МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ВОЛЖСКИЙ ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ (ФИЛИАЛ)
ФЕДЕРАЛЬНОГО ГОСУДАРСТВЕННОГО БЮДЖЕТНОГО ОБРАЗОВАТЕЛЬНОГО
УЧРЕЖДЕНИЯ ВЫСШЕГО ОБРАЗОВАНИЯ
«ВОЛГОГРАДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Н.Н. Короткова

МАТЕМАТИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ПРОГРАММНЫХ СИСТЕМ

Электронное учебное пособие

Волжский

2019

УДК 519.8(07)
ББК 22.18я73
К 687

Рецензенты:

Заведующий кафедрой прикладной математики и информатики Волжского филиала ФГАОУ ВПО ВолГУ,
канд. физ.-мат. наук, доцент
Полковников А.А.,
начальник УКЦ ООО «Ракурс-инжиниринг», канд. тех. наук
Бурцев А.Г.

Издается по решению редакционно-издательского совета
Волгоградского государственного технического университета

Короткова, Н.Н.
Математическое обеспечение программных систем [Электронный ресурс] : учебное пособие / Н.Н. Короткова : ВПИ (филиал) ВолгГТУ, – Электрон. текстовые дан. (1 файл: 779 КБ). – Волжский, 2019. – Режим доступа: http://lib.volpi.ru. – Загл. с титул. экрана.

ISBN 978-5-9948-3238-7

Пособие содержит теоретические сведения и примеры решения задач по теме «Сети Петри». Предназначено для студентов бакалавриата, обучающихся по направлению 09.03.04 «Программная инженерия».

Ил. 45, табл. 4, библиограф.: 6 назв.

ISBN 978-5-9948-3238-7

Оглавление

1.1. Графический метод моделирования систем

Сети Петри используются для моделирования асинхронных систем, функционирующих как совокупность параллельных взаимодействующих процессов. Анализ сетей Петри позволяет получить информацию о структуре и динамическом поведении моделируемой системы.

Причинно-следственная связь событий в асинхронных системах задается множеством отношений вида "условия-события".

Построение моделей систем в виде сетей Петри заключается в следующем:

1. Моделируемые процессы описываются множеством событий (действий) и условий определяющих возможность наступления этих событий, а также причинно-следственными отношениями, устанавливаемыми на множестве пар "события-условия".
2. Определяются события-действия, последовательность выполнения которых управляется состояниями системы. Состояния системы задаются множеством условий, формируемых в виде предикатов. Количественно условия характеризуются величиной, которая выражается числами натурального ряда.
3. Условия, в зависимости от значений их количественных характеристик, могут выполняться или нет. Выполнение условий обеспечивает возможность реализации событий. Условия, с фактом выполнения которых связывается возможность реализации событий, называются предусловиями. Реализация события обеспечивает возможность выполнения других условий, находящихся с предусловиями в причинно-следственной связи. Эти условия называются постусловиями.

В сетях Петри условия – это позиции, а события – переходы. В соответствии с этим граф сети Петри является двудольным ориентированным мультиграфом.

Ориентированные дуги могут соединять только позиции и переходы в прямом и обратном направлении (свойство двудольности). Сеть Петри является мультиграфом, так как допускается кратность дуг между позициями и переходами (вершинами графа). В графах сети Петри количественные характеристики условий (числа натурального ряда) принято изображать числом меток в соответствующих позициях.

Рассмотрим метод моделирования на основе сетей Петри, а также его применение для моделирования параллельных систем взаимодействующих процессов и решения ряда классических задач из области синхронизации процессов.

Простое представление системы сетью Петри основано на двух основополагающих понятиях: событиях и условиях. ***События*** – это действия, имеющие место в системе. Возникновением событий управляет состояние системы. Состояние системы может быть описано множеством условий. ***Условие*** – это предикат или логическое описание состояния системы. Условие может принимать либо значение "истина", либо значение "ложь".

Так как события являются действиями, то они могут происходить. Для того чтобы событие произошло, необходимо выполнение соответствующих условий. Эти условия называют ***предусловиями события***. Возникновение события может вызывать нарушение предусловий и может привести к выполнению других условий, постусловий.

В качестве примера, рассмотрим задачу моделирования простого автомата-продавца. Автомат-продавец находится в состоянии ожидания до тех пор, пока не появится заказ, который он выполняет и посылает на доставку. Условиями для такой системы являются:

a. автомат-продавец ждет;
b. заказ прибыл и ждет;
c. автомат-продавец выполняет заказ;
d. заказ выполнен.

События будут:

1. Заказ поступил.
2. Автомат-продавец начинает выполнение заказа.
3. Автомат-продавец заканчивает выполнение заказа.
4. Заказ посылается на доставку.

Предусловия события 2 (автомат-продавец начинает выполнение заказа) очевидны: (a) автомат-продавец ждет; (b) заказ прибыл и ждет. Постусловие для события 2: (c) автомат-продавец выполняет заказ. Аналогично можно определить предусловия и постусловия для других событий и составить следующую таблицу событий и их предусловий и постусловий:

Таблица 1.

Событие	Предусловия	Постусловия
1	нет	b
2	a, b	c
3	c	d, a
4	d	нет

Такое представление системы легко моделировать сетью Петри. В сети Петри условия моделируются позициями, события – переходами. При этом входы перехода являются предусловиями соответствующего события; выходы – постусловиями. Возникновение события равносильно запуску соответствующего перехода. Выполнение условия представляется фишкой в позиции, соответствующей этому условию. Запуск перехода удаляет разрешающие фишки, представляющие выполнение предусловий и образует новые фишки, которые представляют выполнение постусловий.

Сеть Петри на рисунке 1 иллюстрирует модель приведенного выше автомата-продавца. Каждому переходу и позиции указаны соответствующие событие и условие.

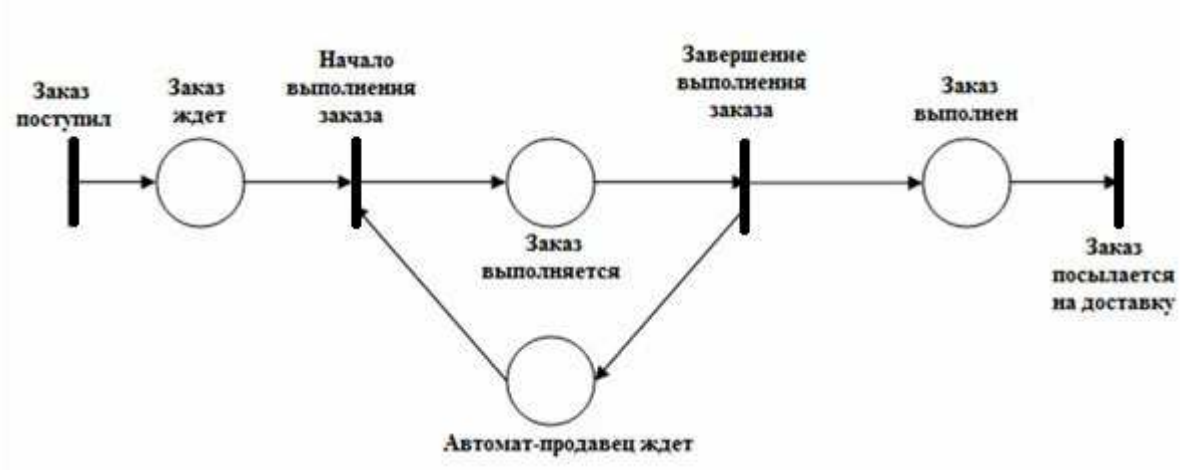


Рис.1. Сеть Петри для простого автомата-продавца

Аналогичный пример можно привести для вычислительной системы, которая обрабатывает задания, поступающие с устройства ввода, и выводит результаты на устройство вывода. Задания поступают на устройства ввода.

Когда процессор свободен и в устройстве ввода есть задание, процессор начинает обработку задания. Когда задание выполнено, оно посылается в устройство вывода; процессор же либо продолжает обрабатывать

другое задание, если оно имеется, либо ждет прихода задания, если устройство ввода еще не получило такового. Эта система может быть промоделирована сетью Петри, показанной на рисунке 2.

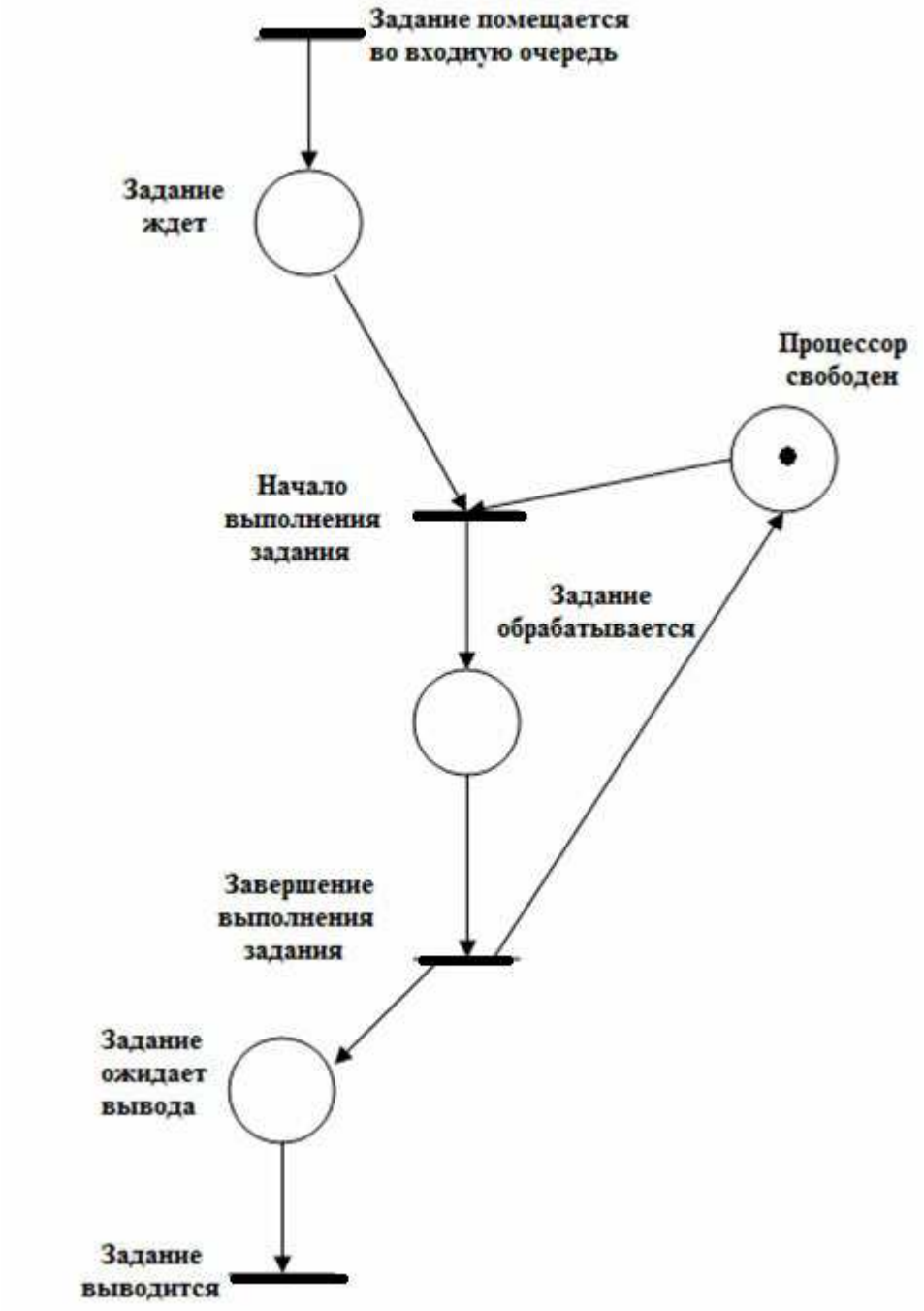


Рис.2. Моделирование простой вычислительной системы

Одной из особенностей сетей Петри и их моделей является свойственный сетям Петри и их моделям параллелизм или одновременность. В модели сети Петри два разрешенных невзаимодействующих события могут происходить независимо друг от друга. Синхронизировать события, пока это не требуется моделируемой системе, нет нужды. Но, когда син-

хронизация необходима, моделировать ее легко. Сети Петри представляются идеальными для моделирования систем с распределенным управлением, в которых несколько процессов выполняются одновременно.

Другая важная особенность сетей Петри – это их асинхронная природа. В сети Петри отсутствует измерение времени или течение времени. В реальной жизни различные события укладываются в различные интервалы времени, и это отражено в модели сети Петри независимостью от времени управления последовательностью событий. Структура сети Петри такова, что содержит в себе всю необходимую информацию для определения возможных последовательностей событий. Таким образом, на рисунке 2 предыдущего шага событие "завершение выполнения задания" должно следовать за соответствующим событием "начало выполнения задания". Однако нет и не требуется никакой информации, связанной с количеством времени, необходимым на выполнение задания.

Выполнение сети Петри рассматривается здесь как последовательность дискретных событий. Порядок появления событий является одним из возможных, допускаемых основной структурой. Это приводит к явной недетерминированности в выполнении сети Петри. Если в какой-то момент времени разрешено более одного перехода, то любой из нескольких возможных переходов может стать "следующим запускаемым". Выбор запускаемого перехода осуществляется недетерминированным образом, т.е. случайно. Эта особенность сети Петри отражает тот факт, что в реальной жизненной ситуации, где несколько действий происходит одновременно, возникающий порядок появления событий – не однозначен; скорее может возникнуть любая из множества последовательностей событий. Однако частичный порядок появления события – единственен.

Из теории относительности известно, что если два события произойдут одновременно (т.е. они не имеют причинной взаимосвязи), то порядок возникновения этих событий для двух разных наблюдателей может оказаться различным.

Такое представление влечет за собой значительные трудности при описании и анализе динамического поведения сети Петри, когда определяется последовательность запусков переходов. Для простоты обычно вводят следующее ограничение. Запуск перехода (и соответствующего события) рассматривается как мгновенное событие, занимающее нулевое время, и возникновение двух событий одновременно невозможно. Моделируемое таким образом событие называется ***примитивным***; примитивные события мгновенны и неодновременны.

Непримитивными называются такие события, длительность которых отлична от нуля. Они не являются одновременными и, следовательно, могут пересекаться во времени.

Недетерминированность и неодновременность запусков переходов в моделировании параллельной системы показываются двумя способами. Один из них показан на рисунке 3.

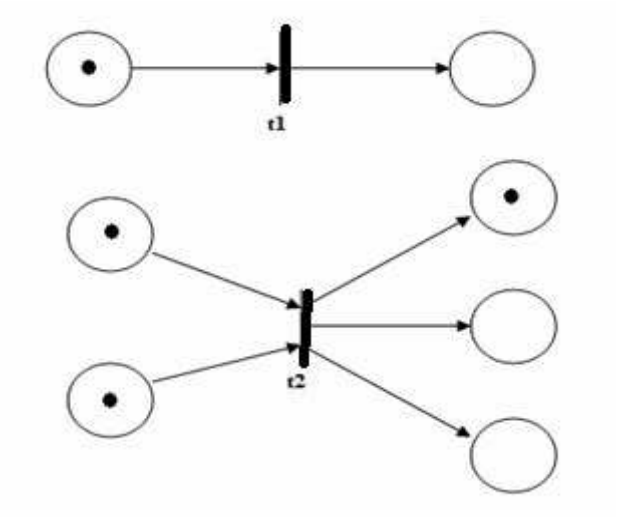


Рис.3. Одновременность

В этой ситуации два разрешенных перехода в любом случае не влияют друг на друга, и в число возможных последовательностей событий входит последовательность, в которой первым срабатывает один переход, и последовательность, в которой первым будет запушен другой переход. Это называется ***одновременностью***.

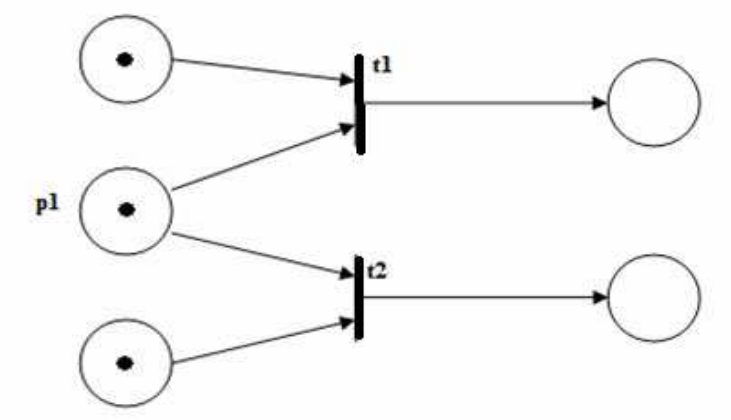


Рис.4. Конфликт

Другая ситуация, в которой одновременное выполнение затруднено и которая характеризуется невозможностью одновременного возникновения событий, показана на рисунке 4.

