**О моделирующих возможностях сетей Петри.**

С точки зрения инженерных приложений наибольший интерес представляет анализ динамики изменения маркировок сети Петри и возникающих при этом ситуаций.

Однако важен и вопрос о том, насколько широкий класс объектов могут моделировать сети Петри. В п. 2.1.3. говорилось о свободном языке сети Петри, который представляет собой некоторое подмножество всех слов в алфавите *Т.* Множество свободных языков всех сетей Петри образует класс свободных языков сетей Петри.

В ряде случаев язык сети Петри можно изменить, связав с некоторыми переходами сети *N* определенные символы из алфавита А, и часть переходов оставить непомеченными (вернее, помеченными пустым символом λ*).* В этом случае говорят о **помеченной сети Петри (***PN,* ∑ *)*, где ∑: Т → *А –* помечающая функция, ставящая в соответствие переходам *tj € T* символы *ак € А.* Ясно, что обычная сеть Петри есть частный случай помеченной сети Петри при *Т = A={t1...tn}.*

Помеченную сеть Петри можно рассматривать как генератор слов и изучать ее возможности с точки зрения математической лингвистики.

Рассмотренные в п. 2.1.5. расширения сетей Петри порождают другие классы языков.

Эти классы языков интересно сравнивать с языками, порождаемыми иными типами абстрактных систем, в частности с языками конечных автоматов и машин Тьюринга. Такое

сравнение позволяет характеризовать моделирующие возможности сетей Петри, их способность адекватно описывать системы со сложной динамикой функционирования.

В [8, 9] доказываются следующие утверждения:

1.Класс помеченных сетей Петри строго мощнее класса конечных автоматов и строго менее мощен, чем класс машин Тьюринга.

2.Классы ингибиторных сетей и сетей с приоритетами строго мощнее класса сетей Петри и равномощны классу машин Тьюринга.

3.Класс раскрашенных сетей при конечном количестве цветов равномощен классу сетей Петри.

4. Класс самомодифицируемых сетей эквивалентен классу ингибиторных сетей и сетей с приоритетами.

**2.3. Моделирование дискретных систем**

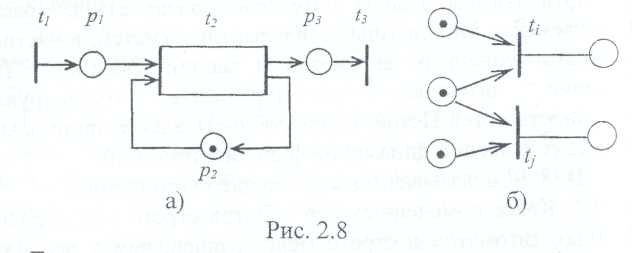
Сети Петри были разработаны и используются для моделирования и исследования сложных систем. С помощью различных модификаций этих сетей можно описать многие системы, в особенности системы с независимыми элементами, например, аппаратное и программное обеспечение ЭВМ, системы телекоммуникаций, физические, химические, социальные идругие системы.

При описании сетей Петри выделяют два понятия: события и условия.

***События*** *-* это действие в системе. В сетях Петри они моделируются переходами.

***Условие*** *-* предикат или логическое описание системы, принимающее значение «истина» или «ложь». Условия моделируются позициями и условиями на дугах. Различаются предусловия и постусловия.

***Предусловие*** *-* это условие до срабатывания перехода, ***постусловие*** *-* соответственно, условие после срабатывания перехода.



Если процесс в системе достаточно сложный, то его подсистемы можно представить в виде ***непримитивных*  *событий.*** Показанный на рисунке 2.8 а составной переход *t2*  непримитивное событие, моделируемое отдельной сетью Петри. При этом процесс моделируется иерархической сетью Петри *(п.* 2.1.5).

