркировок сети Петри, представленной на рис. 7.11

Из нее видно, что имеет место самосинхронизация, так как всегда восстанавливается начальная маркировка $\mu_1 = p_1, p_4$. Вводятся предикаты, разрешающие конфликт:

$$\mathcal{P}_6 = [\text{ошибка} \vee (r \neq s)], \mathcal{P}_5 = \neg\mathcal{P}_6,$$
$$\mathcal{P}_7 = [\text{ошибка} \vee (r = s)], \mathcal{P}_3 = \neg\mathcal{P}_7.$$

Однако из анализа последовательностей срабатывания видно, что если будут постоянно выполняться переходы t_6 и t_7, то протокол будет работать "вхолостую" и, следовательно, будет иметь место блокировка темпа. Решение этой проблемы может быть выполнено с помощью верхнего уровня, куда можно ввести специальный *таймер*, или счетчик времени, в течение которого разрешена блокировка темпа. По истечении интервала времени, измеряемого таймером, можно произвести, например, перекоммутацию на другой (исправный) канал связи.

Модель с потерей сообщений

Усложним модель, предположив, что в канале связи возможно искажение и потеря сообщений. Как уже установлено, асинхронный протокол в случае возможной потери сообщения попадает в дедлок. Для того, чтобы избежать остановки при взаимодействии, предположим, что источник обладает механизмом таймаута, который включает таймер в момент посылки сообщения и, если по истечении критического интервала не приходит подтверждение от приемника, осуществляет повторную посылку. Сеть Петри, моделирующая такой протокол, показана на рис. 7.13.

Переход t_8 моделирует потерю сообщения, t_9 – потерю подтверждения приема сообщения (здесь использованы фрагменты сетей Петри, называемые "поглотителями маркеров"), переход t_{10} – срабатывание механизма таймаута T по истечении критического интервала τ. Предикаты конфликтных событий задаются следующим образом:

$$\mathcal{P}_8 = (\text{потеря}), \mathcal{P}_9 = (\text{потеря}), \mathcal{P}_9 = \neg\mathcal{P}_8,$$
$$\mathcal{P}_{10} = (T = \tau), \mathcal{P}_4 = \neg\mathcal{P}_9 \wedge \neg\mathcal{P}_{10} = \neg(\mathcal{P}_9 \vee \mathcal{P}_{10}).$$

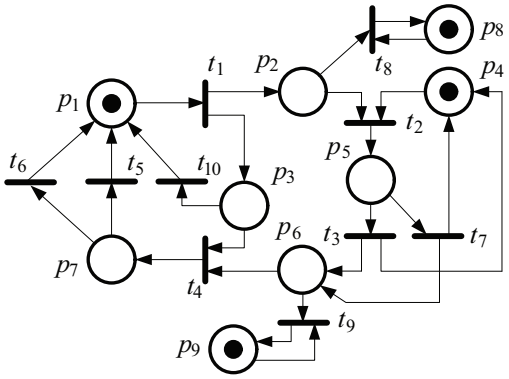


Рис. 7.13. Протокол чередующегося бита с учетом возможности потери сообщения в канале связи

Заметим, что такая модель требует правильного срабатывания механизма таймаута, т.е. лишь в том случае, когда действительно произошла потеря сообщения. Иначе говоря, чтобы сеть Петри работала в соответствии с диаграммой маркировок, показанной на рис. 7.14, необходимо доопределить предикат $\mathcal{P}_{10}$: $\mathcal{P}_{10} = (\mathrm{T} = \tau) \wedge (\mathcal{P}_8 \vee \mathcal{P}_9)$.

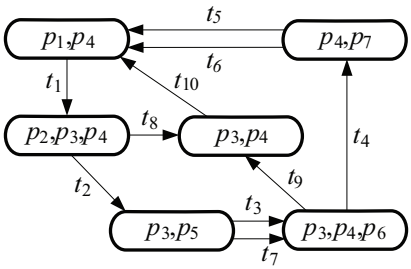


Рис. 7.14. Диаграмма маркировок сети Петри, представленной на рис. 7.13, при срабатывании механизма таймаута только в случае потери сообщения

Другим способом моделирования идеальной работы механизма таймаута служит сеть Петри, изображенная на рис. 7.15.

Поскольку практически невозможно реализовать в источнике условие $\mathcal{P}_{10}$, зависящее от условий $\mathcal{P}_8$ и $\mathcal{P}_9$, так как источник физически не может со стопроцентной гарантией установить, произошла ли потеря сообщения или задержка приема подтверждения оказалась больше критического интервала τ, то реальна ситуация, при которой произойдет срабатывание перехода t_{10} (моделирующего таймаут), и далее будет выдано повторное сообще-

ние, хотя в действительности сообщение не потеряно. Указанная ситуация приведет к появлению в любой из позиций p_2 или p_6 более, чем одной точки.

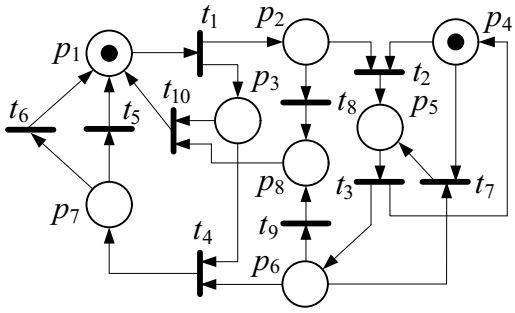


Рис. 7.15. Модификация протокола чередующегося бита

Замечание. Поскольку позиции p_8 и p_9 в "поглотителях точек" (см. рис. 7.13) содержат точки при любой маркировке, то без утраты общности эти позиции в графе достижимых маркировок здесь и далее не будут указываться (для краткости).

Модель с ложным срабатыванием механизма таймаута

Итак, **в случае ложного срабатывания таймаута возможно нарушение безопасности сети Петри**: рост числа маркеров в позициях p_2 и p_6 (рис. 7.13) может быть неограниченным.

Введем следующее ограничение. Пусть таймаут срабатывает лишь один раз, т.е. при выходе из нормального цикла по причине ложного срабатывания таймаута следующее срабатывание таймаута не может произойти до тех пор, пока протокол не возвратится в нормальный цикл. Иными словами, будем считать, что если таймаут сработал ложно один раз, то до момента второго срабатывания либо приходит подтверждение, либо действительно происходит потеря сообщения. В этом случае функционирование протокола будет соответствовать графу маркировок, построенному для сети Петри, представленной на рис. 7.13. При желании читатель может построить этот граф самостоятельно. Поскольку сеть Петри небезопасна, рекомендуем строить граф маркировок без учета возможного накопления маркеров в некоторых позициях.

В графе маркировок можно выделить цикл нормального функционирования для случая, когда нет потери сообщения и таймаут не срабатывает, а также все возможные циклы, в которых нарушается режим нормальной работы.

При работе в нормальном цикле каждое сообщение передается один раз, разумеется, если нет искажения сообщения. В случае ложного срабатывания таймаута каждое сообщение будет передаваться дважды, даже если нет искажений, что является недостатком анализируемого протокола.

Рассмотрим несколько возможных вариантов поведения протокола, представленного сетью Петри на рис. 7.13. Пусть сеть Петри находится в ситуации 1, 4 (маркеры в позициях p_1 и p_4). Источник готов передавать приемнику сообщение с битом $s = 0$. Происходит срабатывание t_1 и переход к маркировке 2, 3, 4. Если в этой ситуации канал работает настолько медленно, что приемник не успевает принять сообщение и срабатывает таймаут (переход t_{10} ведет в маркировку 1, 2, 4), то в канал выдается повторное сообщение с битом $s = 0$ (переход к маркировке 2, 2, 3, 4).

Предположим, что в этот момент приемник принимает первое сообщение (срабатывает t_2, приводя к маркировке 2, 3, 5) и выдает подтверждение на первое сообщение с битом $r = 0$ (срабатывает t_3, приводя к маркировке 2, 3, 4, 6). Тогда после этого источник принимает подтверждение с битом $r = 0$ (маркировка 2, 4, 7), интерпретируя его как потерю одного из двух сообщений и подтверждение другого, уцелевшего сообщения. Источник далее формирует сообщение с битом $s = 1$ (срабатывает t5, маркировка 2, 2, 3, 4). В это время приемник принимает второе посланное источником сообщение с битом $s = 0$ (маркировка 2, 3, 5), но при этом воспринимает его как ошибку, так как он, послав подтверждение с $r = 0$, ожидает новое сообщение с $s = 1$. Отреагировав таким образом (срабатывает t_7, маркировка 2, 3, 4, 6), приемник выдает вновь подтверждение с $r = 0$. Источник, получив это подтверждение (маркировка 2, 4, 7) в ответ на посланное им сообщение с $s = 1$ запоздалое подтверждение с $r = 0$, воспринимает его как ошибку и формирует повторно сообщение с $s = 1$ (срабатывание t_6, маркировка 1, 2, 4).

Далее все сказанное повторяется для бита управления, равного единице. Аналогичные рассуждения можно провести для любого из возможных циклов работы протокола. Заметим, однако, что если при работе в одном из таких циклов произойдет однократная потеря сообщения, то протокол возвращается к нормальному циклу с однократной передачей каждого сообщения (что и требуется). Если при этом предположить, что канал не может по физическим причинам справиться с наличием в нем пары сообщений в одну и ту же или в обе стороны (маркировки, в которых либо пара точек в позициях p_2 или p_6, либо по одной точке в p_2 и p_6), то повторная передача после потери сообщения может выполняться однократно. В этом случае можно сказать, что протокол обладает свойством самокорректировки и самосинхронизации. Вместе с тем, конечно, возможна работа в неэффективных циклах с блокировкой темпа.

Модель без ограничений на емкость канала и число срабатываний таймаута

При анализе вышеописанной модели было введено ограничение на возможность срабатывания механизма таймаута, но при этом все же рассматривалась такая сеть Петри, в которой возможно произвольное количество срабатываний перехода t_{10}. Однако желательно исследовать поведение протокола в общем случае. Итак, что же будет, если таймаут срабатывает неограниченное число раз и нет ограничений на емкость канала связи?

Построение диаграммы достижимости для неограниченной сети Петри – дело нелегкое. Вообще говоря, когда имеется бесконечное число ситуаций, не всегда можно прийти к желаемому результату. Для данного примера удалось найти удобную форму конечного представления бесконечного графа достижимых маркировок в виде диаграммы, являющейся сжатым вариантом полного графа и показанной на рис. 7.16.

В этой диаграмме введены степени позиций, обозначенные ω и γ для позиций p_2 и p_6 соответственно. Текущее значение степени позиции указывает число маркеров, находящихся в данный момент времени в данной позиции. Иначе говоря, всегда можно установить достижи-

мость таких маркировок диаграммы, в которых позиции p_2 и p_6 содержат произвольное число точек (от нуля до бесконечности), а остальные позиции содержат максимум по одной точке и могут входить в состав маркировок только в следующих комбинациях: 1, 4; 3, 4; 1, 5; 3, 5; 4, 7; 5, 7. Дуги между маркировками на диаграмме помечены наименованиями переходов с указание в скобках изменения степени позиции, инцидентной этому переходу.

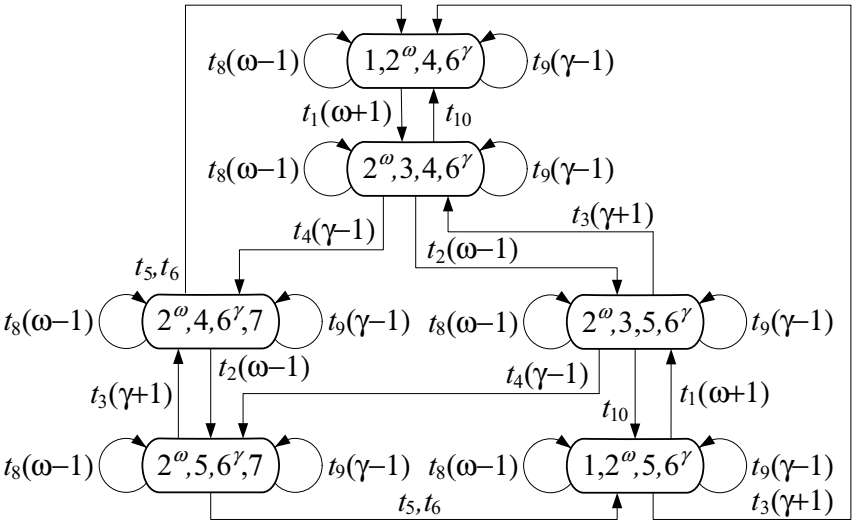


Рис. 7.16. Конечное представление диаграммы маркировок сети Петри, представленной на рис.7.13, для случая, когда число маркеров в позициях 2 и 6 не ограничено

По такому конечному графу можно построить любую часть бесконечного графа маркировок, исходя из начальной маркировки $(1,2^0,4,6^0)=(1,4)$, в которой значения степеней ω и γ равны 0. Необходимо также отметить, что переходы, обозначенные операциями над степенями $\omega-1$ и $\gamma-1$, могут срабатывать при условии $\omega > 0$ и $\gamma > 0$.

Модель с буферированным каналом

Приведенная выше модель протокола в виде сети Петри на рис. 7.13 не совсем адекватна описываемой системе передачи данных, так как в ней не отражены ограничения на число срабатываний механизма таймаута и число сообщений в канале связи. Оказывается, если ввести модель буферированного канала связи, то ряда проблем можно избежать. Сеть Пе-

три, моделирующая буферный канал емкостью в три сообщения, показана на рис. 7.17.

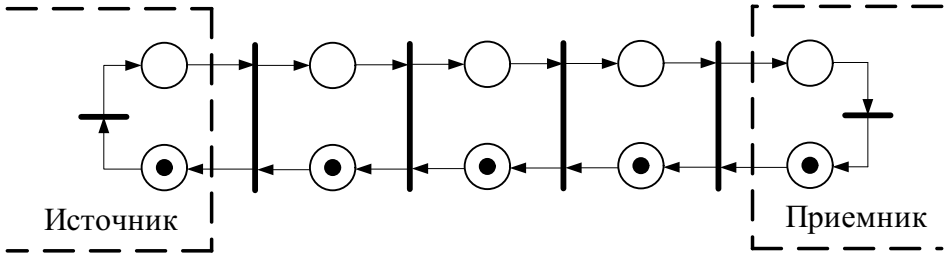


Рис. 7.17. Сеть Петри, задающая буферированный канал связи емкостью в три сообщения

Рассмотри сеть Петри, изображенную на рис. 7.18. Предположим, что канал связи может содержать не более одного сообщения в каждую сторону. Переходы t_8 и t_9 по-прежнему моделируют события, связанные с потерей сообщения. При потере сообщения точка просто перескакивает из позиции p_2 (p_6), моделирующей заполнение канала данными, в позицию p_8 (p_9), моделирующую пустой канал. Легко видеть, что условия нахождения точек в позициях p_2 (p_6) и p_8 (p_9) взаимно ортогональны. Диаграмма маркировок, соответствующая сети Петри на рис. 7.18, содержит 23 маркировки. Предлагается построить такую диаграмму в качестве упражнения.

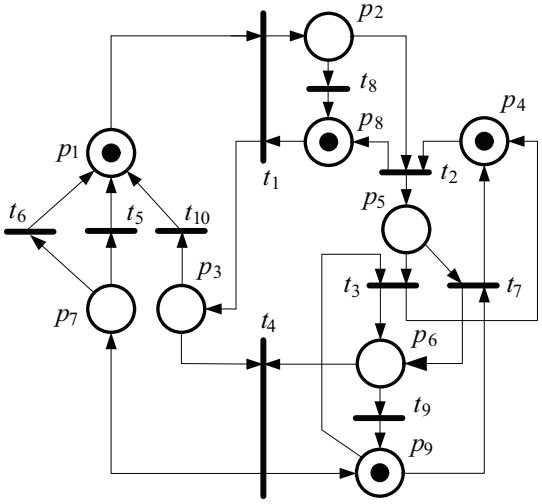


Рис. 7.18. Протокол чередующегося бита с буферированным каналом связи

Приведенная модель протокола не требует никаких дополнительных ограничений, так как сеть Петри в данном случае имеет конечное число достижимых маркировок.

Интересно отметить, что эта модель допускает до трех срабатываний механизма таймаута без получения подтверждения или при потере сообщения. Правда, в случае третьего срабатывания таймаута сеть Петри попадает в маркировку (1, 2, 5, 6) которая при идеальном канале (t_8 и t_9 никогда не могут сработать), является дедлоком. Но поскольку допускается возможность потери сообщения, то эта маркировка не является тупиковой. Кроме того, отметим, что при втором ложном срабатывании таймаута протокол попадает в один из циклов, в котором осуществляется ретрансмиссия одного и того же сообщения (с одним и тем же значением бита управления) трижды, т.е. значительно теряется темп передачи. Возврат к нормальному темпу возможен, как и прежде, при потере двух сообщений.

Итак, были рассмотрены шесть различных моделей протокола чередующегося бита на основе сетей Петри с использованием принципа пошаговой детализации. В ходе анализа были установлены некоторые эффекты асинхронного взаимодействия, возникающие при детализации модели канала связи.

Необходимость применения механизма таймаута продиктована тем, что в случае потери сообщения асинхронный протокол может попасть в ситуацию дедлока. Кроме того, если сеть Петри, задающая интерфейсный протокол, небезопасна, то это может интерпретироваться как наложение сигналов. Поэтому необходимо обеспечить индикацию момента отработки устройств и линий связи на каждое сообщение.

Очевидно, что наложения сигналов (небезопасность) можно избежать поразному. Обычно этот эффект возникает при искусственном введении времени в модель (синхронизация тактовыми сигналами). Так, для устранения дедлока в сугубо асинхронном протоколе вводится механизм таймаута, который задает конечный временной интервал и работает, вообще говоря, независимо от событий, возникающих за пределами источника сообщений. Механизм таймаута нарушает в известной мере принцип асинхронности и, как следствие, приводит к возможности наложения сигналов, или, выражаясь на языке модели, нарушает безопасность сети Петри. Для борьбы с этим эффектом вводится асинхронный буфер на нижнем уровне, который, однако, сам служит причиной возможного дедлока. Но ввиду того, что протокол используется для обмена в заведомо неидеальном канале связи, такой дедлок, конечно, нереален.

7.3.3. Описание протоколов сигнальными графами и диаграммами переходов

Удобными средствами описания интерфейсных протоколов, учитывающими тот факт, что сообщения в межмодульных интерфейсах представляются изменениями логических сигналов на линиях связи, являются сигнальные графы и диаграммы переходов (диаграммы Маллера). Сигнальный граф, используемый для задания протоколов обмена – это маркированный граф, интерпретированный в терминах описываемой интерфейсной функции. Однако модель сигнального графа удобна только для задания протоколов на уровне управляющих сигналов. Некоторые эффекты асинхронного взаимодействия, типичные для несамосинхронных протоколов, требуют учета задержек, вносимых каналами связи и временными характеристиками источника и приемника. Учет задержек приводит к необходимости ревизии первоначальной модели СГ, ее расширения. К числу таких эффектов относится, прежде всего, *явление перекоса задержек*. В чем его суть?

При приеме двух сигналов, передаваемых одновременно по двум параллельным линиям от одного устройства к другому, может возникнуть разница во времени приема этих сигналов, даже если однотипны драйверы линий (усилители), приемники и сами линии (см. рис. 7.19,а). Эта разница, связанная с различными временами распространения сигналов по реальным цепям, называется *перекосом задержек*.

Для интерфейса “Общая шина” перекос задержек гарантируется не более 75 нс. Рис. 7.19,б иллюстрирует зоны неопределенности моментов прихода сигналов на приемник, обусловленные перекосом задержек. Если A и B представляют пару линий шины данных, то максимальная разница во времени приема (перекос) сигналов по этим линиям не превышает 75 нс. Для учета явление перекоса необходимо осуществить синхронизацию параллельной передачи данных таким образом, чтобы приемник своевременно стробировал данные с линий, т.е. фиксировал все необходимые уровни потенциалов (или токов) на линиях, которые были установлены источником. Естественно, это требует введения дополнительной задержки на формирование стробирующего сигнала, компенсирующей перекос сигналов в параллельных линиях. В традиционных асинхронных интерфейсах всегда имеются сигналы (и соответствующие линии) сопровождения и подтверждения данных. Для компенсации перекоса сигнал сопровождения данных выдается источником с задержкой τ_S, равной максимальному значению перекоса, после выставления данных на параллельных линиях.

