Лекция 5. Сети Петри

Сети Петри представляют собой простое и удобное средство для моделирования разнообразных распределенных систем и процессов. Эта модель была придумана немецким ученым Карлом Петри в 1939 году для описания химических процессов. Официальная история сетей Петри началась в 1962 году, когда Карл Петри защитил диссертацию «Kommunikation mit Automaten», в которой им и было введено понятие сети Петри.

0.1 Классические сети Петри

0.1.1 Понятие сети Петри

Сеть Петри представляет собой двудольный ориентированный граф, содержащий вершины двух типов — *места* (обозначаются кружками) и *переходы* (обозначаются прямоугольниками). Любая дуга ведет либо от вершины–места в вершину–переход, либо наоборот. Дуги, соединяющие два места или два перехода, запрещены. Места, у которые нет входящих дуг, называются *входными*. Места, у которых нет исходящих дуг, называются *выходными*.

Каждое место сети Петри может содержать ноль или более *меток* (маркеров, англ. tokens). Все метки считаются одинаковыми и неотличимыми друг от друга. Распределение меток по местам сети называется ее *разметкой*. Работа сети начинается с начальной разметки.

Метки могут переноситься с одного места на другое. Перенос меток выполняется по следующей схеме.

- Переход является *активным*, если каждое его входное место содержит по крайней мере одну метку (более точно — по одной метке на каждую входящую в этот переход дугу).
- Активный переход может *сработать*, при срабатывании переход поглощает по одной метке с каждого своего входного места и размещает по одной метке на каждое свое выходное место (по одной метке на каждую исходящую дугу).
- В каждый момент времени для срабатывания из всех активных переходов недетерминированным образом выбирается один. Если активных переходов нет, то работа сети на этом завершается.

Припишем каждому переходу сети Петри некоторый уникальный символ (например, пронумеруем их). Последовательность символов σ, в которой i-ый символ равен символу перехода, сработавшему на i-ом шаге

работы сети, называется *последовательностью срабатываний* сети Петри. Последовательность срабатываний однозначно определяет последовательность разметок μ_i, где μ_0 является начальной разметкой. Тот факт, что после срабатывания t-го перехода разметка μ преобразуется в разметку μ', будем обозначать кратко $\mu \overset{t}{\longmapsto} \mu'$.

На рисунке 0.1 показан пример работы сети Петри для последовательности срабатываний $\sigma = [t_1, t_3]$. Активные переходы в каждом случае помечены звездочкой. Заметим, что для той же сети имеется еще только одна возможная последовательность срабатываний $\sigma = [t_1, t_2]$.

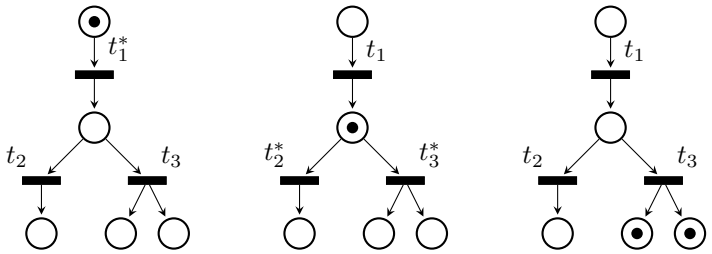

Рис. 0.1 Пример работы сети Петри для $\sigma = [t_1, t_3]$

На рисунке 0.2 приведен пример моделирования сетью Петри простого светофора, в котором цвета переключаются в следующем порядке (*зеленый, желтый, красный*), причем продолжительность каждого сигнала одинакова (и равна одному такту работы сети Петри). Особенностью этой сети является то, что ее работа никогда не завершается.

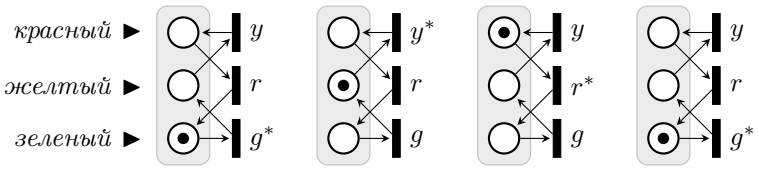

Рис. 0.2 Моделирование работы простого светофора

0.1.2 Формальное определение

Формальным образом сеть Петри определяется как четверка

$$\langle S, T, W, \mu_0 \rangle, \text{ где}$$

- S — конечное множество мест ($|S| = n$);
- T — конечное множество переходов ($|T| = m$, $S \cap T = \emptyset$);
- $W : (S \times T) \cup (T \times S) \to Z_+$ — мультимножество дуг[1];
- $\mu_0 : S \to Z_+$ — начальная разметка сети.