Здесь два разрешенных перехода находятся в ***конфликте***. Может быть запущен только один переход, так как при запуске он удаляет фишку из общего входа и запрещает другой переход. Таким образом моделируются взаимоисключающие события системы.

Рассмотрим ***представление блок-схемы сетями Петри***.

Вырожденным случаем параллельной системы обработки является система с одним процессом. Рассмотрим, как сетью Петри может быть представлен отдельный процесс.

Отдельный процесс описывается программой. Эта программа может быть написана на многих языках. Программа представляет два различных аспекта процесса: вычисление и управление. Вычисление связано с текущими арифметическими и логическими операциями, вводом и выводом, обычными манипуляциями над участками памяти и их содержимым. Управление связано с порядком выполнения выполняемых вычислений.

Сети Петри удачно представляют структуру управления программ. Сети Петри предназначены для моделирования упорядочения инструкций и потока информации, но не для действительного вычисления самих значений. Модель системы является абстракцией моделируемой системы. Поэтому она игнорирует все возможные специфические детали.

Стандартный способ представления структуры управления программ – это блок-схемы. Блок-схема представляет поток управления в программе. Например, программа с рисунка 5 представляется блок-схемой на рисунке 6. Блок-схема не указывает конкретные вычисления, которые надо произвести, а только определяет структуру программы. Таблица показывает, как можно проинтерпретировать блок-схему (рисунок 6) программы, представленной ниже. Каждая последовательная программа может быть представлена в виде блок-схемы. Таким образом, показывая, как блок-схема может быть представлена сетью Петри, можно показать представление сетью Петри программы.

На рисунке 5 представлена программа.

```
begin
  Writeln('Введите y1');
   Readln(y1);
   Writeln('Введите y2');
   Readln(y2);
    y3:=1;
    while y1>0 do
       begin
           if odd (y1) then
               begin
                    y3:=y3*y2;
                    y1:=y1-1;
                end;
                y2:=y2*y2;
                y1:=y1/2;
        end;
      Writeln('y=',y3);
  end.
```

Рис.5. Программа

Блок-схема для этой программы представлена на рисунке 6:

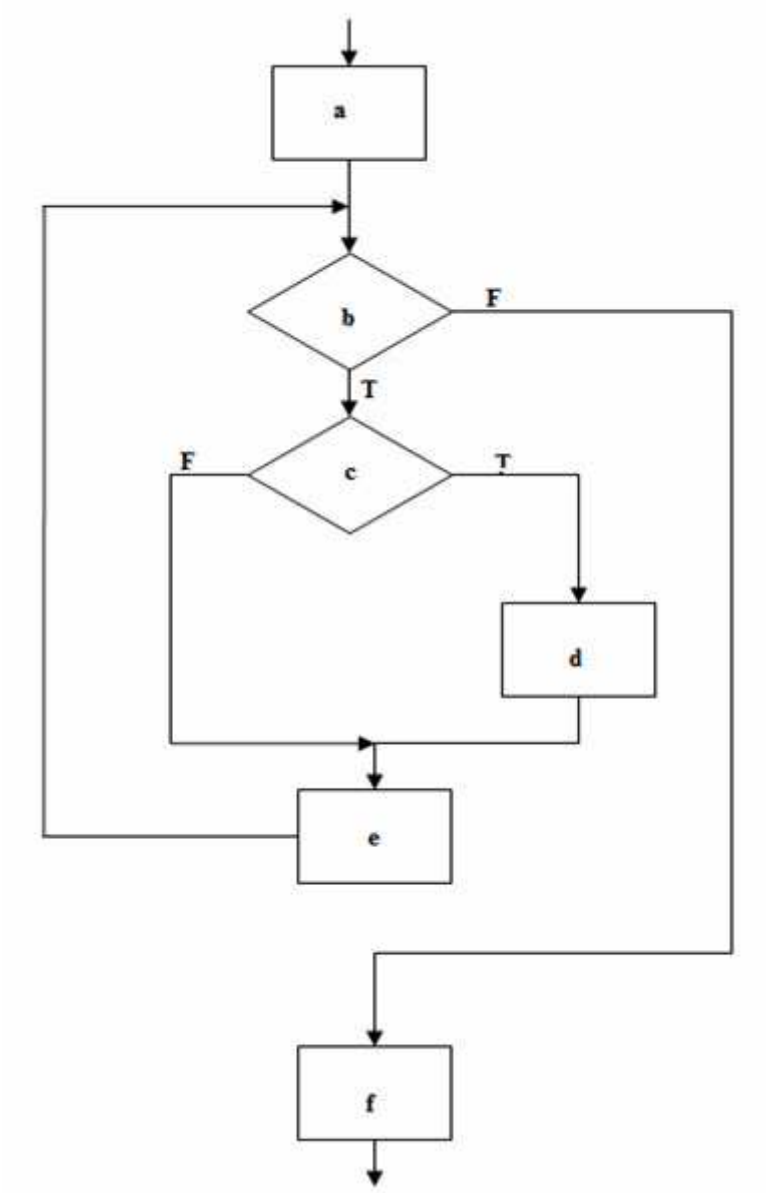


Рис.6. Блок-схема программы

Блок-схема во многом подобна сети Петри: блок-схема представима в виде узлов двух типов (принятия решения, показанные ромбами, и вычисления, показанные прямоугольниками) и дуг между ними. Удобный способ выполнения блок-схемы – введение фишки, которая представляет текущую инструкцию. По мере выполнения инструкций фишка передвигается по блок-схеме. Перевод блок-схемы в сеть Петри заменяет узлы блок-схемы на переходы сети Петри, а дуги блок-схемы – на позиции сети Петри. Каждая дуга блок схемы соответствует точно одной позиции в сети Петри. Узлы блок-схемы представляются по-разному в зависимости от типа узла: вычисления или принятия решения. На рисунке 7 иллюстрируются оба способа перевода.

В таблице 2 приведена интерпретация действий блок-схемы с рисунка 6.

Таблица 2

Действие	Интерпретация
a	Readln(y1); Readln(y2); y3:=1;
b	y1>0 ?
c	Odd (y1) ?
d	y3:=y3*y2; y1:=y1-1;
e	y2:=y2*y2; y1:=y1 /2;
f	Writeln(y3);

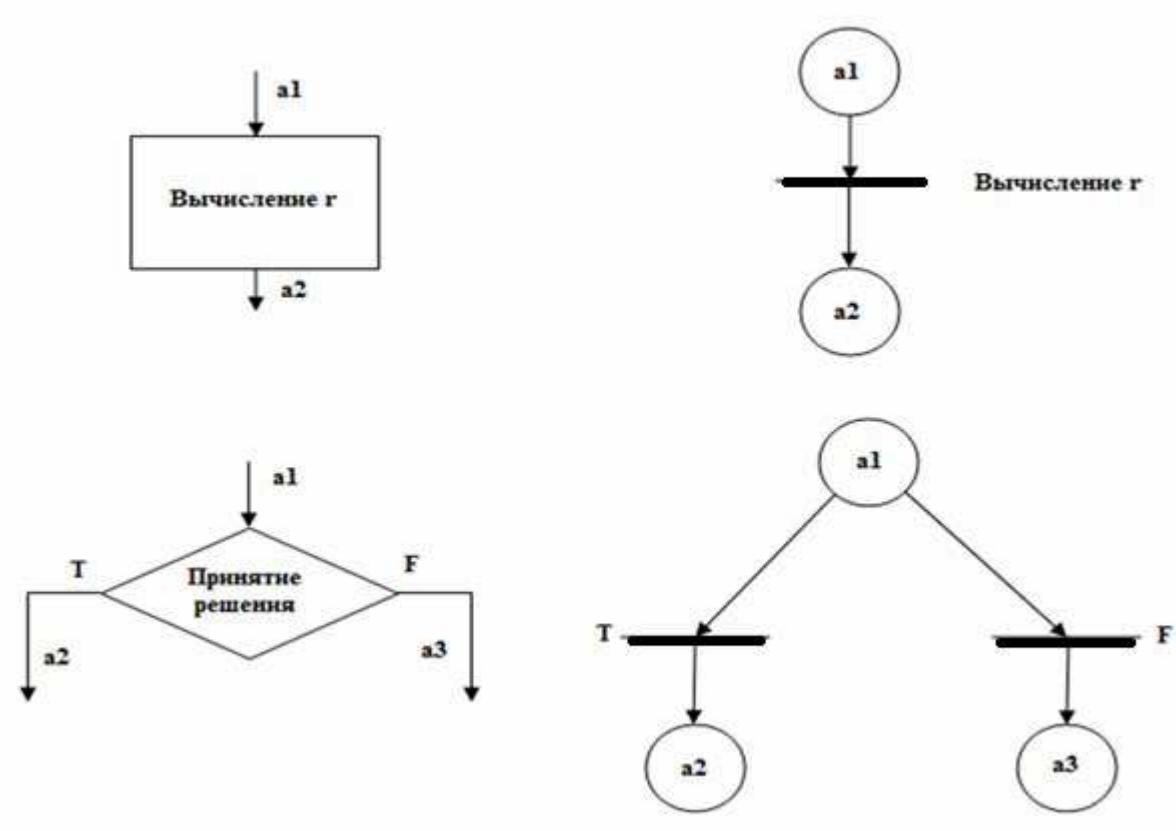


Рис.7. Перевод узлов вычисления и принятия решения в блок-схеме в переходы в сети Петри

На рисунке 8 показан перевод блок-схемы на рисунке 6 в эквивалентную сеть Петри.

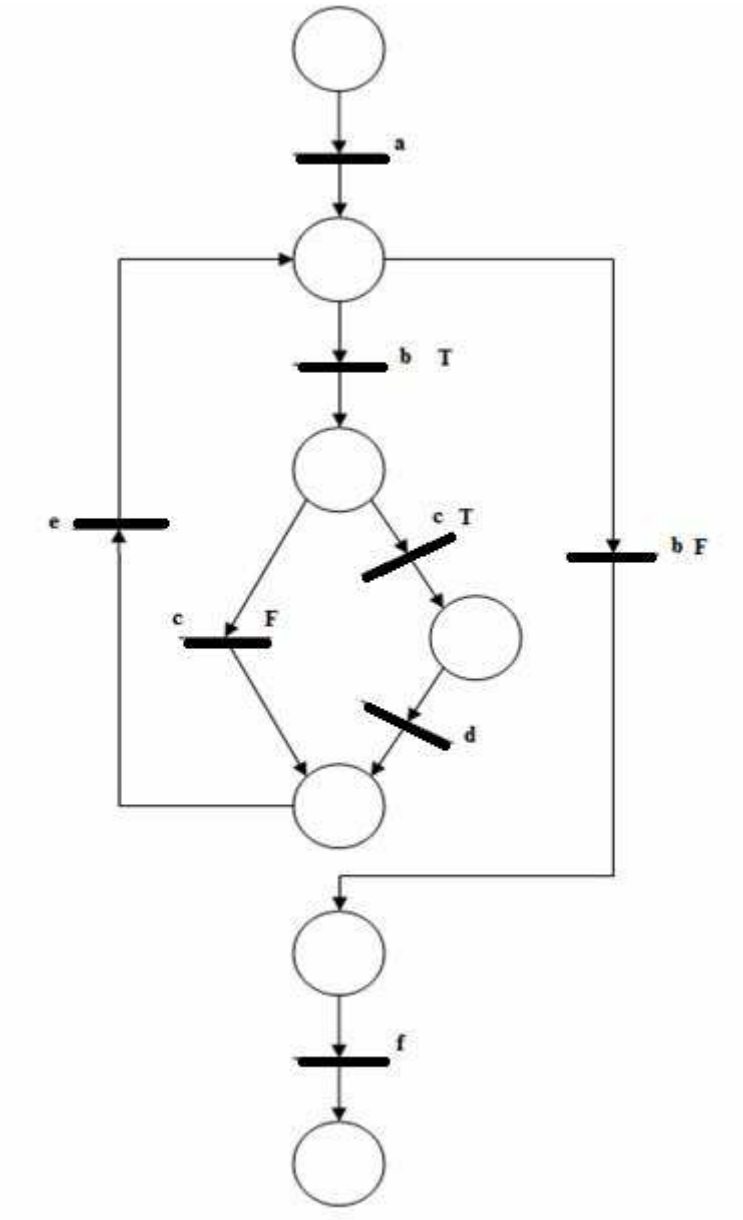


Рис.8. Представление сетью Петри программы

Рассмотрим интерпретацию фишек блок-схемы счетчиком команд. Фишка, находящаяся в позиции, означает, что счетчик команд установлен на готовность выполнения следующей инструкции. Каждая позиция имеет единственный выходной переход, за исключением позиции, которая предшествует принятию решения; такие позиции имеют по два выходных перехода, соответствующих истинному и ложному значению предиката.

Переходы связываются с действиями программы: вычислениями и принятиями решений. Для интерпретации сети Петри необходимо интерпретировать каждый переход. Переходы для вычислений имеют один вход и один выход.

Сетями Петри легко моделируется создание и выполнение параллельных ветвей различных вычислительных процессов. К примеру, на рисунке 9 изображена конструкция **fork/join**, впервые предложенная Деннисом и Ван Хорном. Переход **fork** моделирует "разветвление" – создание из одной ветви выполнения двух параллельных ветвей. Это, как правило, реализуется путем создания одной дополнительной ветви вдобавок к существующей. Переход **join**, в свою очередь, осуществляет "слияние" двух ветвей, по завершению их работы – уничтожение созданной параллельной ветви.

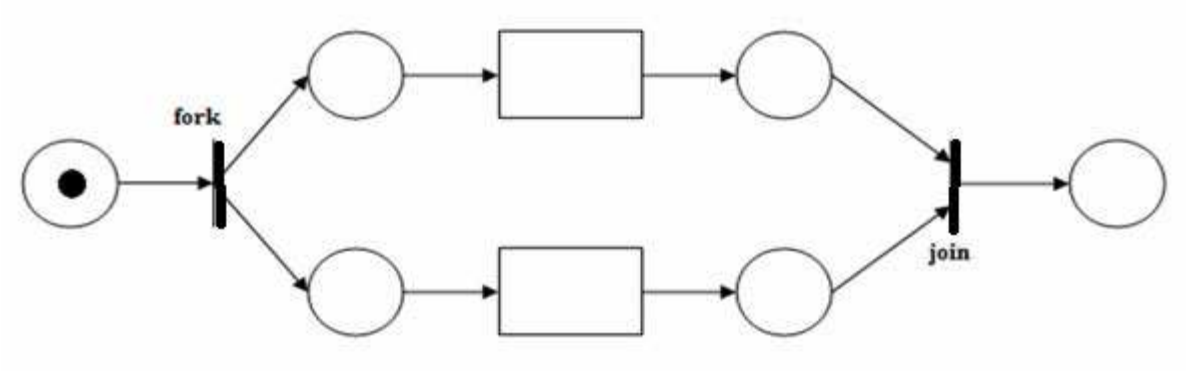


Рис.9. Моделирование конструкции **fork/join** сетью Петри

Другое предложение по введению параллелизма основано на операциях **parbegin** и **parend**. Структура управления была предложена Дейкстрой и имеет вид **parbegin** S_1**;** S_2**; …;** S_n **parend**, где S_i – предложение. Смысл структуры **parbegin/parend** заключается в параллельном выполнении каждого из предложений $S_1,S_2,\ldots,S_n$. Эта конструкция может быть представлена сетью Петри, показанной на рисунке 10.