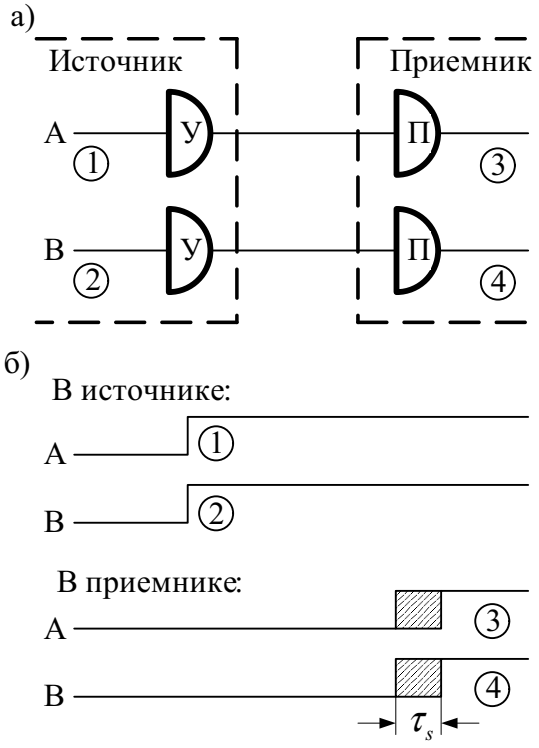


Рис. 7.19. Иллюстрация эффекта перекоса задержек: а) передача сигналов по двум линиям: У – усилители, П – приемники; б) пример перекоса, τ_S – величина перекоса

В *обобщенном сигнальном графе* каждой вершине маркированного графа поставим в соответствие (путем разметки вершин) одно из следующих событий:

- а) изменение состояния некоторой моделирующей переменной;
- б) обозначение встроенной задержки определенного значения, компенсирующей, например, перекос в параллельных линиях, время декодирования адреса выбираемого устройства и т.п.;
- в) "чистые синхронизаторы" – дополнительные вершины, не имеющие семантической интерпретации.

С помощью сигнального графа можно компактно задать упорядочение изменений сигналов на линиях шины, а также некоторых других событий в системе обмена информацией. Для протокола операции чтения данных интерфейса "Общая шина", который был описан неформальными средствами в разд. 7.2, соответствующий сигнальный граф показан на рис. 7.20.

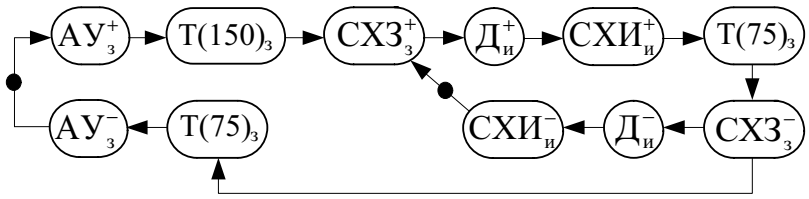


Рис. 7.20. Обобщенный сигнальный граф протокола операции "чтение" интерфейса "Общая шина"

Вершины этого графа помечены изменениями следующих моделирующих переменных:

- **АУ** – переменная, моделирующая состояние шины адреса и управления (состояние АУ = 0 означает, что адрес и биты управления не выставлены на шину, а состояние АУ = 1 – адрес и биты управления выставлены);
- **Д** – переменная, моделирующая состояние шины данных (семантика состояний аналогична переменной АУ);
- **СХЗ, СХИ** – переменные, моделирующие состояния соответствующих линий интерфейса.
- Вершины графа, обозначенные $T(k)$, моделируют встроенную задержку величиной k нс.
- Нижние индексы "з" и "и" в обозначениях вершин соответствуют устройствам (задатчику и приемнику) в зависимости от того, какое из них активизируют данное событие.

Исходная маркировка обобщенного сигнального графа на рис. 7.20 соответствует такому состоянию линий связи, при котором на шинах адреса, управления и данных информация отсутствует, линии СХЗ и СХИ находятся в пассивном состоянии (значение соответствующей переменной равно нулю), временные задержки выключены.

Сравнивая рис. 7.20 с временной диаграммой, представленной на рис. 7.2, легко убедиться в предпочтительности описания протоколов средствами обобщенных сигнальных графов.

Теперь обсудим возможность построения модификации диаграмм переходов, соответствующих обобщенным сигнальным графам, так как в диаграм-

ме Маллера нет средств представления временных задержек определенной величины. Модифицируем диаграмму следующим образом. Например, если после перехода 0–1 сигнала z_i необходимо выдержать время 75 нс, то в тех ситуациях диаграммы, которые соответствуют только что установленному значению $z_i = 1$, в качестве верхнего индекса этого сигнала используется символ d (задержка).

Величина задержки d должна быть указана дополнительно (семантическая информация). Среди непосредственных последователей ситуации, содержащей $z_i = 1^d$, должна быть ситуация, отличающаяся от нее только тем, что переменная z_i равна 1. Типичный пример траектории, которой принадлежат такие ситуации, имеет вид: $...0^*... \rightarrow ...1^d... \rightarrow ...1...$.

Таким образом, модель искусственной задержки является как бы второй разновидностью возбуждения переменной, не требующего переключения значения, в отличие от обычного возбуждения, требующего обязательного переключения значения переменной. Такая диаграмма переходов, показанная на рис. 7.21, задается на множестве состояний, представленных наборами значений моделирующих переменных АУ, СХЗ, СХИ, S. Первые три имеют тот же смысл, что и раньше, а S – переменная, моделирующая сигнал стробирования данных при их чтении с линий шины во внутренний регистр задатчика. Согласно словесному описанию протокола d_1 = 150 нс, а d_2 = 75 нс.

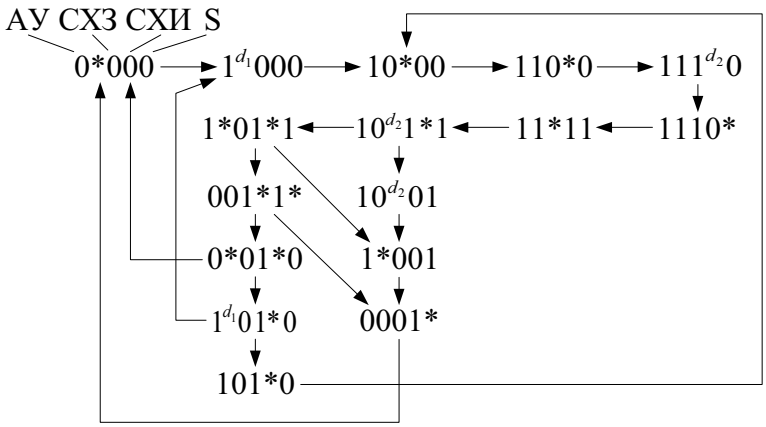


Рис. 7.21. Диаграмма переходов задатчика при операции "чтение" в интерфейсе "Общая шина"

В этой главе были затронуты первые два принципиальных этапа синтеза программно-аппаратных средств сопряжения функциональных модулей

вычислительных систем – спецификация и верификация протоколов. Последующим этапом является выбор способов передачи и кодирования информации, а заключительным этапом – программная или аппаратная реализация сопряжения. Эти этапы будут кратко обсуждены в разд. 8.3.

7.4. Упражнения

1. Объясните, почему протокол, представленный сетью Петри на рис. 7.22, некорректен. Может ли этот протокол быть задан конечным автоматом?

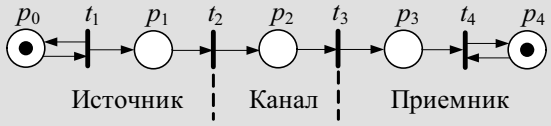


Рис. 7.22. Пример некорректного протокола, в котором возможна неупорядоченная передача пакетов сообщений

2. Пусть протокол асинхронной передачи задан сетью Петри, представленной на рис. 7.23. В нем t_1 интерпретирует генерацию нового сообщения источником, t_2 – передачу сообщения, t_3 – прием и использование сообщения приемником, а t_4 – передачу квитанции.

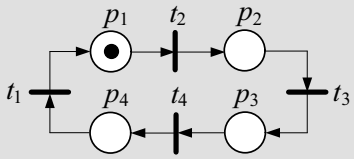


Рис. 7.23. Протокол асинхронного обмена, заданный сетью Петри

Предложите интерпретацию позиций протокола. Изобразите диаграмму маркировок, охарактеризуйте класс СП, задающий протокол. Изобразите этот протокол в виде сигнального графа на уровне обмена управляющими сигналами.

3. Пусть задана система, состоящая из терминала (А) и ВМ (В). Терминал может выдавать сообщение НАЧ (начало обмена), а ВМ – сообщение ГОТ (готов к работе). Оба участника обмена могут выдавать сообщения ИНФ (информация) и НОВ (запрос новой информации).

Для управления обменом в терминале используются четыре команды с верхнего уровня: “Вкл” (включение), “Выкл” (выключение), “Есть В”

(информация для В есть), “Нет В” (информации для В нет). Эти команды можно понимать как директивы с пульта оператора терминала. В ВМ используются команды: “Есть А” (информация для А есть) и “Нет А” (информации для А нет).

Протокол обмена можно описать следующим образом.

1) Обмен инициируется командой “Вкл”. После нее А выдает НАЧ, а В отвечает ГОТ.
2) Если ни А, ни В не желают диалога, то А выдает НОВ и получает в ответ ГОТ. При появлении “Выкл” происходит завершение обмена. Возможен переход к п.1.
3) По установке “Есть В” А выдает ИНФ, а В отвечает НОВ, иначе переход к п.5.
4) Когда А заканчивает передачу последнего сообщения (устанавливает “Нет В”), в ответ на последний запрос из В НОВ А выдает НОВ; после этого переход к п.2.
5) Если В хочет передать информацию (установлено “Есть А”), то он ждет, пока закончит передачу А, а затем отвечает на НОВ из А своим ИНФ, и так продолжается, пока не поступит команда “Нет А”. Иначе – переход к п.2.

Задайте протокол обмена:

а) на языке асинхронных б) на языке сетей Петри; охарактеризуйте класс СП, к которому относится этот протокол.

4. Постройте и проанализируйте диаграмму маркировок для протокола, представленного на рис. 7.18.

5. На рис. 7.24 дано описание протокола непосредственного взаимодействия источника и приемника на языке блок-схем, а на рис. 7.25 тот же протокол задан сетью Петри.

Предложите адекватную интерпретацию событий (переходов) и условий (позиций) этой СП.

6. На рис. 7.26 задан протокол синхронной передачи данных, представленный в рамках модели временной сети Петри, в которой каждому переходу $t_i \in T$ ставится в соответствие пара вещественных чисел $\tau_{\min}, \tau_{\max}$. Если переход стал возбужденным в момент времени τ, то он не может сработать

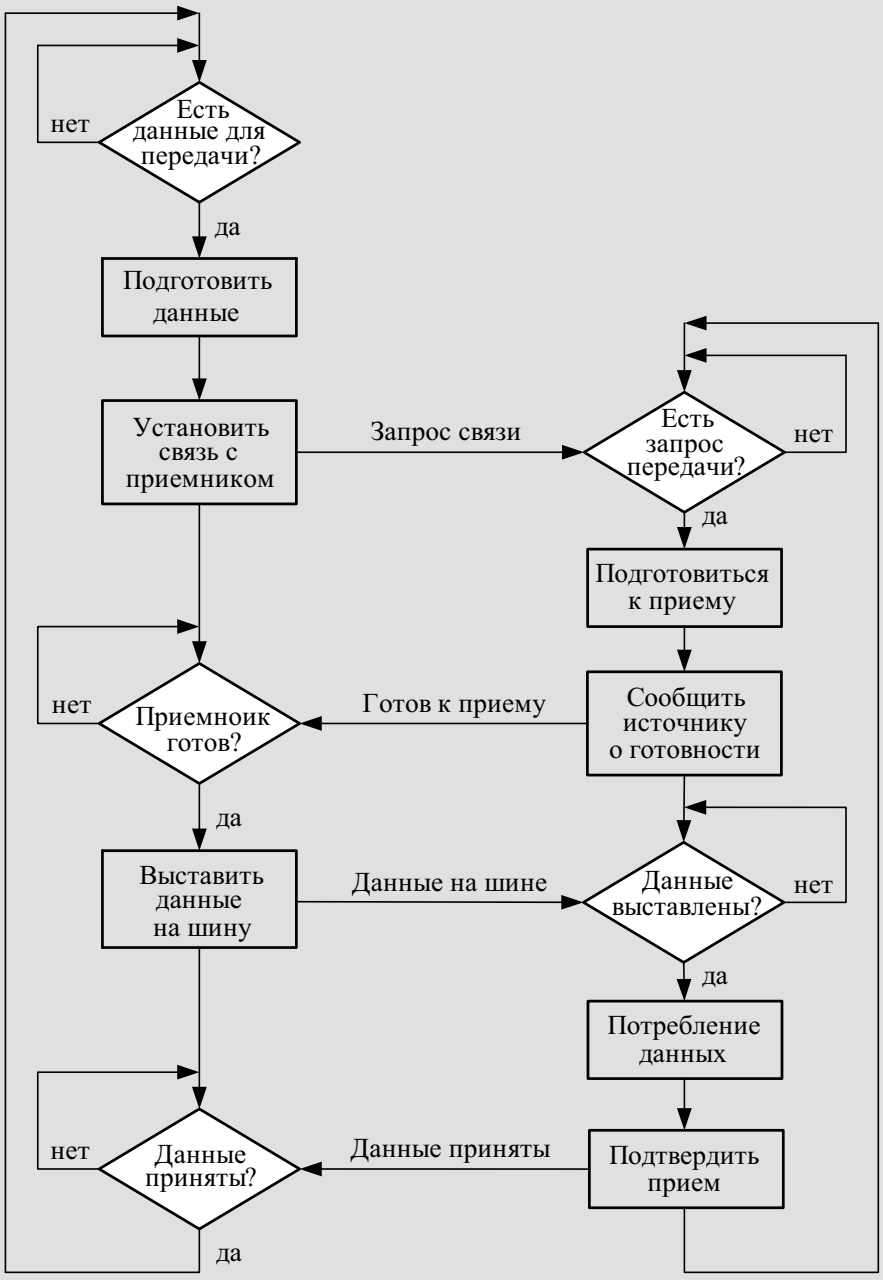


Рис. 7.24. Описание протокола взаимодействия источника и приемника на языке блок-схем

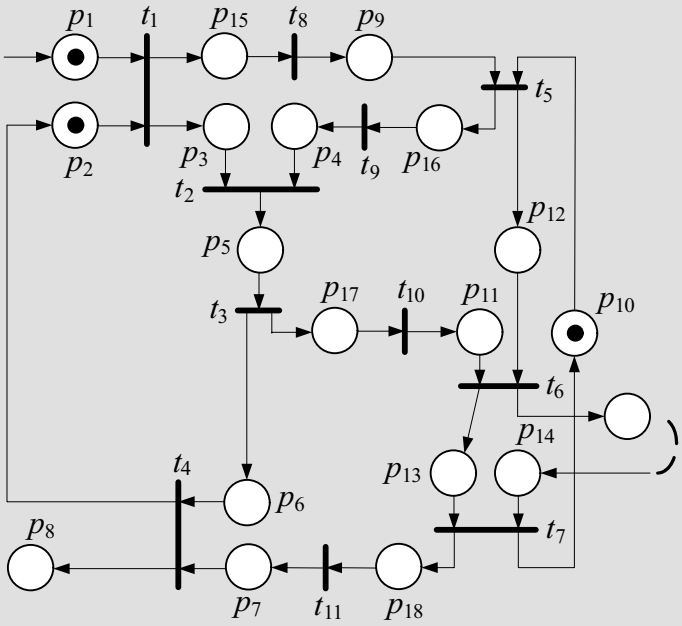


Рис. 7.25. Протокол, заданный сетью Петри

раньше, чем в момент $\tau + \tau_{\min}$ и, более того, должен сработать не позже, чем в момент $\tau + \tau_{\max}$.

Для упрощения будем полагать, что каждому переходу $t_i \in T$ поставлено в соответствие фиксированное время его срабатывания τ_i такое, что $\tau_{\min} \le \tau_i \le \tau_{\max}$.

Определите зависимость τ_1 как функции от τ_3, τ_4 и τ_5, при которой сохраняется безопасность временной СП, т.е. невозможно нарушение порядка следования сигналов от источника к приемнику.

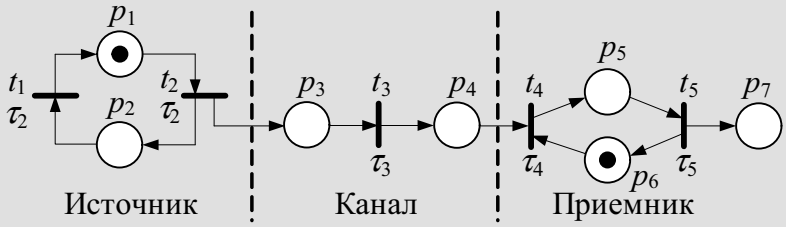
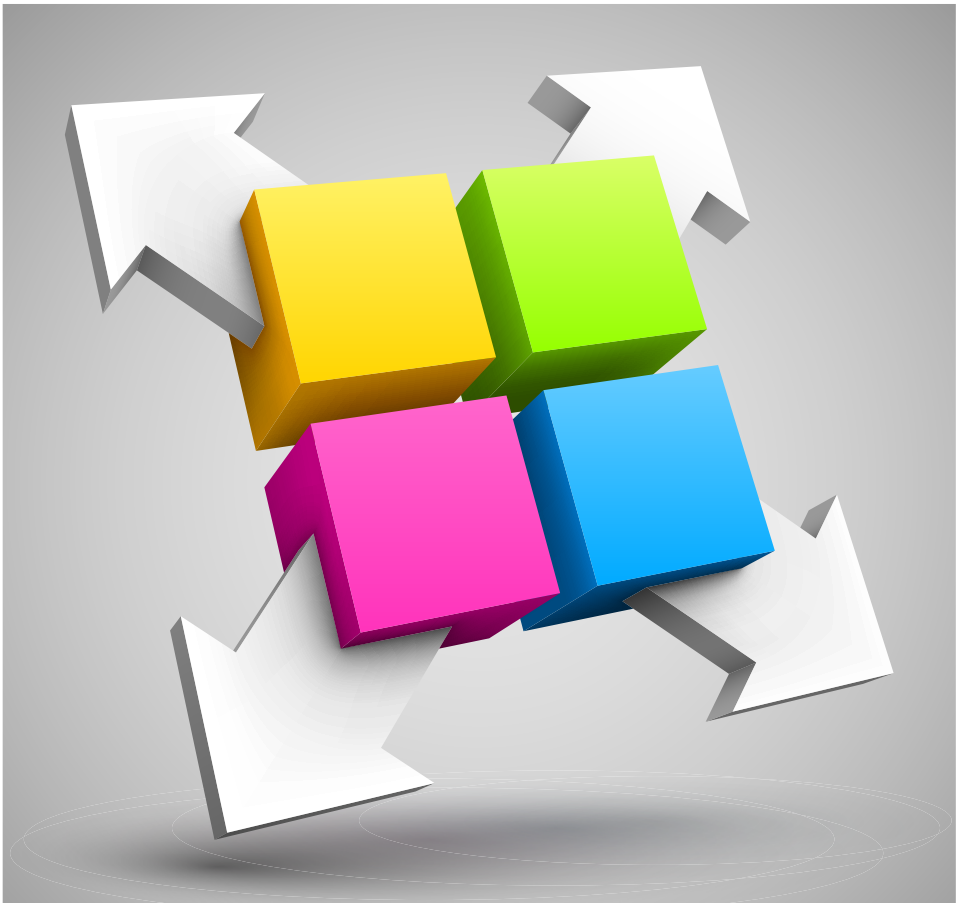


Рис. 7.26. Протокол синхронной передачи данных, заданный временной сетью Петри



Глава 8.

ПРИМЕРЫ РЕШЕНИЯ ПРИКЛАДНЫХ ЗАДАЧ

"Приложение принципа к делу гораздо важнее самого принципа". Д.И. Писарев

Ограниченный объем книги не позволяет сколь-нибудь подробно изложить методологию проектирования программного и аппаратного обеспечения прикладных задач. Такая методология неизбежно должна включать не только материал, составляющий предмет книги, но аккумулировать в себе знания, полученные в рамках смежных дисциплин. Поэтому в этой главе будут приведены только примеры решения практических задач из ряда предметных областей. Эти задачи носят несколько «игрушечный» характер по вполне понятной причине: любая даже сравнительно простая задача оказывается достаточно громоздкой. Здесь действует закон Мьюира: «как только мы захотим отделить какой-нибудь объект от других, мы обнаружим, что он связан со всем на свете». Приведенные примеры, как правило, не доводятся до конца, они лишь иллюстрируют определенные этапы процесса проектирования.

