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 *T.* 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 → *A* – помечающая функция, ставящая в соответствие переходам $t_j \in T$ символы $a_к \in A$. Ясно, что обычная сеть Петри есть частный случай помеченной сети Петри при $T = A=\{t_1...t_n\}$.

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 и другие 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.

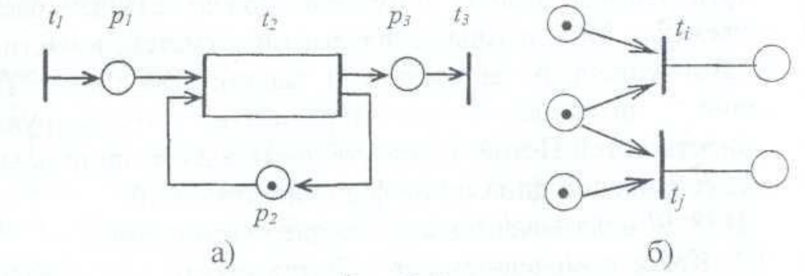

Рис. 2.8

Если процесс в системе достаточно сложный, то его подсистемы можно представить в виде ***непримитивных событий.*** Показанный на рисунке 2.8 а составной переход t_2 непримитивное событие, моделируемое отдельной сетью Петри. При этом процесс моделируется иерархической сетью Петри *(п.* 2.1.5).

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

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

Переходы t, и t_j 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.

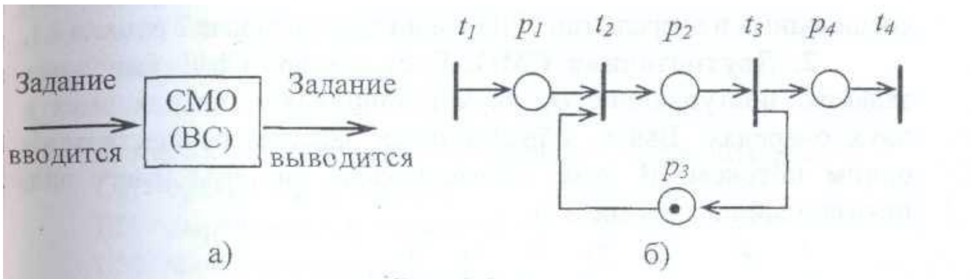

Рис. 2.9

Рассмотрим множество условий и событий, характеризующих систему.

Условия:

P_1 - задание ждет обработки;

P_2 - задание обрабатывается;

P_3 - процессор свободен;

P_4 - задание ожидает вывода.

События:

t_1 - задание помещается во входную очередь;

t_2 - начало выполнения задания;

t_3 - конец выполнения задания;

t_4- задание выводится.

Сеть Петри, моделирующая рассматриваемую систему, показана на рисунке 2.9б.

Поясним работу данной сети. Показанная на рисунке начальная маркировка $M_0 = [0,0,1,01$ соответствует состоянию, когда система свободна и заявки на обслуживание отсутствуют. При срабатывании перехода t_1 (от внешнего источника) поступает задание и получается маркировка $M, = [1,0,1,0]$. При этом может сработать переход t_2, что означает начало обслуживания задания и приводит к маркировке $M_2 = [0.1,0,0]$. Затем может сработать переход t_3, что означает окончание обслуживания задания и освобождение системы, т.е. переход к маркировке $M_3 = [0,0,1,1]$. Переходы t_1 и t_4 могут работать независимо от t_2 и t_3, 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.

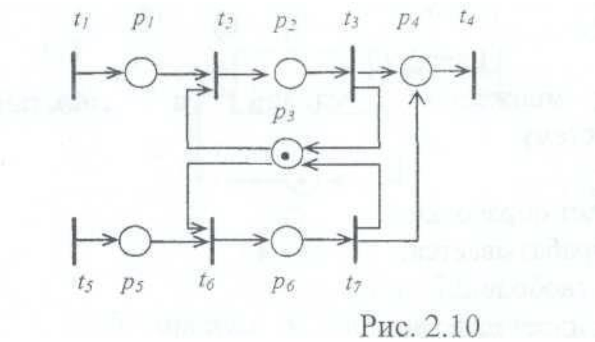

Рис. 2.10

Здесь введены дополнительные условия:

P_5 - задание из второй очереди ждет обработки;

P_6- задание из второй очереди обрабатывается.

Также введены дополнительные события:

t_5- задание помещается во вторую очередь;

t_6- начало выполнения задания из второй очереди;

t_7- 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 $\mu_3=2$ (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.

$A = \pm M_A * 2^{\pm Pa}$,

$B = \pm M_B * 2^{\pm Pb}$.

Здесь ,

M_A, M_B - мантиссы чисел *A и B,*

p_a, p_b - двоичные порядки этих чисел.

