ГЛАВА 7. ОСНОВЫ ТЕОРИИ СЕТЕЙ ПЕТРИ

7.1. Сети Петри – инструмент моделирования, анализа и синтеза систем

Теория сетей Петри делает возможным моделирование системы математическим представлением ее в виде сети Петри. Предполагается, что анализ сетей Петри поможет получить важную информацию о структуре и динамическом поведении моделируемой системы.

Системы могут содержать большое число взаимодействующих элементов, каждый элемент, в свою очередь, также может быть системой с множеством компонентов, которые взаимодействуют друг с другом сложным образом. Примерами подобных систем могут служить экономические системы, юридические, химические, биологические системы, а также системы управления.

Однако при всем разнообразии моделируемых систем можно выделить несколько общих черт:

1. Системы состоят из отдельных взаимодействующих элементов. Каждый элемент также может быть системой.
2. Каждый элемент имеет свое состояние. Состояние элемента системы – формализм, используемый для описания дальнейшего поведения системы. Часто состояние зависит от предыстории этого элемента и может со временем меняться.
3. Параллелизм (одновременность): один элемент действует одновременно с другими элементами. Например, в вычислительной системе могут параллельно действовать печатающее устройство и устройство чтения. В экономической системе – производители, продавцы, покупатели действуют в одно время. Параллелизм создает трудности при моделировании. Необходима синхронизация. А эта синхронизация может привести к тому, что один элемент будет ждать другой. Согласование во времени действий разных элементов может быть очень сложным.

Сети Петри специально разрабатывать для моделирования систем, содержащих взаимодействующие параллельные элементы. Впервые сети Петри предложил Карл Адам Петри в своей докторской диссертации «Связь автоматов», в которой изложены основные понятия.

Итак, сети Петри – инструмент проектирования и анализа систем. Возможны два подхода при использовании сетей Петри.

Первый подход – сеть Петри вспомогательный инструмент анализа. Для построения системы используются общепринятые методы проектирования. Затем построенная система моделируется сетью Петри, и модель анализируется. При анализе выявляются изъяны в проекте. Проект

пересматривается, затем снова моделируется и исследуется. И так до тех пор, пока анализ не даст удовлетворительного результата, происходит постоянное преобразование системы в модель в виде сетей Петри (рис. 7.1*а*).

При втором подходе весь процесс проектирования проводится при помощи теории сетей Петри. Проектируемая система сразу представляется сетью Петри, сеть анализируется, совершенствуется. Получив сеть с необходимыми свойствами, она преобразуется в реальную систему (рис.7.1*б*).

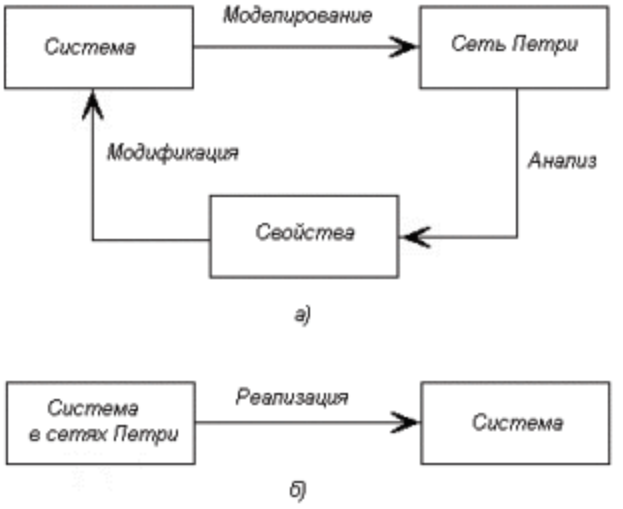


Рис. 7.1. Возможные подходы к использованию сетей Петри:
а) сеть Петри вспомогательный инструмент анализа;
б) представление проектируемой системы сетью Петри

Существующие два подхода требуют решения двух типов задач:

1. Необходима разработка методов моделирования систем сетями Петри.
2. Необходима разработка методов реализации сетей Петри системами.

И в том и в другом случае нужны методы анализа сетей Петри для определения свойств модели. И первое, что требуется, – изучение свойств самих сетей Петри.

Развитие теории сетей Петри проводилось по двум направлениям:

1. Прикладная теория сетей Петри, необходимая для моделирования систем, их анализа.
2. Чистая теория сетей Петри, то есть разработка основных средств, методов, понятий.

7.2. Определение сетей Петри

Сеть Петри состоит из 4-х элементов:

1. Множество позиций P.
2. Множество переходов T.
3. Входная функция I.
4. Выходная функция O.

Входные и выходные функции связаны с переходами и позициями. Определения входных и выходных функций основаны на понятии комплекта.

Комплект – обобщение понятия множества. Комплект – набор элементов, но всякий элемент может входить в него 0, 1, 2 и т. д. раз. В теории множеств элемент есть либо элемент множества, либо не элемент множества. В теории комплектов элемент может входить в комплект любое заданное число раз. Например, $B_1 = \{a,b,c\}$, $B_2 = \{a,b,c,c\}$, $B_3 = \{a,a,a\}$, $B_4 = \{b,c,b,c\}$, $B_5 = \{b,b,c,c\}$. $B_6 = \{a\}$. Комплекты B_1 и B_6 являются множествами. B_5 и B_4 один и тот же комплект. Порядок элементов не важен.

Основное понятие теории множеств – отношение включения, определяющее какие элементы являются членами каких множеств. Основное понятие теории комплектов – функция числа экземпляров элемента в комплекте. Обозначается $\#(x,B)$. Читается: «число x в комплекте B».

Если $0 \le \#(x,B) \le 1$, то получим обычное множество. Множество – частный случай комплекта. В комплекте $\#(x,B) \ge 0$ для всех x и B.

Элемент x член комплекта B $(x \in B)$, если $\#(x,B) > 0$. Аналогично $x \notin B$, если $\#(x,B) = 0$. Комплект называется пустой, если для всех x $\#(x,B) = 0$ (аналогичен пустому множеству).

Мощность комплекта $|B| = \sum_X \#(x,B)$ определяет общее число экземпляров в комплекте.

$A \subseteq B$ (комплект A является подкомплектом комплекта B), если для всякого x $\#(x,A) \le \#(x,B)$. В этом случае каждый элемент комплекта A является элементом комплекта B не большее число раз. $A = B$, если $\#(x,A) = \#(x,B)$ для всех x.

Рассмотрим основные операции над комплектами:

1. Объединение комплектов $A \cup B$:
$$\#(x, A \cup B) = \max(\#(x,A), \#(x,B));$$
2. Пересечение комплектов $A \cap B$:
$$\#(x, A \cap B) = \min(\#(x,A), \#(x,B));$$
3. Сумма $A + B$:
$$\#(x, A + B) = \#(x,A) + \#(x,B);$$
4. Разность $A - B$:

$$\#(x, A - B) = \#(x, A) - \#(x, A \cap B).$$

Пусть M – множество элементов, из которых составляются комплекты. Обозначим M^n – множество всех комплектов, построенных из элементов множества M так, что

$$\#(x, B) \le n, \quad B \in M^n,$$

M^∞ – множество всех комплектов, построенных из элементов множества M без ограничения на число экземпляров элемента в комплекте.

Определение. Сеть Петри – это четверка

$$C = (P, T, I, O),$$

где $P = \{p_1, p_2, ..., p_n\}$ – конечное множество позиций;

$T = \{t_1, t_2, ..., t_m\}$ – конечное множество переходов;

$P \cap T = 0$ (множества P и T не пересекаются).

Входная функция I отображает переход t_j в комплект позиций, называемых входными позициями перехода t_j. Выходная функция O отображает переход t_j в комплект позиций, называемых выходными позициями перехода t_j.

Итак:

$I: T \to P^\infty$ – входная функция – отображение переходов в комплекты позиций;

$O: T \to P^\infty$ – выходная функция – отображение переходов в комплекты позиций.

7.3. Графическое представление сетей Петри

Графическое представление сети Петри – двудольный ориентированный мультиграф с вершинами двух типов:

- позиции (кружок);
- переходы (планка, черточка).

Функции O и I представляются дугами. Дуги ориентированы (стрелки), соединяют позиции и переходы.

Позиции, дуги из которых ведут в переход t_j, называются входными для t_j; позиции, в которые ведут дуги из перехода t_j, называются выходными для t_j. Комплекты входных позиций обозначают $I(t_j)$. Комплекты выходных позиций – $O(t_j)$.

Пример. Рассмотрим сеть, представленную на рис. 7.2.

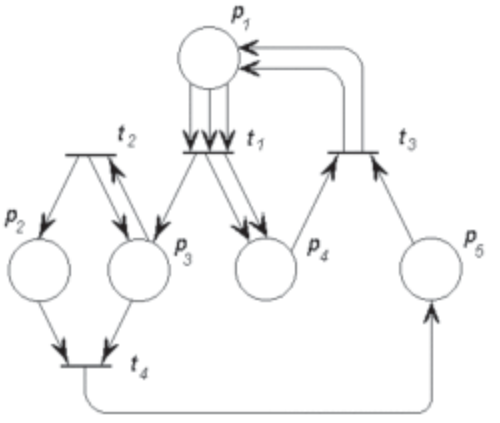


Рис. 7.2. Пример сети Петри

Комплекты входных позиций:

$$I(t_1) = \{p_1, p_1, p_1\};$$
$$I(t_2) = \{p_3\};$$
$$I(t_3) = \{p_4, p_5\};$$
$$I(t_4) = \{p_2, p_3\}.$$

Комплекты выходных позиций:

$$O(t_1) = \{p_3, p_4, p_4\};$$
$$O(t_2) = \{p_2, p_3\};$$
$$O(t_3) = \{p_1, p_1\};$$
$$O(t_4) = \{p_5\}.$$

Граф:

- двудольный – вершины состоят из 2-х множеств – множества позиций и множества переходов;
- ориентированный – дуги направлены (стрелки);
- мультиграф – допускает существование кратных дуг от одной вершины к другой.

Функции O и I удобно обобщить и на отображение из позиции в комплекты переходов $P \to T^{\infty}$ и обозначать комплекты входных и выходных переходов $I(p_i)$ и $O(p_i)$, например, $I(p_3) = \{t_1, t_2\}$, $O(p_3) = \{t_2, t_4\}$.

Все эти понятия относятся к статической структуре сети Петри. Динамические свойства сети Петри определяются с помощью понятия маркировки.

7.4. Маркировка сетей Петри

Маркировка (метка) μ сети Петри $C=(P,T,I,O)$ – это функция, отображающая множество позиций P в множество неотрицательных целых чисел N.

$$\mu: P \to N.$$

Маркировка изображается с помощью точек (маркеров, фишек) или чисел, которые помещаются внутрь позиций. Маркировка может определяться как n-вектор: $\mu=(\mu_1,\mu_2,...,\mu_n)$, где $n=|P|$ и каждое $\mu_i \in N$ $(i=1,2,...,n)$ есть $\mu(p_i)$. Вектор μ определяет для каждой позиции p_i сети Петри количество точек (маркеров, фишек) в этой позиции.

Сеть Петри с маркировкой называется маркированной. Маркированная сеть Петри $M=(C,\mu)$ есть совокупность сети Петри $C=(P,T,I,O)$ и маркировки μ: $M=(P,T,I,O,\mu)$.

Маркировка для сети, изображенной на рис. 7.3,

$$\mu(p_1)=1,\ \mu(p_2)=2,\ \mu(p_3)=\mu(p_4)=0,\ \mu(p_5)=1.$$

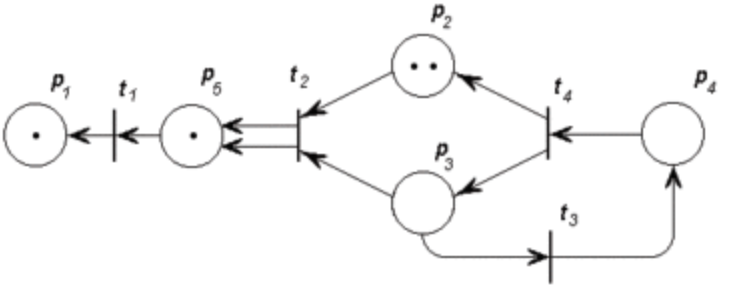


Рис. 7.3. Пример маркированной сети Петри

Маркировка характеризует динамику изменения состояний системы, причем динамика изменения состояний моделируется движением точек по позициям сети. Маркировка может изменяться в результате запуска переходов. Переход t_j маркированной сети с маркировкой μ называется разрешенным, если $I(t_j) \le \mu$, то есть в каждой входной позиции t_j находится не меньше точек, чем из этой позиции исходит дуг в t_j. Всякий разрешенный переход может запуститься. В результате запуска перехода t_j маркировка сети изменяется на новую: из всякой входной позиции p_i перехода t_j удаляется столько точек, сколько дуг ведет из p_i в t_j, а в каждую выходную позицию p_k помещается столько точек, сколько дуг ведет из t_j в p_k. Последовательность запусков переходов называется выполнением сети.