В настоящее время установлено, что проектирование вычислительных и управляющих систем (как и любых других), как правило, сопровождается существенными ошибками, часть из которых удается ликвидировать в процессе разработки, хотя и ценой дополнительных затрат материальных и денежных ресурсов, а часть оказывается скрытой и проявляет себя на последующих стадиях жизненного цикла проектов. Поэтому **одной из передовых концепций сейчас признается концепция гарантоспособности, т.е. обеспечения свойств проектируемых систем, позволяющего обоснованно полагаться на выполнение услуг, для которых она предназначена**. Понятие гарантоспособной системы обязательно включает в себя проверку (верификацию) спецификации проектируемой системы, т.е. ее синтаксическую корректность, и проверку ее адекватности, т.е. ее семантическую корректность. Эти этапы проектирования настолько существенны, что по некоторым данным для систем электронной коммутации в распределении ошибок проектирования ошибки в спецификации составляют около 20%, а ошибки из-за неверного определения условий функционирования (в основном, из-за неточного учета влияния внешней среды) – около 30% (при 50% уровне ошибок, связанных с дальнейшими этапами конструирования).

Значительная доля ошибок проектирования может быть исключена путем применения формальных моделей спецификаций, о которых шла речь в этой книге, и соответствующих методов анализа этих спецификаций. Ниже на ряде примеров показано, как следует составлять спецификацию и верифицировать ее, а также проиллюстрирована возможность перехода от спецификации дискретного устройства к его синтезу. Приведенные решения не претендуют на какую-либо завершенность, исключительность и максимальную эффективность, так как, прежде всего, они рассчитаны на прозрачность восприятия их читателем. Что касается программной реализации спецификаций на некотором наборе модельных средств, рассматриваемых в книге, то она сравнительно легко может быть получена перекодированием этих спецификаций на нужный язык программирования, что выходит за пределы проблематики данного курса.

8.1. Логическое управление

Логическое управление — этот термин часто используется для обозначения проблематики, связанной с построением дискретных (логических) управляющих устройств в таких областях, как промышленная автоматика, гибкие автоматизированные производства и пр.

Первый из рассматриваемых примеров ограничен вопросами составления спецификации на языке сетей Петри по вербальному описанию задачи, где особенно эффективен иерархический подход (отдельно задается функционирование модулей устройства, а также композиции модулей). Это даёт возможность использовать при проектировании принцип модульности (блочное проектирование).

Сети Петри широко используются как средство моделирования динамического поведения процессов самой различной природы. Объекты исследования могут изучаться посредством сетей Петри с разными целями (классификация, выявление различных свойств, реализация и т.п.) на разных уровнях детализации.

Построение описания (спецификация) моделируемой системы является неформальным актом, выполняемым на этапе формулировки технического задания в режиме диалога проектировщика с заказчиком. Этот акт среди всего прочего требует, чтобы некоторым происходящим в реальной или

проектируемой системе событиям к условиям их наступления были поставлены в соответствие понятия СП, т.е. чтобы моделирующая СП была интерпретирована. В интерпретированной СП все или некоторые переходы и (или) позиции могут быть помечены некоторыми символами, удобными для представления семантики функционирования системы, а наличие маркеров в некоторых позициях обычно связывают с истинностью некоторых предикатов. Не останавливаясь на деталях, проиллюстрируем этап построения интерпретированных сетей Петри следующим примером, являющимся типичным для промышленной автоматики.

Первый пример — "Управление манипулятором"

На рис. 8.1,а схематично показан манипулятор, предназначенный для транспортировки груза (детали) w с площадки A на площадку B.

Рука манипулятора с двумя пальцами имеет три степени свободы: она может перемещаться вправо-влево и вниз-вверх при изменении направления вращения двигателей (не показанных на рисунке) горизонтального перемещения X и вертикального перемещения Z, а ее пальцы могут сжиматься или разжиматься посредством двигателя G. Положение руки манипулятора фиксируется по вертикали и горизонтали замыканием концевых переключателей z_1, z_2 и x_1, x_2 соответственно. При захвате детали пальцами срабатывает контакт g_2, при полном разжиме пальцев – контакт g_1. При наличии груза на площадке A или B замыкается контакт d_1 или d_2 соответственно. Транспортировка груза возможна, если d_1 замкнут (на площадке A установлен груз), а d_2 – разомкнут (площадка B свободна). Начальному положению руки соответствуют координаты x_1, z_2 и пальцы разжаты (g_1).

Ясно, что двигатель Z должен останавливаться в положениях z_2 и z_1 и после выполнения условий пуска вниз или вверх работать до новой остановки. Функционирование двигателя может быть описано с помощью различных СП; один из возможных вариантов представлен на рис. 8.1,б.

Наличие маркеров в позициях СП имеет следующую интерпретацию: 1 (2) – выполнены условия запуска в сторону, соответствующую перемещению руки вниз – ВН (вверх – ВВ); 3, c – замкнут концевик z_1; 4, e – замкнут концевик z_2; k – замкнут z_1 или z_2; a – двигатель остановлен; b (d) – двигатель работает, рука движется вниз (вверх), находясь в промежуточном положении между концевиками z_1 и z_2.

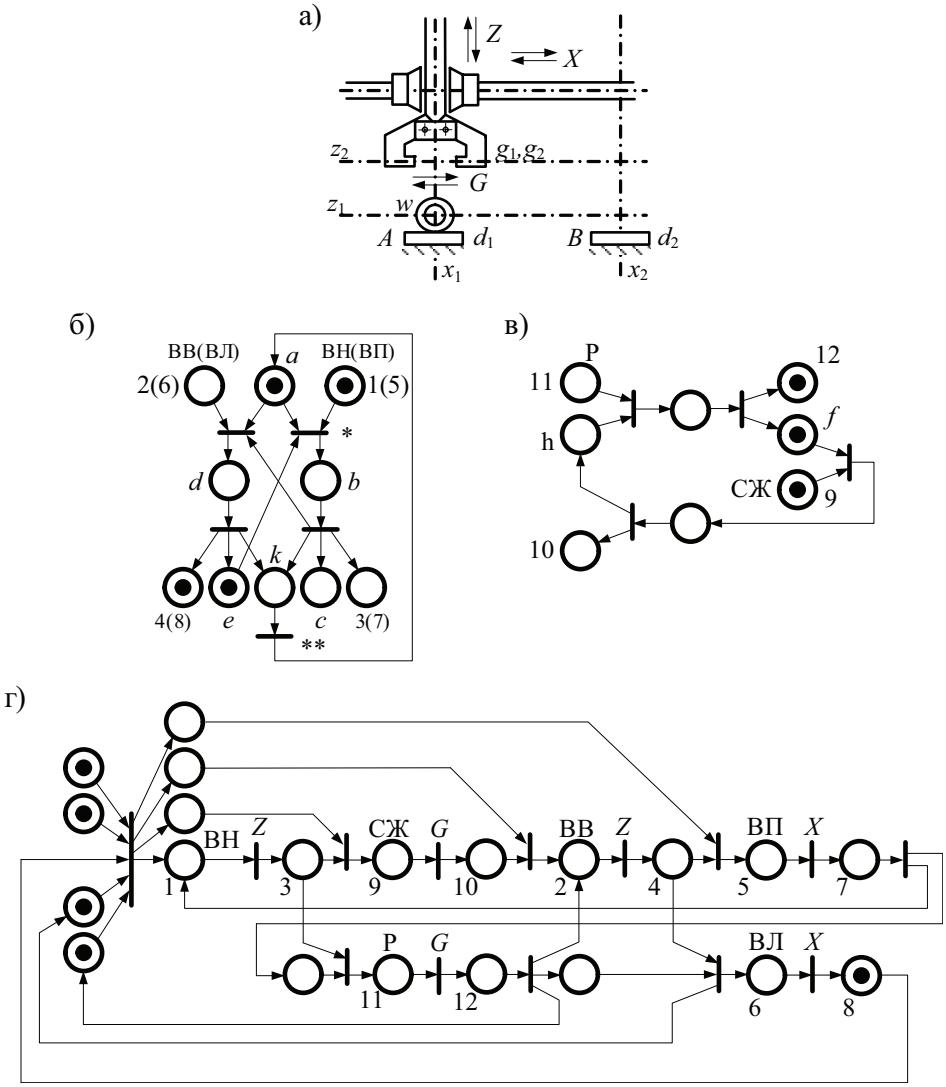


Рис. 8.1. Пример спецификации посредством интерпретированной сети Петри: а) манипулятор, б) управление двигателем вертикального и горизонтального перемещения руки, в) управление двигателем сжимания и разжимания пальцев руки, г) общая модель процесса управления манипулятором

Пусть исходно маркеры находятся в позициях *a*, *e* и 4, а двигатель остановлен в положении z_2. Включение в работу (режим ВН) может произойти только после появления маркера в позиции 1. Тогда после срабатывания перехода, отмеченного одной звездочкой, маркер появятся в позиции *b* и бу-

дет изъят из его входных позиций. После замыкание концевика z_1 маркеры появятся а позициях k, c и 3, а затем, после срабатывания помеченного двумя звездочками перехода, маркер из позиции k переместится в позицию a. В конце режима ВН (вниз) маркеры находятся в позициях a, с. Эта маркировка является исходной для ВВ (вверх), который выполняется аналогично. Появление маркеров одновременно в позициях 1 и 2 исключается в силу того, что возможен лишь один из этих режимов. Заметим, что позиции c, k и 3, (e, k и 4) по смыслу эквивалентны, но если поименованные буквами позиции нужны для организации собственно процессов управления двигателем Z, то отмеченные числами – для организации процесса управления манипулятором в целом.

Работа двигателя X также моделируется рис. 8.1,б; будем иметь в виду, что позиции для этого процесса и названия режимов обозначены символами и числами, заключенными в скобках.

Работа двигателя G описывается проще (рис. 8.1,в), поскольку он должен останавливаться в одном из рабочих состояний – "пальцы сжаты" (позиции h и 10) и «пальцы разжаты» (позиции f и 12). Маркеры в позициях 9 (сжать – СЖ) и 11 (разжать – Р) суть команды на выполнение одного из этих режимов. Два непомеченных условия соответствуют нахождению пальцев в промежуточных состояниях. Исходная маркировка, показанная на рис. 8.1,в, при наличии маркера в позиции 9 переходит в 10, h , а последняя при наличии маркера в позиции 11 – вновь в 12, f.

Управление манипулятором в целом изображается СП, представленной на рис. 8.1,г. На этом рисунке все числовые обозначения позиций соответствуют приведенным на рис. 8.1,*б,в*, а переходы, поименованные большими латинскими буквами Z, X и G, – процессам, изображенным на тех же рисунках. Так, входная позиция 1 и выходная позиция 3 перехода Z – это позиция 1 (ВН) и 3 рис. 8.1,б.

Начальная маркировка, указанная на рис. 8.1,г, соответствует начальному состоянию манипулятора: рука находится над столом A в верхнем положении, пальцы разжаты и выполнены условия d_1 (деталь на месте A) и $\overline{d}_2$ (место B пусто). Из исходного положения работа манипулятора осуществляется по циклу вниз за деталью (ВН) – сжать деталь (СЖ) – вверх с деталью (ВВ) – вправо с деталью (ВП) – вниз с деталью (ВН) – разжать деталь (Р) – вверх без детали (ВВ) – влево без детали (ВЛ). Запуск следующе-

го цикла происходит в том случае, если на месте *A* появится новая деталь, а с места *B* перенесенная ранее деталь убрана.

Рис. 8.1,г демонстрирует возможность задания СП по иерархическому принципу. Такое задание осуществляется за счет соответствующей интерпретации позиций и переходов. Любой переход с инцидентными ему входной и выходной позициями может быть интерпретирован как некоторый подпроцесс, поэтому в модели, отражающей взаимодействие подпроцессов, они могут быть изображены примитивно. При необходимости может быть получена детализация подпроцессов и путем композиции построена общая модель. При желании можно было бы объединить рис. 8.1,б,в и г, но это привело бы к утере наглядности.

Второй пример — "Управление движением вагонеток"

Требуется синтезировать схему управления движением двух передвигающихся по рельсам вагонеток 1 и 2, осуществляющих автоматическую загрузку, транспортировку и разгрузку сыпучего груза (скажем, песка). Эскиз сцены представлен на рис. 8.2.

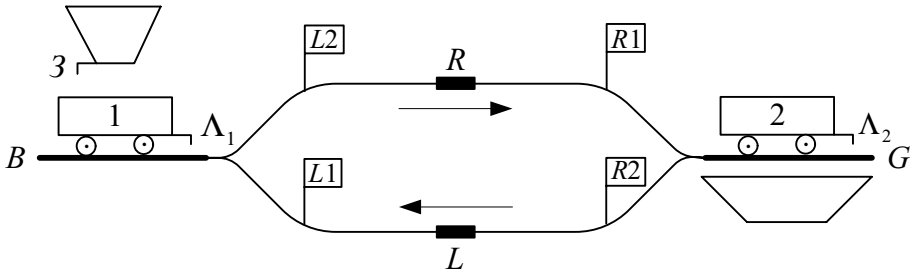


Рис. 8.2. Сцена для задачи о вагонетках

Вагонетка, стоящая в пункте *B* непосредственно под воронкой, загружается посредством открытия заслонки *З* воронки на время, пока песок в вагонетке не достигает верхнего уровня, отмечаемого датчиком. После этого заслонка закрывается.

Разгрузка вагонетки, стоящей в пункте *G*, выполняется открытием находящегося на дне вагонетки люка Λ_i на время, пока песка в вагонетке не останется, что отмечается датчиком нижнего уровня. Груз ссыпается в накопительную емкость.

Закончив загрузку (разгрузку), вагонетка должна переехать в пункт разгрузки (загрузки). Движение осуществляется по двухколейной железнодорожной ветке, но ближайшие подходы к терминальным пунктам являются однопутными. Вагонетки могут одновременно передвигаться либо слева направо по ветке $L2$, $R1$, либо справа налево по ветке $R2$, $L1$, для чего каждая из них снабжена реверсивным двигателем. Естественно, что управление должно содержать механизмы, препятствующие столкновениям вагонеток.

Из соображений, которые станут понятными позже, на рис. 8.2 символами L и R отмечены две промежуточных станции, в которых требуется останавливать вагонетки, если возникает реальная опасность их столкновения, а также четыре семафора: $L1$ (на участке L–B), $L2$ (B–R), $R1$ (R–G), $R2$ (G–L). Семафоры можно толковать как предикаты, принимающие значения 1, если путь открыт, и 0 в противном случае. Кроме того, имеет смысл ввести другие предикаты и пропозициональные переменные, необходимые для адекватного описания функционирования системы:

– переменные x_B, x_R, x_G, x_L, моделирующие датчики, установленные в пунктах B, R, G, L пути и принимающие значение 1 (0), если какая-либо из вагонеток находится в данном пункте (вне него):

– переменные $x^i_{\uparrow}$, $x^i_{\downarrow}$, $i = 1,2$, моделирующие датчики верхнего и нижнего положений сыпучего груза, которыми должна быть оборудована каждая вагонетка, $x^i_{\uparrow} = 1(0)$, если вагонетка полная (неполная), $x^i_{\downarrow} = 1(0)$, если вагонетка непустая (пустая);

– переменная $y_з$, моделирующая состояние заслонки воронки, $y_з = 1(0)$, если заслонка открыта (закрыта);

– предикаты y^i_R, y^i_L, $i = 1,2$, моделирующие состояние исполнительного двигателя i-й вагонетки, $y^i_R = 1(0)$, если двигатель включен для движения вправо (остановлен), $y^i_L = 1(0)$, если двигатель включен для движения влево (остановлен);

– переменные $y^i_\Lambda, i = 1,2,$ моделирующие положение люка i-й вагонетки, $y^i_\Lambda = 1(0)$, если люк открыт (закрыт).

Теперь имеет смысл формально задать процесс p_i движения одной (i-й) вагонетки, i = 1, 2, для чего выберем изобразительные средства сетей Петри. Этот процесс изображен на рис. 8.3.а. Он является чисто последовательным и состоит в циклическом прохождении пунктов B (загрузка), R, G (разгруз-

ка) и *L*. Символами пунктов помечены переходы сети Петри, которая вследствие этого может считаться интерпретированной.

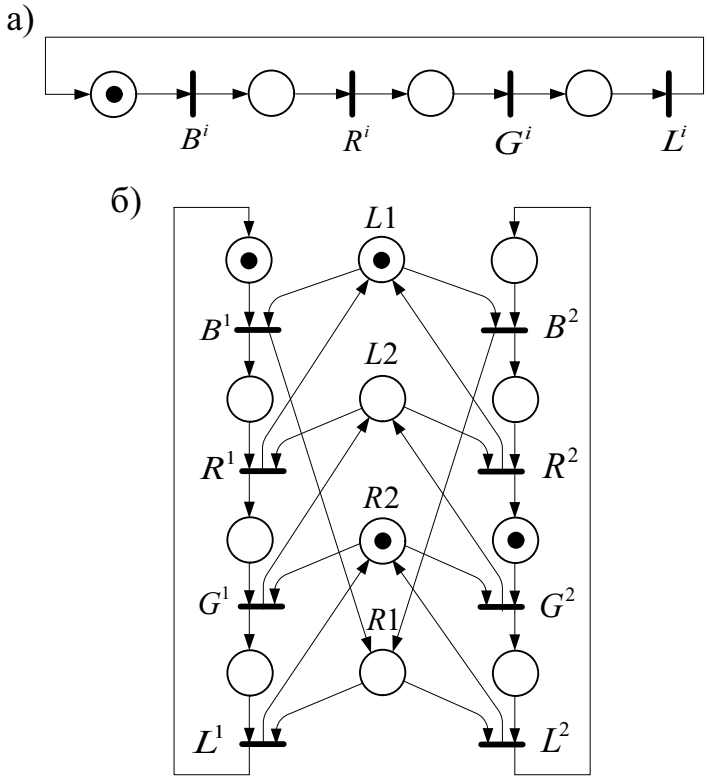


Рис. 8.3. Сети Петри для задания а) процесса движения i-й вагонетки и б) процесса синхронизации вагонеток

Наличие маркера во входной (выходной) позиции перехода трактуется как движение вагонетки из предыдущего (данного) пункта в данный (следующий) пункт, а факт срабатывания переходов *B*, *R*, *G*, *L* означает окончание загрузки, прохождение пункта *R*, окончание разгрузки, прохождение пункта *L* соответственно.

Общая модель синхронизации процессов p_1 и p_2 движения обеих вагонеток, одна из которых исходно находится в пункте загрузки, а вторая – в пункте разгрузки, показана на рис. 8.3.б (верхние индексы в обозначениях переходов СП указывают номера процессов, или вагонеток). Синхронизация осуществляется с помощью позиций, помеченных символами *L*1, *L*2, *R*1, *R*2; механизм реализации синхронизации эквивалентен использованию сема-

форов Э.Дейкстры. Анализ представленной сети Петри показывает, что она является живой, безопасной и устойчивой (невозможно одновременное возбуждение одноименных переходов, т.е. столкновение вагонеток в какомлибо пункте пути). Можно показать, что уменьшение числа семафоров до двух, например, попытка использовать только семафоры $L2$ и $R2$, ведет к нарушению безопасности и устойчивости. Доказательство устойчивости сети Петри, представленной на рис. 8.3,б, следует из рис. 8.4.а, на котором показан граф достижимых маркировок. Начальной маркировке μ_0 на рис. 8.4,а соответствует маркировка, указанная на рис. 8.3,б. Легко видеть, что в этом варианте осуществляется достаточно жесткая синхронизация: выезд одной из вагонеток из данного пункта B, R, G или L возможен только в том случае, когда вторая вагонетка готова к выезду из диаметрально расположенных по отношению к нему пунктов G, L, B или R соответственно. Такое же поведение имеет маркированный граф, представленный на рис. 8.4,б. Чтобы убедиться в этом, достаточно сравнить рис. 8.3,б и рис. 8.4,б.

Наконец, следует отметить, что сигнальный граф, показанный на рис. 8.4,в, который содержит по сравнению с графом на рис. 8.4,б лишние непомеченные вершины, эквивалентен ему, ибо последовательности срабатываний помеченных вершин в обеих графах одинаковы.

Построив формальную модель процесса синхронизации движения, необходимо осуществить детализацию подпроцессов, происходящих “на фоне” движения: включение и выключения двигателей, открытие и закрытие заслонок и люков и пр. Такие подпроцессы удобнее всего задавать на языке сигнальных графов, имея в виду не только способ спецификации, но и дальнейший переход к синтезу схем управления.

На рис. 8.5,а приведен фрагмент сигнального графа (без маркировки дуг), описывающего подпроцесс B^i (G^i), включающий события, связанные с подъездом i-й вагонетки к пункту B (G) и загрузку (разгрузку) песка.

Семантика (интерпретация вершин) ясна из сопровождающих надписей. Знаком «+» отмечены события типа включения, наполнения и открытия исполнительных механизмов, знаком «–" – события типа их выключения, остановки, закрытия и пр.