Требуется получить результат

$C = A + B = \pm M_C * 2^{\pm Pc}$

Как известно, эта операция состоит из следующих этапов:

СП - сравнение порядков;

ВП - выравнивание порядков;

СМ - сложение мантисс;

НР - 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 $Рг_i$ и выходного $Рг_0$

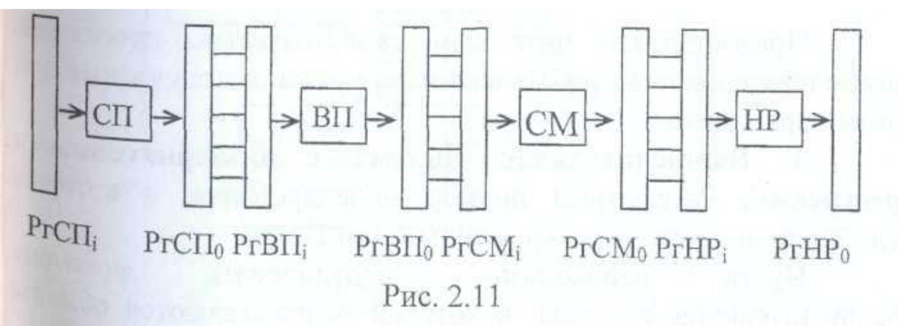

Рис. 2.11

Выпишем для i-го функционального устройства условия:

p_{i1} - входной регистр свободен;

p_{i2} - входной регистр заполнен;

p_{i3} - блок занят;

p_{i4} - выходной регистр свободен;

p_{i5} - выходной регистр заполнен;

p_{i6} - пересылка в следующий блок возможна, и события:

t_{i1} - начало работы i-го блока;

t_{i2} - завершение работы i-го блока;

t_{i3} - начало пересылки в *(i+1)-ыя* блок;

t_{i4} - завершение пересылки в *(i+1)-ый* блок.

При этом модель i-го блока конвейера при начальной маркировке $M_{Oi} = [0,l,0,1,0,0]$ примет вид, показанный на рисунке 2.12.

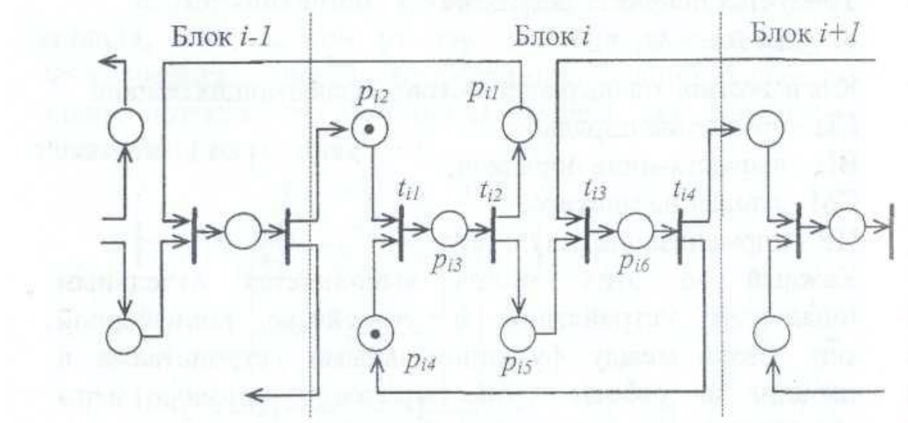

Рис. 2.12

Предоставляем читателям самостоятельно проследит последовательность срабатывания переходов и получаемые при этом маркировки.

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

Пусть необходимо смоделировать фрагмент вычислительной системы, в которой осуществляются обмены между тремя накопителями на магнитных дисках D_1, D_2, D_3 и центральным процессором *ЦП* через два канала C_1, и C_2. При этом требуется, чтобы D_1 использовал канал C_1; D_2 - канал C_2, D_3 - оба канала C_1 и C_2 (рисунок 2.13).