[1] Символом Z_+ будем обозначать множество неотрицательных целых чисел

Сеть Петри можно задать и в компактной векторно-матричной форме. В этом случае структура сети Петри (т. е. ее граф) описывается двумя матрицами W^+ и W^- размера $n \times m$, определяющими наборы дуг, ведущих от мест к переходам (матрица W^-) и обратно (W^+): w^-_{ik} = числу дуг, ведущих из i-го места в k-ый переход; w^+_{ik} = числу дуг, ведущих в i-ое место из k-го перехода. Из матриц W^+ и W^- составляется еще одна матрица $W = W^+ - W^-$, которая обычно используется для вычисления нового состояния (разметки) сети после применения к ней заданной последовательности срабатываний. Начальная разметка сети задается целочисленным вектором μ_0 длины n.

Например, для сети, показанной на рисунке 0.3, матрицы W^+, W^- и W равны

$$W^- = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}, \quad W^+ = \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad W = \begin{bmatrix} -1 & 1 \\ 1 & -1 \\ 1 & -1 \\ 0 & 1 \end{bmatrix},$$

а ее разметка (в левой части рисунка) определяется вектором $\mu = [1, 0, 2, 1]$.

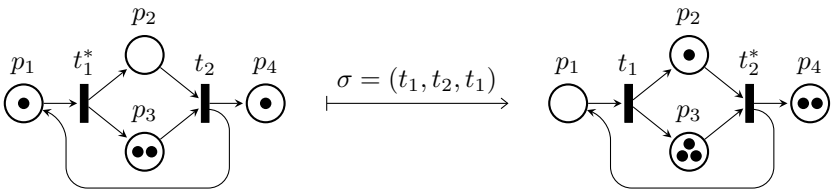

Рис. 0.3 Пример применения последовательности σ к сети Петри

Нетрудно убедиться, что для разметки μ переход $t_k \in T$ является активным, если выполняется условие

$$w^-_k \leq \mu.$$

В последней формуле символ w^-_k обозначает k-ый столбец матрицы W^-, а сравнение двух векторов выполняется поэлементно (т. е. считается, что $a \leq b$, если каждый элемент в a меньше или равен соответствующего элемента в b: $a_i \leq b_i$ для всех возможных i).

Для сети Петри, показанной в левой части рисунке 0.3, легко проверить, что выполняются следующие неравенства

$$w^-_1 \leq \mu: \ [1,0,0,0] \leq [1,0,2,1] \text{ и } w^-_2 \not\leq \mu: \ [0,1,1,0] \not\leq [1,0,2,1].$$

Значит первый переход в данном случае является активным, а второй — нет.

Пусть для некоторой сети Петри задана корректная последовательность срабатываний σ. Обозначим за $v(\sigma)$ вектор, k-ый элемент которого равен числу вхождений символа t_k в σ. Например, для сети с двумя переходами (рисунок 0.3) и последовательности $\sigma = (t_1, t_2, t_1)$ вектор $v = [2, 1]$. Тогда, легко проверить, что применения последовательности σ к начальной разметке μ приводит к разметке μ', векторное представление которой вычисляется по следующей формуле:

$$\mu' = \mu + Wv(\sigma). \tag{1}$$

Например, применение последовательности $\sigma = (t_1, t_2, t_1)$ к сети, показанной в левой части рисунка 0.3 приведет к разметке, которая может быть вычислена по формуле (1) следующим образом:

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

Видно, что результат такого векторно-матричного вычисления полностью соответствует разметке сети в правой части рисунка 0.3.

0.1.3 Свойства сетей Петри

Отдельные элементы сети Петри (места и переходы) могут обладать различными свойствами, на основе которых сначала определяются свойства самих сетей, а затем строится и их классификация.

Простейшим свойством места является количество меток, которые могут в нем располагаться. Если в любой достижимой разметке число меток в заданном месте будет не более одной (0 или 1), то такое место называется безопасным. Сеть Петри называется *безопасной*, если все ее места безопасны. В безопасных сетях состояние каждого места описывается всего одним битом, поэтому такие сети могут быть легко реализованы аппаратно, используя те или иные виды переключателей (триггеров). Кстати, первоначальный вариант определения сети Петри, данный самим Адамом Петри, как раз подразумевал, что сеть является безопасной.

Однако, для большинства приложений требование безопасности сети является чересчур строгим. Его можно ослабить, разрешив каждому месту хранить некоторое ограниченное число меток. Более строго, место называется k-ограниченным, если в любой достижимой разметке в данном месте будет не более k меток. Очевидно, что 1-ограниченное место является безопасным. Место называется ограниченным, если существует такое k, что это место является k-ограниченным. Наконец, сеть Петри является *k-ограниченной*, если любое его место является k-ограниченным, и просто *ограниченной*, если все его места ограничены. Ограниченные сети также допускают эффективную аппаратную реализацию, в которой каждое место представляется уже счетчиком (например, регистром) некоторой заданной емкости. Неограниченные сети имеют, как правило, только теоретический интерес.