На рас. 8.5,б представлен фрагмент сигнального графа, описывающий подпроцесс R^i (L^i). Эти фрагменты непосредственно реализовать нельзя, так как по семантике требуется введение дополнительных переменных $t^i_{R(L)}$ и

$t^i_{B(G)}$, «запоминающих» текущие места пребывания i-й вагонетки на соответствующих участках пути; без них не удается координировать совместное движение вагонеток. Поэтому от фрагментов процессов B^i (G^i) и R^i (L^i), показанных на рис. 8.5,а и б, осуществлен переход к их модификациям $\mathbb{B}^i$ ($\mathbb{G}^i$) и $\mathbb{R}^i$ ($\mathbb{L}^i$), представленным соответственно на рис. 8.5,в и г.

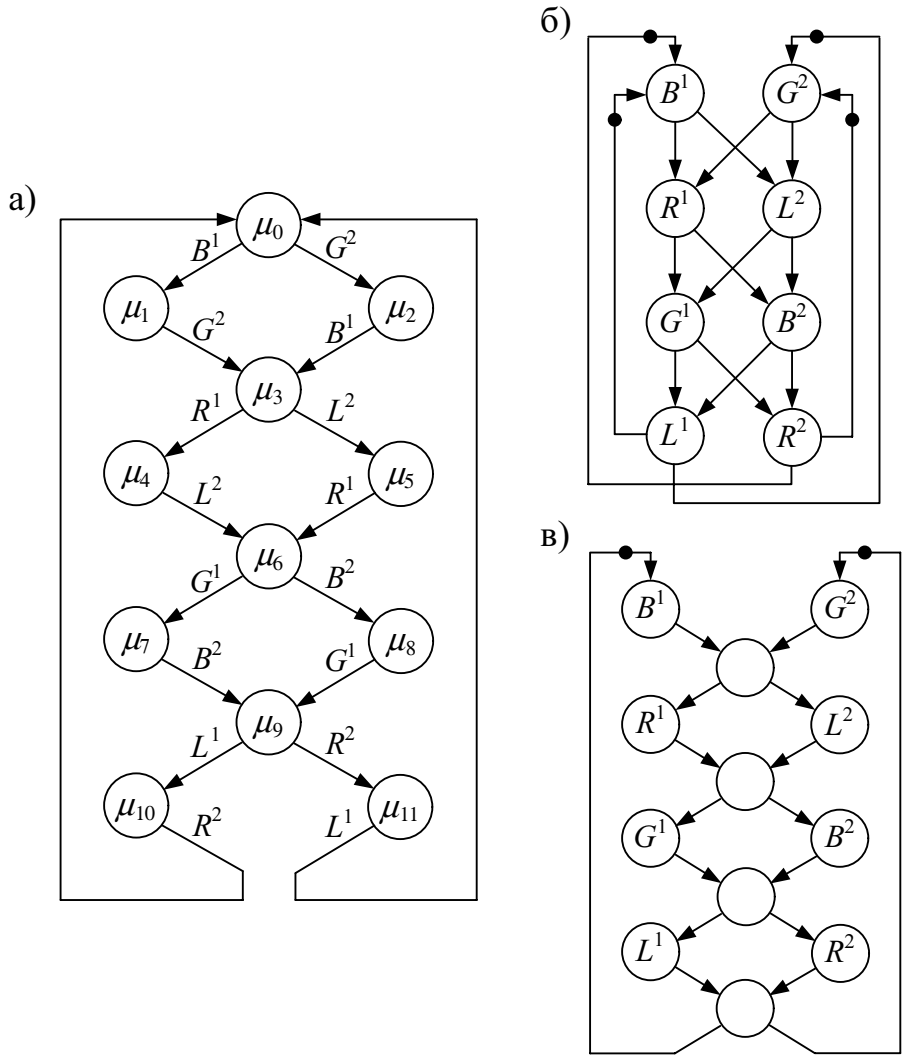


Рис. 8.4. Графы, соответствующие СП на рис.8.3,б: а) диаграмма маркировок б) маркированный граф без дополнительных вершин и в) маркированный граф с дополнительными синхронизирующими вершинами

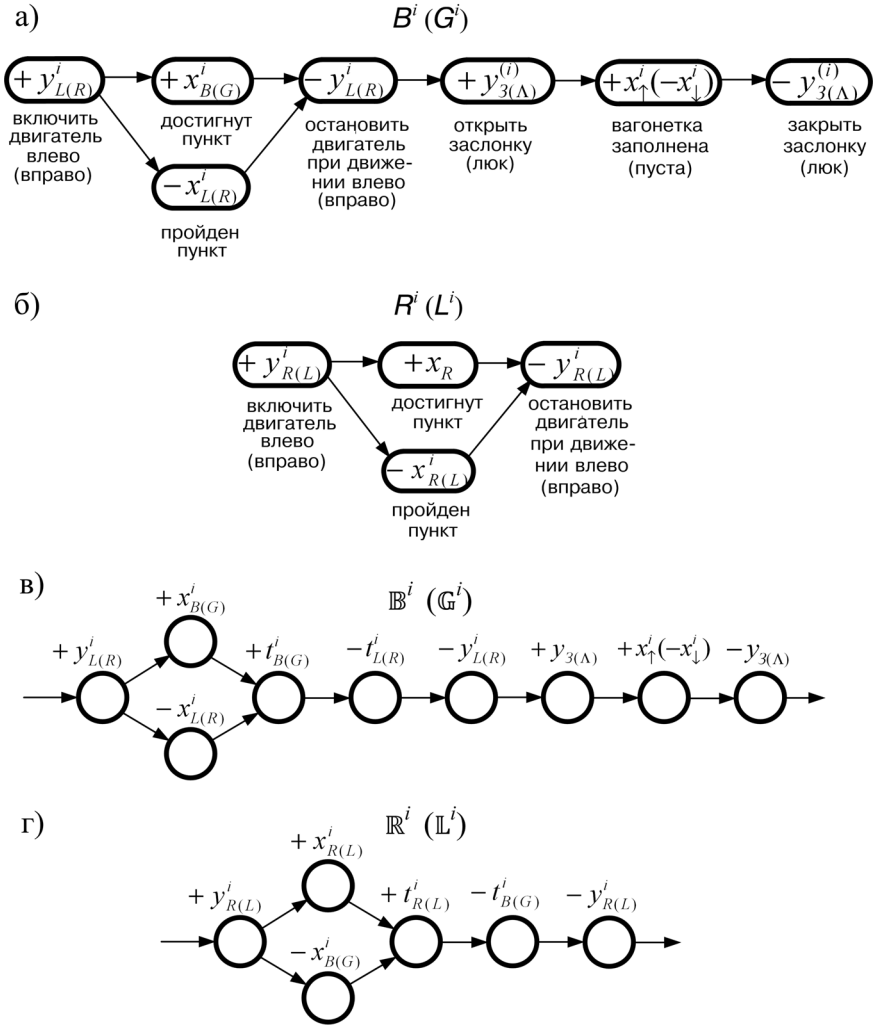


Рис. 8.5.Фрагменты сигнальных графов для задачи о вагонетках: а) процесса $B^i(G^i)$, б) процесса $R^i(L^i)$, в) процесса $\mathbb{B}^i(\mathbb{G}^i)$ и г) процесса $\mathbb{R}^i(\mathbb{L}^i)$

Далее требуется перейти к окончательному представлению сигнального графа, блочно изображенному на рис. 8.6. В нем подпроцессы синхронизированы способом, указанным на рис. 8.4,в; отличие заключается в том, что вместо каждой непомеченной вершины используется пара вершин, обозначенных символами $+s_j$ или $-s_j$. $j = 1,2,3,4$. Введение этих дополнительных переменных обосновывается синтаксическими требованиями к сигнальному графу как спецификации самосинхронных схем. Анализ графа, показан-

ного на рис. 8.6, свидетельствует о том, что он удовлетворяет этим требованиям.

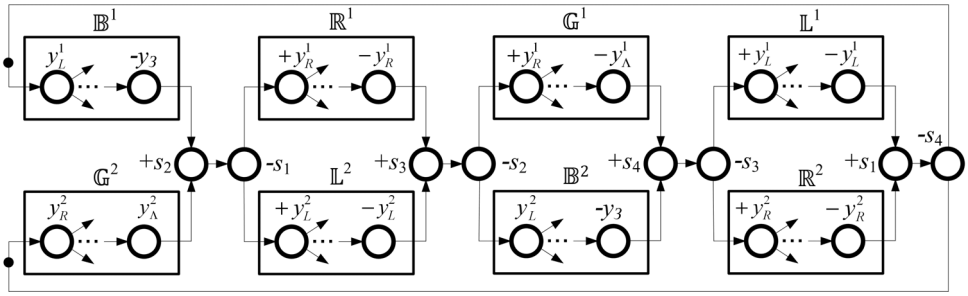


Рис. 8.6. Сигнальная интерпретация задачи о вагонетках

Полученная спецификация не является, естественно, единственной. Можно предложить и другие, в том числе обладающие большей степенью параллелизма. Их существование вызывается неточностью вербального описания задачи, и возможность той или иной детализации спецификации зависит от знаний и интуиции разработчика. Например, можно изменить условие выезда вагонетки. Пусть выезд ее из данного пункта *B*, *R*, *G* или L возможен после выезда второй вагонетки из следующего по направлению движения пункта *G*, *L*, *B* или R. Соответствующий маркированный граф показан на рис. 8.7. Для достижения более высокого уровня параллелизма по сравнению с графом, представленным на рис.8.4,б, здесь потребовалось расщепить каждый процесс на два подпроцесса: а) достижения пункта и работы в нем, б) выезда из пункта.

Переход к самосинхронной реализации также не является однозначным. Здесь выбран стандартный канонический подход, не требующий от проектировщика глубоких познаний в области теории самосинхронных схем, но требующий владения приемами совершенной реализации, при которой каждой переменной ставится в соответствие *RS*-триггер, поведение которого задается автоматным уравнением $z = S \vee \overline{R}z$. Функции возбуждения *S* и *R*, фигурирующие в этом уравнении, должны быть:

- 1) ортогональными, т.е. должно удовлетворяться условие $SR = 0$,
- 2) представлены в сокращенной, а не минимальной нормальной форме.

Оставляя в стороне способ нахождения функций возбуждения *RS*-триггера по сигнальному графу (рис. 8.6 и рис. 8.5,в,г) , ограничимся тем, что приведем функции для каждой из фигурирующих в сигнальной графе переменной.

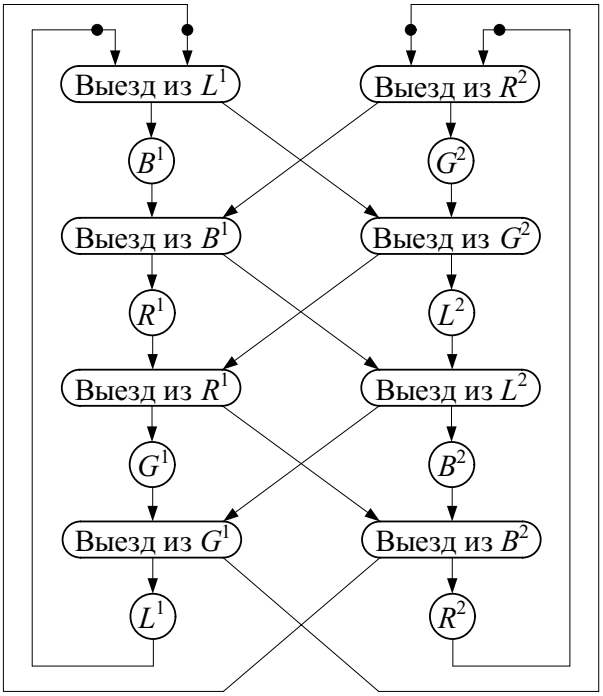


Рис. 8.7. Модифицированный маркированный граф для задачи о вагонетках

$$y^i_{R(L)}: \ S = t^i_{B(G)}\bar{s}_{1(3)} \vee t^i_{R(L)}\bar{s}_{2(4)}, \ \ R = \bar{t}^i_{B(G)}s_{2(4)} \vee \bar{t}^i_{R(L)}s_{3(1)};$$

$$y_з: \ S = \bar{y}^1_L t^1_B \bar{x}^1_\downarrow \vee \bar{y}^2_L t^2_B \bar{x}^2_\downarrow, \ \ R = x^1_\uparrow x^1_\downarrow s_1 \vee x^2_\uparrow x^2_\downarrow s_3;$$

$$y^i_\Lambda: \ S = \bar{y}^i_R t^i_G x^i_\uparrow, \ \ R = \bar{x}^i_\downarrow \bar{x}^i_\uparrow;$$

$$t^i_{R(L,G,B)}: \ S = x_{R(L,G,B)}\bar{x}_{B(G,R,L)}t^i_{B(G,R,L)}, \ \ R = t^i_{G(B,L,R)};$$

$$s_1: \ S = \bar{y}^1_L t^1_L \bar{y}^2_R t^2_R s_4, \ \ R = s_2;$$

$$s_2: \ S = \bar{y}_з x^1_\uparrow \bar{y}^2_\Lambda \bar{x}^2_\downarrow s_1, \ \ R = s_3;$$

$$s_3: \ S = \bar{y}^1_R t^1_R \bar{y}^2_L t^2_L s_2, \ \ R = s_4;$$

$$s_4: \ S = \bar{y}^1_\Lambda x^1_\downarrow \bar{y}_з \bar{x}^2_\uparrow s_3, \ \ R = s_1.$$

Схемы триггеров, реализующих типовые функции приведенной выше системы, представлены на рис. 8.8. Все триггеры переключаются через тран-

зитное состояние (0,0). На рис. 8.8,а показан триггер x_R, который по прибытии вагонетки в пункт R (срабатывании датчика, выдающего на вход триггера сигнал R) устанавливается в состояние $x_R = 1, \bar{x}_R = 0$, а после выезда вагонетки из пункта R – в состояние $x_R = 0, \bar{x}_R = 1$. Аналогично реализуются триггеры $x_L, x_B, x_G, x^i_\uparrow, x^i_\downarrow$. Триггеры y^i_R, y^i_L реализуются в соответствии с рис. 8.8,б, s_1, s_2, s_3, s_4 – в соответствии с рис. 8.8,в, y^i_Λ – в соответствии с рис. 8.8,г, t_R, t_L, t_G, t_L – в соответствии с рис. 8.8,д, $y_з$ – в соответствии с рис. 8.8,е.

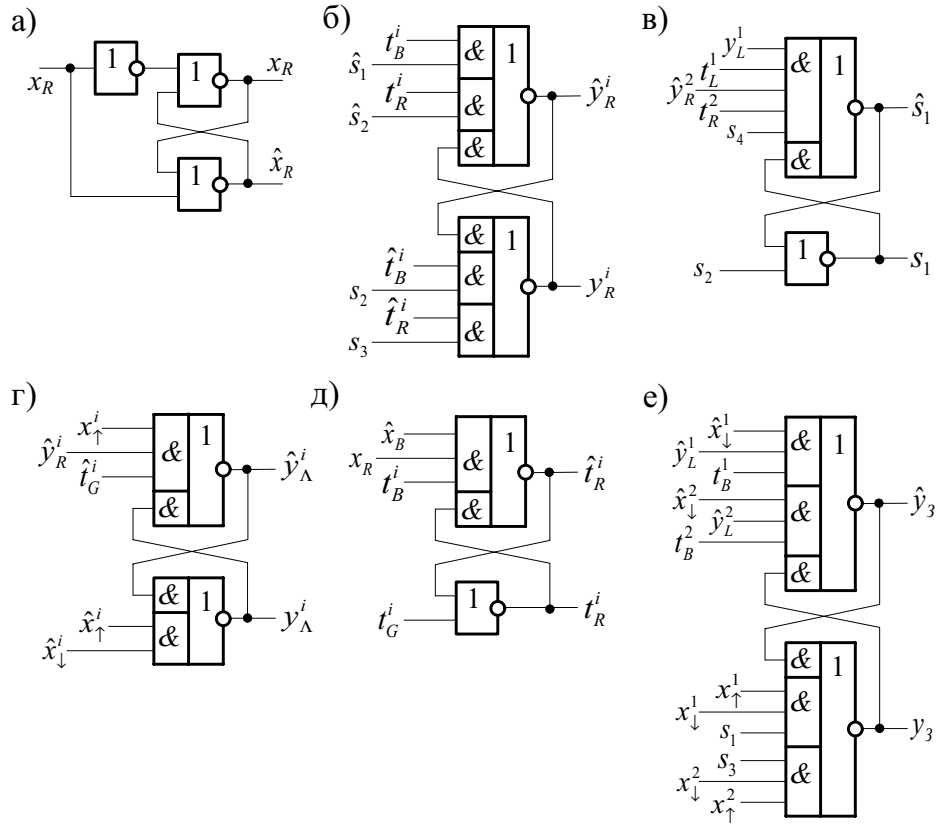


Рис. 8.8. Реализация управления в задаче о вагонетках

Общая схема управления содержит 22 триггера, которые соединяются в соответствии с уравнениями функций возбуждения триггеров, причем вместо переменных, содержащихся в правых частях уравнений в прямом (инверсном) виде используется выход плеча, не содержащего (содержащего) крышку над его обозначением.

Полученное решение должно быть верифицировано. Имеются адекватные средства математического обеспечения (например, FORCAGE или PETRIFY). Система уравнений, соответствующая реализации, вместе с начальным состоянием схемы вводится в компьютер. Результатом анализа является сообщение о том, что анализируемая схема принадлежит классу самосинхронных схем или, в случае неудачи, указываются имевшие место нарушения полумодулярности. После корректировки схемы вновь требуется провести верификацию. При проектировании самосинхронных схем по верифицированным спецификациям с использованием развитой САПР этап верификации схем в принципе не требуется, однако, к сожалению, такой системы автоматизации проектирования самосинхронных схем пока не существует. Даже PETRIFY предусматривает интенсивное человеко-машинное взаимодействие, что требует верификацию действий человека.

8.2. Протоколы асинхронных схем

В этом разделе будет рассмотрен пример, на котором демонстрируется возможность использования сигнальных графов как эффективного средства протоколирования функционирования асинхронных схем. В качестве такого примера используется схема *JK*-триггера на восьми элементах И-НЕ, изображенная на рис. 8.9,а. На рис. 8.9,б приведена временная диаграмма, поясняющая его работу.

Приведем вербальное описание функционирования триггера, заимствованное из авторского свидетельства на изобретение, и обсудим далее это описание.

“При сигнале логической единицы на выходе 8 (4) переключение входного сигнала 9 (10) из логического нуля в логическую единицу вызывает переключение элемента 1 (5) в логический нуль. После этого переключается 2 (6) в логическую единицу, затем элемент 6 (2) в логический нуль и элемент 7 (3) в логическую единицу.

Переключение сигнала 9 (10) из логической единицы в логический нуль вызывает переключение элемента 1 (5) в логическую единицу. После этого одновременно переключаются элементы 3 и 8 (7 и 4) в логический нуль, а затем элемент 4 (8) переключается в логическую единицу. При этом состояние входа 10 (9) не влияет на работу триггера, пока на выходе 4 (8)

действует сигнал логического нуля. Если в момент переключения элемента 4 (8) в логическую единицу на входе 10 (9) поддерживается сигнал логического нуля, то на этом переключение элементов триггера заканчивается, и если на входе 10 (9) действует сигнал логической единицы, то переключение элемента 4 (8) в логическую единицу вызовет переключение элемента 5 (1) в логический нуль. После этого последовательно переключаются элементы 6 (2) в логический нуль, 2 (6) в логический нуль и 3 (7) в логическую единицу...