Следующая особенность Сети Петри – ***одновременность****.* Если переходы *ti*- и *tj* не влияют друг на друга, то в возможный словарь языка сети Петри входят как слова, начинающиеся с *ti* так и слова*,* начинающиеся с *tj.*

Еще одна ситуация называется ***конфликтом.***

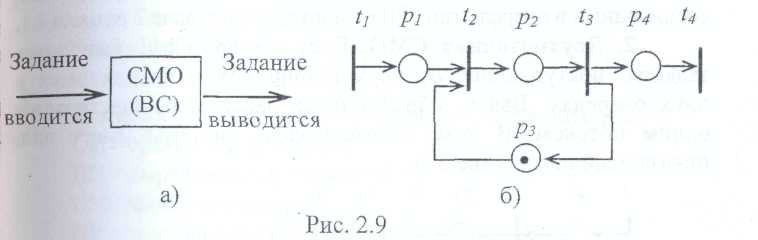
Переходы *t,* и *tj* находятся в конфликте, если запуск одного из них блокирует запуск другого (рис. 2.8 б).

Рассмотрим несколько примеров применения сетей Петри.

**2.3.1. Моделирование вычислительных систем**

**1. Простейшая система массового обслуживания.**

Рассмотрим систему массового обслуживания (например, вычислительную систему), схема которой показана на рисунке 2 9а. Система имеет входной поток заданий, и пока она занята выполнением очередного задания, она не может ввести следующее задание.



Рассмотрим множество условий и событий, характеризующих систему.

Условия:

*P1 -* задание ждет обработки;

*Р2 -* задание обрабатывается;

*Рз -* процессор свободен;

*Р4 -* задание ожидает вывода.

События:

*t1 -* задание помещается во входную очередь;

*t2* - начало выполнения задания;

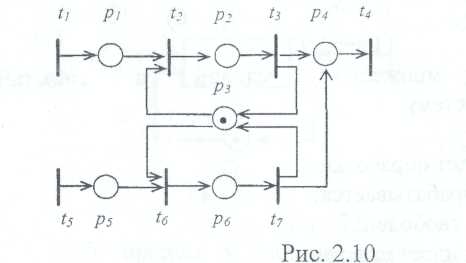
*t3 -* конец выполнения задания;

*t4*- задание выводится.

Сеть Петри, моделирующая рассматриваемую систему, показананарисунке 2.9б.

Поясним работу данной сети. Показанная на рисунке начальная маркировка *М0 = [0,0,1,01* соответствует состоянию, когда система свободна и заявки на обслуживание отсутствуют. При срабатывании перехода *t1* (от внешнего источника) поступает задание и получается маркировка *М,* = *[1,0,1,0].* При этом может сработать переход *t2,* что означает начало обслуживания задания и приводит к маркировке *М2 = [0.1,0,0].* Затем может сработать переход *t3*, что означает окончание обслуживания задания и освобождение системы, т.е. переход к маркировке *М3* = [0,0,1,1]. Переходы *t1* и *t4* могут работать независимо от *t2* и *t3,* моделируя поступление и вывод заданий. Сеть Петри, моделирующая последовательность обслуживающих устройств, соединенных в очередь типа FIFO, приведена в задаче 2 раздела 4.1.

2. **Двухпоточная СМО.** Пусть теперь СМО выполняет задания, поступающие от двух источников и находящиеся в двух очередях. Вывод обработанных заданий осуществляется одним потоком. В этом случае модель системы имеет вид, показанный на рисунке 2.10.



Здесь введены дополнительные условия:

*Р5 -* задание из второй очереди ждет обработки;

*Р6-* задание из второй очереди обрабатывается.

Также введены дополнительные события:

*t5-* задание помещается во вторую очередь;

*t6-* начало выполнения задания из второй очереди;

*t7-* завершение выполнения задания из второй очереди.

Как видно, здесь имеет место конфликт. Одновременно может выполняться только одно задание из любой очереди.

В то же время, если µ*3=2* (это соответствует двухпроцессорной системе), то возможно одновременное выполнение двух заданий из обеих очередей в любой комбинации.

**3. Конвейер.** В качестве следующего примера рассмотрим схему управления асинхронной ЭВМ с конвейерной обработкой.

Поясним работу конвейера на примере операции сложения двух двоичных чисел с плавающей точкой.