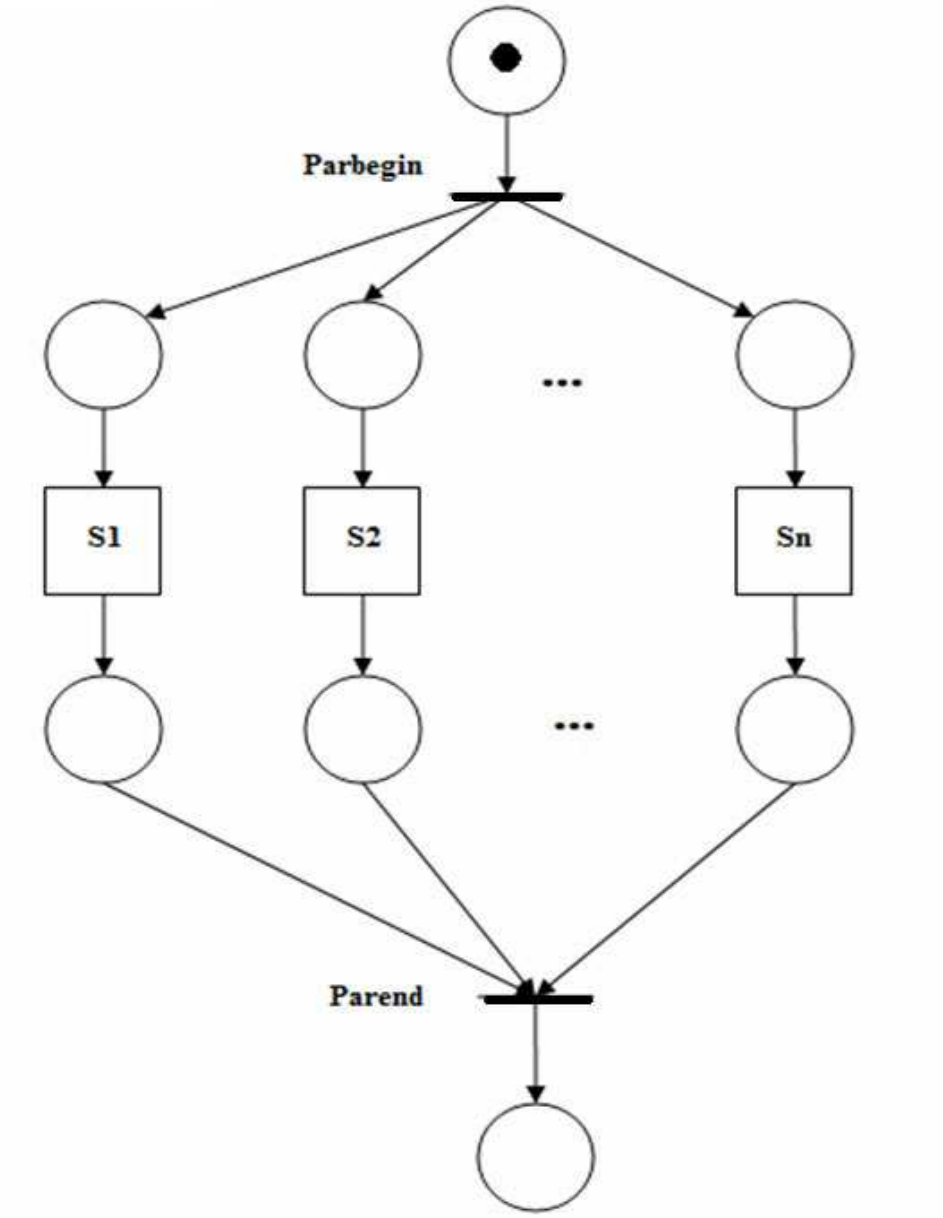


Рис.10. Моделирование структуры **parbegin/parend** сетью Петри

Используемые сегодня объекты синхронизации также легко моделируются сетями Петри. К примеру, на рисунке 11 изображен фрагмент сети Петри, моделирующий барьерную синхронизацию трех процессов.

В начале работы сети разрешенными являются три длительных перехода, характеризующих выполнение некоторых подзадач. Три независимых процесса выполняют эти подзадачи, и лишь после их полного завершения становится разрешенным переход-барьер **wait(all)**. После его срабатывания все три процесса снова продолжают свою работу.

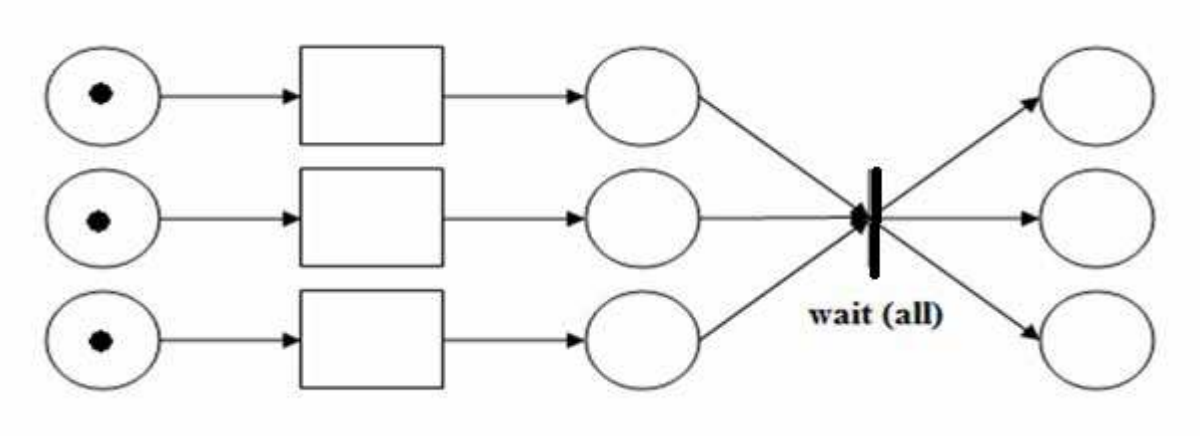


Рис.11. Фрагмент сети Петри, моделирующий барьерную синхронизацию трех процессов

Похожим образом может быть смоделировано ожидание завершения любой из подзадач, первой завершившей свое выполнение. К примеру, на рисунке 12 показан такой фрагмент сети.

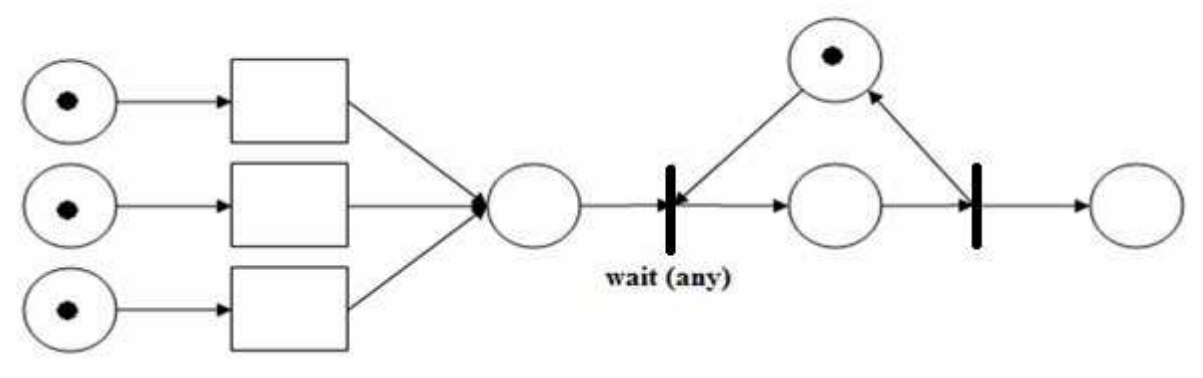


Рис.12. Фрагмент сети Петри, моделирующий ожидание завершения любой из подзадач, первой завершившей свое выполнение

При завершении выполнения любого длительного перехода становится разрешенным переход **wait(any)**. После его срабатывания дальнейшая работа продолжается, в то время как остальные подзадачи завершают свое выполнение. Дополнительная входная позиция перехода **wait(any)** защищает его от срабатывания при завершении остальных подзадач в случае, если в текущий момент ожидание не выполняется. В процессе дальнейшей работы сети фишка в эту позицию возвращается, и тем самым разрешается ожидание и обработка результатов выполнения остальных подзадач.

Введение параллелизма полезно только в том случае, когда компоненты процессов могут взаимодействовать при получении решения задачи. Такое взаимодействие требует распределения ресурсов между процессами. Для гарантии правильности работы системы в целом распределением нужно управлять. Проблемы синхронизации, возникающие при взаимодействии процессов, иллюстрируются многочисленными примерами. Среди задач синхронизации: задача о взаимном исключении, производителе/ потребителе, обедающих мудрецах, чтении/записи.

Последовательности событий отображаются срабатываниями переходов. Выполнение какого-либо условия связано с появлением одной или нескольких меток в соответствующей этому условию позиции. Соглашения о правилах срабатывания переходов является способом выражения причинно-следственных связей между условиями и событиями в системе.

При моделировании гибких производственных систем позиции отражают отдельные операции производственного процесса (например: транспортировка заготовки к конвейеру, передвижение заготовки к станку конвейером, обработку детали) или состояния компонентов гибкой производственной системы (например: робота, конвейера, станка). Наличие метки в одной из позиций соответствует состоянию выполнения некоторой из технологических операций либо состояние, в котором пребывают некоторые из компонентов гибкой производственной системы.

Переходы соответствуют событиям, отображающим начало или завершение моделируемых операций. Например, переход интерпретируется как событие, связанное с завершением операции транспортирования заготовки роботом и ее установки на конвейере, а также с началом операции перемещения заготовки конвейером к станку.

Присвоение меток позициям сети Петри называют маркировкой сети. Динамика сетей Петри связана с механизмом изменения маркировок позиций и соглашениями о правилах срабатывания переходов. Переход срабатывает, если в каждой входной позиции (предусловии) число меток не меньше числа дуг, исходящих из позиции в данный переход. Такие переходы называют возбужденными, их срабатывание может наступить через любой конечный промежуток времени после возбуждения. В результате срабатывания из всех входных позиций перехода исключается число меток, равное числу дуг, выходящих из соответствующей позиции в переход, а в выходные позиции данного перехода добавляется число меток, равное числу дуг, исходящих из перехода в соответствующую выходную позицию. На рис. 13,а) показан возбужденный переход, а перехода на рис.13,б) невозбужден.

а) б)

Рис.13, а) − возбужденный переход, б) − невозбужденный переход

Формализованное описание сетей Петри.

Для описания и математического анализа процессов с точками ветвления и синхронизации взаимодействия разработан аппарат сетей Петри. Структура сети представляется ориентированным двудольным графом. Множество V вершин графа разбивается на два подмножества T и P, $V = T \cup P$, $T \cap P = \varnothing$. Дугами могут связываться вершины из множеств P и T. Динамика развития процессов отражается в вершинах P метками (марками). Распределение меток по вершинам P называют маркированием сети. Каждое маркирование соответствует определенному состоянию сети.

Сеть Петри определяется пятеркой $N = \{P, T, I, O, M_0\}$, где

$P = \{p_i\}$, $i = 1,2,...,n$ - множество позиций;
$T = \{t_j\}$, $j = 1,2,...,m$ - множество переходов;
$I : T \times P \to \{0,1\}$ - функция следования;
$O : P \times T \to \{0,1\}$ - функция предшествования;
$M_0 : P \to Z_0$ - начальное маркирование (состояние) сети;
Z_0 - множество положительных целых чисел.

Функции I и O задают множества дуг $\{t_j, p_i\}$ и $\{p_i, t_j\}$ соответственно.

Дуги, предшествующие позиции p_i, обозначим множеством $I(p_i) = \{(t_j, p_i) | I(t_j, p_i) = 1\}$, а дуги, предшествующие переходу t_j, множеством $I(t_j) = \{(p_i, t_j) | O(p_i, t_j) = 1\}$.

Здесь запись $I(t_j, p_i) = 1$ означает наличие дуги (t_j, p_i), а запись $O(p_i, t_j) = 1$ - дуги (p_i, t_j). Аналогично, дуги, следующие из p_i и t_j, представим множествами $O(p_i) = \{(p_i, t_j) | O(p_i, t_j) = 1\}$, $O(t_j) = \{(t_j, p_i) | I(t_j, p_i) = 1\}$.

Входные позиции перехода t_j объединяются в множества его предшественников $\Pr e(t_j) = \{p_i \in P | O(p_i, t_j) = 1\}$, а выходные позиции – в множества позиций–последователей $Post(t_j) = \{p_i \in P | I(t_j, p_i) = 1\}$.

Маркирование сети представляется вектором $M = \{m(p_i)\}$, где $m(p_i)$ - число меток в позиции p_i. Переход t_j возбужден при маркировании M и может сработать, если выполняется условие $\forall p_i \in \Pr e(y_j) [m(p_i) - O(p_i, t_j) \geq 0]$, то есть число меток $m(p_i)$ больше или равно числу дуг (p_i, t_j), что соответствует $m(p_i) - O(p_i, t_j) \geq 0$.

Срабатывание перехода t_j приводит к тому, что каждая позиция $p_i \in \Pr e(t_j)$ теряет $O(p_i, t_j)$ меток, а каждая из позиций $Post(t_j)$ получает $I(t_j, p_i)$ меток.

Если при маркировании M возбуждено несколько переходов, то порядок их срабатывания не определен, и, следовательно, может быть представлено несколько последовательностей срабатывающих переходов.

1.2. Методы анализа характеристик сетей

Существуют два базовых метода анализа сетей Петри. Один из них основан на использовании графа достижимых маркировок (диаграммы состояний сети), другой – на применении уравнения состояний и методов линейной алгебры. Статические свойства системы определяет графовая часть сети Петри, а динамические – начальное маркирование и правила возбуждения (моделирования).

Анализ заключается в изучении основных свойств сетей Петри: безопасности, ограниченности, сохранении, активности, достижимости и покрываемости.

Активность.

Тупик в сети Петри – это переход (или множество переходов), который не может быть запущен. В связи с понятием тупика определим для сети Петри N с начальной маркировкой M_0 следующие *уровни* активности переходов.

Уровень 0: Переход t обладает *активностью уровня* 0 и называется *мёртвым* или *пассивным*, если он никогда не может быть запущен.

Уровень 1: Переход t обладает *активностью уровня 1* и называется *потенциально живым,* если существует такая $M \in R(N)$, что t разрешён в M.

Уровень 2: Переход t обладает *активностью уровня 2* и называется *живым,* если для всякой $M_0 \in R(N)$ переход t является потенциально живым для сети Петри N с начальной маркировкой M_0.

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

Уровень 4: Переход t обладает *активностью уровня 4* и называется активным, если для всякой разметки $M_0 \in R(N)$ существует такая последовательность запусков, в которых переход t разрешен.

Сеть Петри обладает активностью уровня j ,если каждый её переход обладает активностью уровня j.

Сеть Петри называется *живой*, если все её переходы являются живыми.

В качестве примера, иллюстрирующего уровни активности, рассмотрим сеть Петри на рисунке 14. Переход t_0 не может быть запущен никогда; он пассивен. Переход t_1 можно запустить точно 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. Если мы хотим запустить t_2 пять раз, мы запускаем пять раз t_3, затем t_1 и после этого пять раз t_2. Однако, как только запустится t_1 (t_1 должен быть запущен до того, как будет запущен t_2), число возможных запусков t_2 станет фиксированным. Следовательно, t_2 обладает активностью уровня 2, но не уровня 3. С другой стороны, переход t_3 можно запускать бесконечное число раз, и поэтому он обладает активностью уровня 3, но не уровня 4, поскольку, как только запустится t_1, t_3 больше запустить будет нельзя.

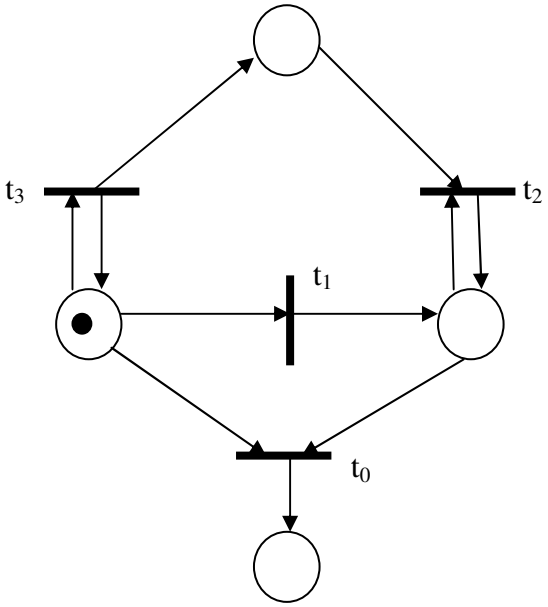


Рис.14. Сеть Петри, иллюстрирующая различные уровни активности

Важной задачей, возникающей при распределении ресурсов, является задача выявления тупиков. Рассмотрим изображенную на рисунке 15 систему, включающую два различных ресурса q и r и два процесса а) и в), нуждающиеся в обоих ресурсах. Каждый процесс запрашивает ресурс, а затем освобождает его. Процесс а) сначала запрашивает ресурс q, затем ресурс r и освобождает их. Процесс в) работает аналогично, но запрашивает сначала ресурс r, а затем q.

Начальная маркировка помечает ресурсы свободными и готовность процессов к работе. Выполнение сети в последовательности t_1,t_2,t_3,t_4,t_5,t_6 или t_4,t_5,t_6,t_1,t_2,t_3 не приводит к тупику. Если начать с переходов t_1,t_4 то выполнение заблокировано, процесс а) обладает ресурсом q и хочет получить r, процесс в) обладает r и хочет получить q.