Еще одним свойством сетей Петри, основанным на подсчете числа меток, является свойство консервативности. Сеть называется *консервативной*, если число меток в любой достижимой разметке сохраняется одним и тем же (равным числу меток в начальной разметке). Такая модель используется, например, в тех случаях, когда метки представляют собой некоторые ресурсы системы, которые не уничтожаются и не создаются. Эти ресурсы могут переходить от одной части системы к другой, но их суммарное количество в процессе работы системы не изменяется. Нетрудно показать, что любой переход, который встречается хотя бы в одной достижимой разметке, должен иметь одинаковое число входных и выходных дуг — сколько он выбрал меток, столько он должен их и поставить.

0.2 Параллельные процессы

В стандартной интерпретации сетей Петри каждый переход трактуется как некоторый процесс, который берет свои входные данные из входных мест, обрабатывает их некоторым образом и помещает результат в свои выходные места. Однако, природа эти процессов, в частности, их продолжительность, никак не конкретизируется. Хотя сети Петри обладают необходимым потенциалом для моделирования параллельно происходящих процессов, очевидно, что говорить о параллельности процессов имеет смысл только тогда, когда они имеют некую продолжительность.

0.2.1 Последовательная обработка процессов

В простейшем варианте сеть Петри может рассматриваться как последовательное устройство, работа которого заключается в реализации некоторой последовательности срабатываний. Такая интерпретация имеет смысл в нескольких случаях. Например, если в модели предполагается непрерывное время, а продолжительность срабатывания перехода является нулевой, то вероятность одновременного срабатывания двух переходов может считаться равной нулю. Очевидно, что в этом случае, важным оказывается как раз порядок (последовательность) срабатывания переходов.

Последовательностная интерпретация работы сети Петри применяется также в тех случаях, когда все процессы (переходы) обрабатываются одним и тем же устройством (исполнителем). Примером такого рода системы может служить однопроцессорный компьютер, в котором выполняется сразу несколько различных процессов. Эти процессы логически могут быть не связанными друг с другом (и рассматриваться пользователем как параллельные), однако, реализуются эти процессы в системе строго последовательно один за другим.

0.2.2 Параллельная обработка процессов

Более адектватной во многих случаях являются интерпретации сетей Петри, в которых процессы могут обрабатываться одновременно. В стандартной модели такой сети предполагается, что время срабатывания всех переходов является одинаковым. Для простоты будем считать, что это время равно единице. Также положим, что срабатывание любого перехода производится в целочисленный момент времени. Таким образом, время в модели оказывается дискретным, его отсчет будем вести с нулевого момента (начальная разметка).

Для того, чтобы описать параллельное поведение сети Петри, необходимо определить понятие конфликта. *Конфликтом* в сети Петри называется ситуация, когда сразу несколько активных переходов претендуют на одну метку некоторого места. При последовательном срабатывании переходов конфликты никак не учитываются, однако при параллельной интерпретации требуется некоторый способ их разрешения. Пример конфликта показан на рисунке 0.4, на котором переходы t_1 и t_2 конфликтуют из-за общей метки в месте p.

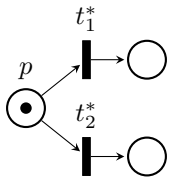

Рис. 0.4 Пример конфликта в сети Петри

Стандартная схема параллельной обработки переходов в сетях Петри выглядит следующим образом: из всех активных в данный момент времени переходов выбирается некоторое их бесконфликтное подмножество (никакие два из выбранных переходов не имеют взаимного конфликта); все эти переходы срабатывают одновременно. Как и выше, если активных переходов нет, то сеть завершает свою работу.

0.2.3 Принцип максимального параллелизма

Недетерминизм сети, заключающийся в выборе переходов для срабатывания, можно существенно понизить, если ввести следующее правило, называемое *принципом максимального параллелизма*: на каждом шаге для срабатывания выбирается такое помножество A' множества A активных переходов, что 1) никакие два перехода из A' не конфликтуют; 2) для любого невыбранного перехода из множества $A \setminus A'$ имеется хотя бы один конфликтующий переход в A'. Другими словами, выбирается такой набор переходов, который не может быть расширен бесконфликтно никакими другими переходами. Будет ли такой набор наибольшим (по числу переходов) не является важным, главное, чтобы он был не расширяемым.

С помощью сетей с максимальным параллелизмом оказывается возможным моделировать системы, в которых необходима синхронизация выполнения отдельных процессов. В качестве примера рассмотрим сеть Петри, моделирующую систему из двух светофоров, работающих согласованно: когда на первом горит красный свет, на втором горит зеленый, и наоборот.