*А = ±МА \* 2±Рa,*

*В = ±МВ \* 2±Рb.*

*Здесь* ,

*МA, Мв -* мантиссы чисел *А и В,*

*ра, рb -* двоичные порядки этих чисел.

Требуется получить результат

*С = А + В = ±МС \* 2±Рc*

Как известно, эта операция состоит из следующих этапов:

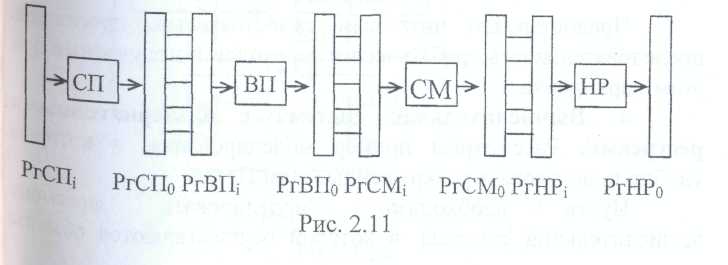
СП - сравнение порядков;

ВП - выравнивание порядков;

СМ - сложение мантисс;

HP - нормализация результата.

Каждый из этих этапов выполняется отдельным функциональным устройством в устройстве конвейерной обработки. Связь между функциональными устройствами и синхронизация их работы осуществляется с помощью пары регистров: входного Рг; и выходного Рг0



Выпишем для i-гo функционального устройства условия:

*рi1* - входной регистр свободен;

*pi2 -* входной регистр заполнен;

*рi3 -* блок занят;

*pi4 -* выходной регистр свободен;

*pi5 -* выходной регистр заполнен;

*pi6 -* пересылка в следующий блок возможна, и события:

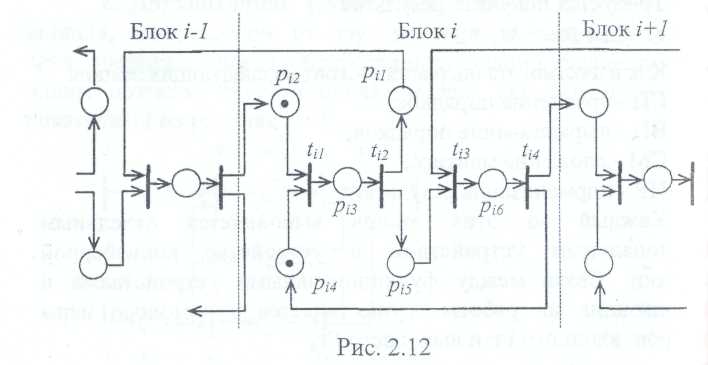
*ti1 -* начало работы *i*-гo блока;

*ti2**-* завершение работы *i*-гo блока;

*ti3 -* начало пересылки в *(i+1)-ыя* блок;

*ti4 -* завершение пересылки в *(i+1)-ый* блок.

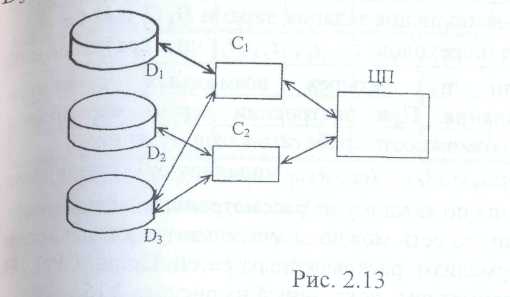
При этом модель *i*-го блока конвейера при начальной маркировке *MOi = [0,l,0,1,0,0]* примет вид, показанный на рисунке 2.12.