Тупик в сети Петри – это переход (или множество переходов), которые не могут быть запущены. Переходы t_2 и t_5 являются тупиковыми. Переход активен в маркировке M, если он потенциально запустим во всякой маркировке из $R(N)$. Следовательно, если переход активен, то всегда возможно перевести сеть Петри из ее текущей маркировки в маркировку, в которой запуск перехода станет разрешенным.

Маркировку $M \in R(N)$ называют тупиковой, если в этом состоянии сети ни один из переходов не может сработать.

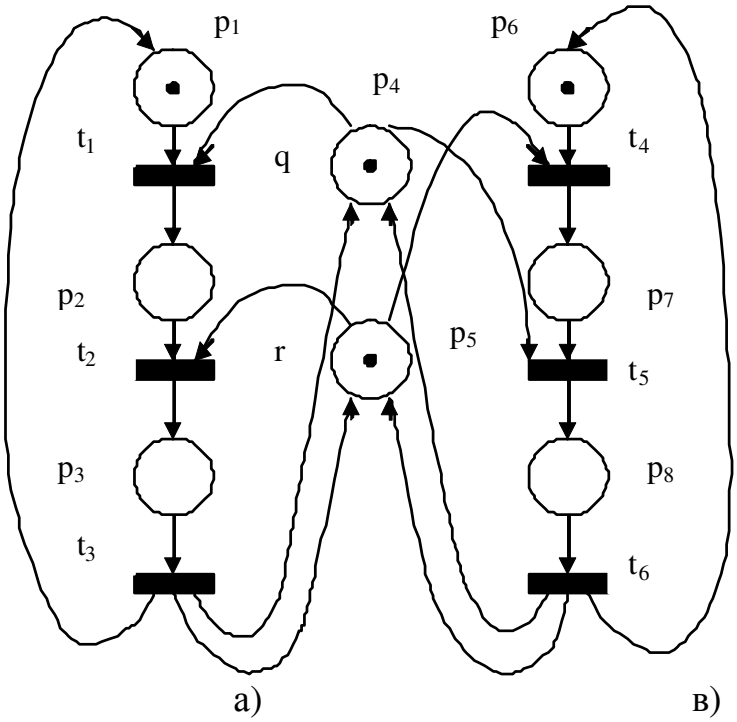


Рис. 15. Иллюстрация наличия тупика

Достижимость.

Маркирование M называют достижимым из маркирования M_0, если существует последовательность срабатывающих переходов $\sigma = t_1, t_2, ..., t_n$, переводящая сеть N из состояния M_0 в M. Отношение следования маркирований, возбуждаемых действием последовательности σ имеет вид: $M_0 \xrightarrow{t_1} M_1 \xrightarrow{t_2} M_2 \xrightarrow{t_3} ... \xrightarrow{t_n} M$.

Переход t_j называют достижимым в сети N из маркирования M, если существует такое маркирование M', достижимое из M, при котором происходит срабатывание перехода t_j.

К задаче достижимости могут сводиться многие задачи. Например, тупик в сети на рисунке 15 может возникнуть, если достижимым является состояние (0, 1, 0, 0, 0, 0, 1, 0).

Покрываемость.

Для сети Петри с начальной маркировкой M_0 и маркировки M определить, существует ли такая достижимая маркировка $M' \in R(N)$, что $M' \geq M$.

(Отношение $M' \geq M$ истинно, если каждый элемент маркировки M' не меньше соответствующего элемента маркировки M.)

Задачи достижимости и покрываемости могут рассматриваться относительно некоторых интересующих нас подмножеств позиций.

Живость.

Переход t_j называют живым, если он достижим в N из любого $M \in R(N)$, где $R(N)$ - множество достижимых маркирований сети.

Сеть N - живая, если все ее переходы живые. Доказав, что сеть является живой, можно гарантировать, что в соответствующей системе выполнимы все элементарные действия процессов при их развитии.

Безопасность.

Позиция $p_i \in P$ сети $N = \{P, T, I, O, M_0\}$ является безопасной, если $m(p_i) \leq 1$ для любой $M \in R(N)$. Сеть Петри безопасна, если безопасна каждая ее позиция.

Сеть N - безопасная, если $\forall M \in R(N): m(p_i) \leq 1$, то есть при всех достижимых маркированиях ее позиции не могут иметь более одной метки. Анализ безопасности проводится для исключения условий одновременного использования ресурса несколькими процессами. Например, если два сообщения о нем могут одновременно передаваться по каналу или два вычислительных процесса не могут одновременно находиться в критической программной секции, имеет место безопасность маркирования в соответствующих позициях сетевой модели.

Безопасность – важное свойство для аппаратной реализации. Безопасная позиция имеет число меток 0 или 1 и может быть реализована одним триггером.

Сети, в которых позиции рассматриваются (интерпретируются) как предусловия событий, маркировка каждой позиции должна быть безопасной.

Ограниченность.

Безопасность – это частный случай более общего свойства ограниченности. Безопасность позволяет реализовать позицию триггером, а в более общем случае можно использовать счетчик. Любой счетчик ограничен по максимальному числу К. Соответствующая позиция также является К-

безопасной или К-ограниченной, если количество меток в ней не может превысить целое число К.

Позиция $p_i \in P$ сети $N = \{P, T, I, O, M_0\}$ является К-безопасной, если $m(p_i) \le K$ для любой $M \in R(N)$..

Позиция называется ограниченной, если она К-безопасна для некоторого К.

Сеть Петри ограничена, если все ее позиции ограничены, то есть для каждой позиции сети существует предельное число меток, которые могут быть там одновременно. Если метки используются для представления промежуточных данных, созданных системой, то, проанализировав ограниченность сети, можно сделать вывод о возможности потери информации в ВС при ее буферировании.

Сохранение.

В сетях Петри, моделирующих запросы, распределения и освобождения ресурсов, некоторые позиции могут представлять состояние ресурсов. Например, если три метки в позиции представляют три устройства (однотипных) в вычислительной системе, то интерес представляет свойство сохранения меток. То есть метки, представляющие ресурсы, никогда не создаются и не уничтожаются.

Сеть Петри называется строго сохраняющей, если для всех $M \in R(N)$ выполняется условие:

$$\sum_{p_i \in P} m(p_i) = \sum_{p_i \in P} m_0(p_i)$$

Сеть Петри должна сохранять ресурсы, которые она моделирует. Однако не всегда имеется однозначное соответствие между меткой и количеством или числом ресурсов. В этом случае метка используется для создания кратных меток (по одной на ресурс), путем запуска перехода с большим числом выходов, чем входов. Поэтому вводятся взвешенные метки, а условие сохранения определяется через взвешенную сумму меток.

Сеть N обладает терминальностью (завершаемостью) процессов, если при всех допустимых начальных маркированиях завершение процессов в ней приводит к тому, что метки покидают сеть или собираются в некоторой терминальной (конечной) позиции. Терминальность гарантирует работу систем при следующей инициализации без всяких побочных эффектов. Терминальность процессов связана с результативностью алгоритмов

(реентерабельностью соответствующих программ). Два перехода, связанные с общей входной позицией p_i, являются конфликтными, если они возбуждаются в результате появления метки в p_i. Конфликт за метку из p_i приводит к случайному (недетерминированному) характеру процессов в сети.

Сеть – непротиворечивая, а процессы в ней – детерминированные, если для всех достижимых маркирований множество переходов $E(M)$, возбуждаемых маркированием M, равно множеству $S(M)$ переходов, срабатывающих при M, то есть $\forall M \in R(N): E(M) = S(M)$.

Методы анализа на основе дерева достижимости

Особый интерес вызывают методы анализа свойств сетей Петри, которые обеспечивают автоматический анализ моделируемых систем. Рассмотрим метод анализа сетей Петри, который основан на использовании *дерева достижимости*.

Дерево достижимости представляется множеством состояний сети.

Рассмотрим следующую сеть Петри.

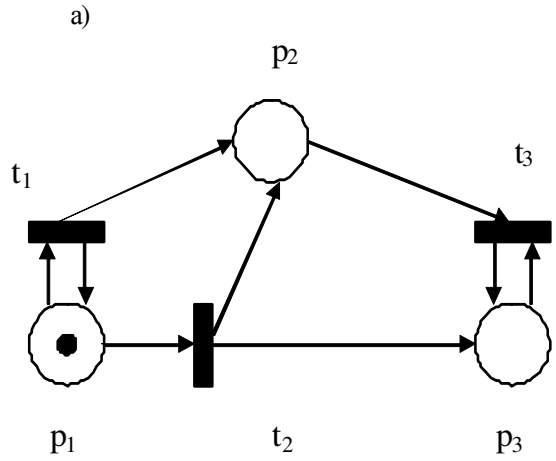


Рис.16. Сеть Петри

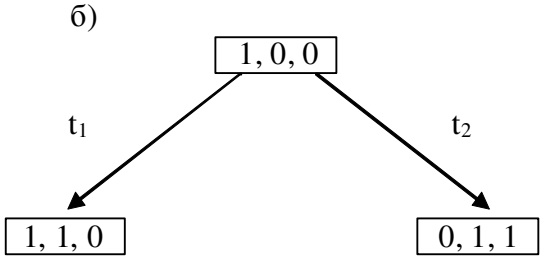


Рис.17. Первый шаг построения дерева достижимости

В начальной маркировке (1, 0, 0) разрешены два перехода t_1 и t_2. Первый шаг построения дерева достижимости показан на рис.17. Из маркировки (1, 1, 0) можно снова запустить t_1 (получая (1, 2, 0)) и t_2 (получая (0, 2, 1)); из (0, 1, 1) можно запустить t_3 (получая (0, 0, 1)). Второй шаг построения дерева достижимости показан на рисунке 18.

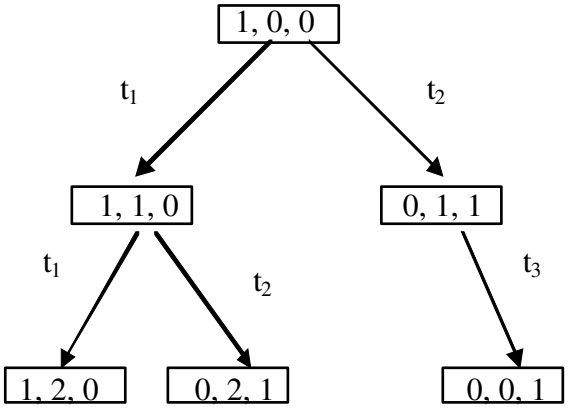


Рис.18. Второй шаг построения дерева достижимости

Продолжая три полученные маркировки (рис 18), на третьем шаге получим следующие маркировки (см. рис.19). Маркировка (0, 0, 1) пассивная: никакой переход в ней не разрешен. Маркировка (0, 1, 1), порождаемая запуском t_3 в (0, 2, 1), была уже порождена непосредственно из начальной маркировки запуском t_2.

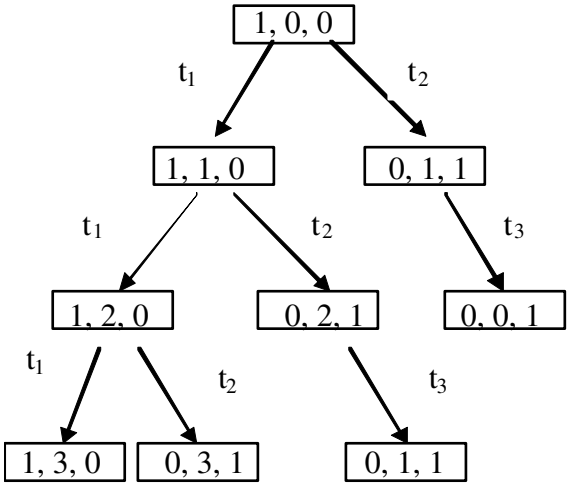


Рис.19. Третий шаг построения дерева достижимости

Если эту процедуру повторять снова и снова, каждая достижимая маркировка окажется порожденной, а полученное дерево может оказаться бесконечным.

Всякий путь в дереве, начинающийся в корне, соответствует допустимой последовательности переходов.

Для превращения дерева в полезный инструмент анализа необходимо найти средства ограничения его до конечного размера. Приведение к конечному представлению осуществляется двумя операциями.

Первая из них – это использование пассивных маркировок, именуемых терминальными вершинами, и маркировок, ранее встречавшихся в дереве, именуемых дублирующими вершинами. В нашем примере маркировка (0, 0, 1) – терминальная, а (0, 1, 1) – дублирующая.

Вторая операция. Рассмотрим последовательность запусков переходов σ, начинающуюся в начальной маркировке M_0 и концом M', $M' > M_0$. Маркировка M' совпадает с маркировкой M_0, за исключением того, что имеет некоторые «дополнительные» метки в некоторых позициях, то есть $M' = M_0 + (M' - M_0)$ и $M' - M_0 > 0$. Поскольку на запуски переходов дополнительные метки не влияют, последовательность σ можно запустить снова, начиная в M', приходя к маркировке M''. Так как действие последовательности переходов σ добавило $M' - M_0$ меток к маркировке M_0, она добавит также $M' - M_0$ меток к маркировке M', поэтому $M'' = M' + (M' - M_0)$ или $M'' = M_0 + 2(M' - M_0)$. В общем случае последовательность σ можно запустить n раз, получив в результате маркировку $M_0 + n(M' - M_0)$.

В нашем примере можно запустить переход t_1 столько раз сколько необходимо для того, чтобы получить произвольное число меток в p_2. Это бесконечное число маркировок обозначим символом ω. Для любого числа a определим

$$\omega + a = \omega, \quad \omega - a = \omega, \quad a < \omega, \quad \omega \le \omega$$

Эти операции с символом ω достаточны для построения дерева.

Алгоритм построения дерева. Каждая вершина i дерева связывается с расширенной маркировкой $M(i)$. В этой маркировке число меток в позиции может быть либо неотрицательным целым, либо ω. Каждая вершина классифицируется как граничная или терминальная, или дублирующая, или внутренняя.

Граничными являются вершины, которые не обработаны алгоритмом. Алгоритм превратит их в терминальные, дублирующие, внутренние.

Алгоритм начинает с начальной маркировки M_0, принимая ее в качестве граничной вершины.

Пусть x – граничная вершина.

1. Если в дереве имеется другая вершина y, не являющаяся граничной, и с ней связана также маркировка $M(x) = M(y)$, то вершина x – дублирующая.
2. Если для маркировки $M(x)$ ни один из переходов не разрешен для всех $t_i \in T$, то x – терминальная вершина.
3. Для всякого перехода $t_i \in T$, разрешенного в $M(x)$, создана новая вершина z дерева. Маркировка $M(z)$ определяется для каждой позиции p_i следующим образом:

а) если $M(x)_i = \omega$, то $M(z)_i = \omega$.

б) если на пути от корневой вершины к x существует вершина y с $M(y) < \delta(M(x), t_j)$ и $M(y)_i < \delta(M(x), t_j)_i$, то $M(z)_i = \omega$, δ - функция следующего состояния.

в) в противном случае $M(z)_i = \delta(M(x), t_j)_i$,

Функция $\delta(M(x), t_j)$ не определена, если t_j не разрешен в маркировке $M(x)$. Если t_j разрешен, то $\delta(M(x), t_j) = M'(x)$, где $M'(x)$ – маркировка, полученная после срабатывания t_j. Дуга, помеченная t_j, направлена от вершины x к вершине z. Вершина x переопределяется как внутренняя, вершина z становится граничной.

Когда все вершины дерева – терминальные, дублирующие или внутренние, алгоритм останавливается.

Для нашего примера с рисунка 16 дерево достижимости показано на рисунке 20.

Имеется доказательство того, что алгоритм конечный и не может создавать новые граничные вершины бесконечно.

Ниже на рисунке 21 приведем еще один пример сети Петри и дерева достижимости для него.

Сеть Петри ограничена тогда и только тогда, когда символ ω отсутствует в ее дереве достижимости. Если сеть ограничена и символ ω отсутствует в дереве достижимости, то сеть представляет систему конечных со-

стояний. Это позволяет решить вопросы анализа простым перебором и проверкой конечного множества всех достижимых маркировок.

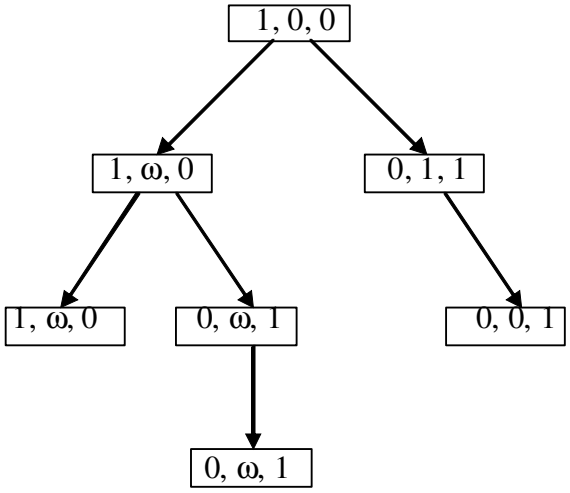


Рис 20. Пример дерева достижимости

Сеть Петри является сохраняющей, если она не теряет и не порождает метки, а просто передвигает их. Свойство сохранения проверяется по дереву достижимости вычислением для каждой маркировки суммы меток. Если метки взвешены, то вычисляется взвешенная сумма. Если сумма одинакова для каждой достижимой маркировки, сеть – сохраняющая.