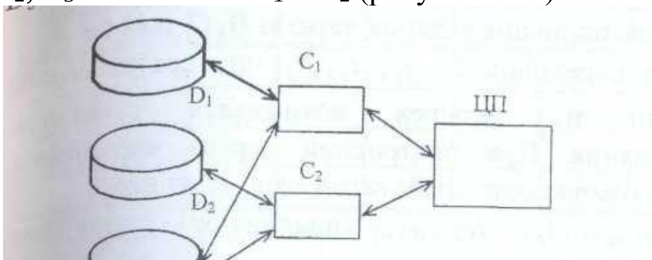

Рис. 2.13

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

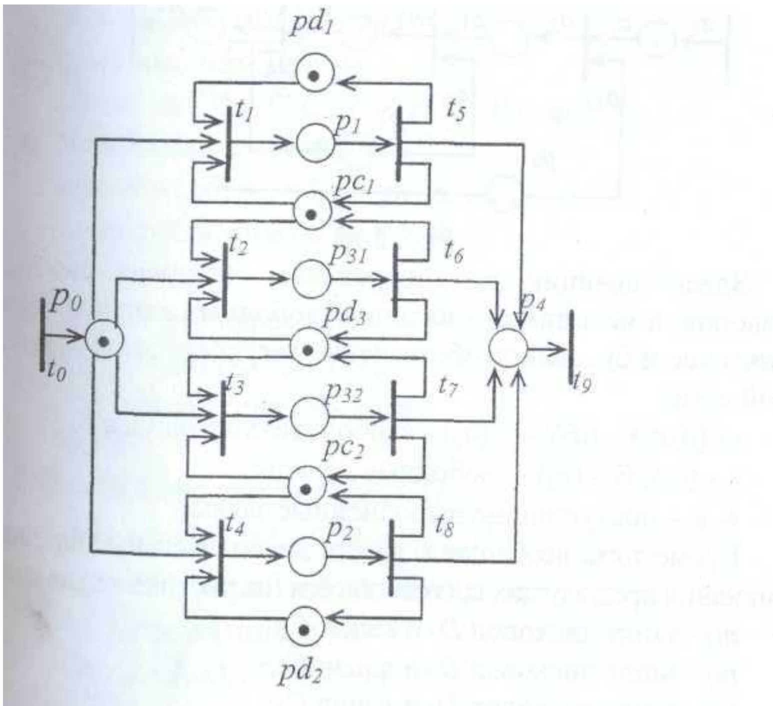

Рис. 2.14

Позиции p_{d1} , p_{d2} , p_{d3} определяют, свободен или занят, соответствующий дисковод D_1, D_2, D_3, позиции p_{c1}, p_{c2} соответственно - свободен или занят соответствующий канал C_1 и C_2. Позиции p_1 и p_2 - выполнение заданий парами D_1C_1 и D_2C_2 позиции p_{31} и p_{32} -выполнение задания парами D_3 C_1; и D_3C_2.

Каждый из переходов t,, t_2, t_3, t_4 при срабатывании определяет один из четырех возможных варианту обслуживания задания. При построении дерева маркировок необходимо дать возможность сработать каждому из них.

Переходы t_5, t_6, t_7, t_8 моделируют завершение выполнения задания по каждому из рассмотренных вариантов.

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

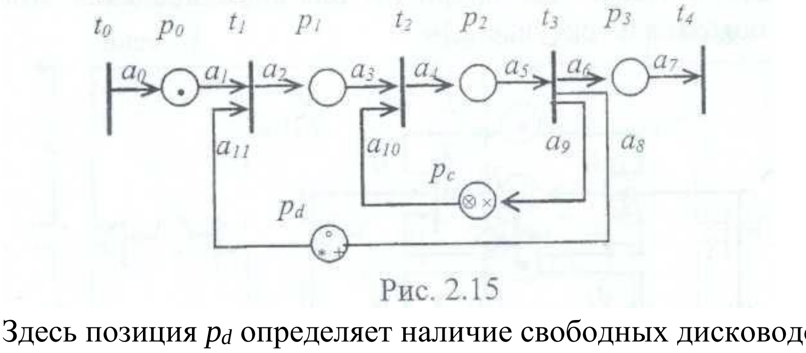

Рис. 2.15

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

◦ - (d_1); * - (d_2); + - (d_3) - свободные дисководы;

× - (c_1);⊗ - (c_2) - свободные каналы;

•- e - поступившие/выполненные заявки.

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

m_1 - занят дисковод D_1 и канал C_1;
m_2 - занят дисковод D_2 и канал C_2;
m_3 - занят дисковод *Ds* и канал C_1;
m_4 - занят дисковод *Ds* и канал C_2;
Правила срабатывания переходов t_1, t_2, t_3 задаются таблицей 2.1.

Таблица 2.1

Переход t_1			Переход t_2			Переход t_3			
p_0	p_d	p_1	p_1	p_c	p_2	p_2	p_3	p_c	p_d
e	d_1	d_1	d_1	c_1	m_1	m_1	e	c_1	d_1
e	d_2	d_2	d_2	c_2	m_2	m_2	e	c_2	d_2
e	d_3	d_3	d_3	c_1	m_3	m_3	e	c_1	d_3
			d_3	c_2	m_4	m_4	e	c_2	d_3

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, c2);
  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: *A = AP* ∪ *AT;*
  type AP = (al,a3,a5,a7,a10,a11);
  type AT = (a0,a2,a4,a6,a8,a9);
  переменные типа дуги:
  var a1,a3ta5,a7,a10,a11: AP;
  var a0,a2,a4,a6,a8,a9: AT,

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

• Цветовая функция

$$C(p) = \begin{cases} colorD \ if \ \ p \in \{pd, p1\}, \\ colorC \ if \ \ p \in \{pc\}, \\ colorE \ if \ \ p \in \{p0, p3\}, \\ colorM \ if \ \ p \in \{p2\}. \end{cases}$$

• Выражения на дугах *E(a)* задаются таблицей 2.2.