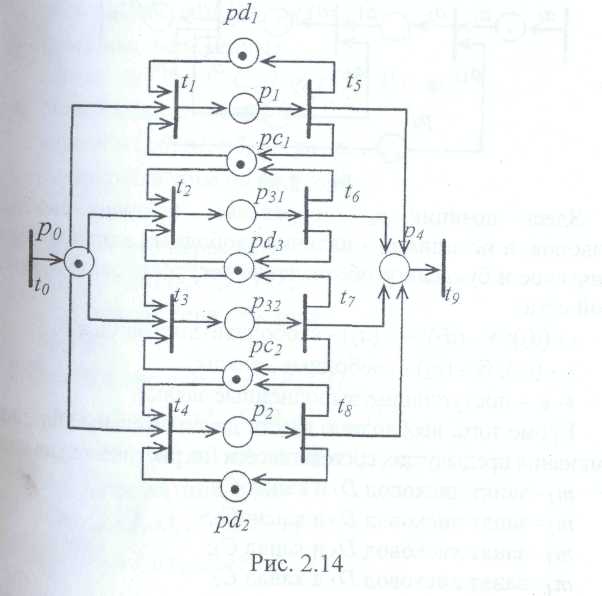
Предоставляем читателям самостоятельно проследит последовательность срабатывания переходов и получаемые приэтом маркировки.

**4. Вычислительная система с альтернативный ресурсами.** Рассмотрим пример моделирования, в которомудобно использовать раскрашенные сети Петри.

Пусть необходимо смоделировать фрагмент вычислительной системы, в которой осуществляются обмены междутремя накопителями на магнитных дисках *D1, D2, D3* и центральным процессором *ЦП* через два канала С*1*, и С2. При этом требуется, чтобы *D1* использовал канал С*1*; *D2 -* канал *С2, D3* - оба канала *С1* и С2 (рисунок 2.13).



Обыкновенная сеть Петри *PN* для моделирования этой системы показана на рисунке 2.14.

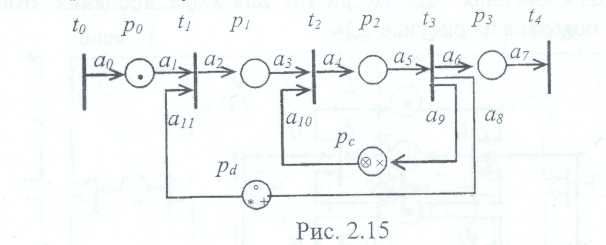


Позиции *pd1 , pd2 , pd3* определяют, свободен или занят, соответствующий дисковод *D1, D2, D3*, позиции *рс1, рс2* соответственно- свободен или занят соответствующий канал *C1* и *С2.* Позиции *р1* и*р2* - выполнение заданий парами *D1C1* и *D2С2* позиции *р31* и *p32* -выполнение задания парами *D3 С1;* и *D3C2.*

Каждый из переходов *t,, t2, t3, t4* при срабатывании определяет один из четырех возможных варианту обслуживания задания. При построении дерева маркировок необходимо дать возможность сработать каждому из них.

Переходы *t5, t6, t7, t8* моделируют завершение выполнения задания по каждому из рассмотренных вариантов.

Рассмотренную сеть можно значительно упростить, еслииспользовать формализм раскрашенных сетей Петри CPN. В этом случае она примет вид, показанный на рисунке 2.15.



Здесь позиция *pd* определяет наличие свободных дисководов, а позиция *рс -* наличие свободных каналов. Введем графическое и буквенное обозначение ресурсов, используемых данной сети:

◦ - (*d1*); \* - *(d2);* + - (*d3*) - свободные дисководы;

× - (*с1*); × - *(с2) -* свободные каналы;

*•- е -* поступившие/выполненные заявки.

Кроме того, необходимо ввести дополнительные фишки длязапоминания предыдущих состояний сети (на рисунке не указаны):

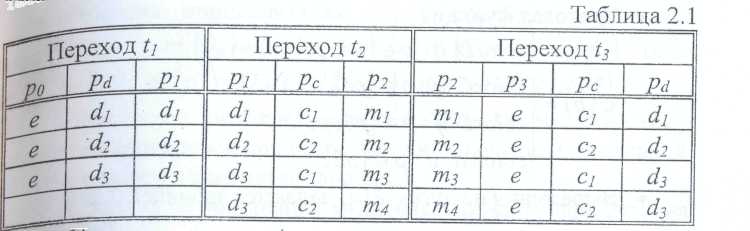
*m1 -* занят дисковод *D1* и канал *С1*;

*m2* - занят дисковод *D2* и канал *C2*;

*m3* - занят дисковод *Ds* и канал *С1*;

*т4 -* занят дисковод *Ds* и канал *С2;*

Правила срабатывания переходов *t1, t2, t3* задаются таблицей 2.1.