Вернемся к сети Петри, изображенной на рис. 7.2. Пусть позиции p_1 и p_3 содержат по одной точке. Рассмотрим выполнение такой маркированной сети. В данной маркировке разрешен только переход t_2. При его запуске точка удалится из p_3, а затем в позициях p_2 и p_3 появится по точке, то есть в результате запуска появится точка еще и в p_2.

Теперь становятся разрешенными переходы t_2 и t_4. Поскольку запуститься может любой разрешенный переход, предположим, что запускается переход t_4. После его запуска из позиции p_2 и p_3 точки удалятся, а в позиции p_5 появится одна точка. В получившейся маркировке μ'' не разрешен ни один переход. На этом выполнение сети Петри заканчивается.

Маркировка μ' называется непосредственно достижимой из μ, если найдется такой переход t_j, разрешенный в μ, что при его запуске получается маркировка μ', то есть

$$\mu \overset{t_j}{\mapsto} \mu'.$$

Маркировка μ' называется достижимой из маркировки μ, если существует последовательность переходов $\sigma = t_{j_1}, t_{j_2}, \ldots, t_{j_k}$, срабатывание которых переводит сеть из маркировки μ в маркировку μ', то есть

$$\mu \overset{\sigma}{\mapsto} \mu'.$$

Множество достижимых из μ маркировок сети Петри C называется множеством достижимости и обозначается $R(C, \mu)$.

Интерпретация сетей Петри основана на понятиях условия и события. Состояние системы описывается совокупностью условий. Функционирование системы – осуществление последовательности событий. Для возникновения события необходимо выполнение некоторых условий, называемых предусловиями.

Возникновение события может привести к нарушению предусловий и к появлению некоторых новых условий, называемых постусловиями. В сети Петри условия моделируются позициями, события – переходами. Предусловия события представляются входными позициями соответствующего перехода, постусловия – выходными позициями. Возникновение события моделируется запуском перехода. Выполнение условий представляется наличием точек в соответствующих позициях, невыполнение – их отсутствием.

7.5. Сети Петри для моделирования

Моделирование является основным применением сетей Петри. С их помощью моделируется аппаратное и программное обеспечение ЭВМ, физические, социальные, экономические системы, а также системы управления.

Пример. Вычислительная система обрабатывает задания, поступающие с устройства ввода, и выводит результаты на устройство вывода.

Задания поступают на устройство ввода. Если процессор свободен и в устройстве ввода есть задание, то процессор начинает обработку задания. Когда задание выполнено, оно посылается в устройство вывода; при этом процессор либо начинает обрабатывать другое задание, если оно имеется, либо ждет прихода задания, если устройство ввода еще его не получило (рис. 7.4).

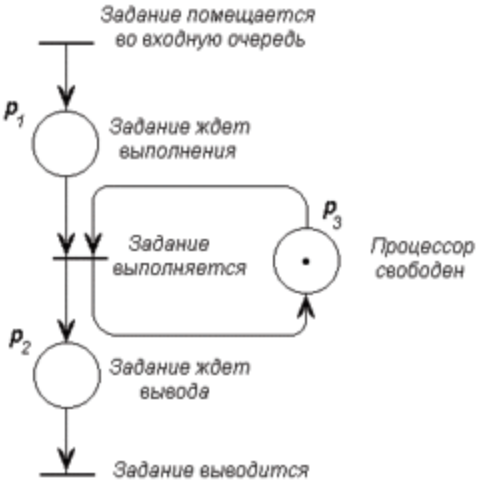


Рис. 7.4. Сеть Петри, моделирующая обработку заданий в вычислительной системе

Особенность систем, моделируемых сетями Петри, – параллелизм или одновременность. Два разрешенных невзаимодействующих события могут происходить независимо друг от друга. Синхронизировать события не надо. Сети Петри позволяют моделировать системы, в которых несколько процессов выполняются одновременно.

Особенность сетей Петри как инструмента моделирования является отсутствие понятия времени. Например, в сети Петри, изображенной на рис. 7.4, нет никакой информации о количестве времени, необходимом для выполнения задания. Срабатывание переходов сети Петри представляет собой последовательность дискретных событий. Порядок появления событий является одним из возможных, допускаемых структурой. Это приводит к явной случайности в выполнении сети Петри. Если в какой-то момент времени разрешено более одного перехода, то сработать может любой переход. Выбор запускаемого перехода осуществляется случайно. Это отражает тот факт, что при одновременности действий возникающий порядок появления события неоднозначен.

Поэтому для простоты обычно вводят следующее ограничение: запуск перехода – мгновенное событие и двух одновременных запусков (двух одновременных событий) не может быть. Моделируемое таким образом событие называется примитивным.

Примитивные события мгновенны и неодновременны. Если мы присвоим каждому событию время возникновения, то вероятность того, что время возникновения двух событий одинаково, равна 0 и, следовательно, события неодновременны.

Непримитивными называются события, длительность которых не равна 0, как, например, событие «задание выполняется» (рис. 7.4). Такие события не являются неодновременными и могут пересекаться во времени. Однако это не приводит к возникновению проблем при моделировании систем.

Непримитивное событие можно представить в виде двух примитивных событий: начало примитивного события, конец примитивного события. Эта ситуация может быть промоделирована с помощью сети, показанной на рис. 7.5*а*.

Карл Петри предложил представлять непримитивные события в сети в виде прямоугольника (рис. 7.5*б*), а примитивные в виде планки (черточки), как мы это раньше делали.

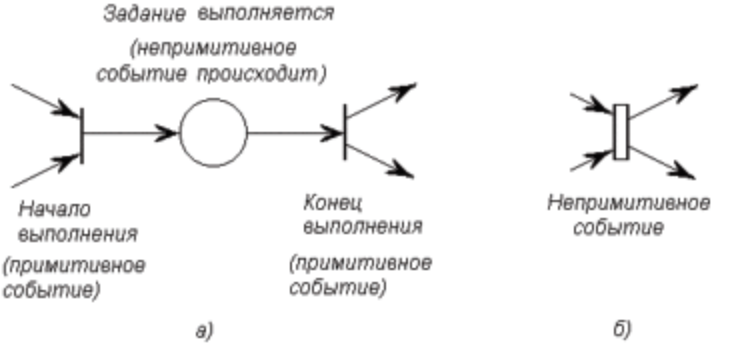


Рис. 7.5. Представление примитивных и непримитивных событий:
а) представление непримитивного события в виде двух примитивных;
б) представление непримитивного события в виде прямоугольника

Случайность и неодновременность запусков переходов в моделировании параллельной системы может привести к двум ситуациям.

1. Переходы t_j и t_k разрешены. Эти переходы не влияют друг на друга (рис. 7.6*а*). Возможна последовательность событий, когда первым срабатывает t_j и последовательность событий, когда первым срабатывает t_k. Эта ситуация называется одновременность или параллелизм.
2. Переходы t_j и t_k разрешены (рис. 7.6*б*). Но запуск одного из них удаляет точку из p_2 и, тем самым, делает запрещенным другой. Такая ситуация

называется конфликт. Таким образом, в этом случае может быть запущен только один переход.

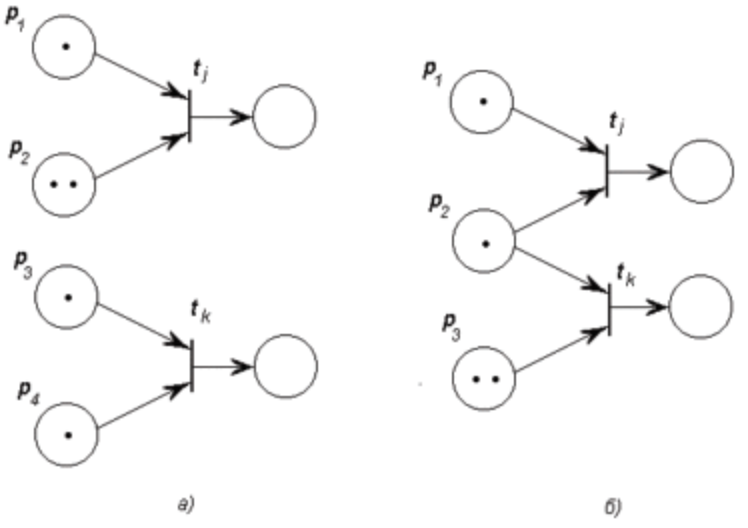


Рис. 7.6. Одновременность (а) и конфликт (б)

Подобные ситуации требуют внимательного изучения систем, которые моделируются сетями Петри.

Рассмотрим применение сетей Петри для моделирования аппаратного и программного обеспечения систем.

7.6. Моделирование аппаратного обеспечения сетями Петри

В качестве примера рассмотрим устройство, преобразующее двоичное число в дополнение до двух. В разделе 5.17 такое устройство было представлено в виде конечного автомата, входной и выходной алфавиты которого состоят из 3-х символов: $0, 1$ и R (рис. 7.7).

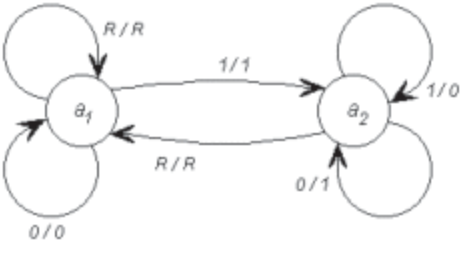


Рис. 7.7. Автоматная модель устройства, преобразующего отрицательное двоичное число в дополнительный код

При моделировании такого конечного автомата сетью Петри каждый входной и выходной символы, а также состояния могут быть представлены позициями (рис. 7.8).

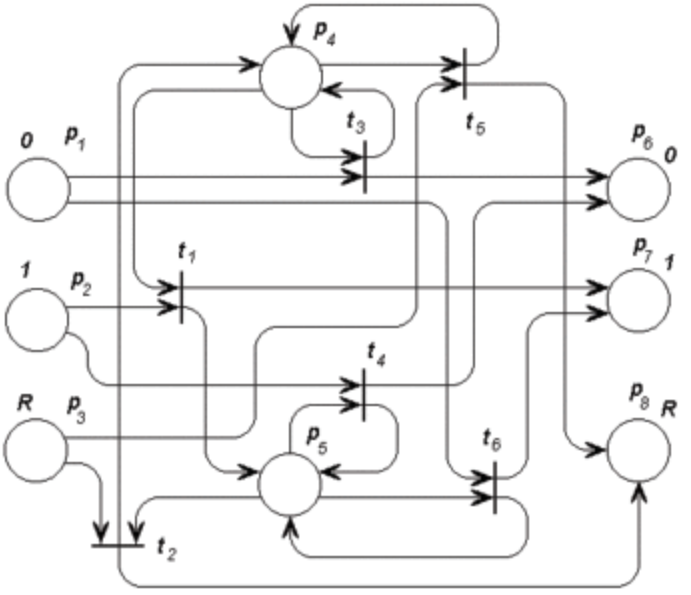


Рис. 7.8. Модель в виде сети Петри устройства, преобразующего отрицательное двоичное число в дополнительный код

Возникает вопрос: зачем здесь нужна сеть Петри? Ведь автоматное описание (рис. 7.7) более понятно и компактно, чем сеть Петри, в которой 6 переходов, 8 позиций, 24 дуги (рис. 7.8). Здесь мы просто показали, что сетью Петри можно описать устройство, представленное автоматом.

Но сети Петри имеют преимущества перед композицией автоматов. Композиция автоматов – это задача построения автоматной модели устройства по известным автоматным моделям отдельных частей этого устройства. Пусть, например, требуется построить систему, вычисляющую дополнение до двух и проверяющую полученное двоичное число на четность. Композиция автоматов, заключающаяся в объединении отдельных автоматных моделей в общую автоматную модель, представляет собой достаточно трудную задачу. При моделировании сетью Петри – это просто совмещение выходных позиций первой сети с входными позициями второй в случае последовательной работы отдельных устройств (рис. 7.9*а*). При параллельной работе устройств модель в виде сети Петри примет вид, показанный на рис. 7.9*б*. Точки во входных позициях, соответствующих входным символам, дублируются и используются одновременно в двух сетях Петри. Выходом является выход первой и выход второй сети.