Задача активности одного перехода формулируется следующим образом: активен ли данный переход $t_j \in T$?

Очевидно, что задача активности сети Петри сводится к задаче активности одного перехода. Для нахождения решения задачи активности мы просто решим задачу активности одного перехода для каждого перехода. Если у нас m переходов, то мы должны решить m задач активности одного перехода.

Задачу достижимости можно также свести к задаче активности. Поскольку варианты задачи достижимости эквивалентны, мы рассмотрим задачу достижимости нуля в одной позиции. Если перед нами стоят какие-либо другие задачи достижимости, их можно свести к задаче достижимости нуля в одной позиции. Теперь, если мы хотим определить, может ли быть позиция p_i нулевой в какой-либо достижимой маркировке для сети Петри $N_1 = \{P_1, T_1, I_1, O_1, M_1\}$ с начальной маркировкой M_1, то построим сеть Петри $N_2 = \{P_2, T_2, I_2, O_2, M_2\}$ с начальной маркировкой M_2, которая будет активна тогда и только тогда, когда нулевая маркировка не будет достижима из M_1.

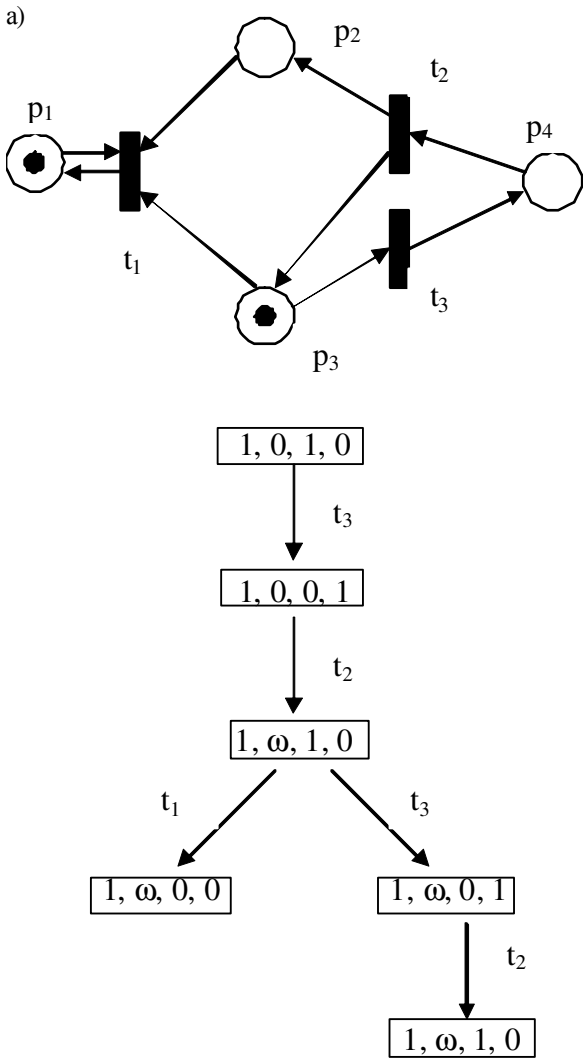


Рис 21. Пример сети Петри и дерева достижимости

Следующая задача, которую можно решить с помощью дерева достижимости, – задача покрываемости.

В задаче покрываемости мы хотим для данной маркировки M определить, достижима ли маркировка $M' \geq M$. Задача покрываемости маркировки M маркировкой M' сводится к поиску на дереве такой вершины x, состояние которой покрывает состояние M. Если такой вершины с маркировкой $M'(x)$ не существует, маркировка M не покрывается никакой достижимой маркировкой. Если такая вершина найдена, M' дает достижимую маркировку, покрывающую M.

Путь от корня к покрывающей маркировке определяет последовательность переходов, которые приводят из начальной маркировки к покрывающей маркировке, а маркировка, связанная с этой вершиной, определяет покрывающую маркировку. Символ ω должен рассматриваться как обозначение бесконечного множества значений. Если компонента покры-

вающей маркировки – маркировка M, то в пути от корня к покрывающей маркировке имеется «цикл». Для увеличения соответствующей компоненты с тем, чтобы она была не меньше, чем в данной маркировке, необходимо достаточное число раз повторить этот цикл.

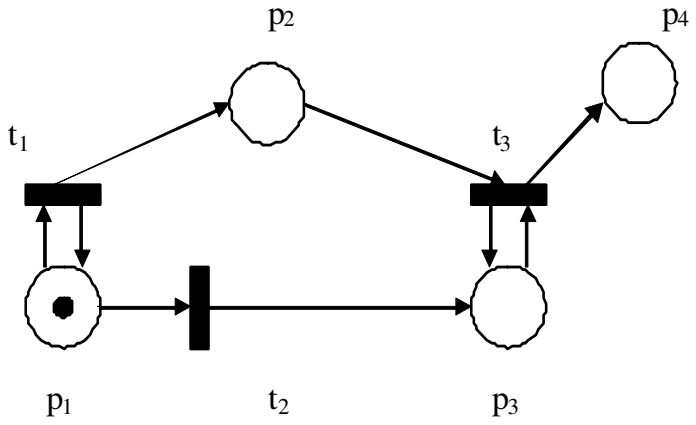


Рис 22. Сеть Петри

Заметим, что, если несколько компонентов покрывающей маркировки пропорциональны, между изменениями маркировки, получающимися в результате прохождения циклов, возможна взаимосвязь. Рассмотрим сеть Петри на рис. 22 и ее дерево достижимости, показанное на рис. 23. Согласно проведенному анализу, маркировка (0,14,1,7) покрывается в множестве достижимости. Путь, порождающий покрывающую маркировку, состоит из некоторого числа переходов t_1, за которыми следует переход t_2, после которого уже следует некоторое число переходов t_3. Задача заключается в определении того, сколько раз нужно запустить переходы t_1 и t_3. Так как мы хотим иметь в позиции p_1 14 фишек, а t_1 помещает в p_2 одну фишку, попытаемся выполнить 14 t_1. Однако нам необходимо выполнить 7 t_3, а каждый запуск t_3 удаляет из p_2 фишку, поэтому в действительности необходимо выполнить не менее 21 t_1 затем t_2 и после этого не менее 7 t_3 (выполнить t_3 такое число раз, чтобы в позиции p_2 осталось не менее 14 фишек). Существует алгоритм, определяющий минимальное число запусков переходов, необходимых для покрытия дайной маркировки.

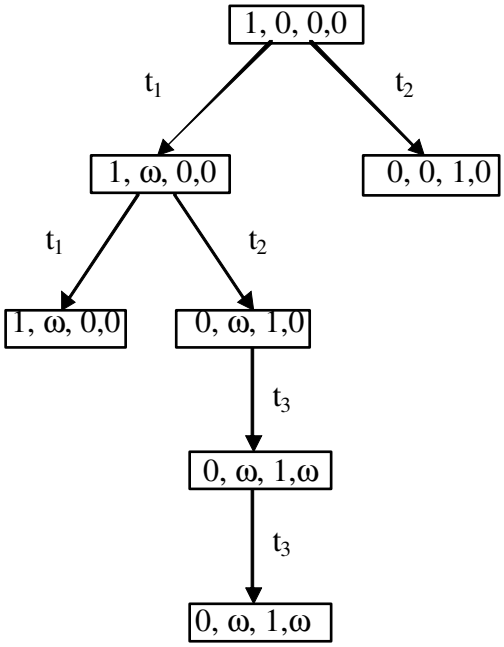


Рис.23. Дерево достижимости для сети Петри на рис.

Таким образом, дерево достижимости можно использовать для решения задач безопасности, ограниченности, сохранения и покрываемости. К сожалению, в общем случае его нельзя использовать для решения задач достижимости и активности, а также для определения возможной последовательности запусков. Решение этих задач ограничено существованием символа ω.

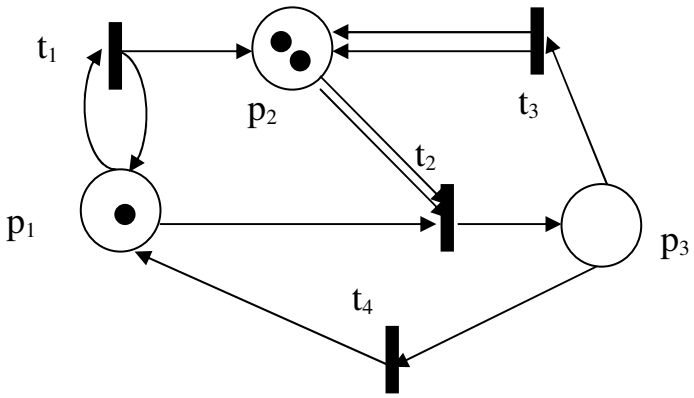


Рис. 24. Пример сети Петри

Символ ω означает потерю информации, конкретные количества меток отбрасываются, учитывается только существование их большого числа. Вместе с тем, в отдельных конкретных случаях дерево достижимости позволяет судить о свойствах достижимости и активности. Например, сеть, дерево достижимости которой содержит терминальную вершину, не активна. Аналогично искомая маркировка M' в задаче достижимости может

встретиться в дереве достижимости, что означает ее достижимость. Кроме того, если маркировка не покрывается некоторой вершиной дерева достижимости, то она недостижима.

В процессе функционирования сети Петри некоторые ее места могут накапливать неограниченное число фишек, что обозначается символом ω. Примером такого места может служить место p_2 в сети, изображенной на рисунке 24. Если интерпретировать места как накопители (буферы) данных, сигналов или деталей в моделируемых системах, то естественно потребовать, чтобы при любом варианте функционирования этих систем не происходило переполнение накопителей, которые в реальных ситуациях имеют конечную, фиксированную емкость.

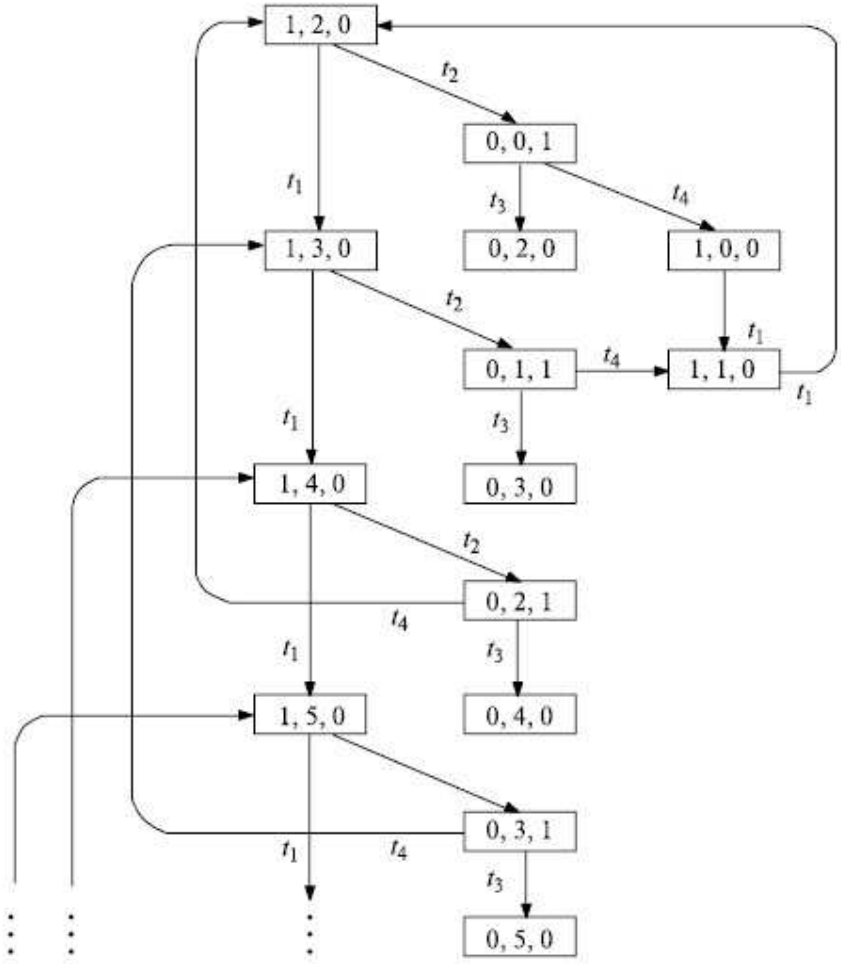


Рис. 25. Граф разметок сети Петри, изображенной на рис.24

Методы анализа на основе матричных уравнений

Проверка свойств сетей может выполняться путем построения и анализа множества достижимых состояний системы. Однако в случае большого количества состояний такой способ затруднителен. Другой способ представляет собой структурный анализ сети на основе заданной матрицы

инцидентности и начального маркирования. Динамика сети в пространстве состояний может быть описана рекуррентным уравнением:

$$M_k = M_{k-1} + A^* U_k; \qquad k = 1,2,...,n,$$

где M_k - состояние, которое следует после состояния M_{k-1} в результате k-го воздействия U_k; M_0 - начальное маркирование; A^* – матрица, полученная транспонированием матрицы инцидентности позиций и переходов, причем ее элементы a_{ji} равны n, $-n$, или 0, когда соответственно переход j имеет n исходящих дуг к позиции i, n входящих дуг в позицию i или не имеет связи с позицией i; U_k – управляющий вектор, компоненты которого $U_{jk} = \{0,1\}$. Если $U_{jk} = 1$, то в k-й момент асинхронного времени происходит срабатывание перехода t_j, если же $U_{jk} = 0$, то срабатывание не происходит.

Для решения рекуррентных уравнений смены состояний сетевых моделей используются методы линейной алгебры. Они позволяют на основе математического исследования структуры биграфа сети и начального маркирования M_0 оценить такие качественные характеристики сети как ограниченность, живость и т.д. Для исследования количественных характеристик сетей можно использовать различные методы теории графов и классического сетевого анализа.

Матричный подход основывается на представлении сети двумя матрицами D^- и D^+, представляющими входную и выходную функции сети. Каждая матрица имеет m строк (по одной на переход) и n столбцов (по одному на позицию). Определим матрицы $D^-(j,i) = K(p_i, I(t_j))$ и $D^+(j,i) = K(p_i, O(t_j))$. D^- описывает входы в переходы, D^+ – выходы из переходов, K – кратность позиции по входам и выходам.

Введём вектор $e(j)$ размерности m, содержащий нули везде, за исключением j-й компоненты. В этом случае переход t_j представляется m – вектором $e(j)$.

Тогда переход t_j в маркировке M разрешён, если $M \geq e(j)D^-$, а результат запуска перехода t_j в маркировке M обозначим функцией $\delta(M, t_j)$ определяющей новую маркировку M':

$$\delta(M, t_j) = M - e(j)D^- + e(j)D^+ = M + e(j)(-D^- + D^+) = M + e(j)D,$$

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

Тогда для последовательности запусков переходов $G = \{t_{j1}, t_{j2}, ..., t_{jk}\}$ имеем:

$$\delta(M, G) = \delta(M, t_{j_1}, t_{j_2}, ..., t_{jk}) = M + e(j_1)D + e(j_2)D + ... + e(j_k)D =$$
$$= M + [e(j_1) + e(j_2) + ... + e(j_k)]D = M + f(G)D .$$

Вектор $f(G) = e(j_1) + e(j_2) + ... + e(j_k)$ называется вектором запуска последовательности $t_{j1}, t_{j2}, ..., t_{jk}$. i-й элемент вектора $f(G)$ – это число запусков перехода t_j в последовательности $t_{j1}, t_{j2}, ..., t_{jk}$. Вектор запусков, следовательно, является вектором с неотрицательными целыми компонентами.

Рассмотрим задачу сохранения при известном векторе взвешивания меток W. Если M_0 – начальная маркировка, а M' – произвольная достижимая маркировка, необходимо, чтобы $M_0 W = M'W$. Тогда существует последовательность запусков переходов G, которая переводит сеть из M_0 в M'. Поэтому,

$$M' = \delta(M_0, G) = M_0 + f(G)D .$$

Следовательно, $M_0 W = M'W = (M_0 + f(G)D)W = M_0 W + f(G)DW$. Исходя из условия сохранения, имеем: $f(G)DW = 0$.

Поскольку это должно быть верно для всех $f(G)$, имеем $DW = 0$.