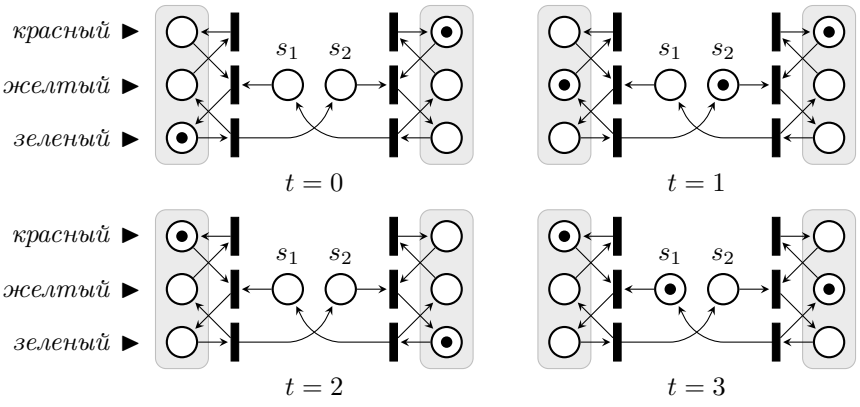

Рис. 0.5 Моделирование согласованного поведения двух светофоров

Для построения такой модели можно взять два светофора, показанных на рисунке 0.2, и связать их с помощью двух дополнительных «безопасных» мест s_1 и s_2 так, как это показано на рисунке 0.5. Т. к. в этой сети

нет конфликтных мест, то в силу принципа максимального параллелизма каждый переход будет срабатывать, как только будут заполнены все его выходные места. В частности, переключение с желтого на красный цвет одного светофора будет происходить одновременно с переключением красного на зеленый другого светофора. На рисунке 0.5 показано первые три такта работы этой сети. На четвертом такте произойдет переход к начальной разметке, т. е. работа вссей системы является периодической с периодом длины 4.

0.2.4 Алгоритмическая универсальность сетей Петри

Известно, что классические сети Петри с полностью недетерминированным поведением не являются полными по Тьюрингу. Интересной особенностью сетей с максимальным параллелизмом является то, что они таким свойством обладают. Чтобы показать алгоритмическую универсальность этого типа сетей (а ниже и некоторых других типов), покажем, как с помощью сети Петри смоделировать работу другой алгоримически универсальной модели — счетчиковой машины Минского.

Напомним, что счетчиковая машина Минского состоит из конечного набора поименнованных регистров (счетчиков), в которых хранятся целые неотрицательные числа. Программа для такой машины состоит из пронумерованной последовательности команд трех типов:

- i : Inc a, k — увеличивает значение регистра a на единицу и переходит к выполнению k-ой команды;
- i : Dec a, k, l — если значение регистра a не равно нулю, то это значение уменьшается на единицу и выполняется переход к k-ой команде, в противном случае (в регистре ноль и это значение не может быть уменьшено) производится переход к l-ой команде;
- i : Halt — команда останова.

Сеть Петри, моделирующая выполнение программы для машины Минского, содержит два типа мест — первые используются для представления регистров и имеют соответствующие этим регистрам имена; вторые однозначно представляют команды и обозначаются соответствующими номерами. Число меток, находящихся в месте a, соответствует числу, хранимому в регистре a. Текущая команда представляется местом с одной меткой, все остальные места–команды меток не имеют.

Команда i : Inc a, k в такой сети представляется с помощью одного перехода t, имеющего одно входное место с номером i и два выходных — с меткой a и номером k (рисунок 0.6). Как только в i-ом месте появится метка, сразу же будет активирован переход t, его срабатывание приведет к добавлению еще одной метки в место a и размещению метки в k-ом месте, что приведет на следующем шаге к активации уже k-ой команды.

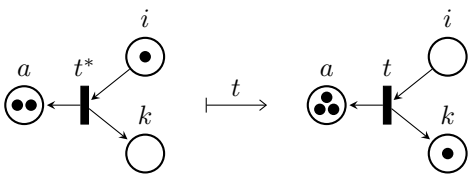

Рис. 0.6 Представление команды i : Inc a, k

Команда условного вычитания i : Dec a, k, l реализуется существенно сложнее (рисунок 0.7).

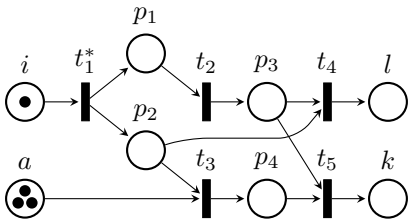

Рис. 0.7 Представление команды i : Dec a, k, l

Если в регистре a находится число $n > 0$, то будет выполнена следующая последовательность переходов $\sigma = (t_1, \{t_2, t_3\}, t_5)$, которая приведет к уменьшению на единицу числа меток в a и появлению одной метки в k-ом месте (вычитание единицы и переход к k-ой команде). Если же регистр a пустой, то последовательность срабатывания переходов будет другой: $\sigma = (t_1, t_2, t_4)$, ее выполнение приведет к появлению метки в l-ом месте (переход к l-ой команде). Далее будет показано, что команда условного вычитания единицы реализуется намного проще, если использовать такие вариации сетей Петри, как сети с приоритетами или ингибиторные сети.

Проще всего устроена реализация команды i : Halt, которая представляется переходом t с одним входным i-ым местом и без выходных мест (рисунок 0.8). Это значит, что активация этой команды приведет к поглощению переходом t метки из места i, после чего выполнение программы будет завершено, т. к. ни одна команда больше не сможет активироваться.