Итак, описать работу системы при помощи композиции автоматов – задача достаточно трудная. Сеть Петри здесь имеет неоспоримое преимущество. Сеть Петри и возникла как связь автоматов.

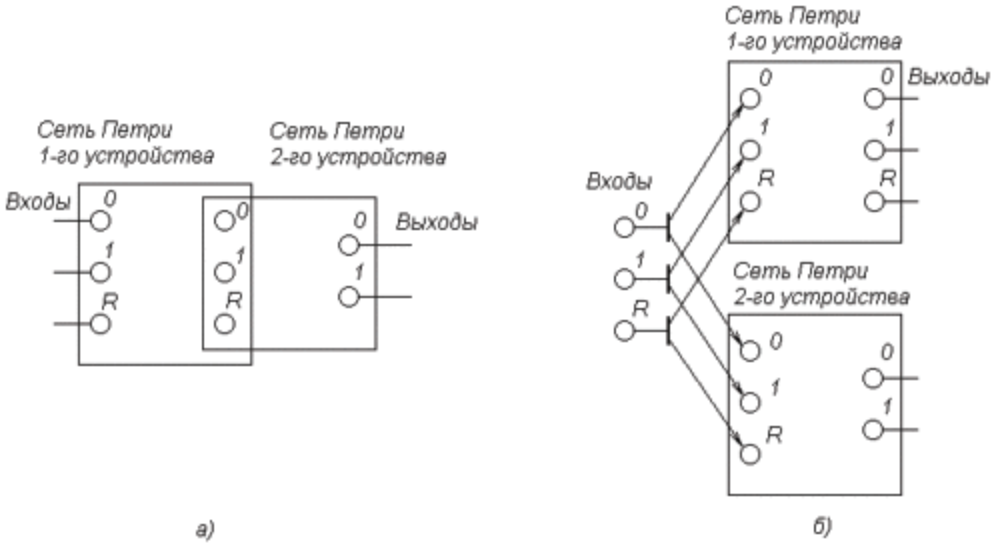


Рис. 7.9. Моделирование сетью Петри совместной работы двух устройств:
а) при последовательной работе; б) при параллельной работе

Рассмотрим еще примеры моделирования при помощи сетей Петри аппаратного обеспечения вычислительных систем. При разработке аппаратуры приходится решать одну из главных задач: как увеличить производительность вычислительной системы? Один из способов – конвейерная обработка данных.

Рассмотрим пример конвейерной обработки данных. Пусть требуется сложить два числа с плавающей точкой. Эту операцию сложения можно разбить на следующие этапы:

- выделить экспоненты;
- сравнить экспоненты;
- сдвинуть точку в числе с меньшей экспонентой;
- сложить дроби;
- нормализовать результат, если это необходимо;
- проверить экспоненту на переполнение.

Например, необходимо сложить два числа: 526.381 и 42.897. Выполним вышеназванные этапы.

1. $526.381 = 0.526381 \cdot 10^3$;
   $42.897 = 0.42897 \cdot 10^2$.
2. Во втором числе экспонента меньше.
3. Сдвигаем точку во втором числе: $0.42897 \cdot 10^2 = 0.042897 \cdot 10^3$.
4. Складывая мантиссы, получим 0.569278.
5. Нормализация результата не нужна.

6. $10^{-78} < 10^{3} < 10^{76}$, то есть экспонента не выходит за допустимые пределы.

Каждый из этих шагов может быть выполнен своим вычислительным блоком, и результаты вычисления передаются от блока к блоку (как бы по конвейеру). Такая организация позволит выполнять до 6 сложений одновременно. Как координировать работу блоков? Можно разными способами.

1. Установить постоянное и фиксированное время t на выполнение каждого шага. Через каждые t единиц времени результат каждого блока перемещается и становится входом следующего блока. Это синхронный принцип обработки. Недостаток – требуемое время для каждого блока может быть разным. И оно, конечно, разное. Кроме того, один и тот же блок может требовать разное время для разных входов (чисел). Например, для выполнения шага «Нормализация» требуется разное время в зависимости от того, на сколько разрядов сдвигать точку. Время же t должно быть выбрано максимальное, т.е. такое, за которое самый медленный блок выполнит обработку самого «трудного» числа. Поэтому, как правило, в этом случае другие блоки оказываются незагруженными и ждут окончания работы самого медленного блока.
2. Асинхронный способ. Процесс обработки может быть ускорен при помощи сигнализации о завершении работы каждого блока и готовности блока передать свой операнд и получить новый.

Результат шага k-ой обработки может быть послан на шаг $k+1$, как только шаг k выполнен, а блок $k+1$ свободен. Рассмотрим произвольный шаг обработки. Очевидно, что нужно иметь место, куда можно поместить входы и выходы. Нужны регистры; блок использует значение своего входного (буферного) регистра, производит обработку и ждет, пока:

- выходной регистр блока не будет очищен, то есть произойдет пересылка его содержимого во входной регистр следующего блока;
- новое входное значение не появится в его входном регистре.

Таким образом, для управления блоком k необходимо знать, когда выполняются следующие условия:

- входной регистр заполнен;
- входной регистр пуст;
- выходной регистр заполнен;
- блок занят;
- блок свободен;
- пересылка осуществлена.

Блок-схема такого процесса представлена на рис. 7.10, сеть Петри, моделирующая процесс, – на рис. 7.11.

Рис. 7.10. Блок схема процесса сложения двух чисел

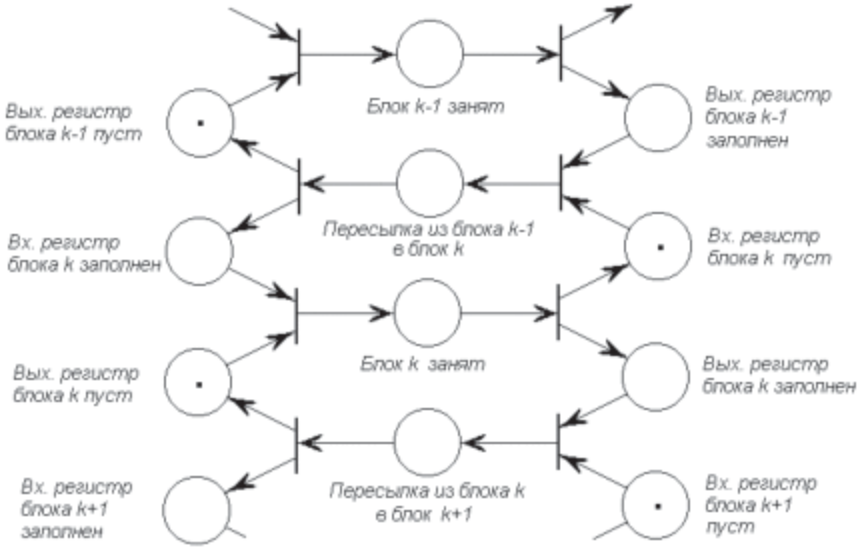


Рис. 7.11. Процесс конвейерной обработки в виде сети Петри на примере сложения двух чисел

7.7. Моделирование сетями Петри программного обеспечения

Отдельный процесс описывается программой. Программа может быть написана на различных языках и представляет два аспекта процесса:

1. Вычисление.

2. Управление.

Вычисление связано с арифметическими и логическими операциями, вводом-выводом. Управление определяет порядок выполнения операций.

Программа может быть представлена в виде сети Петри. Сеть Петри отображает порядок инструкций и поток информации, но не вычисление самих значений и может быть полезной для моделирования параллельных взаимодействующих процессов.

Обычный способ представления структуры управления программ в виде блок-схем во многом подобен сети Петри. Блок-схема содержит узлы и дуги. Узлы отображают действия и могут быть двух типов:

- прямоугольники (вычисления);
- ромбы (принятия решений).

Порядок выполнения действий в блок-схеме определяется дугами. Блок-схема задает единственно возможный порядок действий.

Так как в сети Петри действия моделируется переходами, то при переходе от блок-схемы к сети Петри нужно заменить узлы блок-схемы переходами, а дуги блок-схемы позициями сети Петри. Каждая дуга блок-схемы соответствует точно одной позиции сети Петри. Узлы блок-схемы представляются по-разному в зависимости от типа узла: вычисление или принятие решения. На рис 7.12. показаны переходы сети Петри, соответствующие разным типам узлов блок-схемы.

Переходы сети Петри, моделирующие принятие решения в программе, нагружаются дополнительными условиями (да, нет), и срабатывание таких переходов определяется не только структурой сети, но и выполнением дополнительных условий. Таким образом, на рис. 7.12, *б* сработать может только один переход.

Параллелизм (одновременность) вводится несколькими способами. Например, два параллельных процесса, каждый процесс может быть представлен сетью Петри. Составная сеть – простое объединение этих двух сетей. Начальная маркировка составной сети Петри имеет два маркера, по одному в каждой сети.

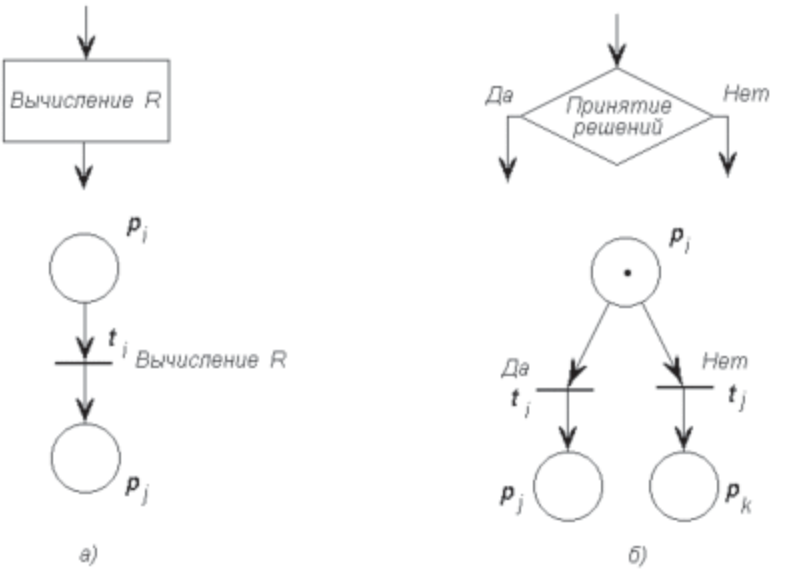


Рис. 7.12. Представление сетью Петри различных типов узлов блок-схемы
а) представление вычисления; б) представление принятия решения

Другой способ введения параллелизма основан на двух операциях: parbegin и parend (рис. 7.13).

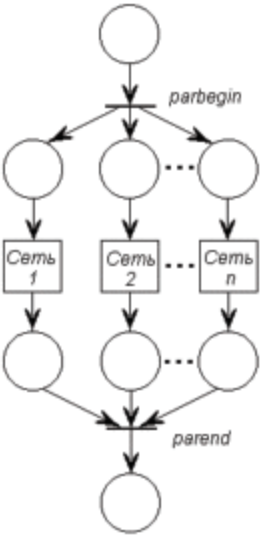


Рис. 7.13. Сеть Петри, моделирующая параллелизм
при выполнении программы

Сеть Петри может использоваться для анализа последовательностей выполнения операторов в программе.

В результате такого анализа выявляются возможные параллельные вычисления, что важно для ЭВМ, которые могут выполнять различные инструкции параллельно.

Например, имеется фрагмент программы, содержащей три оператора:

$$\begin{array}{ll} 10 & x = x + 1 \\ 20 & y = y + 1 \\ 30 & z = x + y \end{array}$$

Программа задает однозначный порядок следования операторов и написана таким образом, что оператор 10 предшествует оператору 20. Но данная последовательность вовсе не является обязательной. Очевидно, что порядок, в котором будут выполнены операторы 10 и 20 (или они будут выполнены одновременно), не имеет значения для программы. Однако оператор 30 должен следовать только после операторов 10 и 20. Сеть Петри, моделирующая программу, исключает ограничения на последовательность выполнения операторов и показывает параллелизм везде, где только он возможен.