Таблица 2

$a0$	$1\text{\`}e$
$a1$	$1\text{\`}e$
$a2$	$if\ d = d1\ then\ 1\text{\`}d1\ else\ if\ d = d2\ then\ 1\text{\`}d2\ else$ $if\ d = d3\ then\ 1\text{\`}d3;$
$a3$	$if\ d = d1\ then\ 1\text{\`}d1\ else\ if\ d = d2\ then\ 1\text{\`}d2\ else$ $if\ d = d3\ then\ 1\text{\`}d3;$
$a4$	$if\ (d = d1\ and\ c = c1)\ then\ 1\text{\`}m1\ else$ $if\ (d = d2\ and\ c = c2)\ then\ 1\text{\`}m2\ else$ $if\ (d = d3\ and\ c = c1)\ then\ 1\text{\`}m3\ else$ $if\ (d = d3\ and\ c = c2)\ then\ 1\text{\`}m4\ else\ empty;$
$a5$	$if\ m = m1\ then\ 1\text{\`}m1\ else\ if\ m = m2\ then\ 1\text{\`}m2$ $else\ if\ m = m3\ then\ 1\text{\`}m3\ else\ 1\text{\`}m4;$
$a6$	$1\text{\`}e$
$a7$	$1\text{\`}e$
$a8$	$if\ m = m1\ then\ 1\text{\`}d1\ else\ if\ m = m2\ then\ 1\text{\`}d2\ else$ $if\ m = m3\ then\ 1\text{\`}d3\ else\ 1\text{\`}d3;$
$a9$	$if\ m = m1\ or\ m = m3\ then\ 1\text{\`}c1\ else$ $if\ m = m2\ or\ m = m4\ then\ 1\text{\`}c2;$
$a10$	$if\ c = c1\ then\ 1\text{\`}c1\ else\ if\ c = c2\ then\ 1\text{\`}c2$
$a11$	$if\ d = d1\ then\ 1\text{\`}d1\ else\ if\ d = d2\ then\ 1\text{\`}d2\ else$ $if\ d = d3\ then\ 1\text{\`}d3;$

- Блокировочная функция истинна для всех переходов: *G(t)=true*.
- Функция инициализации *I(p)*, задающая начальную маркировку $m_0(p_0)=(1\text{\`}e)$; $m_0(p1) = (\varnothing)$; $m_o(p_2) = (\varnothing)$; $m_0(p_3) = (\varnothing)$; $m_0(p_d) = (1\text{\`}d1, 1\text{\`}d2, 1\text{\`}\ d3)$; $m_0(p_c) = (1\text{\`}c1, 1\text{\`}c2)$.

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 тех пор она служит своеобразным тестом для методов решения задач распараллеливания.

Рис. 2.16

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 $(i = 1,...,N)$:

M_i - i-й мудрец гуляет;

E_i - i-й мудрец ест;

C_k - *k-я* палочка свободна;

C_{1i} - *i-й* мудрец держит левую палочку;

C_{2i} - i-й мудрец держит правую палочку.

Рассмотрим возможные события:

t_{i1}- мудрец начинает есть;

t_{i2} - мудрец уходит гулять и освобождает палочки;

t_{i3} - мудрец взял левую палочку;

t_{i4} - мудрец взял правую палочку.

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

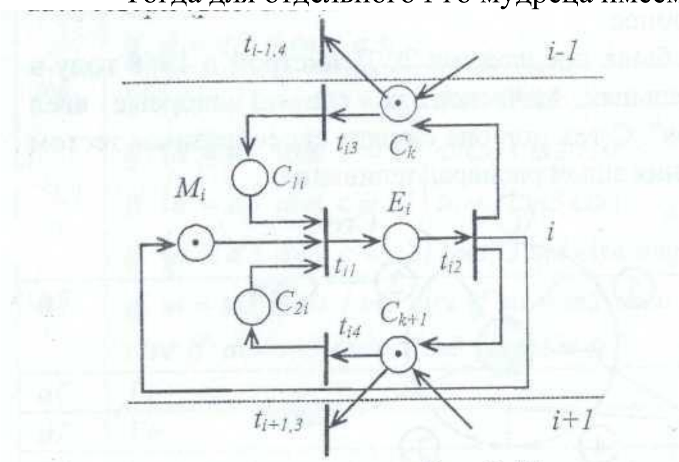

Рис. 2.17

Из рисунка видно, что мудрецы взаимно связаны орудиями питания, т.е. из-за ресурса C_k (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.