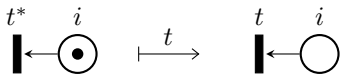

Рис. 0.8 Представление команды i : Halt

0.3 Расширения сетей Петри

В настоящее время имеется большое количество различных вариаций сетей Петри, тем или иным образом расширяющих (а иногда и сужающих) функционал этой модели. Мы рассмотрим несколько наиболее стандартных расширений — сети с приоритетами, ингибиторные, цветные и временные сети Петри.

0.3.1 Сети с приоритетом

Сеть Петри с приоритетами — это стандартная сеть Петри, каждому переходу t которой поставлено в соответствие некоторое число $\Pr t$, называемое *приоритетом* этого перехода. Приоритеты используются для более

полного управления разрешением конфликтных ситуаций. Если два перехода t_1 и t_2 конфликтуют из-за некоторого общего ресурса, то преимущество получит тот, который имеет больший приоритет. В частности, если все переходы данной сети имеют разные приоритеты, то конфликтов в такой сети вообще не будет. На практике, однако, достаточно определить только несколько приоритетов для потенциально конфликтных переходов.

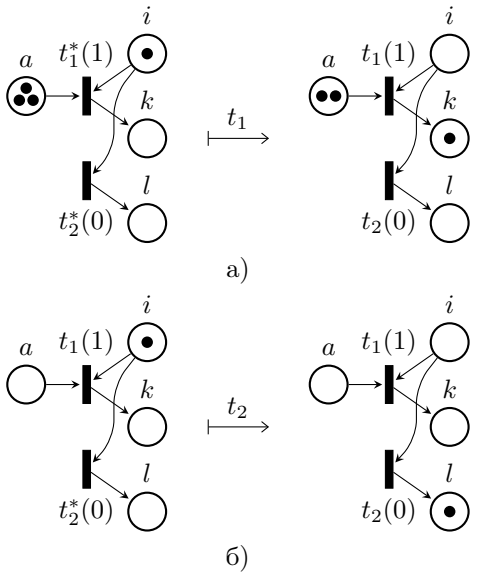

Рис. 0.9 Представление команды i : DEC a, k, l с помощью сети Петри с приоритетом

В качестве примера рассмотрим сеть, реализующую операцию условного вычитания единицы (рисунок 0.9). Сеть содержит два перехода t_1 и t_2 с приоритетами $\Pr t_1 = 1$ и $\Pr t_2 = 0$. Если регистр a содержит хотя бы одну фишку (рис. 0.9а), то активными являются оба перехода, причем они конфликтуют из-за единственной метки в i-ом месте. Т. к. приоритет перехода t_1 больше, то именно он и сработает, что приведет к вычитанию одной фишки из регистра a и активации k-ой команды. Если же регистр a пуст (рисунок 0.9б), то активным будет только переход t_2, срабатывание которого приведет к активации l-ой команды.

0.3.2 Ингибиторные сети

В *ингибиторных сетях Петри* к стандартным дугам, ведущим из мест в переходы, добавляется специальный вид *ингибиторных* (тормозящих) дуг. Такая дуга, при наличии в соответствующем месте хотя бы одной метки, *препятствует* активации соответствующего перехода. Поэтому, такой переход будет активизирован только тогда, когда в данном месте закончатся все метки. Таким образом, тормозящие дуги позволяют явным образом выполнять проверку на отсутствие меток в заданном месте.

На рисунке 0.10 показана реализация операции условного вычитания единицы с помощью одной ингибиторной дуги, которая изображена пунктирной линией.

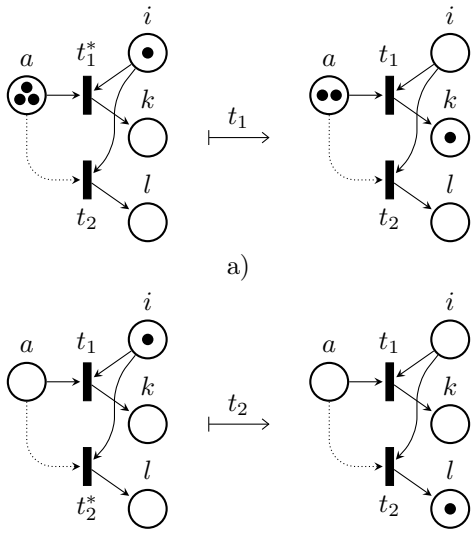

Рис. 0.10 Представление команды i : DEC a, k, l с помощью игибиторной сети Петри

С помощью ингибиторных сетей Петри оказывается возможным реализовать и работу логических элементов. На рисунке 0.11 приведены примеры моделирования четырех базовых логических операций с использованием тормозящих дуг. Предполагается, что пустое место соответствует логическому нулю, а место с одной меткой — логической единице.