Для параллельного выполнения операторов необходимы вычислительные системы с несколькими регистрами и функциональными блоками. Поэтому составляется вторая сеть Петри, моделирующая устройство управления и определяющая ограничения, налагаемые аппаратным обеспечением. Вторая сеть объединяется с сетью, моделирующей программу. Объединенная сеть Петри представляет собой все возможные последовательности инструкций, которые допускаются программой и могут выполняться аппаратурой.

Эта сеть затем выполняется для получения всех таких последовательностей инструкций. Две последовательности образуются всякий раз, когда разрешены одновременно два перехода. Запуск одного перехода будет порождать одну последовательность, запуск другого – другую.

После того, как последовательности получены, вычисляются затраты времени на их выполнение, и наиболее быстрая последовательность генерируется компилятором для действительного выполнения.

7.8. Примеры типовых задач, решаемых с помощью сетей Петри

Рассмотрим задачи, решаемые с помощью сетей Петри и ставшие классическими.

Задача о взаимном исключении

Предположим, что несколько процессов используют общую переменную, запись, файл или элемент данных, который принято называть разделяемый

объект. Разделяемый объект может использоваться процессами следующим образом. Упрощенно это можно представить как:

- чтение значения элемента данных;
- запись нового значения.

Для обновления элемента данных процесс должен сначала считать старое значение, затем вычислить новое и, наконец, записать его на то же место.

Если два процесса имеют доступ к разделяемому объекту (рис. 7.14*а*, где позиции p_2 и p_2' моделируют один и тот же разделяемый объект), то возможна следующая последовательность:

- первый процесс считывает значение x из разделяемого объекта;
- второй процесс считывает значение x из разделяемого объекта;
- первый процесс вычисляет новое значение $x' = f(x)$;
- второй процесс вычисляет новое значение $x'' = g(x)$;
- первый процесс записывает x' в разделяемый объект;
- второй процесс записывает x'' в разделяемый объект, уничтожая при этом x'.

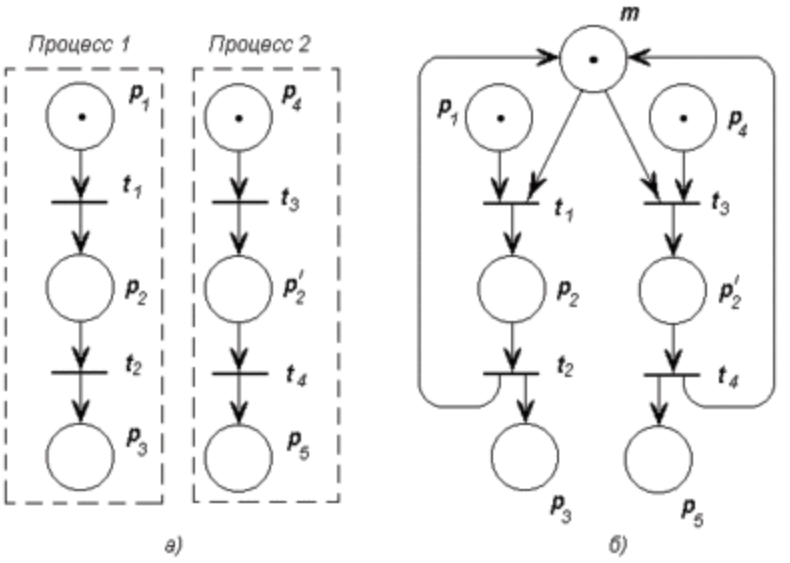


Рис. 7.14. Задача о взаимном исключении:
а) оба процесса имеют доступ к разделяемому объекту;
б) одновременно не более чем один процесс имеет доступ к разделяемому объекту

Итак, результат вычисления первого процесса потерян, так как теперь значение разделяемого объекта является $g(x)$, в то время как им должно быть либо $g(f(x))$, либо $f(g(x))$. Представим себе, что $g(x)$ – снять деньги со счета x, $f(x)$ – поместить деньги на счет x, а процессы 1, 2 – банковские операции.

Для предотвращения таких проблем необходим механизм взаимного исключения. Взаимное исключение – одновременно не более чем один процесс имеет доступ к данным. Идея в том, что когда процесс готов выполнить операции с данными, то он сначала ждет, пока другой процесс не закончит выполнения операций с теми же данными. Затем он блокирует доступ к данным, не давая возможность никакому другому процессу использовать их. После обработки данных они становятся доступны для других процессов.

Эта задача решается с помощью сетей Петри следующим образом (рис. 7.14*б*). Позиция m – разрешение для обращения к данным. Для того чтобы какой-либо процесс мог использовать данные, он должен иметь точку в p_1 или в p_4 соответственно, свидетельствующую о желании их использовать. Кроме того, должна существовать точка и в позиции m, дающая разрешение на вход. Если оба процесса пытаются начать работать с данными одновременно, то переходы t_1 и t_3 вступают в конфликт, и только один из них сможет запуститься. Запуск перехода t_1 запретит запуск перехода t_3 и наоборот. Второй процесс вынужден ждать, пока первый процесс закончит работу и отправит точку обратно в позицию m.

Задача о производителе и потребителе

В задаче о производителе и потребителе также существует совместно используемый объект. Процесс-производитель создает объекты, которые помещаются на склад. Потребитель ждет, пока объект не будет помещен на склад, забирает его оттуда и использует. Такая ситуация может быть промоделирована сетью Петри (рис. 7.15). Позиция p_5 моделирует склад. Точки в этой позиции означают, что произведенная продукция имеется на складе.

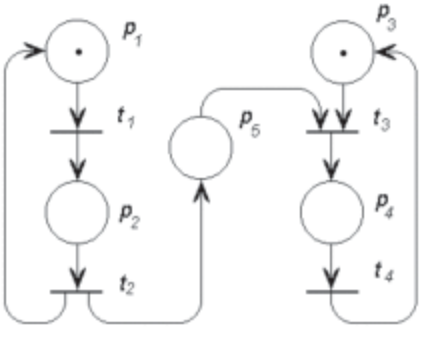


Рис. 7.15. Задача о производителе и потребителе

Возможны различные варианты этой задачи, например, несколько производителей и несколько потребителей. Другой вариант – склад ограниченного размера. При такой постановке задачи склад имеет только n мест для произведенной продукции. Следовательно, производитель не может

постоянно работать с большой скоростью, а будет ждать, если потребитель работает медленно и весь склад заполнен. Сеть Петри, моделирующая эту задачу, представлена на рис. 7.16. Складу ограниченного размера сопоставлены две позиции – p_5 и p_6. Точки в позиции p_5 моделируют количество произведенной продукции, хранящейся на складе и еще не использованной, то есть число заполненных мест, а точки в позиции p_6 – число свободных мест на складе.

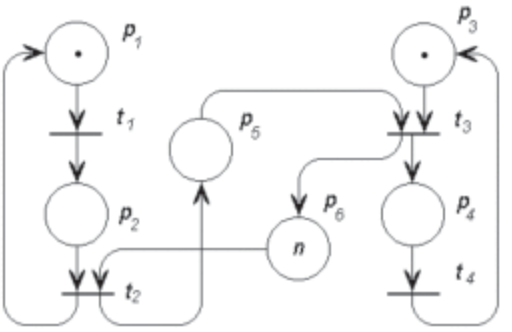


Рис. 7.16. Задача о производителе и потребителе
со складом ограниченного размера

Первоначально p_6 имеет n точек, а p_5 точек не имеет. Когда в p_5 n точек, а производитель пытается поместить еще один элемент на склад, то он будет остановлен, так как в p_6 нет точки (нет свободного места). Переход t_2 становится не разрешен.

Сети Петри могут применяться для моделирования различных систем. Бегло рассмотрим некоторые из них.

PERT-диаграммы (сетевые графики), используемые для планирования

PERT-диаграмма – графическое представление взаимосвязи между различными этапами, составляющими проект. Этапы – узлы. Дуги – связи между узлами. PERT-диаграмма легко преобразуется в сеть Петри. Каждый этап – позиция, причинно-следственные связи – переходы. Сеть Петри – средство для представления параллелизма и причинно-следственной связи. Но PERT-диаграмма – это еще и информация о времени. Сеть Петри не содержит такой информации. Добавление информации о времени позволяет создать мощное средство моделирования, так называемые временные сети Петри.

Производственные системы

Сети Петри применяются для моделирования конвейерной обработки, сборочных линий, использования общего ресурса.

Методология построения систем логического управления гибкими производственными системами (ГПС) и роботизированными технологическими

комплексами использует формальный аппарат сетей Петри и их модификаций как средство описания функционирования технологического объекта управления. Полученное в виде сети Петри описание после выполнения анализа на корректность функционирования проектируемой системы представляется в аналитическом виде и является основой для аппаратной или программной реализации системы управления.

Химические системы

Вещества, участвующие в реакции – позиции, химические уравнения – переходы. В случае необходимости катализатора так же удобно использовать сети Петри.

Юридические системы

Действующие лица – судьи, адвокаты, обвиняемые, свидетели могут одновременно выполнять действия, относящиеся к конкретному делу. Действия и их взаимосвязи могут быть представлены сетью Петри.

7.9. Свойства сетей Петри

Мы рассмотрели, как с помощью сетей Петри можно моделировать системы. Смоделировав систему, необходимо провести анализ модели – ради чего, собственно, и нужна модель. Ведь само по себе моделирование бесполезно.

Анализ сетей Петри заключается в определении: обладает ли сеть необходимыми свойствами? Рассмотрим, какие свойства сети Петри как модели системы могут интересовать разработчика.

Безопасность. Позиция $p_i \in P$ сети Петри $C = (P, T, I, O)$ с начальной маркировкой μ является безопасной, если $\mu'(p_i) \le 1$ для любой $\mu' \in R(C, \mu)$. Другими словами, позиция сети Петри является безопасной, если число точек в ней никогда не превышает 1. Сеть Петри безопасна, если безопасны все позиции сети.

Безопасную позицию можно реализовать одним триггером. Это свойство очень важно при интерпретации позиций как простых условий: если в позиции есть точка, то условие выполняется, если нет, то не выполняется.

Ограниченность. Расширение свойства безопасности. Безопасность – необязательное требование. Безопасность позволяет реализовать позицию триггером, но в более общем случае можно использовать счетчик. Однако любой аппаратно-реализуемый счетчик ограничен по максимальному числу k, которое он может представить.

Позиция $p_i \in P$ сети Петри $C = (P, T, I, O)$ с начальной маркировкой μ k-ограничена или k-безопасна, если $\mu'(p_i) \le k$ для всех $\mu' \in R(C, \mu)$, то есть позиция является k-ограниченной, если количество точек в ней не превышает целое число k. 1-безопасная позиция – просто безопасная позиция.

Сеть, естественно, может содержать позиции, безопасность которых различна. Однако, если позиция p_i k-безопасна, то она и k'-безопасна для всех $k' \ge k$. Поэтому сеть Петри k-безопасна, если каждая ее позиция k-безопасна, где k – верхняя граница безопасности ее позиций.

Часто важно знать, является ли число точек в позиции ограниченным или нет, а не конкретное значение границы. Позиция называется ограниченной, если число точек в ней конечно. Сеть Петри ограниченна, если все ее позиции ограничены. Ограниченную сеть можно реализовать аппаратно, сеть с неограниченными позициями в общем случае реализовать аппаратно нельзя.

Сохранение. Маркировка позиций сети Петри может моделировать некоторые объекты или ресурсы. Для таких сетей важным свойством является сохранение. Необходимо, чтобы точки, представляющие ресурсы, при срабатывании переходов не исчезали и не создавались, то есть общее число точек в сети должно оставаться постоянным.

Сеть Петри $C = (P, T, I, O)$ с начальной маркировкой μ называется строго сохраняющей, если $\forall \mu', \mu' \in R(C, \mu)$ имеет место

$$\sum_{i=1}^{n} \mu'(p_i) = \sum_{i=1}^{n} \mu(p_i), \quad \text{где } n - \text{число позиций в сети.}$$

Строгое сохранение – сильное требование. Из него немедленно следует, что число входов в каждый переход должно быть равно числу выходов: $|I(t_j)| = |O(t_j)|$. Если бы это было не так, то запуск перехода изменил бы число точек в сети. Рассмотрим сеть, представленную на рис. 7.17*а*. Сеть не является строго сохраняющей, поскольку запуск перехода t_1 уменьшит число точек на единицу, а запуск перехода t_2 увеличит число точек (добавит точку к маркировке). Но мы можем легко преобразовать эту сеть в сеть строго сохраняющую (рис. 7.17*б*).