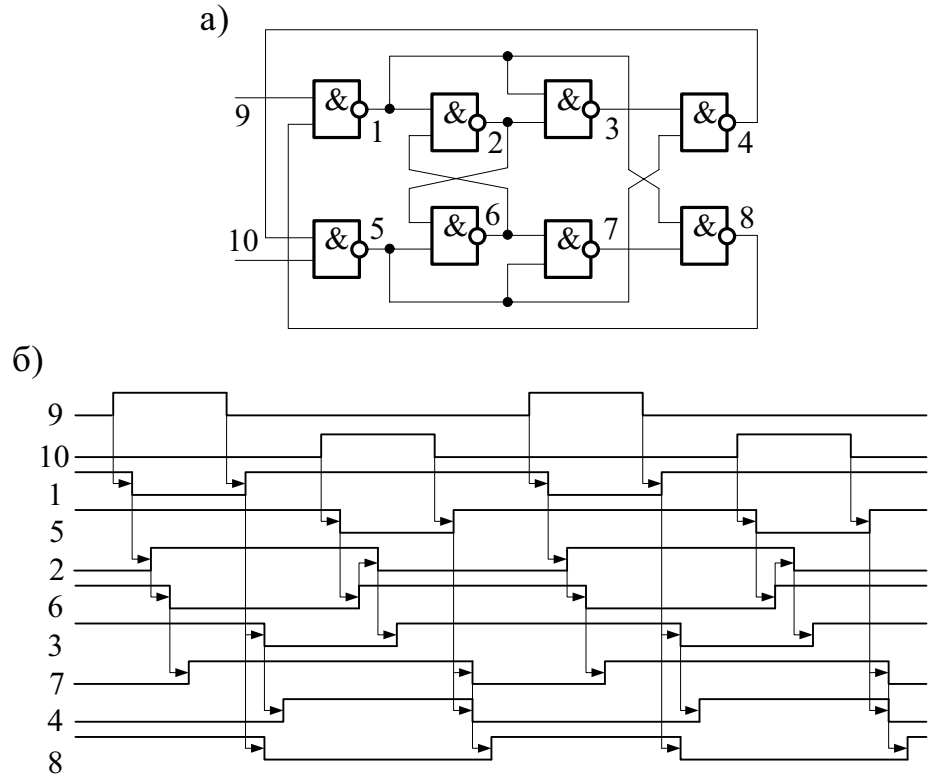


Рис. 8.9. Асинхронный JK-триггер: а) логическая схема и б) временная диаграмма ее работы

Как видно из временной диаграммы (рис. 8.9,б), минимальная длительность паузы входных импульсов для предлагаемого устройства, построенного на логических элементах И-НЕ, определяется временем переключения только трех логических элементов (1, 3, 4 или 5, 7, 8) вместо четырех для прототипа".

Использование триады "принципиальная схема – текстовое (вербальное) описание – временная диаграмма" является типичным в научно-технической литературе. Но, к сожалению, как будет далее показано на примере, трудно адекватно понять, как эти объекты функционируют в динамике. Наиболее уязвимы в этой триаде две последние компоненты. В части текстового описания это связано с лингвистическими особенностями самого языка, которым порождаются только цепочки слов, а параллельные события не могут описываться одновременно.

Громоздкость, неполнота, противоречивость, избыточность, недетерминированность и др. особенности естественного языка в разной степени препятствуют пониманию смысла текстов, в данном случае описаний изобретения. Опасной является сама иллюзия легкости понимания описания, которая порождает возможность неправильного или неоднозначного понимания принципа взаимодействия частей (элементов) устройства (схемы). Именно поэтому в практике описания объектов изобретений сложилась устойчивая традиция иллюстрировать текстовое описание такими средствами, каким являются *временные диаграммы, циклограммы* и т.п.

Отметим, что эти средства являются столь известными и привычными для всех контингентов людей, имеющих дело с электроникой (и не только для них), что определения соответствующих понятий не найти даже в широкодоступных справочниках, не говоря уже о ГОСТ, ЕСКД и специальной литературе. Однако кажущаяся естественность временных диаграмм часто оборачивается недоразумениями.

Для представления сложных параллельных процессов временные диаграммы приходится модифицировать: используют так называемые *пространственно-временные диаграммы*, в которых изменения (фронты) сигналов, являющиеся причиной последующих изменений, отмечаются дугой, идущей от причины к следствию. Наличие более одной такой дуги, ведущей к одному фронту, означает синхронизацию изменений сигналов-причин. Это помогает более точной трактовке последовательности событий, но существенно загромождает диаграмму, делая ее неудобочитаемой.

Проанализируем вышеприведенное словесное описание, используя модель (язык) сигнальных графов. Второй абзац описания приводит к фрагментам сигнальных графов, представленных на рис. 8.10, а и б. Третий абзац трактуется в терминах сигнальных графов так, как показано на рис. 8.10,в и г, но такая трактовка требует пояснений. Действительно, неясно, как следует по-

нимать фразу: «После этого одновременно переключаются элементы 3 и 8 (7 и 4) в логический нуль, а затем элемент 4 (8) переключается в логическую единицу»? По-видимому, слово “одновременно” следует понимать в данном контексте как сообщение о том, что после этого параллельно начинаются процессы переключения элементов 3 и 8 (7 и 4). Но тогда неясно, когда затем переключается 4 (8) (после переключений обоих элементов 3 и 8 (7 и 4) или какого-то одного из них?). Рассмотрение соответствующих фрагментов временной диаграмм (рис. 8.9,б) проясняет эту неясность: оказывается, что лишь –3 (–7) вызывает +4 (+8).

Следующая фраза второго абзаца словесного описания: "Если в момент переключения элемента 4 (8) в логическую единицу на входе 10 (9) поддерживается сигнал логического нуля, то на том переключения элементов триггера заканчивается". Эта фраза трудно поддается интерпретации в терминах сигнальных графов, и поэтому соответствующие ей средства на рис. 8.10,а-г не приведены.

Рассматривая фрагменты сигнального графа, показанного на рис. 8.10,а-г, убеждаемся в том, что они:

1) несвязны (не соединены друг с другом), так что текстовое описание является фрагментарным и уже потому неполным;
2) пересекаются, т.е. содержат общие подфрагменты (действительно, фрагменты “в” и “г” рис. 8.10 полностью включают в себя фрагменты “б” и “а” соответственно), поэтому текстовое описание избыточно и отличается тавтологичностью;
3) противоречивы (дважды изображены вершины +9 и +10);
4) не обладают полнотой описания, так как отсутствует механизм, моделирующий процесс изменения сигналов на входах 9 и 10 схемы.

Теперь рассмотрим сигнальный граф, показанный на рис. 8.10,д. Он корректно описывает работу схемы и лишен недостатков: несвязность и неполнота устранены с помощью дуг, отмеченных пунктирными линиями, произведено склеивание фрагментов, введена начальная маркировка, отражен механизм изменения входных сигналов, с помощью которого совместное функционирование источника сигналов и схемы изображается автономным сигнальным графом.

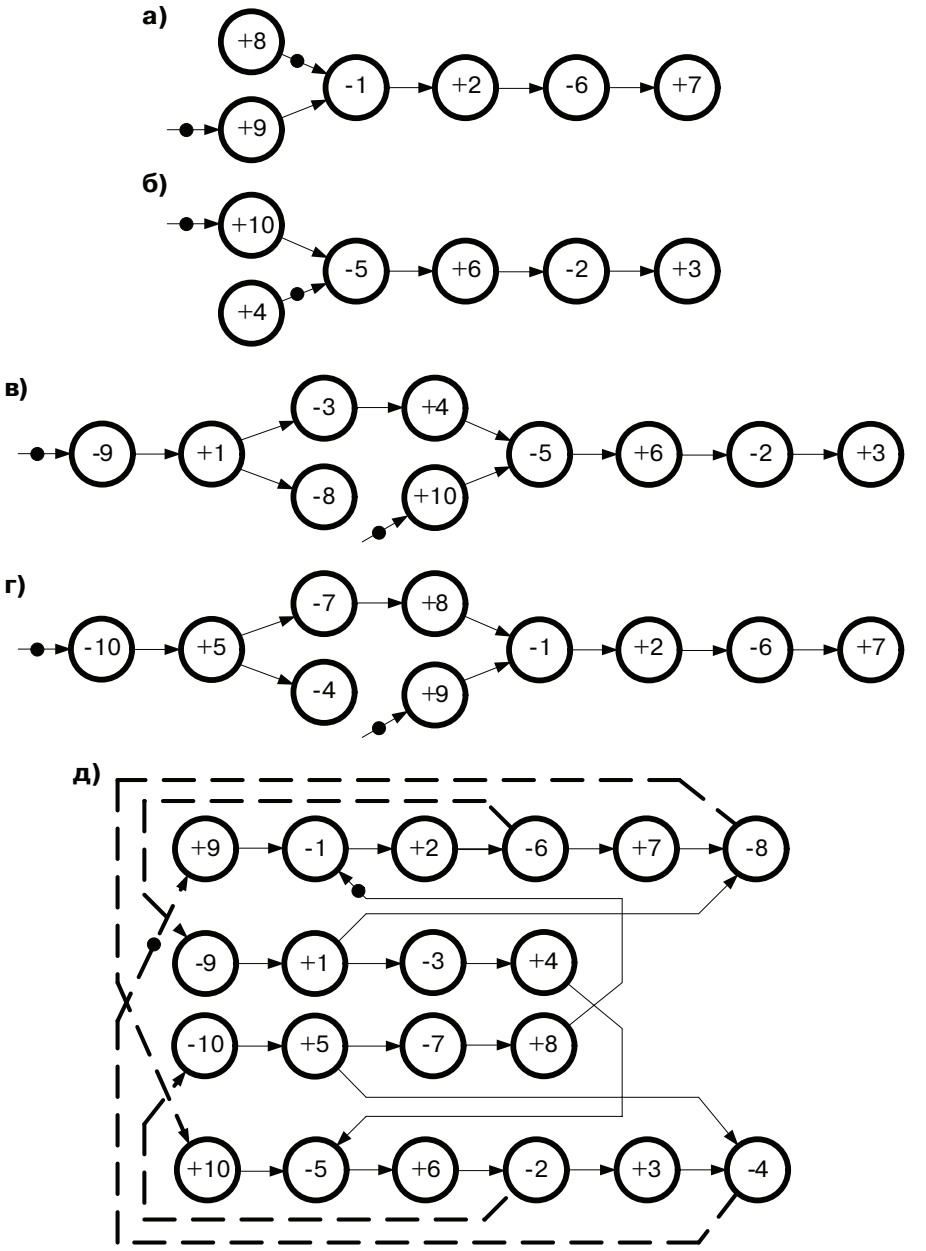


Рис. 8.10. Фрагменты сигнального графа, иллюстрирующие вербальное описание работы JK-триггера на рис.8.9,а: а) – г) в динамике и д) полный протокол

В ходе построения сигнального граф на рис. 8.10,д установлен еще один факт, не отраженный в описании изобретения, а именно то, что дугами соединены вершины +7 и –8 (+3 и –4). Только этот факт позволяет установить достижение автором изобретения первой цели (повышение быстродействия по сравнению с прототипом). Минимальный промежуток времени между изменениями значений сигналов на входах 9 и 10 определятся суммарным временем срабатывания элементов цепочки –1+2–6 (–5+6–2) для изменения входов 9 (10) из логической единицы в логический нуль и цепочки +1–3+4 (+5–7+8) для обратного изменения входов 9 (10).

Вторая часть обоснования достижения цели в описании отсутствует, и, как следствие, не рассматривается механизм взаимодействия триггера с внешней средой (репозиция), который изображается дугами, связывающими вершины, представляющие изменения сигналов на выходах элементов схемы, с вершинами, представляющими изменения входных сигналов. В данном случае – это помеченные пунктиром дуги на рис. 8.10,д.

Итак, сигнальный граф, представленный на рис. 8.10,д, адекватно отражает поведение схемы рис. 8.9,а в динамике. По существу он является протоколом функционирования асинхронной схемы. Такой протокол является альтернативным способом описания динамики, не требующим её вербального описания и задания временной диаграммы. К тому же сигнальный граф явно проще временной диаграммы. Но главное преимущество предлагаемого подхода состоит в том, что непосредственно по сигнальному графу (в данном случае, рис. 8.10,д) можно выполнить формальный анализ (вручную или с помощью компьютера), целью которого является верификация функционирования схемы.

Из ранее приведенного материала ясно, что сигнальные графы способны протоколировать схемы с ограниченным параллелизмом, а именно дистрибутивные схемы, если говорить о самосинхронных реализациях. Полумодулярные, но недистрибутивные схемы не могут быть полностью запротоколированы с помощью сигнальных графов, однако их дистрибутивные фрагменты удобно описываются сигнальными графами.

8.3 Информационный обмен

В этом разделе будет приведен еще один пример, проливающий свет на методику непосредственного перехода от спецификации к схеме. Этот пример относится к построению устройств сопряжения в распределенных системах. Специфика этой проблемной области заставляет использовать некоторые средства расширения модели сигнального графа.

Дополним модель СГ возможностью использования еще одного типа вершин, обозначенных через $T(k)$, которые моделируют введение встроенной задержки, равной k единицам времени; естественно также допустить обозначения $T(\geq k)$, $T(\leq k)$ и $T(k_1 \leq, \leq k_2)$, смысл которых понятен без объяснений. Будем также допускать вершины типа δx_i, снабженные временной атрибутикой типа $k_1 \leq t(\delta x_i) \leq k_2$. Такую модификацию графа называют *временным* СГ.

Обычно протоколы информационного обмена задаются временными диаграммами. Так временная диаграмма протокола операции “чтение” интерфейса “Общая шина”, использующая сигналы АУ (моделирование состояние шины адреса и управления), Д (состояние шины данных), СХЗ и СХИ (состояние управляющих линий интерфейса задатчика и исполнителя соответственно), выглядит так, как показано на рис. 8.11,а.

Необходимость указания с помощью стрелок мест синхронизации и без того осложняет достаточно громоздкую форму представления протоколов. Предлагаемый подход, связанный с применением временных СГ, существенно упрощает средства описания и само понимание протокола, который для данного примера изображен на рис. 8.11,б.

Для облегчения работы с протоколом вершины, обозначенные овалами и прямоугольниками, используются для обозначения сигналов исполнителя и задатчика соответственно. Временные задержки 150 нс и 75 нс, вносимые вершинами $T(\geq 150)$ и $T(\geq 75)$ выбраны в стандарте “Общая шина” для компенсации явления “перекоса задержек”, о котором будет сказано далее специально.

Итак, на временном СГ вершина типа $T(k_1 \leq , \leq k_2)$ (чисто временная вершина) или $\delta x_i (k_1 \leq t(\delta x_i) \leq \text{k2})$ (с атрибутом времени) срабатывает не ранее, чем через k_1 единиц времени, и не позже, чем через чем через k_2 единиц (после своего возбуждения).

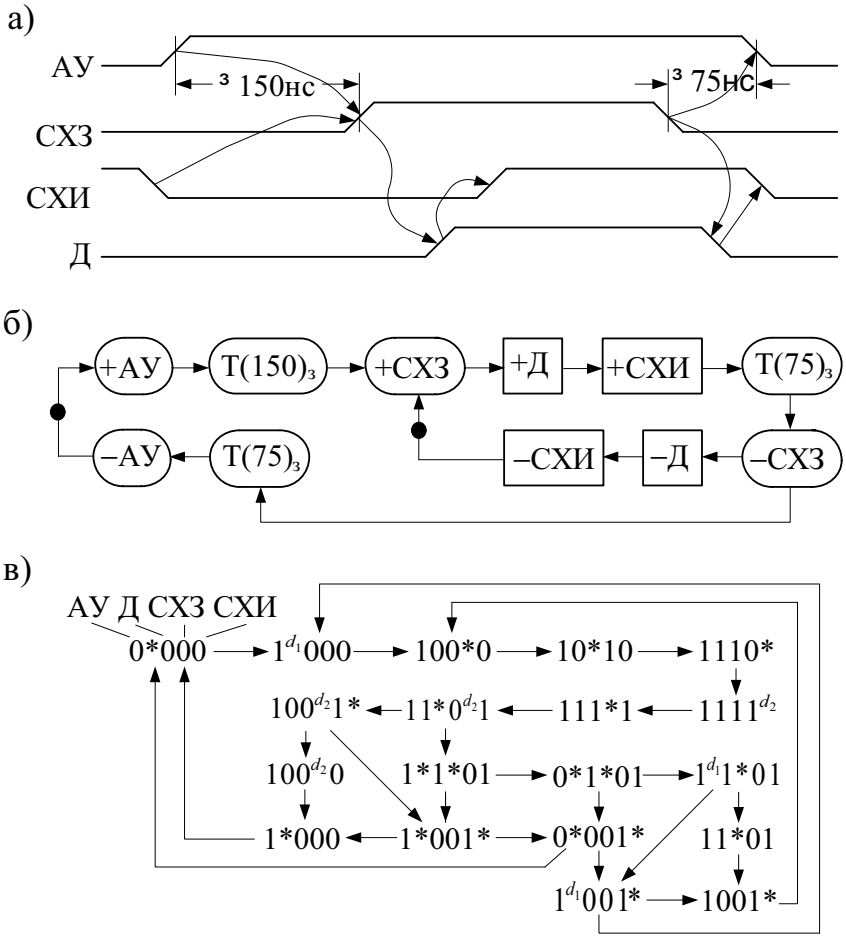


Рис. 8.11. Протокол операции "чтение" интерфейса "Общая шина": а) временная диаграмма, б) временной сигнальный граф, в) временная диаграмма переходов, $d_1 \geq 150$ нс, $d_2 \geq 75$ нс

В общем случае процедура проектирования модели протокола состоит в следующем. На первом этапе строится общий сценарий взаимодействия, который включает определение перечня основных сигналов или сообщений, а также базовой последовательности этих сигналов. На этом этапе создания протокола строится первичная модель обмена в виде СГ. На втором этапе она уточняется путем введения новых сигналов в силу необходимости учета реальных параметров линий связи, способа кодирования сигналов и т.д. Третий этап (построение окончательной модели) обычно связан с последующей реализацией протокола. Здесь требуется учет влияние про-

токолов верхнего уровня и более точная спецификация самих участников обмена с учетом их структуры: регистры, драйверы, чувствительные элементы и пр.

В процессе синтеза протокола необходимо осуществлять поэтапный анализ (верификацию) полученной модели, о которой говорилось в главе 7. Не претендуя на полноту освещения в предыдущей главе всех вопросов анализа и верификации протоколов, достаточно хорошо представленных в литературе, уделим в данной главе основное внимание проблеме самосинхронизации. Для детального рассмотрения этой проблемы необходимо продемонстрировать способы перехода от спецификации на языке СГ непосредственно к самосинхронной реализации с учетом того, что она должна обладать свойством полной *самопроверяемости* и, следовательно, возможностью диагностирования константных консервативных неисправностей. Это делает реальным аппаратное решение задачи *реконфигурации* деградирующей системы путем организации *саморемонта*.

От временного СГ можно перейти к соответствующей диаграмме переходов, которая должна содержать средства, учитывающие фиксированные интервалы задержек. Поэтому в такой диаграмме помимо символа звездочка используется символ задержки *d*, величина которой должна быть указана дополнительно. Например, если после переключения сигнала x_i необходимо выдержать 75 нс, то в тех состояниях диаграммы, которые соответствуют только что установленному значению $x_i = 1$, в качестве верхнего индекса используется символ *d*. Среди непосредственных последователей состояния, содержащего $x_i = 1^d$, должно быть состояние, отличающееся от него только тем, что переменная x_i в нем равна 1. Таким образом, в отличие от обычного возбуждения, требующего обязательного переключения возбужденной переменной, в полумодулярных диаграммах вершина типа задержки не связана с переключением значения переменной.

Возвратимся к примеру протокола, заданного временным СГ на рис. 8.11,б. От этого СГ можно перейти к временной ДП, показанной на рис. 8.11,в. Далее хотелось бы, непосредственно работая с протоколом, заданным временной ДП, “оттранслировать” его в описание схем сопряжения, т.е. получить схемотехническую реализацию задатчика и исполнителя, причем такую, поведение которой не зависело бы от задержек элементов. Однако выполнить такую трансляцию формально невозможно в силу следующих причин:

1) сам протокол не содержит почти никакой информации о структуре участников обмена; действительно, описывается не поведение участников взаимодействия, а лишь процесс обмена сигналами на стыке между ними;

2) ДП содержит кратные состояния, т.е. порождающий эту ДП сигнальный граф не является нормальным;

3) не ясно, как учесть состояния, отмеченные символом d, которые в некотором смысле тоже являются кратными, поскольку существующие методы синтеза самосинхронных схем (реализации полумодулярных ДП) не предполагает наличие временных факторов.

Методика "трансляции" описания в схему иллюстрируется рассматриваемым примером, в котором поставлена ограниченная задача – построить схему задатчика.

1. С целью упрощения предположим, что структура задатчика состоит из регистра (Рг1), в котором хранятся адрес и команда, регистра приема данных (Рг2), стробируемого некоторым сигналом *S*, и схемы управления с входным сигналом СХИ, формирующей помимо сигналов *S*, СХЗ и сигнал АУ, устанавливающий содержимое регистра Рг1 на шины адреса и управления. Поэтому для полной спецификации реализуемого поведения задатчика во временной сигнальный граф рис. 8.11,б требуется ввести пару вершин $+S$ и $-S$; вставляем их в разрывы некоторых его дуг. Однако место введения этих вершин должно быть выбрано с учетом семантики. В частности, ясно, что стробирующий сигнал $+S$ должен быть введен после приема +СХИ и выдержки времени, равной 75 нс. Кроме того, вершины +Д и –Д могут быть исключены из СГ, поскольку эти вершины не несут никакой информации, необходимой для реализации схемы управления задатчика.