******

Приведем теперь формальное описание данной CPN в нотации К. Йенсена (см. п. 2.2).

• Множества цветов:

*type colorD* = *(d1,d.2,d3)*;

*type colorC - (c1, с2)*;

*colorE - (e)*;

*type colorM ={ml,m2,m3,in4).*

• Цветовые переменные:

*Var d:D; Var c:C; Var x:E; Var m* :*M.*

*•* Множество позиций:

*type P = (p0,pl}p2,p3,pd,pc);*

переменная типа позиции:

*Var p:P.*

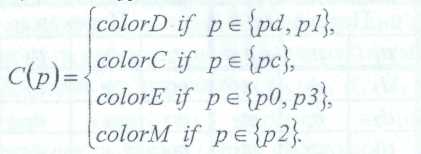
* Множество переходов:  
  *typeT = (t0,t1,t2,t3,t4);*переменная типа перехода:  
  *Var t : T.*
* Множество дуг: *А = AP* U *AT;  
  type AP = (al,a3,a5,a7,a10,a11);  
  type AT = (a0,a2,a4,а6,а8,а9)*;  
  переменные типа дуги:

*var a1,a3ta5,a7,a10,a11: AP;*

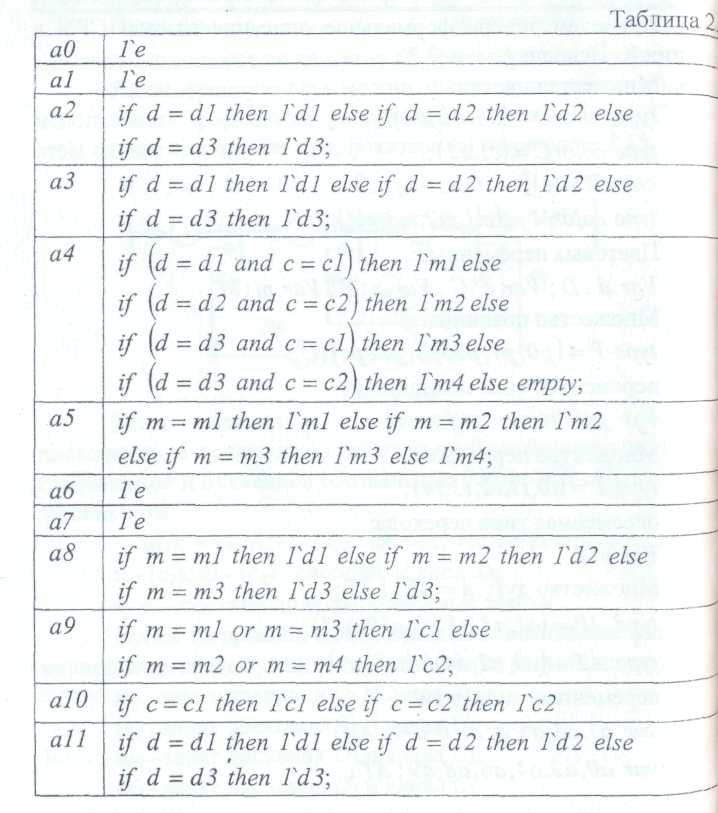
*var a0,a2,a4,a6,a8,a9: AT,*

где типы *АР* и *AT* описаны, соответственно, выражениями (2.14) и (2.15).

• Цветовая функция



• Выражения на дугах *Е(а)* задаются таблицей 2.2.