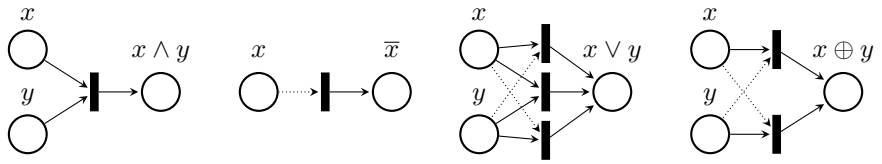

Рис. 0.11 Реализация логических операций с использованием тормозящих дуг

0.3.3 Цветные сети Петри

В традиционных сетях Петри все метки по определению являются одинаковыми, следовательно, неразличимыми. Однако в тех системах, которые моделируются сетями Петри, метки часто представляют собой различные объекты. В *цветных сетях Петри* каждая метка имеет свое значение (цвет), что дает возможность отличать одни метки от других. Значение метки может быть простым, или любого сколь угодно сложного типа.

Переходы в цветных сетях Петри определяют отношения между значениями входных меток и выходных меток. Для этого входным дугам каждого перехода приписываются предусловия, которые определяют, метки с какими значениями поглощаются данным переходом. Выходные дуги перехода определяют (с помощью выражений) значения меток, которые будут помещены в соответствующие места.

По своей выразительности цветные сети Петри уже сопоставимы с таким наглядным средством описания алгоритмов, как блок–схемы. В качестве примера на рисунке 0.12 показана цветная сеть Петри, вычисляющая факториал числа n.

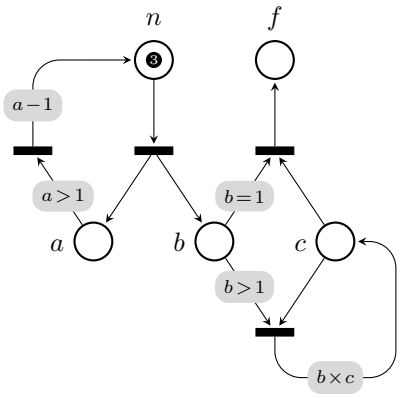

Рис. 0.12 Цветная сеть Петри для вычисления $n!$

0.3.4 Временные сети Петри

Для моделирования систем, в которых отдельные подпроцессы имеют различную продолжительность, можно использовать *временные сети Петри*. В таких сетях каждая метка получает дополнительный атрибут — время задержки. Если метка появилась в каком-то месте в момент времени t и имеет время задержки τ, то она будет доступна для всех переходов (связанных с данным местом) начиная с момента времени $t + \tau$.

Задержки, которые назначаются меткам, приписываются дугам, ведущим от переходов. Соответственно, все метки, поставляемые по какой-либо дуге, будут получать одну и ту же задержку, которая приписана этой дуге. По умолчанию считается, что задержка каждой дуги является единичной.

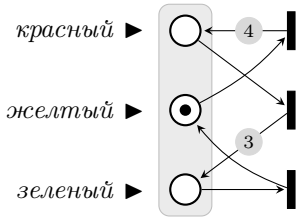

Рис. 0.13 Моделирование работы светофора с помощью временной сети Петри

На рисунке 0.13 приведен пример моделирования с помощью временной сети Петри светофора, у которого продолжительность красного сигнала равна четырем тактам, желтого — одному такту (значение по умолчанию), зеленого — трем тактам.

0.4 Сети Петри и параллельные вычисления

Сети Петри по своему определению обладают встроенным параллелизмом, поэтому они традиционно используются для моделирования разнообразных параллельных процессов, как искусственных (например, вычислительные сети и системы), так и естественных (например, сети химических реакций). Это делает сети Петри с одной стороны адекватным инструментом анализа параллельных процессов, с другой — удобным объектом для реализации на параллельных вычислительных системах.

0.4.1 Моделирование параллельных вычислений

Стандартная интерпретация сетей Петри с точки зрения параллельных вычислительных процессов заключается в том, что метки представляют собой данные, места — память, в которых хранятся данные, а переходы — подпрограммы, которые обрабатывают эти данные. Каждая подпрограмма ожидает, когда будут готовы все ее входные данные, после чего производит необходимые вычисления и помещает результаты на свои выходные места. Если какие-либо два перехода не конфликтуют из-за данных, то при некоторых условиях соответствующие им подпрограммы могут быть выполнены одновременно, т. е. параллельно.

Например, на рисунке 0.14 показана сеть Петри, моделирующая две стандартные операции **fork** и **join** для управления параллельными процессами. Переход *fork* служит для разветвления данного процесса на n независимых процессов $t_1, \ldots, t_n$, переход *join* — для их синхронизации и слияния обратно в один последовательный процесс.

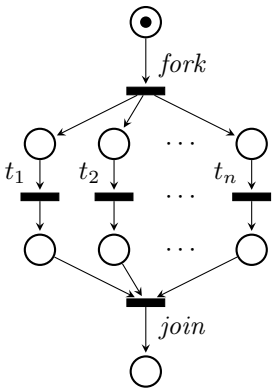

Рис. 0.14 Моделирование операций **fork** и **join** сетью Петри

Так же легко сети Петри позволяют моделировать и конвейерную обработку данных. Простейший пример такой модели показан на рисунке 0.15. Хотя в данном случае каждая метка должна быть обработана тремя переходами, за счет конвейеризации пять меток будут обработаны не за $3 \times 5 = 15$ тактов, а за 7 тактов, т. е. с некоторым ускорением (чем больше данных и длиннее конвейер, тем больше будет ускорение).