Сеть Петри должна сохранять ресурсы, которые она моделирует. Однако строгое взаимно-однозначное соответствие между точками и ресурсами не обязательно. В общем случае можно ввести вес w_i позиции p_i и вектор взвешивания $W = (w_1, w_2, \ldots, w_n)$ с целыми компонентами, строго большими 0.

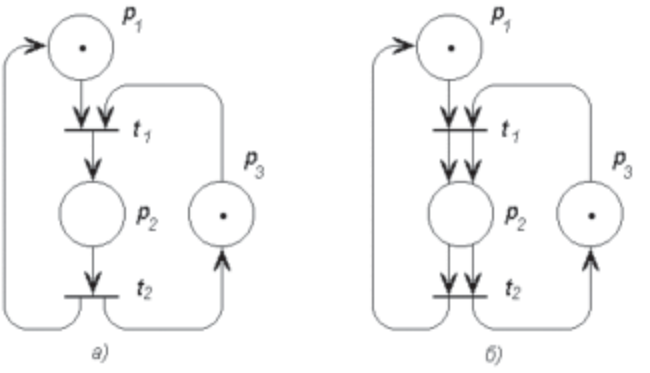


Рис. 7.17. Преобразование сети, не являющейся строго сохраняющей (а), в строго сохраняющую сеть (б)

Сеть Петри с начальной маркировкой μ сохраняющая по отношению к вектору взвешивания, если сумма произведений веса w_i и маркировки μ' постоянна для всех $\mu' \in R(C,\mu)$, то есть $\sum_{i=1}^{n} w_i \mu'(p_i) = \sum_{i=1}^{n} w_i \mu(p_i)$, где n – число позиций в сети.

Для нашего примера (рис. 7.17 *а*):

$$w_1 = 1;\ \ w_2 = 2;\ \ w_3 = 1;$$

$$\sum_i w_i \mu'(p_i) = const \text{ во всех маркировках.}$$

Сеть Петри сохраняющая, если она является сохраняющей по отношению к некоторому вектору взвешивания. Строго сохраняющая сеть Петри является сохраняющей по отношению к вектору взвешивания (1,1,...,1). Все сети являются сохраняющими по отношению к вектору взвешивания (0,0,...,0). Поэтому и возникает ненулевой вектор взвешивания $W > 0$ ($w_i > 0$, $i = 1,2,...,n$, где n – число позиций в сети). Нулевая компонента вектора W означает, что сохранение проверяется без учета соответствующей позиции.

Живучесть. Сеть Петри называют «живой» при заданной начальной маркировке, если:

1. Для любой пары маркировок (μ_i, μ_j), принадлежащих множеству $R(C,\mu)$, имеет место $\mu_i \overset{\sigma}{\mapsto} \mu_j$, то есть существует последовательность переходов $\sigma = t_{j_1}, t_{j_2}, ..., t_{j_k}$, срабатывание которых переводит сеть из маркировки μ_i в маркировку μ_j.

2. Для любого перехода t_k в множестве $R(C,\mu)$ существует такая пара маркировок (μ_n, μ_m), что $\mu_n \overset{t_k}{\mapsto} \mu_m$, то есть срабатывание перехода t_k переводит сеть из маркировки μ_n в маркировку μ_m.

Живые и безопасные сети называются правильными сетями. Правильность сети важна тогда, когда сетью Петри описывается технологический процесс. В этом случае позициям сопоставляются операции, а переходам – условия смены операций. При этом попадание точки в позицию ассоциируется с началом операции, а удаление точки из нее – с концом операции. Для обеспечения корректности дискретного технологического процесса описывающая его сеть должна быть правильной.

Активность. При моделировании систем предметом многих исследований являются тупики. Тупик в сети Петри – это переход или множество переходов, которые не могут быть запущены в некоторой маркировке μ' и последующих из μ' маркировках. Переход называется активным, если он нетупиковый (не заблокирован). Это не означает, что он разрешен, скорее он может быть разрешенным.

Переход t_j в сети Петри C называется потенциально запустимым в маркировке μ, если существует маркировка $\mu' \in R(C,\mu)$, в которой t_j разрешен. Переход активен в маркировке μ, если потенциально запустим во всякой маркировке из $R(C,\mu)$. Следовательно, если переход активен, то всегда можно перевести сеть Петри из ее текущей маркировки в маркировку, в которой запуск перехода станет разрешенным.

Существуют и другие, связанные с активностью понятия, например, уровни активности. Их можно определить для сети C с маркировкой μ следующим образом.

Уровень 0. Переход t_j обладает активностью уровня 0, если он никогда не может быть запущен. Переход, обладающий активностью уровня 0, называется пассивным.

Уровень 1. Переход t_j обладает активностью уровня 1, если он потенциально запустим, то есть если существует такая $\mu' \in R(C,\mu)$, что t_j разрешен в μ'.

Уровень 2. Переход t_j обладает активностью уровня 2, если существует последовательность запусков, в которой t_j присутствует, по крайней мере, n раз.

Уровень 3. Переход t_j обладает активностью уровня 3, если существует бесконечная последовательность запусков, в которой t_j присутствует неограниченно часто.

Уровень 4. Переход t_j обладает активностью уровня 4, если он разрешен в любой маркировке $\mu' \in R(C, \mu)$.

Сеть Петри обладает активностью уровня i, если каждый ее переход обладает активностью уровня не ниже i.

Пример. Рассмотрим сеть, изображенную на рис. 7.18.

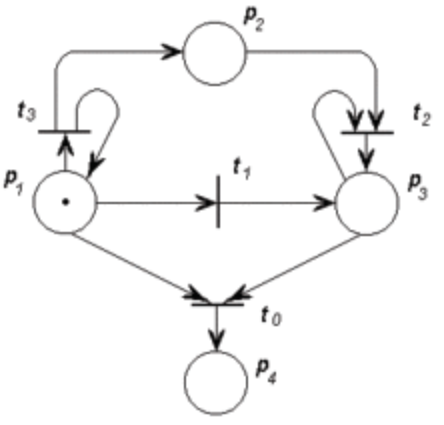


Рис. 7.18. Сеть Петри с различной активностью переходов

Переход t_0 не может быть запущен никогда; он пассивен. Переход t_1 можно запустить только один раз; он обладает активностью уровня 1. Переход t_2 может быть запущен произвольное число раз, но это число зависит от числа запусков перехода t_3, например, чтобы 5 раз запустить переход t_2, необходимо сделать 5 запусков t_3, затем t_1 и после этого 5 раз t_2. Однако, как только запустился t_1 (t_1 должен быть запущен до того, как будет запущен t_2), число возможных запусков t_2 станет строго фиксированным. Следовательно, t_2 обладает активностью уровня 2, но не уровня 3. Переход t_3 можно запустить бесконечное число раз; он обладает активностью уровня 3, но не уровня 4, так как если запустится t_1, то t_3 запустить больше будет нельзя.

7.10. Задачи анализа сетей Петри

Анализ сети Петри, кроме выявления ее свойств, предполагает также решение определенных задач. Рассмотрим эти задачи.

Задача достижимости. Наиболее важная задача анализа – задача достижимости: для данной сети Петри C с маркировкой μ и маркировки μ' определить, принадлежит ли маркировка μ' множеству достижимых маркировок, то есть $\mu' \in R(C, \mu)$?

Многие задачи анализа можно сформулировать в терминах задачи достижимости. Например, в сети Петри возникает тупик (что недопустимо), если достижимой является такая-то маркировка. Или другой пример: для правильного функционирования моделируемой системы необходима достижимость определенных маркировок.

Задача покрываемости. Задача покрываемости является расширением задачи достижимости и имеет следующую формулировку: для данной сети Петри C с начальной маркировкой μ и маркировкой $\mu' \in R(C, \mu)$ определить, существует ли маркировка $\mu'' \in R(C, \mu)$, такая что $\mu'' \geq \mu'$?

В других возможных задачах типа достижимости интерес ограничен некоторыми позициями, а маркировка остальных позиций не важна. Эти задачи называются задачами достижимости подмаркировки и покрываемости подмаркировки.

Последовательность запусков. При анализе сетей Петри важен вопрос: может ли переход быть запущен (иначе, не является ли он пассивным)? В более общем случае возникает необходимость определить, возможна ли заданная последовательность запусков переходов?

Задачи оптимизации и эквивалентности. Пусть сети Петри присуще некоторое поведение, определяемое множеством последовательностей запусков переходов или ее множеством достижимости. Можно ли оптимизировать (минимизировать) сеть, не изменяя ее поведения? Оптимизировать или минимизировать сеть – это значит, например:

- удалить пассивные переходы (которые никогда не удастся запустить);
- удалить пассивные позиции (которые никогда не будут маркированы);
- переопределить некоторые переходы.

Задача эквивалентности – могут ли две различные маркированные сети Петри с одинаковым числом переходов (но с разным числом позиций) порождать одинаковые последовательности запусков переходов или две различные маркированные сети Петри с одинаковым числом позиций (но с разным числом переходов) порождать одно множество достижимости?

7.11. Методы анализа сетей Петри

Существуют два основных метода анализа сетей Петри:

1. Дерево достижимости.
2. Матричный метод.

Дерево достижимости

Дерево достижимости содержит множество достижимости и представляет собой граф, вершины которого маркировки, а дуги – переходы; при

срабатывании переходов данная маркировка изменяется на новую. Рассмотрим маркированную сеть (рис. 7.19*а*). Начальная маркировка (1,0,0) является корнем дерева (рис. 7.19*б*).

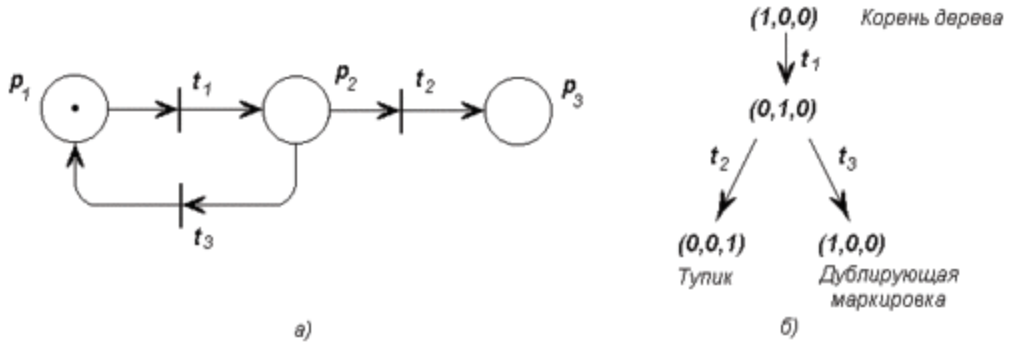


Рис. 7.19. Сеть Петри (а) и соответствующее ей дерево достижимости (б)

Поскольку мы хотим рассмотреть все маркировки, определим вершины в дереве достижимости для маркировок, получающихся в результате запуска каждого из переходов. В начальной маркировке разрешен только один переход: t_1. Дуга, помеченная запускаемым переходом, изменяет начальную маркировку (1,0,0) на новую (0,1,0), непосредственно достижимую из начальной маркировки.

Теперь необходимо рассмотреть все маркировки, достижимые из новой маркировки.

Из маркировки (0,1,0) можно запустить t_2 и получим (0,0,1), но можно запустить и t_3, получив (1,0,0). Таким образом, последовательно, начиная с корневой вершины, ярус за ярусом строится дерево, и определяются новые маркировки. Маркировка (0,0,1) – пассивная, никакой переход в ней не является разрешенным, поэтому никакие новые маркировки из этой пассивной маркировки в дереве порождаться не будут. Маркировка (1,0,0), порождаемая запуском t_3 в маркировке (0,1,0), уже встречалась в дереве.

Назовем вершины (и соответствующие маркировки), построенные на очередном шаге алгоритма, граничными. Если в какой-либо граничной маркировке нет больше разрешенных переходов, то соответствующие граничные вершины называются терминальными или тупиковыми и исходящих дуг они не имеют.

Если какая-либо граничная вершина имеет маркировку, которая уже встречалась в дереве, то такие вершины называются дублирующими. Если строить дуги, исходящие из дублирующих вершин, то будем порождать маркировки, которые уже встречались в сети, и новой информации мы не получим. Поэтому на дублирующих вершинах построение дерева также заканчивается. Если вершина не терминальная и не дублирующая, то для нее продолжаем строить дерево дальше и эта вершина становится внутренней. И

так до тех пор, пока все полученные вершины не станут терминальными и дублирующими. Таким образом, для ограниченной сети Петри дерево достижимости будет конечно.