Таким образом, сеть Петри является сохраняющей тогда и только тогда, когда существует такой положительный вектор W, что $DW = 0$. Это обеспечивает простой алгоритм проверки сохранения, а также позволяет получать вектор взвешивания W для сохраняющей сети. Для этого нужно решить систему $DW = 0$ относительно W.

На рис.24 изображена сеть Петри, на примере которой поясним данные выше определения. В этой сети $P = \{p_1, p_2, p_3\}$, $T = \{t_1, t_2, t_3, t_4\}$.

Функция инцидентности задается с помощью следующих двух матриц D^- и D^+, представляющими входную и выходную функции сети., в которых на пересечении строки x и столбца y стоит кратность позиции по входам и выходам:

$$D^+ = \begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \\ t_4 \end{array} \begin{array}{c} \begin{array}{ccc} p_1 & p_2 & p_3 \end{array} \\ \begin{pmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 2 & 0 \\ 1 & 0 & 0 \end{pmatrix} \end{array} \qquad D^- = \begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \\ t_4 \end{array} \begin{array}{c} \begin{array}{ccc} p_1 & p_2 & p_3 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 \\ 1 & 2 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix} \end{array}$$

Начальная разметка M_0 задается следующим образом: $M_0(p_1)=1$, $M_0(p_2)=2$, $M_0(p_3)=0$, или в векторной форме: $M_0 = \begin{pmatrix} 1 \\ 2 \\ 0 \end{pmatrix}$.

При разметке M_0 могут сработать переходы t_1 и t_2, так как

$$M_0 = \begin{pmatrix} 1 \\ 2 \\ 0 \end{pmatrix} > D^-(t_1) = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

$$M_0 = \begin{pmatrix} 1 \\ 2 \\ 0 \end{pmatrix} \geq D^-(t_2) = \begin{pmatrix} 1 \\ 2 \\ 0 \end{pmatrix}$$

Переходы t_3 и t_4 не могут сработать, так как вектор начальной разметки M_0 не покрывает векторы $D^-(t_3) = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$, и $D^-(t_4) = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$.

В результате срабатывания перехода t_1 разметка M_0 сменяется на разметку (1, 3, 0),

$$M' = (1 \quad 2 \quad 0) + (1 \quad 0 \quad 0)(D^+ - D^-) = (1 \quad 2 \quad 0) + (1 \quad 0 \quad 0) *$$

$$* \left(\begin{pmatrix} 110 \\ 001 \\ 020 \\ 100 \end{pmatrix} - \begin{pmatrix} 100 \\ 120 \\ 001 \\ 001 \end{pmatrix} \right) = (120) + (1000) \begin{pmatrix} 0 & 1 & 0 \\ -1 & -2 & 1 \\ 0 & 2 & -1 \\ 1 & 0 & -1 \end{pmatrix} =$$

$$= (1 \quad 2 \quad 0) + (0 \quad 1 \quad 0) = (1 \quad 3 \quad 0).$$

В результате срабатывания перехода t_2 разметка M_0 сменяется на разметку (0, 0, 1) .

$$M' = (1 \quad 2 \quad 0) + (1 \quad 0 \quad 0)(D^+ - D^-) = (1 \quad 2 \quad 0) + (0 \quad 1 \quad 0) *$$

$$* \left(\begin{pmatrix} 110 \\ 001 \\ 020 \\ 100 \end{pmatrix} - \begin{pmatrix} 100 \\ 120 \\ 001 \\ 001 \end{pmatrix} \right) = (120) + (0100) \begin{pmatrix} 0 & 1 & 0 \\ -1 & -2 & 1 \\ 0 & 2 & -1 \\ 1 & 0 & -1 \end{pmatrix} =$$

$$= (1 \quad 2 \quad 0) + (-1 \quad -2 \quad 1) = (0 \quad 0 \quad 1).$$

Обе новые разметки непосредственно следуют после M_0 в рассматриваемой сети. Можно представить возможные изменения разметок сети N, происходящие в результате срабатывания ее переходов, в виде графа разметок – ориентированного графа, множество вершин которого образовано множеством $R(N)$ достижимых в сети N разметок. Из вершины M в вершину M' ведет дуга, помеченная символом перехода t_2, если и только если $M' \triangleright M'$. На рис.25 показан начальный фрагмент графа разметок сети, изображенной на рис.24. Этот граф бесконечен, так как множество $R(N)$ достижимых разметок бесконечно для рассматриваемой сети.

Разметка $M \in R(N)$ называется тупиковой, если в сети N не существует ни одного перехода, который может сработать при этой разметке. Для рассматриваемой сети тупиковыми являются разметки (0, 2, 0), (0, 3, 0), (0,4,0),..., (0, n, 0)...

Легко видеть, что если выделить путь по дугам графа разметок, начинающийся в вершине M и заканчивающийся в вершине M', и выписать подряд все встречающиеся символы переходов, то полученное слово образует последовательность срабатываний, ведущих от M к M'. Множество всех слов, которые получены выписыванием символов переходов вдоль путей, начинающихся в M_0, образует множество последовательностей срабатываний сети, или ее свободный язык. Так, язык рассматриваемой сети включает слова

$$\{\in, t_1, t_2, t_1t_1, t_1t_2, t_2t_3, t_2t_4, t_1t_1t_1, t_1t_1t_2, t_1t_2t_3, t_1t_2t_4, t_2t_4t_1, t_1t_1t_1t_1, t_1t_1t_1t_2, ...\}$$

Матричная теория является инструментом для решения проблемы достижимости. Пусть маркировка M' достижима из M_0, тогда существует последовательность (возможно пустая) запусков переходов G, которая

приводит из M_0 к M'. Это означает, что $f(G)$ является неотрицательным целым решением следующего матричного уравнения для X:

$$M' = M_0 + X \cdot D \ . \qquad (1)$$

Следовательно, если M' достижима из M_0, тогда уравнение (1) имеет решение в неотрицательных целых. Если уравнение не имеет решения, тогда M' недостижима из M_0.

Покажем возможности применения уравнения (1) для анализа сети, приведённой на рисунке 26.

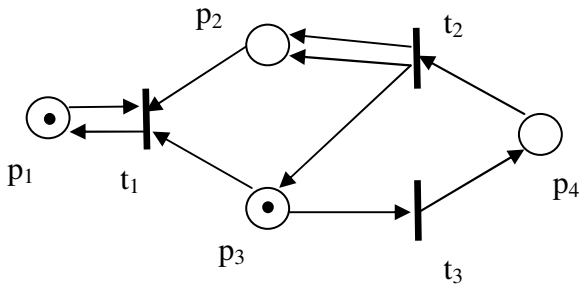


Рис. 26. Пример сети Петри

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

$$D = D^+ - D^- = \begin{pmatrix} 0 & -1 & -1 & 0 \\ 0 & 2 & 1 & -1 \\ 0 & 0 & -1 & 1 \end{pmatrix}$$

В начальной маркировке $M_0 = (1 \quad 0 \quad 1 \quad 0)$ переход t_3 разрешён и приводит к маркировке M', где:

$$M' = (1 \quad 0 \quad 1 \quad 0) + (0 \quad 0 \quad 1) * \begin{pmatrix} 0 & -1 & -1 & 0 \\ 0 & 2 & 1 & -1 \\ 0 & 0 & -1 & 1 \end{pmatrix} = (1 \quad 0 \quad 1 \quad 0) + =$$

$$+ (0 \quad 0 \quad -1 \quad 1) = (1 \quad 0 \quad 0 \quad 1)$$

Последовательность $G=\{t_3,t_2,t_3,t_2,t_1\}$ представляется вектором запусков $f(G)=(1,2,2)$ и получает маркировку M'':

$$M''=(1\quad 0\quad 1\quad 0)+(1\quad 2\quad 2)*\begin{pmatrix}0 & -1 & -1 & 0\\ 0 & 2 & 1 & -1\\ 0 & 0 & -1 & 1\end{pmatrix}=(1\quad 0\quad 1\quad 0)+$$

$$+(0\quad 3\quad -1\quad 0)=(1\quad 3\quad 0\quad 0)$$

Для определения того, является ли маркировка (1, 8, 0, 1) достижимой из маркировки (1, 0, 1, 0), имеем уравнение:

$$(1\quad 8\quad 0\quad 1)=(1\quad 0\quad 1\quad 0)+X*\begin{pmatrix}0 & -1 & -1 & 0\\ 0 & 2 & 1 & -1\\ 0 & 0 & -1 & 1\end{pmatrix}$$

или

$$(0\quad 8\quad -1\quad 1)=X*\begin{pmatrix}0 & -1 & -1 & 0\\ 0 & 2 & 1 & -1\\ 0 & 0 & -1 & 1\end{pmatrix}$$

Получаем систему

$$\begin{cases}0\cdot x_1+0\cdot x_2+0\cdot x_3=0\\ -1\cdot x_1+2\cdot x_2+0\cdot x_3=8\\ -1\cdot x_1+1\cdot x_2+-1\cdot x_3=-1\\ 0\cdot x_1-1\cdot x_2+1\cdot x_3=1\end{cases}$$

которая имеет решение $X=(0,4,5)$. Это соответствует последовательности $G=\{t_3,t_2,t_3,t_2,t_3,t_2,t_3,t_2,t_3\}$.

Можно показать, что маркировка (1, 7, 0, 1) недостижима из маркировки (1, 0, 1, 0) поскольку матричное уравнение:

$$(1\quad 7\quad 0\quad 1)=(1\quad 0\quad 1\quad 0)+X*\begin{pmatrix}0 & -1 & -1 & 0\\ 0 & 2 & 1 & -1\\ 0 & 0 & -1 & 1\end{pmatrix}$$

$$\begin{pmatrix} 0 & 7 & -1 & 1 \end{pmatrix} = X * \begin{pmatrix} 0 & -1 & -1 & 0 \\ 0 & 2 & 1 & -1 \\ 0 & 0 & -1 & 1 \end{pmatrix}$$

не имеет решения.

Недостатки матричных методов.

1. Матрица D теряет информацию о ситуациях, когда переходы имеют входы и выходы из одной позиции (петли).
2. Отсутствие информации о последовательности в векторе запуска. Хотя и известно число переходов, порядок их запуска неизвестен.
3. Решение уравнения (1) является необходимым для достижимости, но недостаточным.

2. **СП-модели параллельных вычислительных систем**

В качестве примеров рассмотрим сети Петри, описывающие некоторые структуры параллельных вычислительных систем в соответствии с классификацией Флинна.

ОКОД (один поток команд – один поток данных). Представителями подобной организации архитектуры являются обычные ЭВМ фон Неймана, ЭВМ CDC 6600, ЭВМ CDC 7600 с конвейерной арифметикой, ЭВМ AMDAHL 470/6 с конвейерной обработкой команд. На рис.27,а изображен синхронный конвейерный процессор с архитектурой типа ОКОД, а на рис.28,а – асинхронный конвейерный процессор с архитектурой аналогичного типа.

СП, моделирующая синхронный конвейерный процессор, представлена на рис.27,б. При этом позиции и переходы получили следующую интерпретацию:

t_{1k} - синхронизация приема данных процессорными элементами;

t_{i2} - начало работы ii-го ПЭ (процессорного элемента);

t_{i3} - окончание работы i-го ПЭ;

t_{02} - запись результатов работы конвейерного процессора в ОП (общую память);

p_{i1} - признак готовности входных данных для i-го ПЭ;

p_{i2} - признак приема входных данных i-м ПЭ;

p_{i3} - признак протекания процесса обработки данных i-м ПЭ;

p_{01} - признак готовности процессора для записи результатов в ОП;

p_{02} - признак окончания цикла обработки данных на конвейерном процессоре.

СП-структура, моделирующая асинхронный конвейерный процессор с промежуточными буферами, представлена на рис.28,б. Интерпретация вершин СП следующая:

t_{i1} - считывание данных i-го ПЭ из промежуточного буфера;

t_{i2} - обработка данных i-м ПЭ;

t_{i3} - запись результатов работы i-го ПЭ в промежуточный буфер;

t_{01} - пересылка данных из ОП во входной буфер 1-го ПЭ;

t_{02} - запись результатов работы конвейерного процессора в ОП;

p_{i1} - признак нахождения данных во входном буфере i-го ПЭ;

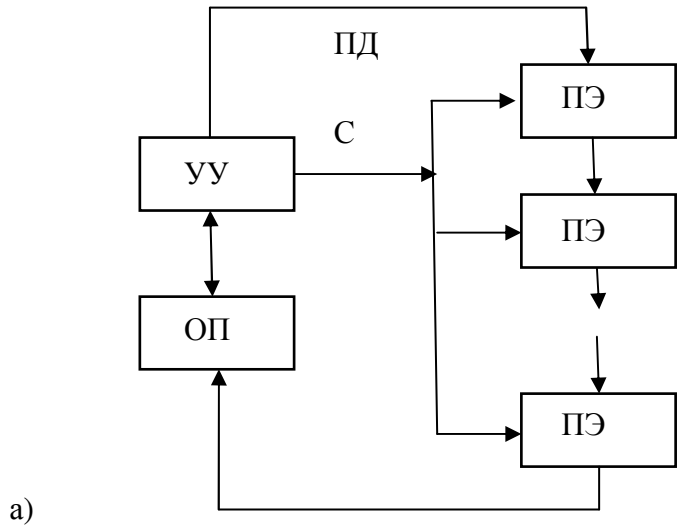
p_{i2}, p_{i3} - физический смысл данных позиций аналогичен смыслу позиций СП-структуры синхронного процессора;

p_{i4} - признак готовности i-го ПЭ для считывания новых данных;

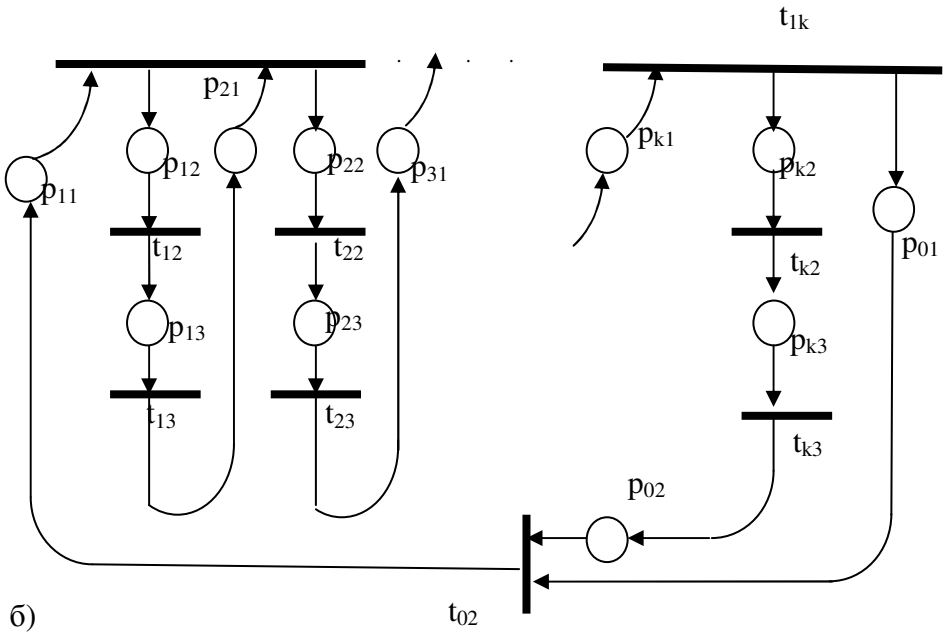
$p_{0i(i+1)}$ - признак незанятости буфера между i-м и i+1-м ПЭ;

p_{01} - моделирование шины обмена данными с ОП;

p_{01} - признак окончания цикла обработки данных на конвейерном процессоре.