2. Условие нормальности СГ нарушается на дуге между вершинами –АУ и +АУ. Очевидно, СГ можно привести к нормальному виду путем вставки между этими вершинами вершины $-S$. Заметим, что в результате кратные состояния в ДП оказались устраненными.

3. Исключаем из СГ вершины, помеченные символами *T*, имея ввиду, что временное упорядочение инцидентных им вершин будет обеспечено специальными схемотехническими приемами, например, введением

специальных элементов, осуществляющих задержку соответствующего сигнала.

В результате выполнения вышеуказанных действий получен невременной нормальный СГ, показанный на рис. 8.12,а, который порождает дистрибутивную диаграмму переходов, изображенную на рис. 8.12,б.

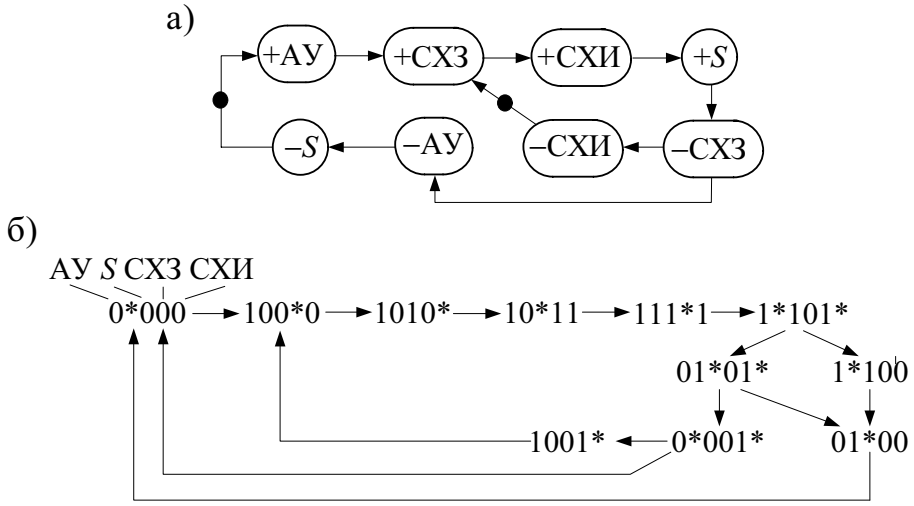


Рис. 8.12. Модифицированный протокол операции "чтение", ориентированный на схемную реализацию: а) правильный сигнальный граф и б) диаграмма переходов для него

Этой диаграмме, в свою очередь, соответствует система уравнений (схема Маллера):

$$\text{АУ} = \overline{S} \vee \text{СХЗ}; \quad S = \text{АУ} \cdot S \vee \text{СХЗ} \cdot \text{СХИ};$$
$$\text{СХЗ} = \text{АУ} \cdot \overline{S} \cdot \overline{\text{СХИ}} \vee \text{СХЗ} \cdot \overline{S}; \quad \text{СХИ} = \overline{\text{СХЗ}}.$$

Реализация системы уравнений в заданном базисе осуществляется известными в теории самосинхронных схем способами, изложенными, например, в главе 4. Отметим, что использование самосинхронной схемотехники для аппаратной реализации схем сопряжения интерфейсов, специфицированных с применением временных СГ и ДП, не может "покрыть" концептуальные недостатки таких протоколов. Ее использование связано лишь с необходимостью обеспечить функционирование сопряжения, не привнося дополнительных факторов, которые могли бы исказить предписанное поведение, в частности, исключить всевозможные состязания и парировать обыч-

но постулируемые и достаточно жесткие ограничения на временные параметры элементной базы.

8.4. Интеллектуальные системы

В системах и машинах обработки знаний большое значение приобретают методы логического (дедуктивного) вывода, развитые в теории автоматического доказательства теорем. Известно, что проблемы дедуктивного вывода являются *труднорешаемыми задачами*. Основные их трудности связаны с комбинаторным ростом числа промежуточных результатов вычислений при увеличении размерности задачи. Поэтому создание методов решения, свободных от экспоненциального взрыва, имеющего место при выводе утверждений в задачах практической сложности, является одной из важнейших проблем в теории дедуктивного вывода. Некоторый выход из этого положения видится в создании параллельных методов дедуктивного вывода, в том числе с применением аппарата теории сетей Петри.

В дедуктивных моделях обработки знаний решаемая проблема записывается в виде утверждений в некоторой формальной системе. Цель проблемы записывается также в виде утверждения, справедливость которого следует установить или опровергнуть на основании аксиом и правил вывода формальной системы. В подавляющем большинстве случаев в качестве такой формальное системы используют исчисление предикатов первого порядка. Для пояснения сказанного приведен пример, где для простоты используется исчисление высказываний.

Пример 1

Некий детектив расследует криминальный случай с двумя действующими лицами: горничной и дворецким. Известна следующая информация: а) горничная сказала, что она видела дворецкого в гостиной; б) гостиная находится рядом с кухней; в) выстрел раздался на кухне и мог быть услышан во всех близлежащих комнатах; г) дворецкий, обладающий хорошим слухом, сказал, что он не слышал выстрела.

Детектив хочет доказать следующую теорему: если горничная сказала правду, то дворецкий лжет.

Обозначим утверждения следующим образом: p – горничная сказала правду; q – дворецкий был в гостиной; r – дворецкий был рядом с кухней; s – дворецкий слышал выстрел; u – дворецкий сказал правду.

Тогда имеют место следующие посылки:

$p \to q$: если горничная сказала правду, то дворецкий был в гостиной;

$q \to r$: если дворецкий был в гостиной, то он был рядом с кухней;

$r \to s$: если дворецкий был рядом с кухней, то он слышал выстрел;

$u \to \overline{s}$: если дворецкий сказал правду, то он не слышал выстрела.

Теорема записываете как:

$$(p \to q)(q \to r)(r \to s)(u \to \overline{s}) \to (p \to \overline{u}),$$

где стрелкой обозначена *операция импликации*: запись $a \to b$ эквивалентна логическому выражению $\overline{a} \vee b$.

Для доказательства теоремы достаточно показать, что значение вышеприведенного выражения при всех возможных значений переменных равно 1 (т.е. истинно). Первый очевидный подход к решению этой проблемы заключается в использовании обычных процедур булевой алгебры, которые основаны на таких правилах, как склеивание, поглощение и правила Де Моргана. Из исходного выражения имеем:

$$\overline{(\overline{p} \vee q)(\overline{q} \vee r)(\overline{r} \vee s)(\overline{u} \vee \overline{s})} \vee \overline{p} \vee \overline{u} =$$
$$p\overline{q} \vee q\overline{r} \vee r\overline{s} \vee us \vee \overline{p} \vee \overline{u} = \overline{q} \vee \overline{r} \vee \overline{s} \vee s \vee \overline{p} \vee \overline{u} \equiv 1.$$

Эта формула равна 1 и, следовательно, она общезначима. В общем случае для логики предикатов первого порядка сложность доказательства таких теорем очень велика (NP-полная проблема).

Задачей дедуктивного вывода будем называть выяснение вопроса логического следования формулы B из заданного множества формул (посылок) F_1, F_2,..., F_n, что равносильно доказательству общезначимости формулы $F_1 \cdot F_2 \cdot \ldots \cdot F_n \to B$, как в нашем примере, или противоречивости (невыполнимости) формулы $F_1 \cdot F_2 \cdot \ldots \cdot F_n \to \overline{B}$. Причем процедура установлении невыполнимости формулы называется процедурой опровержения.

В общем случае невыполнимое множество формул записывается в виде набора утверждений типа $B_1 \cdot B_2 \cdot ... \cdot B_n \to A_1 \vee A_2 \vee ... \vee A_n$, что равносильно ложности выражения $\overline{B}_1 \vee \overline{B}_2 \vee ... \vee \overline{B}_n \vee A_1 \vee A_2 \vee ... \vee A$, где B_i – условия, а A_j – заключения утверждения, выраженные предикатами.

Утверждения, имеющие как условия, там и заключения, представляют собой общие законы (*аксиомы*), описывающие предметную область. Утверждения без условий – это факты (*константы*), соответствующие этой области. И наконец, утверждения без заключений – это отрицания фактов или цели, которые нужно доказать.

Для решении задачи дедуктивного вывода создано большое число методов решения, из которых наиболее известными является *обратный метод Маслова* и *принцип резолюций* с его многочисленными модификациями.

К сожалению, все эти методы мало пригодны для аппаратной реализации в параллельных машинах вывода, так как они не обеспечивают требуемого распараллеливания процесса вычислений. Рассматриваемый далее подход предпочтителен именно с этой точки зрения.

Обратимся к простейшему фрагменту сети Петри, показанному на рис. 8.13,а.

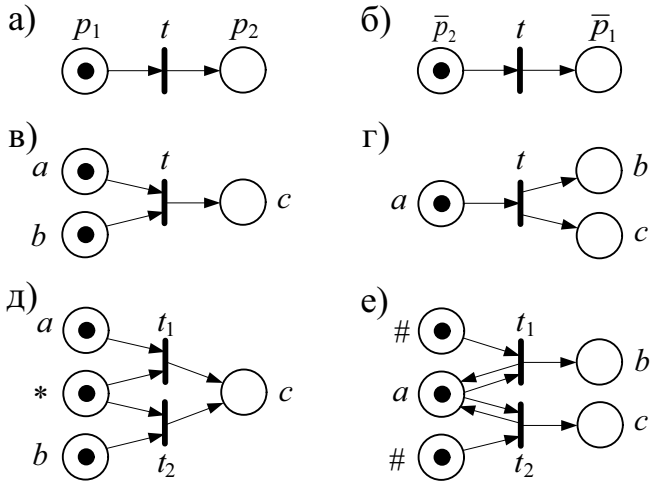


Рис. 8.13. Шесть фрагментов сети Петри

Будем считать, что p_1 и p_2 – не только имена позиций, но и *пропозициональные переменные*. В этом случае мы имеем дело не с фрагментом обычной

СП, а с его *интерпретацией* вследствие дополнительного использования отображения $P \to A$, где A – алфавит, состоящий из переменных p_1 и p_2. При этом позиция p_1 с одним маркером может рассматриваться как логическое условие, а позиция p_2 также с маркером – как логическое следствие, которое реализуется после срабатывания перехода (наступление нового события).

Благодаря такой интерпретации позиций фрагмент СП, показанный на рис. 8.13,а, может быть представлен формулой $p_1(p_1 \to p_2) \to p_2$ или, используя приведенную выше нотацию, следующим образом:

$$\to p_1 \text{ (факт);}$$

$$p_1 \to p_2 \text{ (общий закон);}$$

$$\to p_2 \text{ (цель).}$$

Запись в такой нотации ориентирована на процедуру опровержения, удобную для доказательства противоречивости формул. В дальнейшем эта нотация будет сохранена, хотя вывод на сетях Петри использует процедуру достижимости, которую следует рассматривать как процедуру доказательства общезначимости формул.

Нетрудно видеть, что здесь записано хорошо известное в теории дедуктивного вывода правило *modus ponens*. Для представления правила *modus tollens*, показанного на рис. 8.13,б, достаточно перейти от формулы вида $\overline{b}(a \to b) \to \overline{a}$ к формуле вида $b(\overline{b} \to \overline{a}) \to \overline{a}$, так как из $a \to b$ следует $\overline{b} \to \overline{a}$.

На рис. 8.13,в-е показаны четыре фрагмента СП, которые интерпретируются следующими невыполнимыми формулами:

в) $\to a$
$\to b$
$ab \to c$
$c \to$

г) $\to a$
$a \to bc$, что равносильно (a→b)(a→c)
$bc \to$

д) $\to a \vee b$
$a \vee b \to b$, что равносильно (a→c)(b→c)
$c \to$

е) $\to a$
$a \to b \vee c$
$b \vee c \to$, что равносильно $\overline{\overline{b}\overline{c}}$.

Первые два фрагмента не требуют пояснений. Что касается третьей формулы, то на рис. 8.13,д предусмотрена дополнительная позиция, отмеченная

звездочкой. Она не соответствует переменной из алфавита A, но ее применение необходимо для обеспечения свойства безопасности СП. В этом случае в позиции c будет находиться единственный маркер после срабатывания любого из переходов t_1 или t_2. Во фрагменте на рис. 8.13,е каждый переход может сработать только один раз, вследствие чего окончательная маркировка позиций b и c будет следующей: 10, 01, 11. В результате позиция a для переходов t_1 и t_2 имеет как выходную, так и входную дуги.

Фрагменты СП, показанные на рис. 8.13, вместе с использованием перехода от формулы $a\bar{b} \to c$ к формуле $a \to b \vee c$ позволяют строить сети Петри, имеющие взаимооднозначное соответствие с любой пропозициональной формулой. Процедура для перехода от пропозициональной формулы к соответствующей СП здесь опущена, но может быть пояснена на рассмотренном выше примере.

Для этого примера решение проблемы показано на рис. 8.14,а. Очевидно, что результат непосредственно следует из фрагментов СП, приведенных на рис. 8.12,в-е, и соответствующих им формул. Маркер из начальной позиции p достигает конечной позиции $\bar{u}$, что доказывает теорему или невыполнимость следующего множества утверждений: $\to p$, $p \to q$, $q \to r$, $r \to s$, $s \to \bar{u}$, $\bar{u} \to$. Диаграмма маркировок для этого примера показана на рис. 8.14,б.

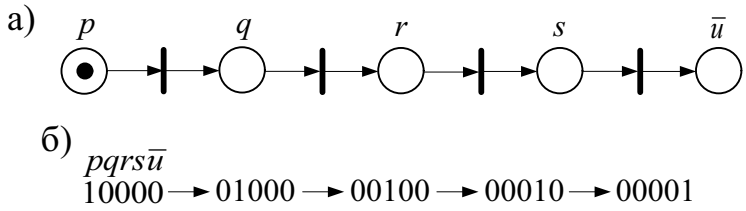


Рис. 8.14. Пример первый: а) сеть Петри и б) граф маркировок для нее

Проблема вывода в логике предикатов первого порядка (по крайней мере, без функциональных символов) может быть описана аппаратом СП. Это утверждение требует проведения дальнейших специальных исследований. Для иллюстрации возможных путей решения проблемы рассмотрим еще один пример.

Пример 2

Имеем следующее множество утверждений:

1) → Мужчина (Дик)
2) → Мужчина (Фред)
3) → Мужчина (Билл)
4) → Женщина (Эмма)
5) → Женщина (Анна)
6) → Отец (Дик, Фред)
7) → Отец (Дик, Фред)
8) → Отец (Дик, Анна)
9) → Мать (Эмма, Фред)
10) → Мать (Эмма, Билл)
11) → Мать (Эмма, Анна)
12) Отец (p, c) → Ребенок (c, p)
13) Мать (p, c) → Ребенок (c, p)
14) Ребенок (c, p)·Мужчина (c) → Сын (c, p)
15) Сын (x, Эмма) →

Здесь утверждения 1–11 представляют собой факты из области семейных отношений. На рис. 8.15 им поставлены в соответствие позиции СП, каждая из которых имеет один маркер, и инцидентные им переходы. Общим законам 12–14 соответствуют три фрагмента СП, а цели 15 (Кто сыновья Эммы?) – один фрагмент СП.

Каждому позиции на рисунке поставлен в соответствие предикат. Отдельные фрагменты СП затем соединяются так, что позиции, соответствующие фактам, соединяются переходами с позициями, соответствующими посылкам общих законов и цели. Эти переходы должны быть взвешены символами индивидных или предметных переменных с использованием правил подстановки входных символов вместо выходных. Процесс подстановки входных символов вместо выходных будем называть *унификацией*, причем разрешается заменять предметную переменную только константой или другой переменной. Некоторые позиции могут иметь более одной входной дуги, что приводит к необходимости использования дизъюнктивных символов. В общем случае некоторые позиции могут не использоваться в про-

цессе вывода, как например позиции, соответствующие фактам 4 и 5. Как и ранее, процесс вывода состоит из движения маркеров по сети, однако унификация изменяет правило срабатывания перехода.

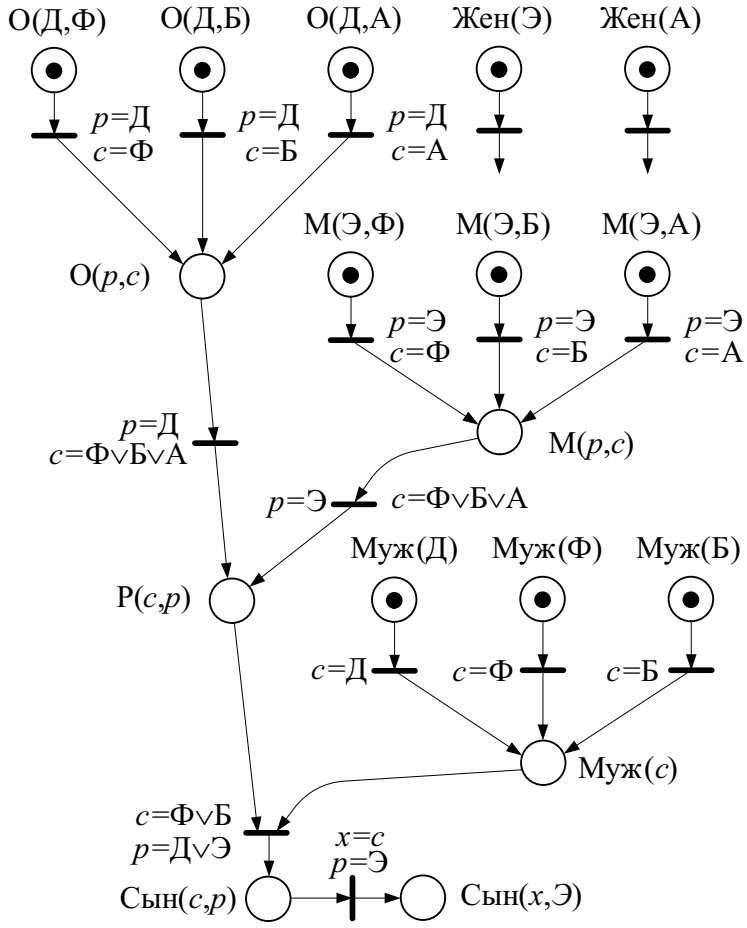


Рис. 8.15. Сеть Петри для второго примера: О – отец, Р – ребенок, М – мать, Д – Дик, Ф – Фред, Б – Билл, Э – Эмма, А – Анна

Теперь правило срабатывания формулируется следующим образом:

а) если периферийная входная позиция, т.е. позиция, не имеющая входов, как например Отец (Дик, Фред), имеет маркер и его переход взвешен (скажем, *p* = Дик, *c* = Фред), то после срабатывания его выходная позиция будет содержать раскрашенные маркеры, каждый из которых имеет свое собственное имя (в данном случае, Дик и Фред). Эти маркеры могут рассма-

триваться как обычные маркеры с атрибутами, причем маркеры в периферийных позициях воспринимаются как смешанные атрибуты;

б) если в одной позиции одновременно находятся два или более маркеров одного цвета, то включается механизм "слияния" и вместо нескольких маркеров одного цвета остается только один (в нашем случае, когда уже сработали все три входных перехода позиции Отец(p, c) имеется четыре маркера: Дик, Фред, Билл и Анна;

в) если переход имеет две или более входных позиции, которым приписаны предикаты, то данный раскрашенный маркер достигнет выходной позиции перехода тогда и только тогда, когда маркер с тем же самым цветом (именем) будет находиться во всех входных позициях (это имя является собственным именем переменной предиката выходной позиции) и отсутствует любое противоречие других переменных.

В нашем примере имеется пара позиций Ребенок (c, p) и Мужчина (с), причем первая позиция содержит два маркера Эмма, Дик типа p и три маркера Фред, Билл, Анна типа c, а вторая позиция содержит три маркера Фред, Билл, Дик типа c. Тогда выходная позиция Сын (c, p) будет содержать два маркера Фред, Билл типа c и два маркера Эмма, Дик типа p. В результате продвижения маркеров будем иметь два маркера цветов Фред и Билл на позиции Сын (х, Эмма). Цель выводы достигнута.

Здесь возникает несколько вопросов. Очевидно, что трансформированная модель СП отличается от обычной *другим правилом срабатывания переходов* и, кроме того, она стала раскрашенной. К тому же раскрашенные маркеры могут сливаться и появляются мультидуги. Но, вероятно, можно показать, что новая модель не является существенным обобщением первоначальной. Важнее подчеркнуть проблему возникающей здесь унификации. Это проблема заключается в том, что желательно реализовать процесс унификации посредством самой модели, а не внешней относительно СП процедурой.