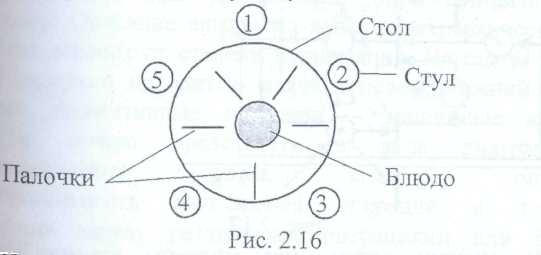
* Блокировочная функция истинна для всех переходов:  
  *G(t)=true.*
* Функция инициализации *I(p),* задающая начальную маркировку *т0(р0)=(1`е); т0(р1) =* (Ø)*; то(р2) =* (Ø); *т0(p3*) = (Ø); *т0(pd*) = *(1`d1,1`d2,1` d3)*; *т0(рс*) = *(1`с1,1`с2).*

Предлагаем читателю самостоятельно проследить схему изменения маркировок при обслуживании поступивших заданий всем четырем возможным вариантам.

**5. Задача об обедающих мудрецах.** Введение параллелизма : полезно только в том случае, когда компоненты процессов могут взаимодействовать при решении задачи. Управление взаимодействующими процессами называют синхронизацией.

Имеется ряд классических задач в этой области: о взаимном исключении, производитель/потребитель, чтения/записи, об обедающих мудрецах. Последнюю рассмотрим подробнее.

Эта задача была предложена Э. Дейкстрой в 1968 году в статье о параллельных вычислениях, где он впервые ввел понятие "семафора". С тex пор она служит своеобразным тестом для методов решения задач распараллеливания.



Имеется *N* китайских мудрецов, которые то гуляют по парку обедают. Каждый из них действует совершенно независимо. Проголодавшись, он идет в столовую, садится на свободный стул за круглый стол, на котором стоит блюдо с рисом берет две палочки и ест. Но палочек всего *N.* Если свободных палочек нет, то мудрец ждет, когда освободятся соседние палочки. Насытившись, он кладет палочки на место уходит. На рисунке 2.16 показана схема столовой для *N = 5 ,*

Обозначим состояния, относящиеся к произвольному i-му мудрецу *(i = 1,...,N):*

*Mi* - i-й мудрец гуляет;

*Ei -* i-й мудрец ест;

*Сk - k-я* палочка свободна;

*Сli - i-й* мудрец держит левую палочку;

*С2i* - i-й мудрец держит правую палочку.

Рассмотрим возможные события:

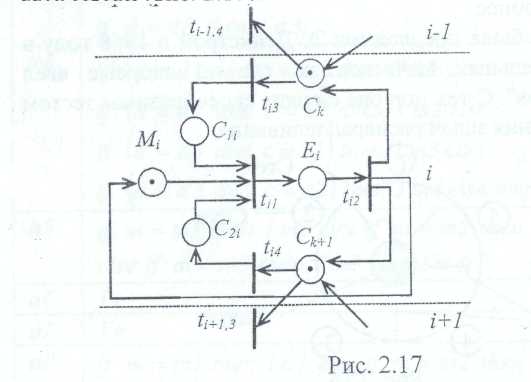
*ti1-* мудрец начинает есть;

*ti2 -* мудрец уходит гулять и освобождает палочки;

*ti3 -* мудрец взял левую палочку;

*ti4 -* мудрец взял правую палочку.

Тогда для отдельного i-го мудреца имеем обыкновенную сеть Петри (рис. 2.17).



Из рисунка видно, что мудрецы взаимно связаны орудиями питания, т.е. из-за ресурса Ck (палочек) имеется конкуренция.

В соответствии с расположением мест за обеденным столом (рис. 2.16) сети Петри для 1-го и N–го мудрецов должны соединяться. При нумерации мест по часовой стрелке правая палочка 1-го мудреца является левой для *N -го.* Таким образом, полный граф *PN* для данного примера представляет собой кольцо,образованное сетями для отдельных мудрецов.

Среди всевозможных маркировок данной сети Петри существуют тупиковые - когда все мудрецы сидят за столом и в руки по одной палочке (например, правой). В этом случае они обречены на голодную смерть, т.к. они никогда не дождутся второй палочки и не смогут начать обед. Этот модельный пример свидетельствует о том, что в реальных параллельно работающих системах должен быть механизм синхронизации, способный разрешить подобные конфликты.