Это что касается ограниченных сетей. Как быть с неограниченными сетями? Как в этом случае остановить рост дерева? Ведь если маркировка позиции может расти до сколь угодно большой величины, то мы никогда не получим терминальную или дублирующую вершину. Естественно, для удобства анализа сети желательно иметь конечное дерево, но любое ограничение дерева приводит к потере информации. Поэтому надо найти такое ограничение, при котором потери информации были бы минимальными.

Для неограниченных сетей вводят специальный символ ω, означающий, что в позиции точек может быть сколько угодно много. При этом справедливы такие соотношения:

$$\omega + a = \omega;$$
$$\omega - a = \omega,$$

где a – некоторое целое положительное число.

Итак, число точек в позиции может быть либо конкретное неотрицательное целое, либо ω.

Будем использовать при построении дерева достижимости следующее правило. Пусть граничная вершина μ не является терминальной или дублирующей. Для каждого разрешенного перехода t_j в маркировке μ построим дугу, исходящую из μ. Дугу будем помечать переходом t_j. Маркировка μ' новой вершины определится следующим образом:

1. Если $\mu(p_i) = \omega$, то $\mu'(p_i) = \omega$;
2. Если на пути от корневой вершины к μ' существует вершина μ'', такая что в μ' число точек в каждой позиции не меньше, чем в μ'', а в позиции p_i строго больше (то есть $\mu'(p_i) > \mu''(p_i)$), то $\mu'(p_i) = \omega$.

   В противном случае $\mu'(p_i)$ – число точек в позиции p_i, получающееся после запуска t_j из μ.

Построим дерево для сети, изображенной на рис. 7.20*а*. Начальная маркировка – корень дерева (рис. 7.20*б*). Может сработать t_1 и t_2. При срабатывании t_1 получим маркировку, которая во всех позициях не меньше, чем начальная, а в p_2 строго больше. Поэтому на место 2-й компоненты ставится символ ω. Это отражает тот факт, что мы можем повторять запуск перехода t_1 неограниченное число раз и сделать число точек в позиции p_2 сколь угодно большим.

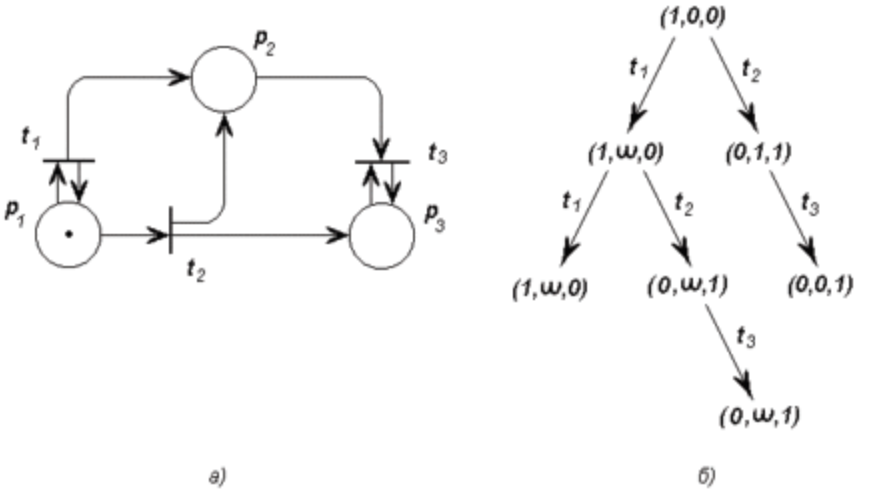


Рис. 7.20. Неограниченная сеть Петри (а)
и соответствующее ей дерево достижимости (б)

При срабатывании t_2 в начальной маркировке получим маркировку (0,1,1). Первый ярус дерева построен. Так как полученные вершины не являются терминальными или дублирующими, переходим к построению следующего яруса. Определяем новые маркировки, которые получаются при срабатывании переходов из полученных маркировок (рис. 7.20*б*).

На основании свойств символа ω и на правилах введения этого символа в маркировку можно утверждать, что дерево достижимости конечно и для неограниченных сетей.

Дерево достижимости – полезный инструмент анализа сетей Петри, позволяет определить свойства сетей.

Безопасность и ограниченность можно проверить с помощью дерева достижимости, просмотрев все его вершины. Если компоненты маркировок не превышают 1, то сеть безопасна. Если ω отсутствует в дереве, то сеть ограничена, и простым перебором можно определить саму границу k.

Символ ω в дереве означает, что сеть не ограничена. Кроме того, положение символа ω показывает, какие именно позиции не ограничены. Для неограниченных сетей по дереву достижимости можно определить границы для тех позиций, которые являются ограниченными.

Свойство сохранения также эффективно проверяется с помощью дерева достижимости. Для каждой вершины составляется уравнение вида

$$\sum_{i=1}^{n} w_i \mu_j(p_i) = S,$$

где $j = 1,2,\ldots,k$ вершины дерева достижимости;

$i = 1,2,\ldots,n$ позиции сети Петри.

Получаем систему k линейных уравнений с $n+1$ неизвестными. Этими неизвестными являются веса w_i позиций и взвешенная сумма S, которая должна быть постоянна для всех достижимых маркировок. Решение такой системы – посильная задача. Если решение существует и решение представляет целые положительные и не равные нулю числа, то есть сохраняющая по отношению к вектору взвешивания.

Если в дереве присутствует символ ω, то соответствующая сеть неограниченна, и сохраняющей быть не может. В этом случае можно считать веса неограниченных позиций равными нулю и говорить лишь о сохранении оставшихся ограниченных позиций.

Задача покрываемости: (для сети с данной маркировкой μ' определить, достижима ли маркировка $\mu'' \geq \mu'$?) решается проверкой дерева достижимости. Ищем любую вершину с маркировкой большей μ'. Если такой вершины нет, маркировка μ' не покрывается никакой достижимой маркировкой.

Путь от корня к покрывающей маркировке определяет последовательность переходов, которые приводят из начальной маркировки к покрывающей маркировке. Если покрывающая маркировка содержит символ ω, означающий, что в позиции может быть сколь угодно много точек, то в пути от корня к покрывающей маркировке имеется цикл. Для увеличения компоненты ω необходимо повторить этот цикл. Если покрывающая маркировка содержит несколько символов ω, то для увеличения соответствующих компонент необходимо выполнить несколько циклов. При этом может оказаться, что выполнение одного цикла увеличивает число точек в одной неограниченной позиции и уменьшает в другой. Например, в сети на рис. 7.21a для получения маркировки, покрывающей маркировку (0,3,1,5), нужно сначала 8 раз запустить переход t_1, затем t_2 и после этого 5 раз переход t_3.

Рассмотрим недостатки дерева достижимости, которые ограничивают его применение для анализа сетей.

Дерево достижимости в случае неограниченных сетей нельзя использовать для решения задач достижимости, покрываемости, активности и последовательности запусков.

Ограничивает решение этих задач символ ω. Символ ω указывает не конкретное количество точек в позиции, а только возможность существования сколь угодно большого их числа.

Рассмотрим еще раз сеть и ее дерево достижимости, представленные на рис. 7.21. Это же дерево достижимости будет иметь такая же сеть, как и на рисунке, но содержащая вместо одной две дуги, ведущие из перехода t_1 в позицию p_2. Маркировка (1,1,0,0), достижимая в исходной сети, оказывается недостижимой в сети с двумя дугами $t_1 p_2$. Дерево достижимости, одинаковое для этих разных сетей, не позволяет выявить данный факт.

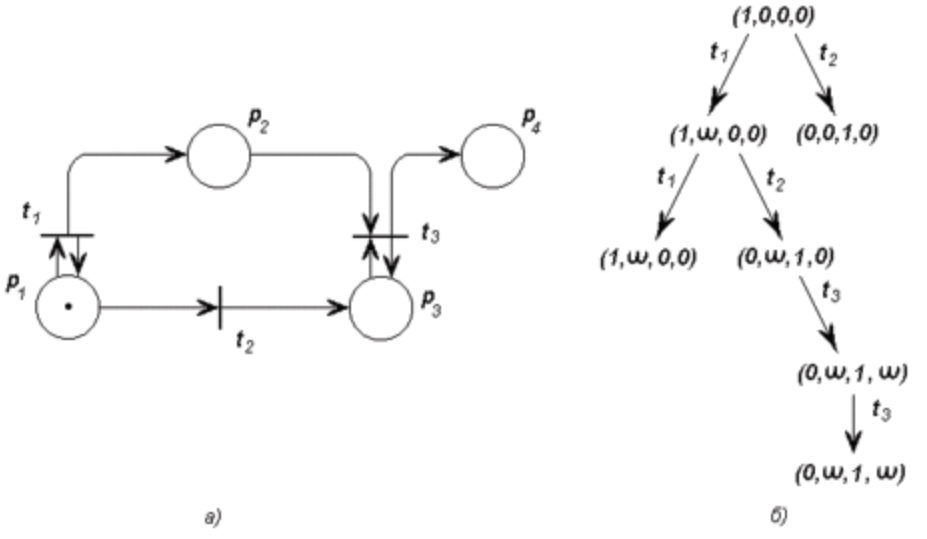


Рис. 7.21. Пример неограниченной сети Петри (а) и её дерева достижимости (б)

Задача покрываемости для неограниченных сетей также не может быть решена с помощью дерева достижимости.

Тот факт, что под символом ω может скрываться любое значение (в том числе и 0), делает невозможным и решение задачи активности. Так в сети на рис. после срабатывания последовательности переходов t_1, t_2, t_3 тупик и дальнейшее срабатывание перехода t_3 невозможно. Однако, судя по дереву достижимости, тупика в сети нет. Символ ω не позволяет обнаружить тупик.

Матричный метод

Второй подход к анализу сетей Петри основан на матричном представлении сетей Петри. Введем две матрицы: входную D^- и выходную D^+, которые представляют входную и выходную функции. Каждая матрица имеет m строк (по одной на переход) и n столбцов (по одному на позицию). Определим элементы матриц.

1. $D^-[j,i]=\#(p_i, I(t_j))$ – определяет входы в переходы (число появлений позиции p_i во входном комплекте перехода t_j).
2. $D^+[j,i]=\#(p_i, O(t_j))$ – определяет выходы из перехода (число появлений позиции p_i в выходном комплекте перехода t_j).

Определение сети Петри в матричной форме: сеть Петри

$$C = (P, T, D^-, D^+).$$

Пусть $e[j]$ – вектор-строка, компоненты которого соответствуют переходам и все равны нулю за исключением j-й компоненты, которая соответствует j-му переходу и равна 1. Переход t_j представляется m-вектором $e[j]$. Тогда переход t_j в маркировке μ разрешен, если $\mu \geq e[j] \cdot D^-$ (μ рассматривается как вектор). Результат запуска перехода t_j в маркировке μ записывается как

$$\delta(\mu, t_j) = \mu' = \mu - e[j] \cdot D^- + e[j] \cdot D^+ = \mu + e[j] \cdot (D^+ - D^-) = \mu + e[j] \cdot D,$$

где $D = D^+ - D^-$ – составная матрица изменений.

Для последовательности запусков переходов $\sigma = t_{j_1}, t_{j_2}, ..., t_{j_k}$ имеем:

$$\delta(\mu, \sigma) = \delta(\mu, t_{j_1}, t_{j_2}, ..., t_{j_k}) = \mu' = \mu + e[j_1] \cdot D + e[j_2] \cdot D + ... + e[j_k] \cdot D =$$
$$= \mu + (e[j_1] + e[j_2] + ... + e[j_k]) \cdot D = \mu + f(\sigma) \cdot D,$$

где $f(\sigma) = e[j_1] + e[j_2] + ... + e[j_k]$ – вектор запусков последовательности $\sigma = t_{j_1}, t_{j_2}, ..., t_{j_k}$.

Вектор запусков – это вектор с неотрицательными целыми компонентами, определяющими количества срабатываний каждого перехода. Например, последовательность $\sigma = t_3, t_2, t_3, t_2, t_1$ представляется вектором запусков $f(\sigma) = (1,2,2)$, компоненты которого означают, что первый переход сработал один раз, второй – два раза и третий тоже два раза.