Для организации взаимодействия параллельных процессов применяются различные схемы и механизмы синхронизации, критические секции, се-

мафоры, операции обмена и т. д., которые относительно просто и наглядно моделируются с помощью сетей Петри.

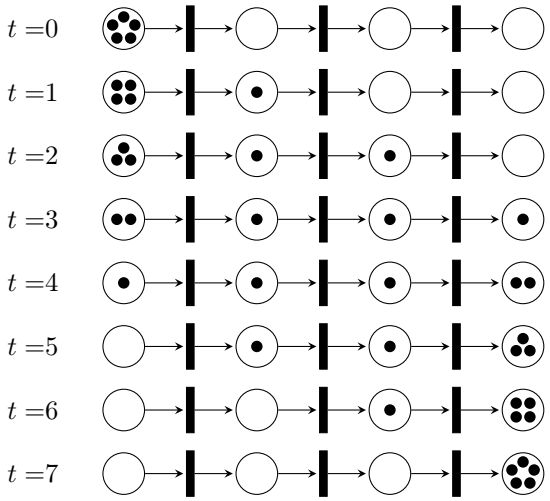

Рис. 0.15 Моделирование конвейерной обработки данных

На рисунке 0.16 показан фрагмент сети Петри, который реализует механизм взаимного исключения, применяемый для обеспечения корректного доступа нескольких процессов к одному разделяемому ресурсу. В качестве такого ресурса может выступать какая-нибудь структура данных (например, файл) или устройство (принтер). Часть процесса, которая используется для доступа и модификации разделяемого объекта, называется *критической секцией*. Выполнение процессом критической секции должно блокировать выполнение другими процессами своих критических секций, связанных с тем же разделяемым объектом.

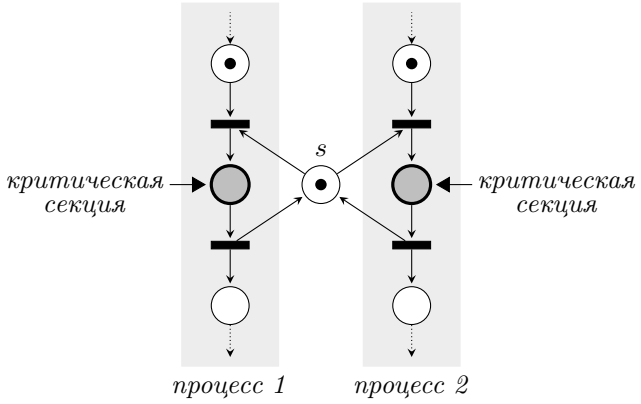

Рис. 0.16 Моделирование процесса взаимного исключения

0.4.2 Параллельная реализация сетей Петри

Реализация сетей Петри на параллельных вычислительных системах в общем случае является всеьма нетривиальной задачей. Если речь идет о

сетях, в которых метки не содержат никакой дополнительной информации (классические сети с различными схемами синхронизации переходов), то можно выделить два основных подхода к организации вычислений.

В первом случае используется матрично-векторные операции над матрицами W, W^+ и W^- и вектором разметки μ. На каждой итерации сначала выполняется активация переходов, эта операция может быть выполнена независимо для всех переходов. Далее неоходимо выбрать подмножество активных переходов, которые сработают в текущий момент времени. После этого формируется вектор σ и производится его умножение на матрицу W — стандартная операция для распараллеливания на любых типах вычислительных параллельных систем. Преимущество матрично–векторного подхода заключается в его концептуальной простоте и в том, что для его реализации используются стандартные инструменты.

Второй способ можно называть объектно–ориентированным — каждый переход и, возможно, каждое место представляются отдельными объектами, которые взаимодействуют тем или иным способом. Все объекты распределяются по некоторой схеме по процессорам вычислительной системы. На каждой итерации переходы запрашивают информацию о метках у своих входных мест, решают между собой возникающие конфликты и поставляют метки на свои выходные места. Как и в предыдущем случае, операции активации и срабатывания могут выполняться одновременно для всех переходов, требуется только обеспечить синхронизацию доступа (для чтения и записи) к местам сети. Большим преимуществом этого подхода является его универсальность — он легко расширяется практически на все типы сетей Петри.

Узким местом при параллельной реализации сетей Петри является разрешение возникающих между переходами конфликтов. В общем случае эту операцию проще всего реализовать, собрав информацию о текущем состоянии сети на одном процессоре и централизованно решить все имеющиеся конфликты. Очевидно, что такая схема очень негативно должна сказываться на общей производительности. В частных случаях можно сделать процесс разрешения конфликтов распределенным. Например, для сетей, в которых каждый переход участвует не более, чем в одном конфликте, разрешение конфликтов может быть решено каждой парой переходов независимо от других переходов. К сожалению, в общем случае конфликты (даже если они парные) могут образовывать весьма сложные цепочки, которые не так просто анализировать распределенным образом.