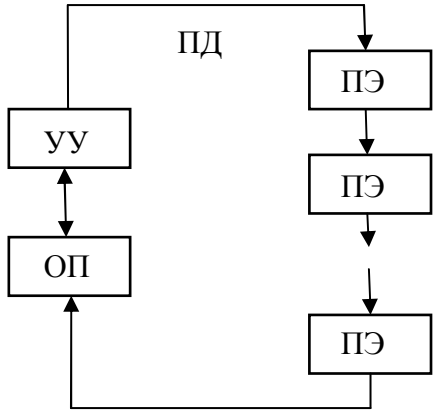
а)



б)

Рис.27. Синхронный конвейерный процессор с архитектурой ОКОД: структурная схема (а), СП-модель (б)

а)



б)

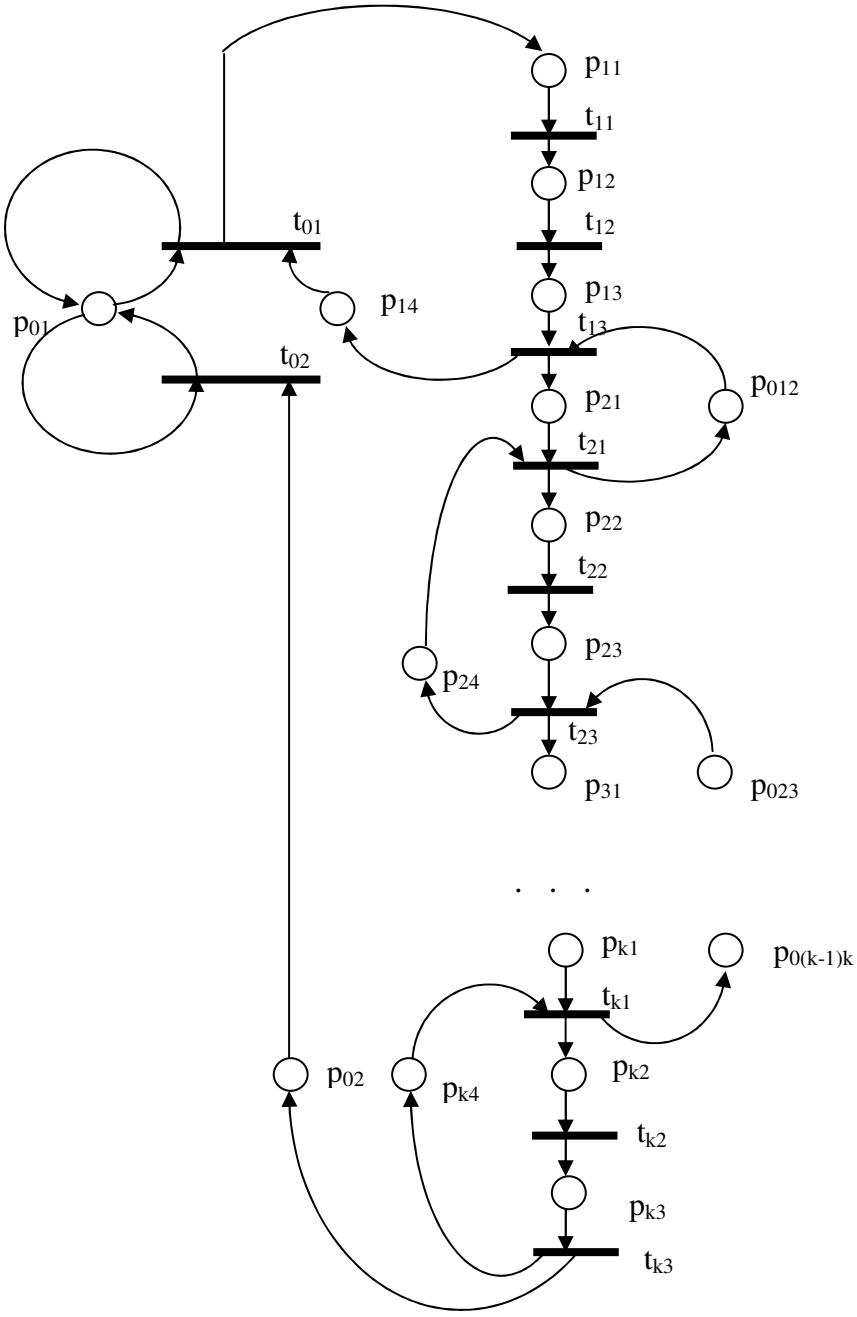


Рис.28. Асинхронный конвейерный процессор с архитектурой ОКОД: структурная схема (а), СП-модель (б)

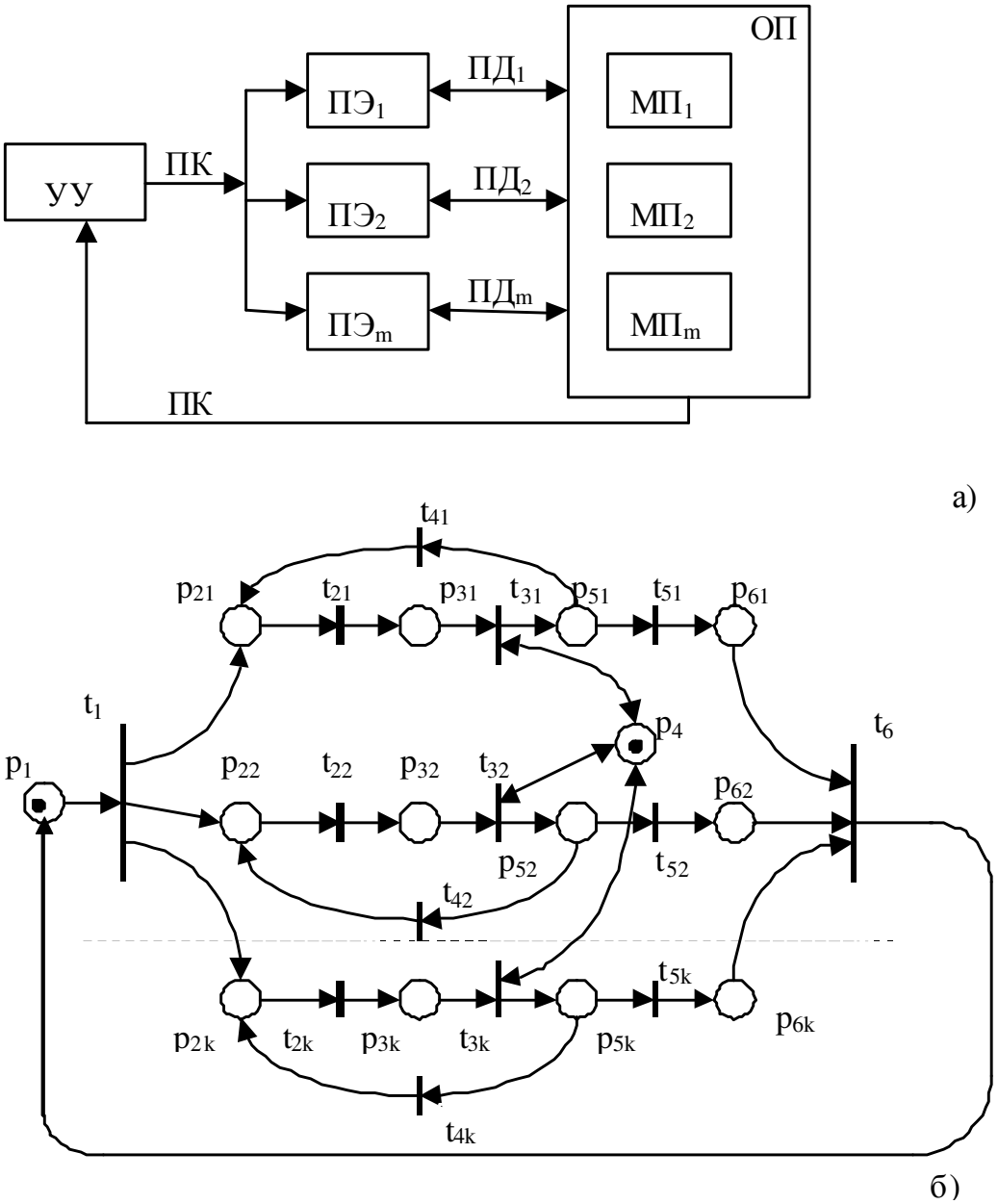


Рис. 29. Процессор с SIMD-архитектурой: структурная схема (а), СП-модель (б)

ОКМД (один поток команд – множественный поток данных или SIMD-архитектура). На рис.29,а представлен процессор с SIMD-архитектурой (один поток команд – множество потоков данных), где УУ – устройство управления, ПЭ – процесссорный элемент, МП – модуль памяти, ОП – общая память, ПК – поток команд, ПД – поток данных. Типичными представителями SIMD-архитектуры являются ЭВМ CRAY-1 (конвейерная векторизация), ILLIAC IV (процессорная матрица), параллельный процессор (MPP) НАСА, распределенный матричный процессор (DAP) фирмы ICL, многопроцессорная ВС "Эльбрус-1". Команды и данные в этих системах хранятся в глобальной оперативной памяти. Центральное устройство управления направляет операции всем ПЭ, связывая их глобальной сетью распространения и синхронизируя их работу.

СП-структура, моделирующая процессор с SIMD-архитектурой, представлена на рис.29,б. При этом позиции и переходы СП получили следующую интерпретацию:

t_1 - действия УУ при координации всех ПЭ;

t_{2i} - функционирование i-го ПЭ при обработке команд от УУ;

t_{3i} - обращение i-го ПЭ к общей памяти;

t_{4i} - передача данных из общей памяти в локальную память i-го ПЭ;

t_{5i} - выработка признака окончания обработки данных i-м ПЭ;

t_6 - выработка признаков окончания обработки команды параллельным процессором;

p_1 - признак готовности параллельного процессора к обработке команды;

p_{2i} - признак поступления данных на i-й ПЭ;

p_{3i} - признак запроса данных из общей памяти i-м ПЭ;

p_4 - арбитр, координирующий обращение ПЭ к участкам общей памяти;

p_{5i} - признак окончания считывания данных из общей памяти i-м ПЭ;

p_{6i} - признак окончания обработки данных i-м ПЭ.

МКМД (множественный поток команд – множественный поток данных или MIMD-архитектура). На рис.30,а представлен процессор с MIMD-архитектурой. Представителями MIMD-архитектуры являются ЭВМ Cray X-MP, Cray-2,3,4, Cyber-205, ETA-CF10, МАРС-М. СП, моделирующая процессор с MIMD-архитектурой, представлена на рис.30,б. При этом позиции и переходы СП получили следующую интерпретацию:

t_{i1} - выдача i-м УУ потока команд;

t_{i2} - прием i-м ПЭ команды из потока команд;

t_{i3} - чтение i-м ПЭ данных из ОП, выполнение команды и запись результатов в МП;

t_{i4} - подготовка i-го ПЭ к приему очередной команды из потока;

t_{i5} - подготовка i-го ПЭ к приему нового потока команд;

p_{i1} - признак, сигнализирующий i-му УУ о начале передачи потока команд;

p_{i2} - признак поступления на вход i-го ПЭ команды из входного потока;

p_{i3} - признак готовности i-го ПЭ к выполнению операции обмена данными с МП;

p_{i4} - признак окончания обработки команды i-м ПЭ;

p_{01} - арбитр, координирующий обращение к ОП параллельных процессов.

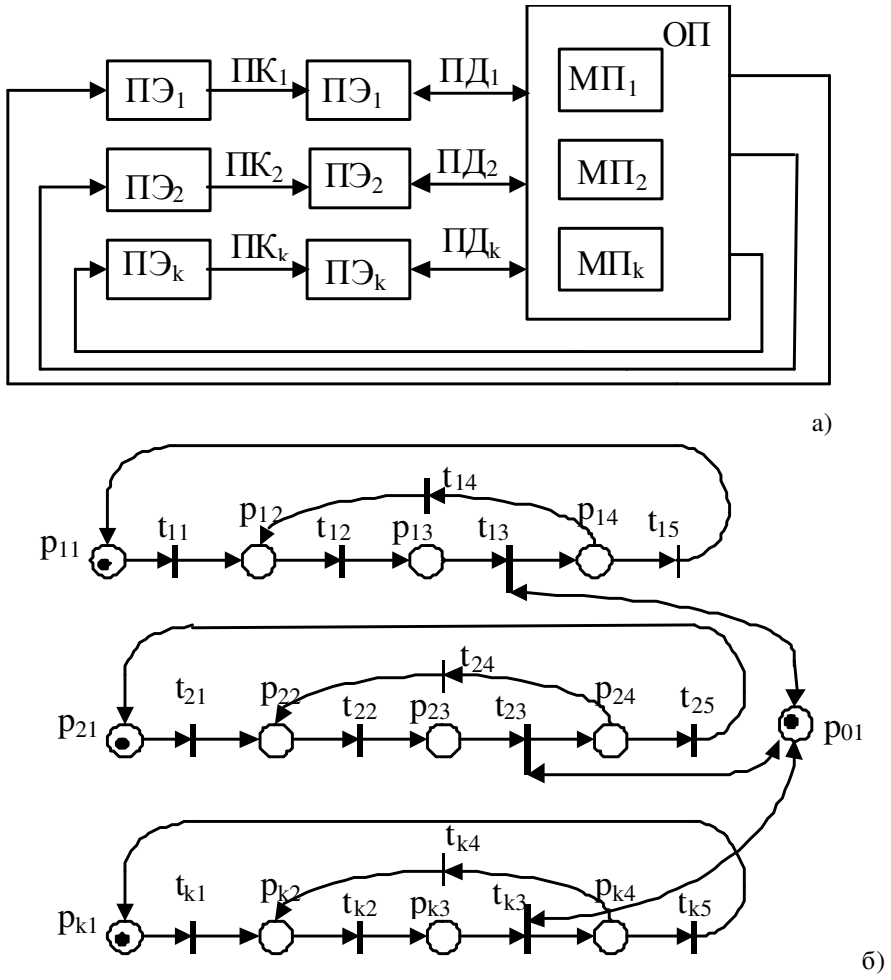


Рис. 30. Процессор с MIMD-архитектурой: структурная схема (а); СП-модель (б)

Матричное описание сетей Петри

Рассмотрим использование данных способов описания СП на примере ВС, состоящей из двух вычислительных устройств (ВУ), одно из которых главное (ВУ1), второе – подчиненное (ВУ2). Рабочий цикл ВУ складывается из трех этапов: начало работы *(BEGIN),* прием или посылка сообщений *(INIT),* окончание работы *(END)*. На этапе *BEGIN* ВУ1 посылает устройству ВУ2 сигнал о начале работы и переходит в состояние ожидания ответа. ВУ2, получив сигнал о начале работы, переходит в активное состояние и выдает подтверждающее сообщение о готовности. ВУ1, получив подтверждающий сигнал, также переходит в активное состояние. На этом этап *BEGIN* заканчивается. Находясь в состоянии *INIT* , ВУ1 может либо передавать задания для выполнения устройству ВУ2, либо перейти в

состояние *END*. В свою очередь ВУ2 может также передавать устройству ВУ1 задания для обработки, но самостоятельно в состояние *END* перейти не может. Если ВУ1 (ВУ2) передало задание устройству ВУ2 (ВУ1) для обработки, то оно переходит в состояние ожидания. Только после того, как будет получен подтверждающий сигнал, ВУ1 (ВУ2) может выполнить действия по инициализации новых заданий. Инициатива по переходу в состояние *END* может исходить только от ВУ1. При этом ВУ1 посылает устройству ВУ2 сигнал о завершении работы и переходит в неактивное состояние.

Построим основанную на понятиях сетей Петри модель, которая описывает функционирование рассмотренной выше ВС. Подобные модели в дальнейшем будем называть СП-моделями. СП-модели системы ВУ1 и ВУ2 в соответствии с описанным выше примером представлены на рис.31 и рис.32. При этом позиции и переходы получили следующую интерпретацию.

Этап *BEGIN*:

p_{11} - ВУ1 находится в неактивном состоянии, но готово перейти в состояние BEGIN;

p_{14} - ВУ1 ожидает подтверждения о начале работы от ВУ2;

p_{15} - ВУ1 послало сигнал устройству ВУ2 о начале совместной работы;

p_{16} - ВУ1 получило подтверждающий сигнал от ВУ2 о начале совместной работы;

t_{11} - переход ВУ1 в состояние BEGIN;

$N1$ - действия ВУ2 при переходе в состояние BEGIN;

t_1 - переход ВУ1 в состояние INIT;

p_{21} - ВУ2 находится в неактивном состоянии, но готово перейти в состояние BEGIN;

t_{21} - переход ВУ2 в состояние INIT.