Рассмотрим удобство такого матричного представления сетей Петри.

Свойство сохранения. Для того чтобы показать сохранение, необходимо найти вектор взвешивания (ненулевой), для которого взвешенная сумма по всем достижимым маркировкам постоянна.

Пусть вектор взвешивания W – вектор-столбец размерности $n \times 1$. Тогда, если μ – начальная маркировка, а μ' – любая достижимая маркировка, необходимо, чтобы $\mu W = \mu' W$. Так как μ' – достижима, то существует последовательность запусков переходов σ, которая переводит сеть из μ в μ'. Поэтому можно записать

$$\mu' = \delta(\mu, \sigma) = \mu + f(\sigma) D.$$

Следовательно, $\mu W = \mu' W = (\mu + f(\sigma) D) W = \mu W + f(\sigma) DW$.

Итак, $\mu W = \mu W + f(\sigma) DW$, поэтому $f(\sigma) DW = 0$.

Поскольку последнее соотношение должно быть справедливо для всех $f(\sigma)$, значит $DW = 0$. Таким образом, сеть Петри является сохраняющей тогда и только тогда, когда существует такой вектор W с целыми и строго большими нуля компонентами, что $DW = 0$.

Вот простой способ проверки сохранения, а также это уравнение позволяет найти и сам вектор взвешивания.

Достижимость. С помощью матричного представления сети Петри можно решать проблему достижимости. Предположим, что маркировка μ' достижима из маркировки μ. Тогда существует последовательность запусков переходов σ, которая переведет сеть из μ в μ'. Это означает, что должно существовать целое неотрицательное решение уравнения

$$\mu' = \mu + xD,$$

где $x = f(\sigma)$.

Итак, если μ' достижима из μ, то необходимо, чтобы уравнение имело решение в неотрицательных целых. Если μ' недостижима из μ – матричное уравнение либо не имеет решений, либо имеет решения, но решения отрицательные или дробные.

Рассмотрим пример. Пусть дана маркированная сеть Петри с начальной маркировкой (1,0,1,0) (рис. 7.22). Покажем возможности матричного метода анализа.

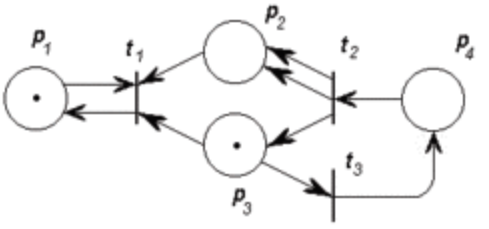


Рис. 7.22. Сеть Петри, анализируемая матричным методом

Запишем входную D^- и выходную D^+ матрицы:

$$D^- = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \qquad D^+ = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Матрица изменений $D = D^+ - D^- = \begin{bmatrix} 0 & -1 & -1 & 0 \\ 0 & +2 & +1 & -1 \\ 0 & 0 & -1 & +1 \end{bmatrix}.$

В начальной маркировке $\mu = (1,0,1,0)$ переход t_3 разрешен и его запуск приводит к маркировке μ'.

$$\mu' = (1,0,1,0) + (0,0,1)\cdot\begin{bmatrix} 0 & -1 & -1 & 0 \\ 0 & +2 & +1 & -1 \\ 0 & 0 & -1 & +1 \end{bmatrix} = (1,0,1,0) + (0,0,-1,+1) = (1,0,0,1).$$

Последовательность $\sigma = t_3, t_2, t_3, t_2, t_1$, представляется вектором запусков $f(\sigma) = (1,2,2)$ и получается маркировка:

$$\mu' = (1,0,1,0) + (1,2,2)\cdot\begin{bmatrix} 0 & -1 & -1 & 0 \\ 0 & +2 & +1 & -1 \\ 0 & 0 & -1 & +1 \end{bmatrix} = (1,0,1,0) + (0,3,-1,0) = (1,3,0,0).$$

Определим: являются ли достижимыми маркировки (1,8,0,1), (1,7,0,1)?

Для определения достижимости маркировки (1,8,0,1) из маркировки (1,0,1,0), имеем уравнение:

$$(1,8,0,1) = (1,0,1,0) + x\cdot\begin{bmatrix} 0 & -1 & -1 & 0 \\ 0 & +2 & +1 & -1 \\ 0 & 0 & -1 & +1 \end{bmatrix}$$

или

$$(0,8,-1,1) = (x_1, x_2, x_3)\cdot\begin{bmatrix} 0 & -1 & -1 & 0 \\ 0 & +2 & +1 & -1 \\ 0 & 0 & -1 & +1 \end{bmatrix}.$$

Получим систему линейных уравнений

$$\begin{cases} -x_1 + 2x_2 = 8; \\ -x_1 + x_2 - x_3 = -1; \\ -x_2 + x_3 = 1, \end{cases}$$

решение которой $x_1 = 0$; $x_2 = 4$; $x_3 = 5$ представляет целые положительные числа.

Таким образом, найден вектор запусков $f(\sigma) = (0,4,5)$.

К маркировке (1,8,0,1) приводит последовательность

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

Вектор запусков определяет не саму последовательность, а только количества срабатываний каждого перехода.

Достижима ли маркировка (1,7,0,1)?

Матричное уравнение:

$$(1,7,0,1) = (1,0,1,0) + x \cdot \begin{bmatrix} 0 & -1 & -1 & 0 \\ 0 & +2 & +1 & -1 \\ 0 & 0 & -1 & +1 \end{bmatrix}$$

или

$$(0,7,-1,1) = x \cdot \begin{bmatrix} 0 & -1 & -1 & 0 \\ 0 & +2 & +1 & -1 \\ 0 & 0 & -1 & +1 \end{bmatrix},$$

где $x = (x_1, x_2, x_3)$, не имеет решения в неотрицательных целых. Маркировка (1,7,0,1) недостижима.

Недостатки матричного подхода к анализу сетей Петри:

1. Матрица D не полностью отражает структуру сети Петри. Переходы, имеющие одни и те же входные и выходные позиции (рис. 7.23), представляются соответствующими элементами матриц D^+ и D^-, которые затем взаимно уничтожаются в матрице $D = D^+ - D^-$ (позиция p_1 и переход t_1 в рассмотренном примере на рис. 23).

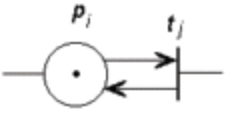


Рис. 7.23. Переход, имеющий одну и ту же входную позицию

2. Отсутствие информации о последовательности в векторе запуска. Например, вектор запуска $f(\sigma) = (1,1,0,2,0,1)$ может иметь последовательность переходов t_1, t_2, t_4, t_4, t_6, а может и t_1, t_4, t_2, t_4, t_6. Хотя и известно число запусков переходов, порядок их запусков неизвестен.
3. Решение матричного уравнения $\mu' = \mu + xD$ является необходимым условием для достижимости, но недостаточным.

Рассмотрим простую сеть Петри (рис. 7.24).

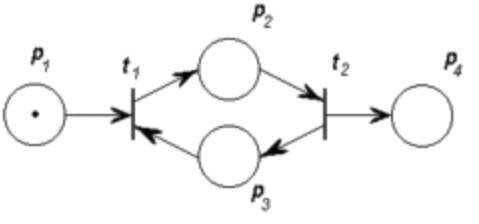


Рис. 7.24. Пример недостаточности решения матричного уравнения для определения достижимости маркировки

Если мы хотим определить, является ли маркировка (0,0,0,1) достижимой из (1,0,0,0), необходимо решить уравнение

$$(0,0,0,1) = (1,0,0,0) + f(\sigma) \cdot \begin{bmatrix} -1 & +1 & -1 & 0 \\ 0 & -1 & +1 & +1 \end{bmatrix}$$

и найти вектор запусков $f(\sigma) = (x_1, x_2)$.

Решение этого уравнения сводится к решению системы линейных уравнений:

$$\begin{cases} -x_1 = -1; \\ x_1 - x_2 = 0; \\ -x_1 + x_2 = 0; \\ x_2 = 1. \end{cases}$$

Получаем

$$x_1 = 1;$$
$$x_2 = 1.$$

Решение $f(\sigma) = (1,1)$ соответствует двум последовательностям: $\sigma_1 = t_1, t_2$ и $\sigma_2 = t_2, t_1$. Но ни одна из этих двух последовательностей переходов невозможна, поскольку в (1,0,0,0) ни t_1, ни t_2 не разрешены. Решение уравнения недостаточно для доказательства достижимости.

7.12. Сложные (иерархические) сети Петри

Сложные сети Петри – средство описания иерархии дискретных процессов.

Сложная сеть Петри – частично упорядоченное множество $C = \{C_0, C_1, ..., C_n\}$ сетей Петри, удовлетворяющее следующим условиям.

1. Все сети C_i, где $i > 0$, обязательно имеют особые переходы: переходы источники, в которые не ведет ни одной дуги и переходы стоки, из которых не исходит ни одной дуги. Сети с такими переходами называют блоками.

2. Отношения между сетями графически представляется деревом с вершинами-прямоугольниками, в которые помещаются компоненты C_i $(i = 0,1,...,n)$. Корню дерева соответствует старшая сеть C_0.

3. Для любых двух сетей C_i и C_j, соответствующих входной и выходной вершине относительно дуги дерева, в сети C_i присутствует особая позиция, называемая дублером блока C_j.

4. Между входными (выходными) переходами дублера и источниками (стоками) дублируемого блока C_j существует взаимно-однозначное соответствие. При этом каждый входной (выходной) переход и соответствующий ему источник (сток) срабатывают одновременно. Множество переходов, которые срабатывают только одновременно, называют связкой переходов.

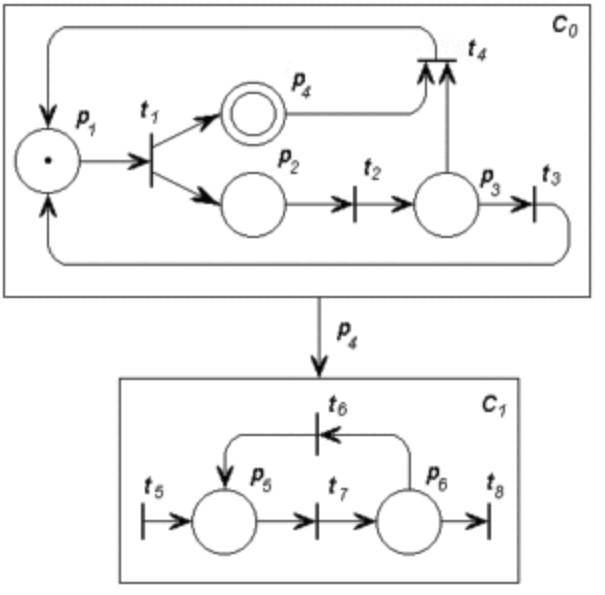
5. Начальные маркировки сетей, образующих множество C, согласованы таким образом, что точка в дублере находится тогда и только тогда, когда не все позиции соответствующего блока пусты.

6. На каждой дуге дерева, соединяющей вершину C_i с C_j, проставляется обозначение дублера из сети C_i, раскрываемого блоком C_j.

7. Позиции-дублеры изображаются двойными кружками.

8. Графическое изображение дополняется списком связок переходов.

Пример сложной сети показан на рис. 7.25.



Связки переходов: $\{t_1, t_5\}, \{t_4, t_8\}$.

Рис. 7.25. Пример сложной (иерархической) сети Петри

В обозначении переходов возможно использование двух индексов, например, t_j^i, где нижний индекс означает номер перехода, верхний – номер сети.

В сложной сети Петри различают два вида маркировки: полную μ_n, учитывающую размещение точек во всех позициях, включая дублеры, и основную $\mu_{осн}$, определяемую размещением точек только в простых позициях.

Множество последовательностей полных маркировок, реализуемое сложной сетью, называется ее полным поведением. Если из каждой последовательности этого множества исключить позиции дублеры, то получим множество последовательностей, называемое основным поведением сложной сети.

Дублер пустой тогда и только тогда, когда пусты все позиции дублируемого блока.

Для дискретных процессов сложная сеть Петри приобретает вполне определенный смысл: каждому дублеру соответствует сложная операция, т.е. несколько операций. Эта сложная операция сама является дискретным процессом, описание которой задается дублируемым блоком. Если в дублируемом блоке содержится свой дублер, то ему также соответствует сложная операция, но уже более низкого уровня сложности. Соотношение операций по сложности определяется деревом отношений.