Рассмотренные примеры показывают, что идея использовании методов СП применима к дедуктивному выводу. Пока еще рано говорить о детальной реализации механизмов вывода на СП, важно лишь подчеркнуть настоятельную необходимость развитии этого направления.

Так, для программной реализации могут быть использованные рассмотренные здесь подходы для решения проблемы достижимости на СП. Соответ-

ствующие программы могут опираться на методы поиска в глубину и в ширину, а также на матричный подход.

Для аппаратной реализации целесообразно построение специального клеточного процессора, в котором поведение каждой отдельной клетки моделируется фрагментом СП. Клетка может быть реализована как специальный триггер, содержащий несколько вентилей (элемент Давида). Клеточный процессор позволяет организовать вывод как асинхронную и параллельную обработку без возврата вследствие асинхронной реализации. Формирование специального сигнала завершения вывода необходимо для управления обработкой на сетях в реальном времени.

8.5. Упражнения

1. Прокомментируйте рис. 8.16.

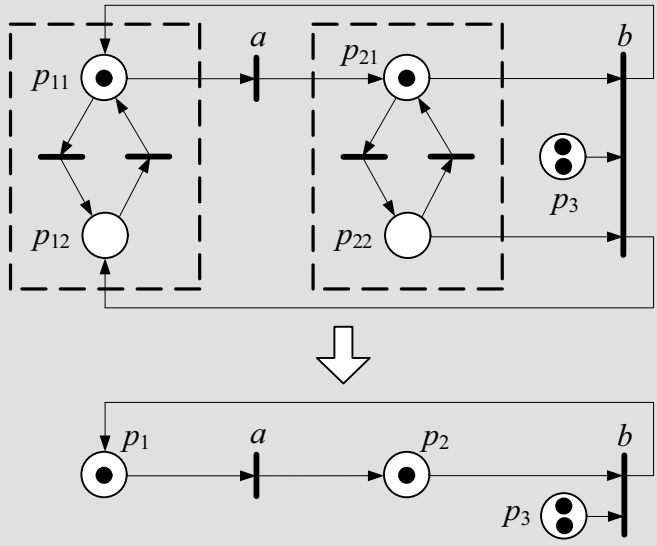
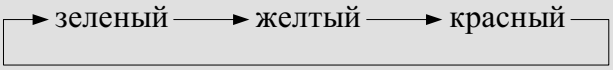


Рис. 8.16. Пример морфизма сетей Петри

2. Разработайте спецификацию в виде сети Петри для схемы автоматического управления переключением светофора, установленного на пересечении главной магистрали с пересекающей улицей. Примите, что по

каждому направлению движения транспорта циклическая последовательность зажигания ламп светофора имеет вид



(желтый свет на переходе от красного к зеленому не включается).

3. На вербальном уровне задана следующая спецификация задачи об управлении дозатором в некотором производстве (см. рис.8.17).

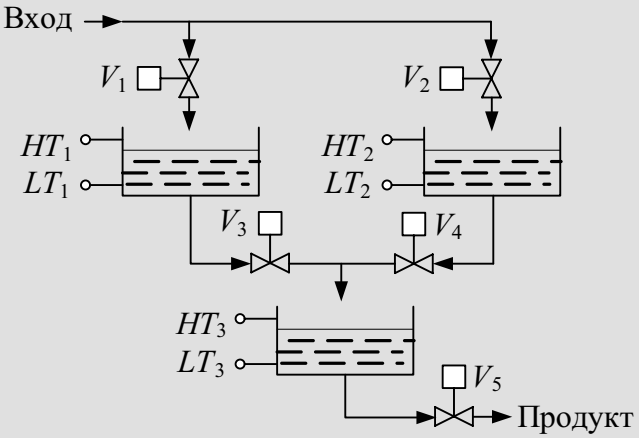


Рис. 8.17. К задаче об управлении дозатором

Имеются три емкости T_i, $i = 1, 2, 3$, каждая из которых снабжена датчиками верхнего HT_i и нижнего LT_i уровней. Первоначально эти емкости пусты и вентили (краны) V_1–V_5 закрыты. Затем открываются вентили V_1 и V_2, которые остаются в открытом состоянии, пока поступающая жидкость не включит контакты датчиков верхнего уровня емкостей T_1 и T_2. После этого вентили V_1 и V_2 закрываются, а вентили V_3 и V_4 открываются. Когда жидкость достигнет либо а) высокого уровня в T_3, либо б) нижнего уровня в обеих емкостях T_1 и T_2, то V_3 и V_4 закрываются, а вентиль V_5 открывается. Когда жидкость в T_3 достигнет нижнего уровня, V_5 закрывается, и процесс возвращается в исходное состояние.

Постройте СП, соответствующую этой спецификации, и проверьте отсутствие поведенческих аномалий.

4. Решите следующую головоломку. В автокемпинг приехали пять автолюбителей, каждый водитель которых поселился в своей палатке. Известны следующие факты:

1. Смирнов приехал на белой “Волге”.
2. Москвич путешествует на одноименном автомобиле.
3. Фамилия рижанина – Соколов.
4. Киевлянин живет рядом с серой палаткой.
5. У Жукова машина красного цвета.
6. Ленинградец приехал на машине серого цвета, и он играет в шахматы с обитателем серой палатки.
7. Иванов живет в зеленой палатке.
8. Одессит живет в синей палатке.
9. Белая палатка расположена правее красной.
10. У обитателя средней палатки синий автомобиль.
11. Ковалев живет справа от владельца «Жигулей».
12. Слева от Иванова живет владелец «Победы».
13. Киевлянин живет в первой палатке.
14. У обитателя белой палатки желтый автомобиль.

Составьте спецификацию задачи на языке сетей Петри и выполните логический вывод для ответа на два вопроса:

1) Откуда приехал владелец “Запорожца”?

2) В палатке какого цвета живет автотурист, у которого белый автомобиль?

5. Решите задачу, которая стоит перед детективом, расследующим убийство. Подозреваемые: Алан, 25 лет, футболист; Алан, 25 лет, мясник; Берт, 55 лет, плотник; Джон, 25 лет, вор-карманник и одна женщина – Барбара, 22 лет, парикмахер. Обстоятельства дела: Барбара состоит в любовной связи с Джоном и Бертом, потерпевшая по имени Сузан – с Джоном; Сузан была убита тяжелым предметом. Мотивы преступления: ограбление или ревность. Установлено, что в день убийства костюм Барбары был испачкан кровью, а брюки Алана грязью. Берт хромает, одна нога у него – деревянная. Джон – владелец пистолета. В качестве тяжелого предмета могут фигурировать деревянная нога, деревянный брусок, пистолет, футбольная бутца, но не ножницы или нож мясника.

Составьте спецификацию задачи на языке сетей Петри с целью установления личности подозреваемого в убийстве и проведите логический вывод.

Послесловие авторов (о В.И. Варшавском и становлении советской/ российской научной школы)

- Какая польза от вашего нового изобретения?
- А какая польза от новорожденного младенца?
Бенджамин Франклин.

Следующий ниже текст использует язык деловой прозы, базирующейся на весьма скупой информации о видном ученом и экстраординарном инженере — Викторе Ильиче Варшавском, разбавленной некоторыми штрихами о его личности, а также деталями развития науки, описанию которой посвящена данная книга. Прототипом типажа В.И., наверное, можно было бы назвать профессора Полежаева в исполнении Н.К.Черкасова в фильме "Депутат Балтики". В.И. в жизни был обаятельным человеком, хотя чудаком явно не был. Его очень ценили друзья и коллеги. Он был всеобщим любимцем.



В.И. родился в Ленинграде 23 февраля 1933 года, в день Красной Армии. Смеем предполагать, что его бойцовский характер был не только генетически запрограммирован, но и астрологически подтвержден "правильной" датой рождения, а его таланты и научно-инженерная жилка, возможно, передалась по наследству.

Место Варшавского в науке и сама наука в контексте деятельности Варшавского

Поэт Лонгфелло как-то заметил: "Мы оцениваем самих себя по ощущениям того, на что мы способны, тогда как другие судят о нас по тому, что мы сде-

лали". Коллега В.И. по ЛОЦЭМИ АН СССР Л.А.Оганесян однажды сказал, что ученые бывают двух типов – "лужи" и "колодцы", имея в виду, что первые ползают по поверхности разрабатываемой темы, а вторые копают глубоко. В.И. останется известным тем, что он, затронув разнообразные области, "вырыл" по крайней мере четыре глубоких колодца.

Колодец номер 1. В.И. был пионером в пороговой логике. В этой области он получил основополагающие результаты, которые были несколько "смазаны" его не слишком значительной с инженерной точки зрения ошибкой. Он неосторожно "залез" в чужую область, асимптотических оценок сложности схем, и сформулировал теорему, согласно которой любая булева функция n переменных может быть реализована схемой из (n+1) пороговых элементов. Фактически было доказано, что это справедливо только для монотонных функций. Ошибка дорого стоила автору, но, закусив удила, Виктор со своими учениками (Б.Л.Овсиевичем, И.Н.Боголюбовым и Л.Я.Розенблюмом) с успехом занялись трехзначной пороговой логикой и одновременно мажоритарной логикой (двузначной и трехзначной), где были получены весьма серьезные результаты. Сюда же можно отнести занятия нейронными сетями, к которым В.И. вернулся в последние годы.

Колодец номер 2. Исследования по коллективному поведению автоматов послужили основой второй докторской диссертации Виктора Ильича (первая была "зарублена" ВАКом опять же по неосторожности диссертанта, который на Экспертном Совете ВАКа заявил, что по теме диссертации он в дальнейшем работать не планирует). Его книга на эту тему знаменовала оформление новой многообещающей области информатики и искусственного интеллекта. Им были разработаны не только вопросы общих закономерностей целесообразного поведения отдельных автоматов, но и проблемы оценки эффективности поведения коллектива автоматов, управления сложными системами, соотношения категорий централизованного и децентрализованного управления с учётом роста сложности систем. Нам кажется, что в последнее время близкие к коллективному поведению автоматов идеи прослеживаются в работах лауреатов Нобелевской премии в области экономики. Вырос также интерес к поведению стохастических автоматов с переменной структурой. Много позже В.И. вместе с Д.А.Поспеловым опубликовали книгу по принципам организации

децентрализованного управления сложными системами (в популярном изложении).

Колодец номер 3. Работая над теорией однородных структур (цепочек и сетей из однотипных элементов с регулярными связями), Виктор (вместе с В.А. Песчанским и В.Б. Мараховским) включился в решение так называемой задачи Дж. Майхила о синхронизации цепи стрелков, носящей почти спортивный характер. Им удалось до минимума сократить сложность реализации (число состояний автомата). Далее последовали решение задачи голосования и обобщение тематики, предложенное в книге 1973 года.

Колодец номер 4. Речь будет идти о лебединой песне В.И. в области, которую нынче стали называть "асинхроникой". (Эта тематика лишь весьма кратко затронута в данной книге.) В.И. довольно неожиданно переключился на новую малоисследованную тогда область – создание асинхронных электронных устройств и систем, в которых не используются "часы". Такие системы обладают рядом полезных свойств, а методы их построения были неизвестны. В.И. явился одним из пионеров этого направления, что безоговорочно признано в "асинхронном сообществе" за рубежом. По-видимому, асинхроника проистекает из основополагающей работы Д.Хаффмена (1954г.), который предложил модель асинхронного конечного автомата. Его труд вызвал к жизни тысячи работ, посвященных противогоночному кодированию автоматов и расширениям модели. Между тем, все известные модели асинхронных автоматов базируются на допущении, что инициатором перехода автомата в следующее состояние является изменение входного символа, а новый входной символ может быть подан лишь после завершения переходных процессов предыдущего такта. В противном случае будут иметь место сбои. Другим существенным ограничением моделей хаффменовского типа являются жесткая дисциплина смены входных сигналов, сильно ограничивающая механизм взаимодействия автомата с внешней средой (обычно – соседние переходы). Это ограничение ставило под вопрос возможность композиции асинхронных автоматов и, в конце концов, стало тормозом на пути широкого практического использования подхода.

В начале 1970-х годов В.И. по счастливому стечению обстоятельств пытался вместе с одним из своих аспирантов разобраться, казалось бы, в тривиальном вопросе: как формально синтезировать схему асинхронного тригге-

ра, известного как гарвардский триггер. Странно, но тогда они сделать этого не смогли (это произошло несколько позже). В то время В.И, его коллеги и ученики не имели абсолютно никакого представления о зарубежных работах, инициированных Маллером. Единственным выходом из затруднения для Варшавского было переключение на изобретательство, что часто помогало ему в жизни. Его неимоверно развитая инженерная интуиция была тем волшебным паровозиком, который толкал В.И. к теоретическому осмыслению решаемых задач.

Покрыв сотни листов бумаги схемами, кубами (которые он любил использовать для представления и минимизации булевых функций) и формулами, В.И. самостоятельно пришел к мысли о неоходимости расщепления входных последовательностей на две фазы – активную (рабочую) и неактивную (спейсер), так как при этом все переходы в последовательностях становятся монотонными. Он предложил конструкцию простейшего триггера с индикацией моментов окончания переходных процессов, названную тригероидом, который работает независимо от реальных задержек его элементов. Триггероид неспособен хранить записанную в него информацию при некоторых значениях входов, но в совокупности с двумя дополнительными триггероидами (без индикаторов) его использование решало задачу создания самосинхронного счетного триггера.

В.И. продолжал развивать свою идею и инициировал тотальный поиск литературы по асинхронике, в который погрузились все члены его команды. К 1975 г. был сделан не только скрупулезный анализ результатов западных ученых, но и развит общий подход к построению самосинхронных схем и устройств, интегрирующий все известные результаты. Практически были созданы основы общей теории самосинхронизации, интегрирующей известные фрагментарные подходы. В 1976 г. под редакцией В.И. на русском языке вышла в свет первая книга, в которой систематически изложены проблемы и решения в области самосинхронизации. Это издание в то время на Западе не было замечено, да этого и трудно было бы ожидать, поскольку еще не существовало ни асинхронного сообщества как такового, ни перевода книги на английский язык.

На фоне некоторого застоя в теории автоматов в 1960-е годы появилась и стала интенсивно развиваться новая научная дисциплина – теория сетей Петри. Развитие этой теории было относительно самостоятельным, но фактически не было автономным. Скорее всего, она должна рассматриваться как ветвь общей теории автоматов, которая занимает ранее неисследован-

ную нишу между конечными автоматами и машинами Тьюринга. Подавляющая часть работ по сетям Петри "обслуживает" проблематику общей теории систем и параллельного программирования. Тем не менее, идейная сторона вопросов, связанных с выделением подкласса живых и безопасных сетей Петри и их исследованием, весьма близка к проблемам самосинхронизации. В книге 1976 г. была продемонстрирована возможность прямой трансляции сети Петри из указанного подкласса в асинхронную схему в обход процедуры противогоночного кодирования состояний автомата.

Дальнейшие исследования команды В.И. в области управления асинхронными процессами нашли отражение во второй книге, законченной в 1984 году и изданной в 1986 году под несколько скучным названием, навязанным редакцией. В 1990 г. ее перевод на английский стал доступен асинхронному сообществу, явившись предметом детального изучения в научных группах двух ведущих в области вычислительной техники университетов мира – в Беркли и Стэнфорде.

В.И. внес фундаментальный вклад во многие направления асинхроники. Многие его результаты приоритетны. Нам представляется разумным обратить внимание научного сообщества на вклад В.И. и предупредить, что терминология, использованная в первоисточниках, часто отличается от современной. Одна из причин этого состоит в применении разных языков, а вторая – в желании многих авторов использовать собственную терминологию для того, чтобы почувствовать себя первооткрывателями.

Микроэлектронный кузнец

Эта рубрика никоим образом не отражает всего, что было сделано Варшавским в области "железа", т.е. устройств и приборов, которые можно пощупать руками. Она ограничена 1988 годом, когда наши пути разошлись, хотя контакты никогда не прерывались.

В.И. в некотором смысле был бессребренником от науки. Он не занимался саморекламой, считая, что и так широко известен, а его результаты работают на него. Но подспудно инженерная душа и изобретательская жилка толкали его на разработку трудоемких проектов, связанных с "железом". Это отнимало кучу времени, которое он мог бы посвятить решению кардинальных проблем. Более того, работа на "железку" практически лишала его сотрудников возможности сконцентрироваться на реализации их собственных намерений. Им, как и шефу-трудоголику, приходилось работать, не по-

кладая рук, почти что в режиме “шарашки“. Впрочем, авторитет шефа и особенно его манера чуть ли не каждый божий день “озадачивать“ членов команды новыми, чаще всего очень заманчивыми, задачками и целыми проблемами, не вызывали шумных протестов и заговоров, а развивали эрудицию и квалификацию исполнителей идей шефа.

Многие попытки В.И. “внедриться” (обратите внимание на это слово – оно предполагает что-то насильственное) в военно-промышленной комплекс (других вариантов просто не было) оказались в конечном счете неудачными. В некотором смысле стратегия нашего шефа была самоубийственной. Первая причина заключалась в том, что нам всем приходилось “вкалывать” на два фронта, т.е. сочетать научные исследования, написание отчетов, статей, заявок на изобретения, книг и т.п. с проектированием “железки”. Вторая причина – полная бессмысленность разработок “на сторону” в условиях, мягко сказать, странностей финансовой системы АН СССР. Заключив контракт со сторонней организацией на определенную сумму, мы на следующий год должны были либо продолжить работу с тем же заказчиком или искать нового, потому что академический бюджет автоматически урезался на сумму хоздоговора. Но получая скромную зарплату за наши академические достижения, мы под водительством нашего шефа безуспешно пытались сломать “внутренний” железный занавес, отделяющий науку от практики. Нельзя добавить “с упорством, достойным лучшего применения», потому что при общении с заказчиками находились новые задачи, приходили свежие идеи и рекрутировались кадры.

Не только злой рок был причиной неудач. Инновационные разработки всегда вызывали недоверие традиционалистов. А в условиях партийного и бюрократического (министерского) контроля микроэлектроника буксовала на месте и как отрасль промышленности была полностью дисфункциональной. Последующие попытки В.И. запустить аснхронику на зарубежную орбиту Великобритании, Финляндии, Японии, США, Израиля заставили обратить внимание на ранее неизвестные разработки его команды, не привели к внедрению. Развитие технологии тогда и в еще большей степени теперь позволяют рынку обойтись традиционными схемотехническими решениями. Так что работы и поделки В.И. признавались оригинальными и в академической среде оценивались весьма высоко, но на практике остались невостребованными.

Впрочем, время, не останавливающееся ни на минуту, как говорил коммунист Джанни Родари, возможно, когда-нибудь устроит переоценку ценно-

стей и установит нерукотворный памятник недооцененному первопроходцу.

В. И. обладал уникальным инженерным даром. Как ему удавалось синтезировать изящные типовые схемы в стандартном базисе – такие, как триггеры, полусумматоры, автомат повторного вхождения, счетчики, буферные устройства и т. п., – загадка. Применение формальных методов в подавляющем большинстве случаев не позволяло воспроизвести оригинал даже при одинаковой начальной спецификации. Варшавского не интересовали "некрасивые" схемы – он оттачивал их до блеска. Во время посещения Советского Союза Нобелевский лауреат Поль Дирак прочел лекцию по философии физики, в ходе которой подошел к доске и написал: "Физические законы должны быть математически красивыми и простыми". Ту же мысль он высказал ранее в одной из своих лекций в Принстоне: "Вся моя жизнь – это написание красивых формул". Большая часть жизни В.И. состояла в изобретении красивых схем. Эта сторона деятельности Виктора Ильича, наверное, являлась главной в его творчестве. Если внимательно проанализировать многие асинхронные блоки, нашедшие практическое применение, то в них при желании можно увидеть прототипы схем, изобретенных Варшавским.

Нельзя не отметить, что "железные игры» были для Варшавского глотком свежей воды в пустыне, неисчерпающим источником для серьезных и свежих научных идей.

В.И. всегда дымил, как паровоз, и в короткие промежутки времени, когда под прессингом свой жены Натальи пытался "завязать" с курением, постоянно сосал сигарету, засунутую в рот "нештатным" концом. Наверное, пагубная привычка сыграла не последнюю роль в том, что оборвалась его яркая жизнь, но работать без сигарет он вряд ли бы смог.

В.И. обладал неистребимым чувством юмора. Память его друзей и сотрудников сохранила множество свидетельств сногсшибательного юмора Виктора. Приведем несколько примеров.

В телепередаче "Три шага за горизонт", которую вел его отец, известный писатель-фантаст, обсуждалась проблема бессмертия. Все участники были убеждены, что эту проблему человечество рано или поздно решит. Виктор настаивал на том, что ее вообще не нужно решать. Его последний аргумент, содержащий аполитичный намек, – "Представьте себе, что вашей кафедрой

100 лет заведует один и тот же человек" – стоил бы редактору потери работы. Передача в эфир не вышла.