Литература

[1] S. Achasova, O. Bandman, V. Markova, et al., *Parallel Substitution Algorithm. Theory and Application.*, Singapore: World Scientific, 1994.

[2] L. M. Adleman, *Molecular Computation Of Solutions To Combinatorial Problems*, Science, 266, 11, pp. 1021–1024, 1994.

[3] E. Alba, B. Dorronsoro, *Cellular Genetic Algorithms*, Springer, 2008.

[4] D. Boneh, C. Dunworth, R. J. Lipton, J. Sgall, *On the Computational Power of DNA*, DAMATH: Discrete Applied Mathematics and Combinatorial Operations Research and Computer Science, 71, 1996.

[5] C. Detrain, J. L. Deneubourg, *Self-Organized Structures in a Superorganism: Do Ants Behave Like Molecules?*, Physics of Life Reviews 3, no. 3, pp. 162-187, 2006.

[6] M. Dorigo, M. Birattari, T. Stutzle, *Ant Colony Optimization*, Technical Report No. TR/IRIDIA/2006-023, September 2006.

[7] S. Goss, S. Aron, J.-L. Deneubourg, J. M. Pasteels, *Self-organized shortcuts in the Argentine ant*, Naturwissenschaften, vol. 76, pp. 579–581, 1989.

[8] J. H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, MI, 1975.

[9] J. Kennedy, R. C. Eberhart, *Particle swarm optimization*, Proceedings of IEEE International Conference on Neural Networks, Piscataway, NJ, pp. 1942–1948, 1995.

[10] K. M. Passino, *Biomimicry of bacterial foraging for distributed optimization and control*, IEEE Control Systems Magazine, 22, pp. 52–67, 2002.

[11] Gh. Paun, *Computing with Membranes*, Journal of Computer and System Sciences, 61, 1 (2000), 108-143.

[12] Gh. Paun, *P Systems with Active Membranes: Attacking NP Complete Problems*, CDMTCS Research Report Series, CDMTCS-102, May 1999.

[13] D. T. Pham, A. Ghanbarzadeh, E. Koc, S. Otri, S. Rahim, M. Zaidi, *The Bees Algorithm — A Novel Tool for Complex Optimisation Problems*, Proceedings of IPROMS 2006 Conference, pp. 454–461, 2006.

[14] P. Prusinkiewicz, A. Lindenmayer, *The Algorithmic Beauty of Plants*, Springer-Verlag, 1996 (http://algorithmicbotany.org/papers/abop)

[15] C. W. Reynolds, *Flocks, herds and schools: a distributed behavioral model*, Computer Graphics, 21, 4, pp. 25-34.

[16] G. Rozenberg, A. Salomaa, *The Mathematical Theory of L Systems*, Academic Press, 1980

[17] T. Stutzle, H. Hoos, *MAX–MIN Ant System*, Future Generation Computer Systems, vol. 16, no. 8, pp. 889–914, 2000.

[18] *The Oxford handbook of membrane computing*, edited by G. Paun, G. Rozenberg and A. Salomaa, Oxford University Press, 2010.

[19] T. Weise, *Global Optimization Algorithms – Theory and Application*, http://www.it-weise.de/.

[20] D. Whitley, *A Genetic Algorithm Tutorial*, Statistics and Computing (4): 65-85, 1994.

[21] S. Wolfram, *A New Kind of Science*, Wolfram Media, 2002.

[22] Hsu-Chun Yen, *Introduction to Petri Net Theory*, Recent Advances in Formal Languages and Applications, 2006, pp. 343-373.

[23] М. Гарднер, *Крестики–нолики*, М.: Мир, 1988

[24] В. Котов, *Сети Петри*, М.: Наука, 1984.

[25] А. А. Марков, Н. М. Нагорный, *Теория алгорифмов*, М.: Наука, 1984.

[26] М. Минский, С. Пейперт, *Персептроны*, М.: Мир, 1971.

[27] Дж. фон Нейман, *Теория самовоспроизводящихся автоматов*, М.: Мир, 1971

[28] Г. Паун, Г. Розенберг, А. Саломаа, *ДНК-компьютер. Новая парадигма вычислений*, М.: Мир, 2003.

[29] Дж. Питерсон, *Теория сетей Петри и моделирование систем*, М. : Мир, 1984.

[30] Ф. Розенблатт, *Принципы нейродинамики: Перцептроны и теория механизмов мозга*, М.: Мир, 1965.

[31] Т. Тоффоли, Н. Марголус, *Машины клеточных автоматов*, М.: Мир, 1991

[32] Ф. Уоссермен, *Нейрокомпьютерная техника: Теория и практика*, М.: Мир, 1992.

[33] С. Хайкин, *Нейронные сети: Полный курс*, М.: «Вильямс», 2006.