Уровень сложности операций тем выше, чем ближе к корню дерева находится сеть, содержащая дублер, соответствующий сложной операции. Таким образом, сложная сеть отражает иерархию операций в дискретных процессах. Самая старшая сеть C_0 (корень дерева) – это сеть наименее детализированная, то есть самая общая.

В сложных сетях Петри применяют операцию замещения дублера и операцию замещения локализованного узла.

Операция замещения дублера p_i в сети C_i блоком C_j состоит в объединении C_i и C_j путем замены каждой связки одним переходом и последующим удалением позиции дублера p_i вместе с входящими и выходящими из нее дугами.

Пример. Выполнить операцию замещения дублера в сети, изображенной на рис. 7.25. Результат замещения в сети C_0 дублера p_4 блоком C_1 представлен на рис. 7.26.

Для переходов, полученных заменой связок, сохраняется то обозначение, которое имел входной (выходной) переход по отношению к дублеру p_i.

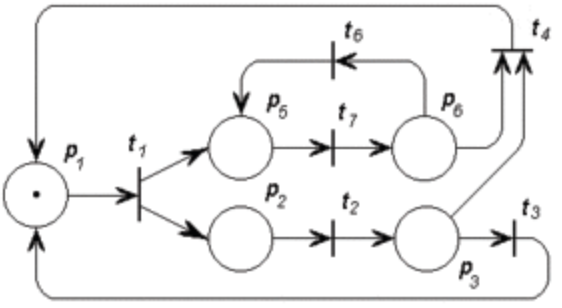


Рис. 7.26. Результат замещения в сети C_0 дублера p_4 блоком C_1

Если исходная сеть содержит несколько дублеров, то в результате однократного применения операции замещения дублера вновь образуется сложная сеть. Путем последовательного замещения дублеров (в произвольном порядке) из исходной сложной сети будет получено семейство сложных сетей и сеть, которая не будет содержать дублеров, то есть простая сеть. Все сети будут иметь одинаковое поведение. Две сложные сети называются эквивалентными, если после проведения в них всех замещений дублеров блоками образуется одна и та же простая сеть.

Операция замещения локализованного узла. Пусть $P = \{p_1,...,p_n\}$, $T = \{t_1,...,t_m\}$ – множество позиций и множество переходов в сети Петри и пусть $P' \subseteq P$ – некоторое заданное подмножество позиций, $T' \subseteq T$ – подмножество всех переходов, инцидентных позициям из P'.

Узлом сети Петри называется фрагмент сети, содержащий все позиции из P', все переходы из T' и все дуги, соединяющие позиции из P' с переходами из T'.

Узел локализованный, если каждый переход из подмножества T' принадлежит одному и только одному типу из трех:

1. В переход ведут дуги только из внешних позиций (принадлежащих $P \setminus P'$), а из перехода – только в позиции из P' (источник).
2. Переход связан только с позициями из P' (внутренний переход узла). Внутренний переход не имеет дуг, связывающих его с внешними позициями.
3. В переход ведут только дуги из P', а из перехода только во внешние позиции (сток узла).

Операция замещения локализованного узла определяется следующей процедурой:

- в сети C_i, содержащей локализованный узел C_j, вводится позиция p_i – дублер блока C_j;

- каждый переход сети C_i, являющийся по отношению к узлу C_j источником (стоком), соединить выходящей (входящей) дугой с дублером;
- в полученной сети каждый переход, связанный с дублером, заменить связкой из двух переходов так, чтобы один из них связывал дублер с позициями, не принадлежащими P', а второй являлся бы источником (стоком) в блоке C_j.

Пример. Заменим локализованный узел, построенный на базе позиций p_1 и p_2, позицией дублером p_4 (рис. 7.27, а).

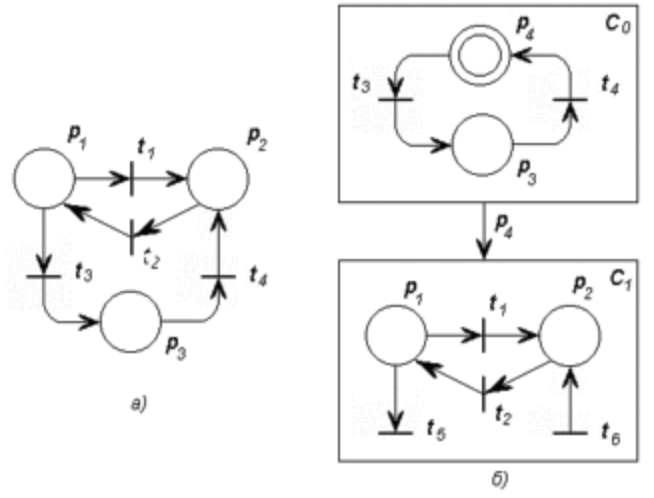


Рис. 7.27. Замена локализованного узла,
построенного на базе позиций p_1 и p_2 (а), позицией дублером p_4 (б)

Результат замещения локализованного узла представлен на рис. 7.27, *б*. Связки переходов $\{t_3, t_5\}$, $\{t_4, t_6\}$. Переходы t_1, t_2 – внутренние переходы узла, t_3, t_4 – переход сток и переход источник соответственно.

Применение операции замещения узла дает возможность преобразовать простую сеть в сложную. Причем, делая такие преобразования, можно получить различные сложные сети, эквивалентные простой сети. Возникает вопрос, какого вида сложную сеть мы хотели бы получить в результате таких преобразований? Естественно, хотелось бы получить такую сложную сеть, которую можно было бы легко анализировать. И мы умели бы это делать. Желательно получить какой-то стандартный, определенный вид сложной сети, чтобы исключить многозначность. Для этого рассмотрим одну из форм представления сложных сетей – стандартную автоматную сеть с параллелизмом: АП-сеть.

7.13. Автоматные сети с параллелизмом (АП-сети)

АП-сеть – такая сложная сеть Петри, в которой каждый блок принадлежит к одному из 2-х типов:

- автоматному (А-блок);
- блоку с параллелизмом (П-блок),

причем дублер каждого А-блока содержится в П-блоке и, наоборот, дублер каждого П-блока содержится в А-блоке либо в старшей сети C_0.

А-блок – блок, содержащий не менее 2-х позиций, такой, что:

- все его переходы имеют не более одной входящей дуги и одной выходящей (такие переходы называются автоматными);
- сеть, представляющая блок, либо сильно связанная, либо становится сильно связанной при добавлении к блоку одной позиции, соединенной с каждым источником (стоком) выходящей (входящей) дугой;
- в начальной маркировке блока имеется не более одной точки.

Пример А-блока показан на рис. 7.28. В начальной маркировке точка содержится в позиции p_1, а все остальные позиции пусты.

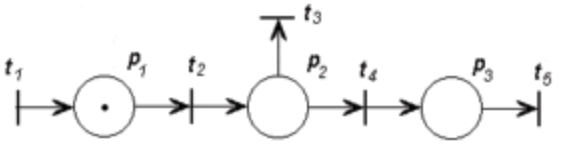


Рис. 7.28. Пример А-блока

П-блок – блок, содержащий не менее двух позиций, такой что:

- все переходы либо источники, либо стоки и других переходов нет;
- каждый источник (сток) связан с каждой позицией блока одной и только одной выходящей (входящей) дугой;
- в начальной маркировке блока либо все позиции пусты, либо в каждой позиции есть по одной точке.

Пример П-блока представлен на рис. 7.29.
Все позиции в начальной маркировке пусты.

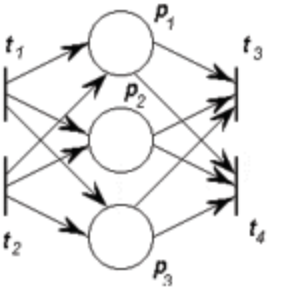


Рис. 7.29. Пример П-блока

Итак, все составляющие АП-сети, (кроме, может быть, старшей составляющей) являются А-блоками или П-блоками и дублер каждого А-блока (П-блока) содержится в П-блоке (А-блоке) либо в старшей компоненте.

Пример АП-сети показан на рис. 7.30, где p_2, p_3 – П-блоки, p_7, p_8 – А-блоки. Связки переходов: $\{t_2, t_5\}$; $\{t_1, t_6\}$; $\{t_4, t_7, t_8, t_{10}, t_{14}\}$; $\{t_3, t_9, t_{13}, t_{16}\}$.

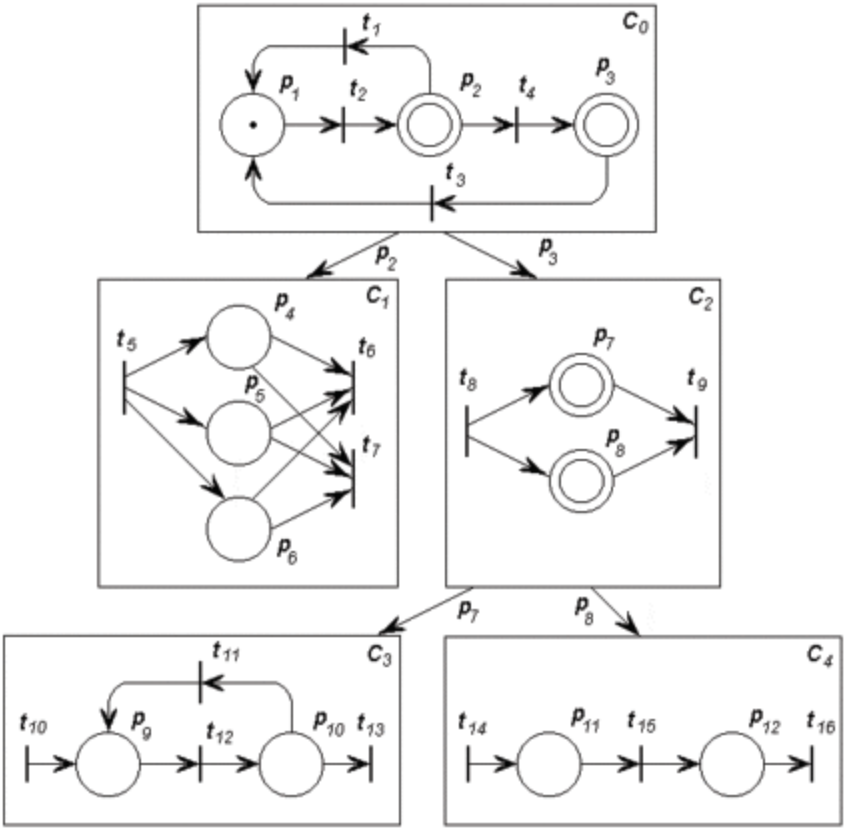


Рис. 7.30. Пример АП-сети

Локализованный узел сети Петри, являющийся А-блоком, будем называть А-узлом, а узел, являющийся П-блоком – П-узлом. Заметим, что всякий П-узел является локализованным, так как все его переходы либо источники, либо стоки и нет переходов, которые могли бы быть связаны входными и выходными дугами как с внутренними, так и с внешними позициями.

Анализ АП-сетей заметно упрощается. Свойства живости и безопасности выявляются по старшей компоненте АП-сети.

АП-сеть C безопасна тогда и только тогда, когда безопасна ее старшая компонента C_0.

АП-сеть C является живой, тогда и только тогда, когда ее старшая компонента C_0 – живая и в C нет тупиковых стоковых наборов относительно П-блоков.

Стоковым набором относительно П-блока C_i называется набор позиций, соединенных со стоками, принадлежавших А-блокам $C_{j_i},...,C_{j_k}$ по одному от каждого А-блока. Стоковый набор относительно П-блока тупиковый, если в нем и во всех достижимых из него наборах, стоки, исходящие из содержащихся в наборе позиций, не входят ни в одну из связок переходов.

Например, АП-сеть, изображенная на рис. 7.30, безопасная. Безопасность следует из безопасности ее старшей компоненты. Кроме того, старшая компонента живая, а стоковый набор (p_{10}, p_{12}) относительно П-блока C_2 не является тупиковым, так как стоки t_{13}, t_{16}, исходящие из позиций p_{10}, p_{12}, содержатся в связке переходов $\{t_3, t_9, t_{13}, t_{16}\}$. Следовательно, вся сеть живая.

При анализе произвольной сложной сети она преобразуется в эквивалентную ей АП-сеть и исследуется на безопасность и живость ее старшая компонента. Исследование в общем случае осуществляется с помощью дерева достижимых маркировок.