Летом 1991 года, незадолго до путча ГКЧП, на фуршете в гостинице "Прибалтийская" вице-президент крупной зарубежной фирмы доверительно сказал В.И.: "Мы скептически относимся к инвестированию в научные исследования. За последнее время мы инвестировали 500 миллионов франков и получили нулевой результат". "Напрасно, – парировал В.И. – Вы могли бы инвестировать в десять раз меньше в советскую науку и иметь тот же самый результат! "

Получив почетный знак "Изобретатель СССР", он моментально отреагировал на это событие, произнеся фразу: "Оказывается, я изобрел СССР!".

Анекдоты хлестали из него, как из пожарного брандспойта. Его розыгрыши, мгновенная реакция на происходящее, острые ремарки, убийственно меткие характеристики людей и событий вошли в научный фольклор. Он был незаменимым тамадой на банкетах, и многие его друзья, которые сами по себе были замечательными спичмейкерами, в его присутствии слегка сникали. Однажды в журнал "Известия АН СССР. Техническая кибернетика" была послана одна за другой серия из четырех статей по мажоритарной логике, а на третью из них был получен отрицательный отзыв анонимного резензента, основанный на мнении его аспирантов. Придя в раздраженное состояние, В.И. предложил написать в редакцию короткий ответ и настаивал, чтобы там были такие слова: "Надо уметь подбирать аспирантов!". Хотя стиль ответа вовсе не был академическим, редакция под его впечатлением подписала статью в печать. Она даже вышла в свет раньше второй статьи, из-за чего аспиранты рецензента, наверное, совсем утратили недоразвитую способность что-нибудь понять.

Всего не перечислить. Но, наверное, больше всего он любил свое дело. Книги, статьи и изобретения, которые он оставил, мы надеемся, еще послужат людям.

Организаторская деятельность

В. И. сыграл большую роль в становлении кибернетики, информатики и искусственного интеллекта в нашей стране.

Удивительной была способность Варшавского создавать команду единомышленников. Его окружение работало, как хорошо отлаженный меха-

низм, который никогда не давал катастрофических сбоев, хотя отдельные зависания и тупики, конечно, случались. В.И. никого не отпускал в автономное плаванье – он выдавал задания, обсуждал возможные пути их решения и постоянно был в курсе их выполнения. Такой контроль не вызывал обычно отрицательных эмоций, так как осуществлялся он по-дружески. Матерые сотрудники, правда, иногда втихомолку роптали – у них были собственные идеи, но прессинг они считали нормальным, поскольку репутация босса была хорошей защитой от неприятных внешних воздействий.

На всех этапах своей жизни В.И. не переставал учиться и учить. Во многом его мировоззрение сформировалось на зимних школах под Ленинградом, которые он сам создал в начале 1960-х годов. Ему удалось привлечь таких выдающихся ученых и мыслителей, работающих в разных областях науки, как М. Л. Цетлин, М. М. Бонгард, Л. И. Розоноер, В. С. Гурфинкель, С. М. Осовец, Д. А. Поспелов, Я. А. Альтман и др. Эти школы более 10 лет питали мыслями и идеями всех своих участников, специализирующихся в области математики, биологии, теории автоматов, коллективного поведения, распознавания образов, вычислительной техники, передачи информации и смежных областях. В.И. принимал активное участие в работе школ М. А. Гаврилова, общепризнанного в СССР пионера в области теории автоматов, который всегда активно поддерживал работы команды В.И.

В.И. был ключевой фигурой и в подготовке ядра эстонских национальных кадров высшей квалификации в области теории автоматов и ее приложений. Можно упомянуть также множество постоянно действующих семинаров, которыми он руководил.

В.И. был экстраординарной фигурой и в общечеловеческом плане. У него было громадное количество друзей и знакомых, которые любили и глубоко уважали его. Варшавсий охотно поддерживал контакты и всегда был готов помочь советом или делом.

Никто и не пытался сосчитать число его аспирантов (что-то около 50), защит, на которых он с блеском выступал в качестве оппонента, или статей и книг, которые он отрецензировал (часто – с сарказмом и сногшибательной критикой).

В.И. не был самовлюбленным человеком. Он, естественно, знал себе цену, но никогда не рвался к власти. Позиции профессора и заведующего лабораторией его вполне устраивали. Его призванием была работа, и он явно от-

носился к разряду трудоголиков. Того же он требовал и от своего окружения. Может показаться странным, но во многом интуитивный отбор кадров в подавляющем большинстве случаев не подводил его. Он не всегда с первого захода принимал результаты и мнения своих коллег и учеников. Его надо было сломить или, по крайней мере, выждать некоторое время. Возможно, это происходило оттого, что он беспрестанно генерировал новые и новые идеи, развитием которых хотел озадачить свое окружение.

В. И. оставил множество последователей, которые развили его идеи и обогатили науку и практику новыми идеями и разработками. Если ранее они работали только в СССР и социалистических странах Восточной Европы, то к настоящему времени многие из них мигрировали в более дальние края и с честью влились в научное сообщество. Возможно, В.И. наметил некоторые пути развития асинхроники как любимого своего детища; это нашло отражение в идеях, связанных с построением самосинхронных схем, нечувствительных к задержкам как в логических элементах, так и проводах, а так же схем управления параллельными асинхронными процессами. 20–25 лет тому назад казалось, что асинхроника должна вот-вот победить синхронный подход. Этого не произошло, наверное, потому, что рутинная компьютерная технология развивалась куда более быстрыми темпами, чем асинхроника. Комбинация обоих подходов, возможно, станет в будущем реально продуктивной.

Один из поздних учеников Варшавского Вадим Смоленский назвал его кремниевым Моцартом и так же озаглавил свои воспоминания о Варшавском. С неподдельныи интересом прочитав их, мы обратились к “Моцарту и Сальери“ Пушкина, где наткнулись на фразу, вложенную в уста Моцарта: “Нас мало избранных...”. Теперь мы знаем, как закончить это послесловие: Виктор Ильич Варшавский был несомненно одним из избранных.

Предметный указатель

А

Автомат 19
- конечный 114
- линейно-связный 114
- магазинный 114
- многократного вхождения 290
- недетерминированный 22

Аксиома 371
Аномалия
- колебательная 294
- метастабильная 294
- поведения системы 239
Арбитр 292
- неограниченный 296
- ограниченный 295

Арбитраж 292

Б

Бифуркатор 125
Блокировка 231
- темпа 327
Блок
- операционный 265
- управляющий 265
Блок-схема параллельная 124

В

Верификация 322
Вершина 24
- внутренняя 92
- граничная 92
- маркированного графа 199
  -- живая 202
- тупиковая 92

Выбор 22
Вывод
- дедуктивный 369
- логический 369

Г

Гамак 39
Гарантоспособность 344
Голодание 227
Граф
- маркированный 87
- ориентированный 185
- сигнальный 204
  -- когерентный 208
  -- нормальный 209
  -- обобщенный 337
  -- регулярный 209

Д

Дедлок 98
- структурный 225
Детализируемость 255

Диаграмма
- временная 151
- переходов 185
  -- бинарная 185
  -- двоичная 185
  -- дистрибутивная 189
  -- Маллера 193
  -- полумодулярная 189
  -- последовательная 189
  -- противоречивая 189
- пространственно-временная 360
- Хассе 207
Дуга тормозящая 143

Е

Е-сети 161

З

Задатчик 303
Задача
- взаимного исключения 233

- достижимости 89
-- подмаркировки 91
-- позиции 91
-- полная задача 90
-- пустой маркировки 90
-- пустой позиции 91
- живости 89
-- общая 96
-- одного перехода 96
- неразрешимая 96
- о курильщиках 245
- о спящем парикмахере 260
- производитель-потребитель 235
- труднорешаемая 369
- читатели-писатели 240
Задержка
- инерциальная 192
- чистая 193
Закон инволюции 129
Замыкание АП 61
Затягивание фронтов 294

И

Импликация 370
Инвариант 103
Инициатор 22
- несущественный 39
Интервал критический 231
Интерпретация
-модельная 56
- предметная 56
Интерфейс 302
Исполнение справедливое 257

К

Квитирование 316
Классификация
- интерфейсов 304
- сетей Петри 82
Класс эквивалентности 28
- заключительный 28
- начальный 28
Код самосинхронный 266
Композиция 32
- параллельная 46
- последовательная 49
Компонента ситуации внутренняя 41
- входная 41
- выходная 42
Контур 24
Конфликт 78

Л

Линия 304
Ловушка 99
- маркированная 99
Логика
- динамическая 184
- модальная 184

М

Маркировка 71
- активная 96
- начальная 71
Маршрутное выражение 252
Матрица инцидентности 102
- переходов 103
Машина
- абстрактная 114
- Тьюринга 114
Метамодель 18
Метод Маслова обратный 371
Механизм интерпретации 56
- рандеву 246
- синхронизации 221
Множество рабочих состояний 197
Модель
- в глобальных состояниях 18
- динамическая 72
- математическая 18

- неинтерпретированная 55
- объектная 18
- сетевая 235
- событийная 57
- статическая 81
- структурная 318
- управляющей структуры 55

Монитор 243
Мощность моделирующая 173

П

Переход
- асинхронного процесса 21
- возбужденный 71
- сети Петри 70

Плотность вероятности 159
Позиция 70
- инверсная 139
- свободного выбора 86

Последовательность допустимая 28
Постусловие 78
Пошаговая детализация 254
Предикат 27
Примитивы Е-сети 163
Принцип резолюции 371
Приоритет 148
Проблема арбитража 293
Проверка на нуль 139
Программирование параллельное 221
Продукция 112
Протокол 25
- асинхронного процесса 25
- чередующегося бита 324

Процесс
- асинхронный 18
- автономный 26
- детерминированный 29
- конвейерный 33
- неприведенный 42
- приведенный 32
- простой 26
- согласующий 222
- управляемый 26
- эффективный 26

Процессы взаимодействующие 222

Р

Развертка сигнального графа 211
Разметка
- конфликтная 208
- знакосогласованная 209

Распознаватель 170
Распределение ресурсов 147
Расширение асинхронного процесса 62
Реализация
-монотранзитная 275
- совершенная 275

Реверс асин 62
Редукция АП 39
- по входной компоненте 41
- по выходной компоненте 42
- по инициатору 39
- по результату 40

Режим
- запрос-ответный 266
- хендшейка 267

Результант 22
- несущественный 40

Репозиция 30
- непростая 126
- простая 126

Решетка 189
- дистрибутивная 207
- полумодулярная 208

Ряд Фиббоначи 181

С

Самоконтролируемость 323
Сборка 125

Сводимость 95
Связность 205
Селектор 92
Семантика
- взаимодействующих процессов 250
- перестановочная 253
- причинности 253
Семафор
- Агервалы и Вентилборга 234
- Бринч-Хансена 234
- двоичный 234
- Дейкстры 232
- Цикритзиса 234
Сентенция 113
Сервис 313
Сеть Петри 70
- автоматная 87
- активная 85
- антидвойственная 128
-- бесконфликтная 87
-- временная 150
- двойственная 128
-- диссипативная 129
-- живая 104
-- инвариантная 103
-- ингибиторная 143
-- консервативная 85
-- немаркированная 101
-- ограниченная 85
-- ординарная 137
-- перекрашивающая 176
-- раскрашенная 140
-- реверсная 128
-- самодвойственная 128
- оценочная 161
- потоков данных 168
- самомодифицируемая 151
-- 1-ограниченная 84
-- k-ограниченная 84
-- селективная 176
-- сеть свободного выбора 86
-- сигнальная 204
-- синхронная 151
-- со взвешенными дугами 136
-- согласованная 103
-- с тормозящими дугами 143
-- стохастическая 155
-- устойчивая 85
-- цветная 140
-- чистая 102
Сигнал
- завершения 265
- запроса 265
- ответа 265
- подтверждения 265
- фазовый 265
Синхронизатор 125
Синхронизация 246
Синхроцикл 201
Ситуация 21
- дополнительная 30
- структурированная 34
Склейка 170
Скорость перехода 155
Следствие логическое 372
Смерть голодная 228
Смеситель 170
Событие 39
Сопряжение 302
Состояние 187
- бифуркантное 187
- детонантное 187
- конфликтное 188
- кратное 187
- метастабильное 293
- рабочее 196
Состязание логическое 193
Спецификация 22
Срабатывание перехода 71
Среднее время

- нахождения в маркировке 159
- пребывания маркера в позиции 161

Степень отношения 24
Структура 57
Суппорт инварианта 108

Схема асинхронная 192
- желобковая 128
- конвейерная 128
- магистральная 128
- Маллера 191
- пайплайновая 128
- параллельного сжатия 281
- самосинхронная 265
- синхронная 293
- управляющая 267

Схемотехника самосинхронная 168

Т

Таймаут 311
Тактирование 19
Теорема Коммонера-Хака 100
Теория
- автоматов 64
- доказательства теорем 369
- общая 173
  -- специальная 175
- Петри 65
- сетей 65

Терминал 112
Триггер 125

У

Унификация 374
Уравнение СП фундаментальное 103
Уровень интерфейса
- логический 305
- механический 302
- электрический 302

Условие 78
- логическое 372

Ф

Фаза 152
- гашения 266
- наблюдения 152
- рабочая 266
- срабатывания 152

Факт 21

Форма
- нормальная дизъюнктивная 275
- сокращенная 276

Функция
- бесповторная-весовая 117
- задержки 148
- запрета 143
- кратности 153
- наименований 116
- повторная 117
- разметки 204
- распределения предикатов 123

Ц

Цепь марковская 156
Цикл 196
- рабочий 196
- синхронизационный 224

Циклограммы 308

Ч

Числа Фиббоначи 64

Ш

Шина 304

Э

Элемент задержки 149
Эффект
- аномальный 225
- каннибализма 226

Я

Язык автоматный 70
- Ада 246
- контекстно-зависимый 70
- контекстно-свободный 70
- Милнера 252
- OKKAM 252
- регулярных выражений 70
- типа 0 113
-- типа 1 113
-- типа 2 113
-- типа 3 113
- Хоара 241

Ячейка Давида 283
- конвейерный аналог 286

Англ.

C

C-элемент 279

M

Modus ponens 372
- tollens 372

СПИСОК ЛИТЕРАТУРЫ

Scribitur ad narrandum, non ad probandum —
Пишут для того, чтобы рассказать, а не для того, чтобы доказать.
Квинтилиан, Обучение оратора.

Основные рекомендуемые источники

Приведенные ниже ссылки на ставшие раритетами три книги, изданные в 1986-1984 годах. Но их можно скачать с интернета в различных форматах (к сожалению, иногда за плату).

1. В.И. Варшавский (ред.). Автоматное управление асинхронными процессами в ЭВМ и дискретных системах. — М.: Наука, 1986.

2. Л.Я.Розенблюм. Сети Петри. Известия АН СССР. Техническая кибернетика, №5, –М., 1983, стр.12 – 40. (http://is.ifmo.ru/works/2013/rosenblum-petri-nets.pdf)

3. Петерсон Дж. Теория сетей Петри и моделирование систем. — М.: Мир, 1984. (Перевод с английского книги: J. Peterson, Petri net theory and the modeling of systems.)

4. Котов В.Е. Сети Петри. — М.: Наука, 1984.

5. М.Минский. Вычисления и автоматы. — М. , Мир, 1971.

6. М.А.Айзерман, Л.А.Гусев, Л.И.Розоноэр, И.М.Смирнова. А.А.Таль. Логика. Автоматы. Алгоритмы. — М., Физматгиз, 1963.

Учебные и методические пособия

Нижеследующие пособия, изданные позже основных источников, могут в определенной степени дополнить материал, изложенный в данной публикации.

1. Карпов Ю.Г. Model Checking. Верификация параллельных и распределенных программных систем. - СПБ: БХВ - Петербург, 2010.

2. Федотов И.Е. Некоторые приемы параллельного программирования: Учебное пособие. — М.: МГИРЭА(ТУ), 2008.

3. Кулагин В.П. Моделирование структур параллельных вычислительных систем на основе сетевых моделей: Учебное пособие. — М.: МГИРЭА(ТУ), 2008.

4. Зайцев Д.А. Математические модели дискретных систем: Учебное пособие. — Одесса: ОНАС им. А. С. Попова, 2004.

5. Безбедова Л.Г., Ключко В.И. Теория вычислительных процессов: Методическое пособие. — Краснодар, 2004.

Дополнительный список библиографических ссылок на английском языке

Авторы выбрали лишь небольшое число ссылок из моря книг, посвященных тематике сетей Петри и родственных моделей в разных предметных областях. Они могут дать лишь некоторое представление о развитии рассматриваемых в данной книге подходов и по существу отражают, как правило, результаты поисков более глубоких, скорее академических решений. Заинтересованные читатели могут найти на интернете не только рекламную информацию о в текущем столетии книг, но и прочесть или скачать интересующие их статьи и главы. Коллекция, в которую входят книги из перечисленного ниже списка, доступна на многих сайтах, из которых самым мощным является http://books.google.com

1. W. Aalst, C. Stahl. Modeling Business Processes: A Petri Net-Oriented Approach. — MIT Press, 2011.

2. K. Jensen, S. Donatelli, M. Koutny. Transactions on Petri Nets and Other Models of Concurrency. — Springer, 2011.

3. I. Koch, W. Reisig, F. Schreiber. Modeling in Systems Biology: The Petri Net Approach. — Springer, 2011.

4. W. Reisig. Elements of Distributed Algorithms: Modeling and Analysis with Petri Nets . — Springer, 2010.

5. R. David, H. Alla. Discrete, Continuous, and Hybrid Petri Nets. — Springer, 2010.

6. P. Haas. Stochastic Petri Nets: Modeling, Stability, Simulation. — Springer, 2010.

7. M. Zhou, N. Wu. System Modeling and Control with Resource-Oriented Petri Nets. —CRC Press, 2010.

8. W. Penczek, A. Pólrola. Advances in Verification of Time Petri Nets and Timed Automata: A Temporal Logic Approach. — Springer, 2010.

9. K. Jensen and L.M. Kristensen: Coloured Petri Nets. Modeling and Validation of Concurrent Systems. Springer-Verlag, 2009, 384 pp.

10. C. Girault, R. Valk. Petri Nets for Systems Engineering: A Guide to Modeling, Verification, and Application. — Springer, 2009.

11. M. Diaz. Petri Nets: Fundamental Models, Verification and Applications. — Wiley & Sons, 2009.

12. B. Hrúz, M. Zhou. Modeling and Control of Discrete-event Dynamic Systems with Petri Nets and Other Tools. — Springer, 2007.

13. D. Kinniment. Synchronization and Arbitration in Digital Systems. —Wiley and Sons, 2007.

14. M. Iordache, P. Antsaklis. Supervisory Control of Concurrent Systems: A Petri Net Structural Approach. — Birkhäuser, 2006.

15. E. Villani, P. Miyagi, R. Valette. Modeling and Analysis of Hybrid Supervisory Systems: A Petri Net Approach . — Springer, 2006.

16. R. German. Performance Analysis of Communication Systems: Modeling with Non-Markovian Stochastic Petri Nets. — Springer, 2006.

17. G. Stremersch. Supervision of Petri Nets: a Petri net structural approach. — Springer, 2006.

18. J. Desel, J. Esparza. Free Choice Petri Nets. — Cambridge University Press, 2005.

19. J. Cortadella, A. Yakovlev, G. Rozenberg. Concurrency and Hardware Design: Advances in Petri Nets. — Springer, 2002.

20. J. Cortadella, M. Kishinevsky, A. Kondratyev, L. Lavagno, A. Yakovlev. Logic Synthesis of Asynchronous Controllers and Interfaces. — Springer, 2002.

21. G. Agha, F. Cindio, Grzegorz Rozenberg. Concurrent Object-Oriented Programming and Petri Nets, Advances in Petri Nets. — Springer, 2001.

22. E. Best, R. Devillers, M. Koutny. Petri Net Algebra. — Springer, 2001.

23. J. Sparsø, S. Furber. Principles of asynchronous circuit design - A systems perspective. — Kluwer, 2001.

24. A. Yakovlev, L. Gomes, L. Lavagno. Hardware Design and Petri Nets. — Kluwer, 2000.

25. David Padua, Editor-in-Chief, Encyclopedia of Parallel Computing, Springer Science+Business Media, LLC 2011.

Замеченные опечатки, возникшие при производстве:

Стр. 291: В формулах вместо символа ^ пропечатан символ €

Стр. 283: В формулах вместо символа ^ пропечатан символ €

Стр. 275: В формулах вместо символа ^ пропечатан символ €

Стр. 253. Немного смещена со своего места формула

Рис.1.4 Между ситуациями S9 и S10 стрелка наоборот (СТР. 29).