Этап *INIT*:

p_{12} - ВУ1 готово послать сообщение устройству ВУ2;

p_{13} - ВУ1 готово принять сообщение от ВУ2;

p_{17} - ВУ1 ожидает подтверждения от ВУ2 о приеме сообщения;

p_{18} - ВУ1 послало устройству ВУ2 сообщение на обработку;

p_{19} - ВУ1 приняло от ВУ2 подтверждение о приеме посланного сообщения;

t_{13} - переход ВУ1 в состояние ожидания после передачи сообщения устройству ВУ2 на обработку;

t_{14} - переход ВУ1 в состояние готовности для передачи ВУ2 следующего сообщения;

$N2$ - действия ВУ2 при обработке принятого от ВУ1 сообщения;

p_{100} - ВУ1 обрабатывает принятое от ВУ2 сообщение;

t_{15} - прием устройством ВУ1 сообщения от ВУ2;

t_{16} - окончание обработки ВУ1 принятого от ВУ2 сообщения;

p_{22} - ВУ2 готово принять сообщение от ВУ1;

p_{23} - ВУ2 готово послать сообщение устройству ВУ1;

p_{24} - ВУ2 обрабатывает принятое от ВУ1 сообщение;

p_{25} - ВУ2 послало сообщение устройству ВУ1;

p_{26} - ВУ2 приняло от ВУ1 подтверждение о приеме посланного сообщения;

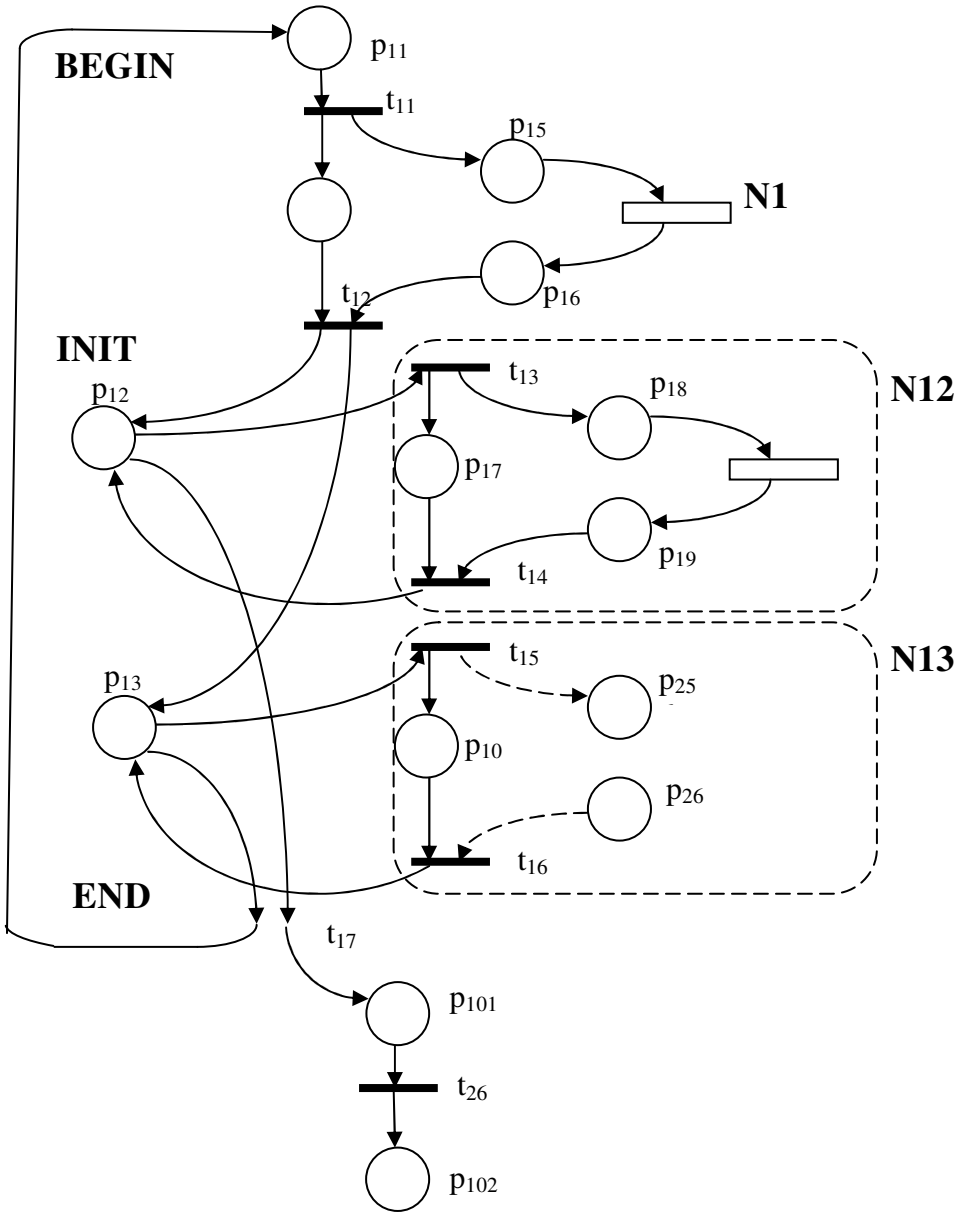


Рис. 31. СП-модель ВУ1

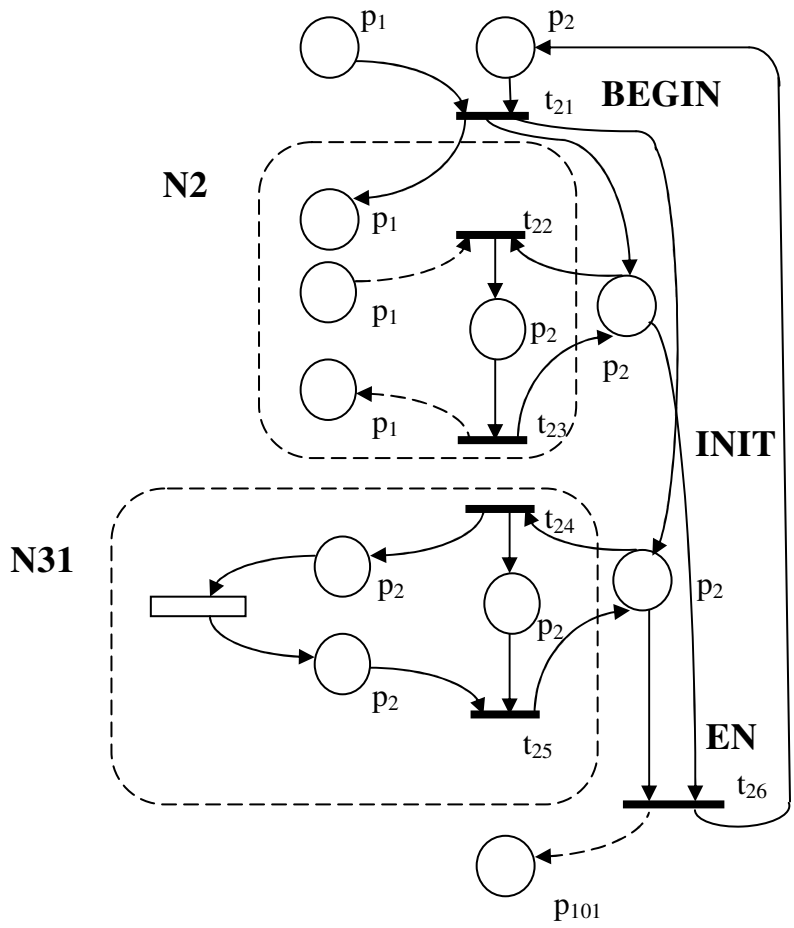


Рис. 32. СП-модель ВУ2

p_{27} - ВУ2 ожидает подтверждения от ВУ1 о приеме посланного сообщения;

t_{22} - прием ВУ2 сообщения от ВУ1;

t_{23} - окончание обработки ВУ2 принятого от ВУ1 сообщения;

t_{24} - переход ВУ2 в состояние ожидания после передачи сообщения устройству ВУ1 на обработку;

t_{25} - переход ВУ2 в состояние готовности для передачи устройству ВУ1 следующего сообщения;

$N13$ - действия ВУ1 при обработке принятого от ВУ2 сообщения.

Этап *END*:

p_{101} - ВУ1 послало сообщение устройству ВУ2 об окончании совместной работы;

p_{102} - система ВУ завершила свою работу;

t_{17} - завершение работы ВУ1;

t_{26} - завершение работы ВУ2.

Матричное описание данных СП-моделей представлено ниже. Связь ВУ1 и ВУ2 в СП отображена наличием переходов N1, N2, N13. Использование ИСП позволяет исследовать работу ВУ1 (или ВУ2) независимо от ВУ2 (ВУ1). Если объединить СП, представленные на рис.31 и 32, то можно будет исследовать совместную работу ВУ1 и ВУ2.

Матричное описание ВУ1

Матрица O(p,t)

	t_{11}	N1	t_{12}	t_{13}	N2	t_{14}	t_{15}	t_{16}	t_{17}	t_{26}
p_{11}	1	0	0	0	0	0	0	0	0	0
p_{14}	0	0	1	0	0	0	0	0	0	0
p_{15}	0	1	0	0	0	0	0	0	0	0
p_{16}	0	0	1	0	0	0	0	0	0	0
p_{12}	0	0	0	1	0	0	0	0	1	0
p_{17}	0	0	0	0	0	1	0	0	0	0
p_{18}	0	0	0	0	0	1	0	0	0	0
p_{19}	0	0	0	0	1	0	0	0	0	0
p_{13}	0	0	0	0	0	1	0	0	0	0
p_{100}	0	0	0	0	0	0	1	0	1	0
p_{101}	0	0	0	0	0	0	0	1	0	0
p102	0	0	0	0	0	0	0	0	0	1

Матрица I(t,p)

	p_{11}	p_{14}	p_{15}	p_{16}	p_{12}	p_{17}	p_{18}	p_{19}	p_{13}	p_{100}	p_{101}	p_{102}
t_{11}	0	1	1	0	0	0	0	0	0	0	0	0
N1	0	0	0	1	0	0	0	0	0	0	0	0
t_{12}	0	0	0	0	1	0	0	0	1	0	0	0
t_{13}	0	0	0	0	0	1	1	0	0	0	0	0
N2	0	0	0	0	0	0	0	1	0	0	0	0
t_{14}	0	0	0	0	1	0	0	0	0	0	0	0
t_{15}	0	0	0	0	0	0	0	0	0	1	0	0
t_{16}	0	0	0	0	0	0	0	0	1	0	0	0
t_{17}	1	0	0	0	0	0	0	0	0	0	1	0
t_{26}	0	0	0	0	0	0	0	0	0	0	0	1

Вектор начальной разметки M_0

p_{11}	p_{14}	p_{15}	p_{16}	p_{12}	p_{17}	p_{18}	p_{19}	p_{13}	p_{100}	p_{101}	p_{102}
1	0	0	0	0	0	0	0	0	0	0	0

Матричное описание ВУ2

Матрица O(p,t)

	t_{21}	t_{22}	t_{23}	t_{24}	t_{25}	N13	t_{22}
p_{21}	1	0	0	0	0	0	0
p_{22}	0	0	1	0	0	0	0
p_{24}	0	1	0	0	0	0	0
p_{23}	0	0	1	0	0	0	0
p_{25}	0	0	0	1	0	0	0
p_{26}	0	0	0	0	0	1	0
p_{27}	0	0	0	0	1	0	0

Матрица I(t,p)

	p_{21}	p_{22}	p_{24}	p_{23}	p_{25}	p_{26}	p_{27}
t_{21}	0	1	0	1	0	0	0
t_{22}	0	0	1	0	0	0	0
t_{23}	0	1	0	0	0	0	0
t_{24}	0	0	0	0	1	0	1
t_{25}	0	0	0	1	0	0	0
N13	0	0	0	0	0	1	0
t_{22}	1	0	0	0	0	0	0

Вектор начальной разметки M_0

p_{21}	p_{22}	p_{24}	p_{23}	p_{25}	p_{26}	p_{27}
1	0	0	0	0	0	0

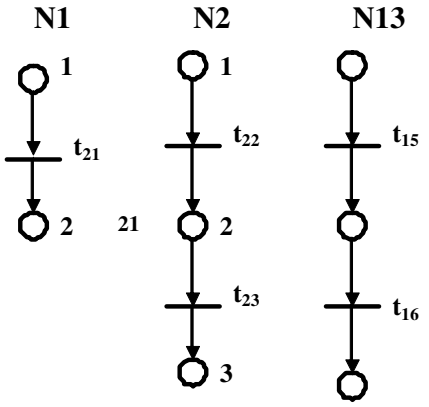


Рис.33. Структуры ИПР N1, N2, N13

Иногда бывает полезно отдельно изучить работу некоторого блока ВУ1 (или ВУ2). В этом случае исследуемый блок описывается как ИПР. В нашем примере ИПР N1, N2, N13, содержание которых представлено на рис.33, описывают работу блоков при переходе ВУ1 и ВУ2 в новое состояние и обработке принятых сообщений.

Для анализа взаимодействия отдельных блоков ВУ1 и ВУ2 на верхнем уровне ИСП, которые представлены на рис.31 и 32, могут быть описаны в виде, представленном на рис.33 и рис.34. Подобное представление абстрагируется от внутреннего содержания ИПР N1, N2, N11, N12, N13, N31, структура которых представлена на рис.33 и рис.36. Можно заметить, что данные ИПР включают в качестве составных частей другие ИПР (N1, N2, N13).

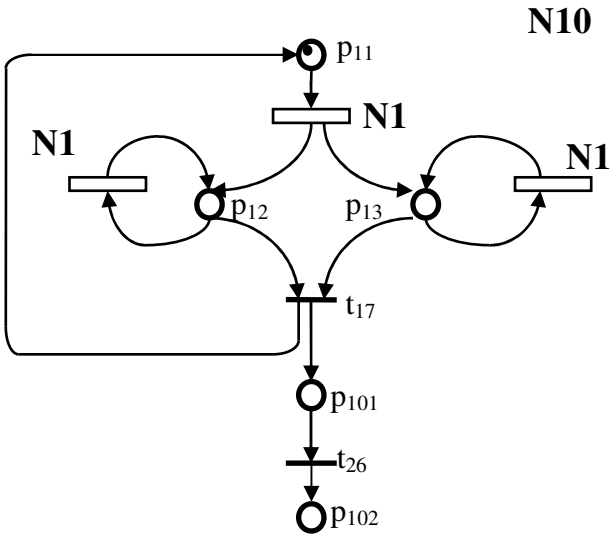


Рис.34. Иерархическая СП ВУ1

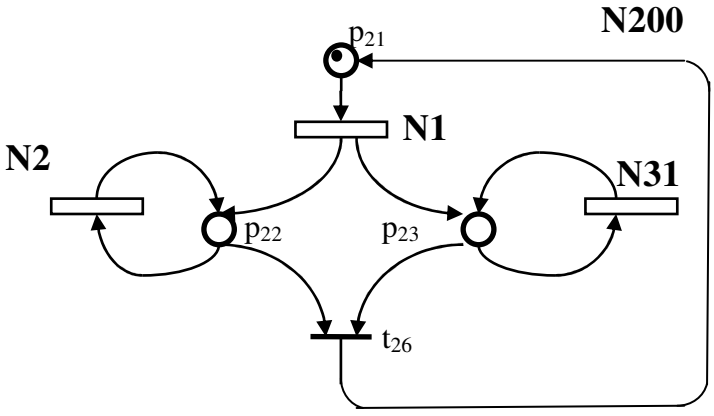


Рис.35. Иерархическая СП ВУ2

Приведенные примеры показывают, что применение ИСП дает возможность разработчику описывать и анализировать проектируемое устройство или систему на различных уровнях детализации.

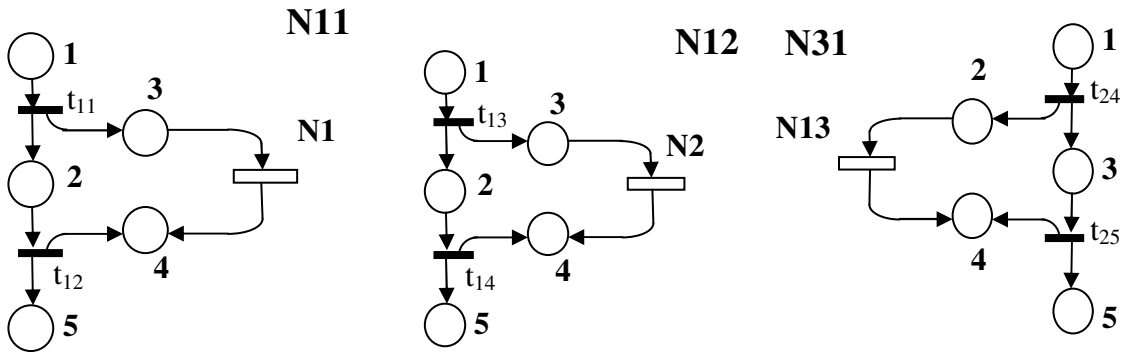


Рис.36. Структуры ИПР N11, N12, N31

Алгебраическое описание сетей Петри

Одной из наиболее актуальных задач в области создания инструментальных средств для анализа и синтеза параллельных ВС является задача разработки средств формального описания СП, моделирующих работу исследуемой ВС. Основным требованием, предъявляемым к формальной модели, является адекватность представления моделей ВС, для которых характерны параллелизм, асинхронность, недетерминированное взаимодействие процессов. Кроме того, модель должна обладать развитым арсеналом средств, пригодных для исследования и преобразования моделей ВС. Существенную помощь в построении подобного математического аппарата может оказать алгебраический подход к описанию СП.

В настоящее время предложено несколько алгебраических подходов для описания как простых, так и расширенных СП. В качестве примера подобного математического аппарата рассмотрим алгебру СП.

Алгебра сетей Петри базируется на алгебре *регулярных СП*. В целях увеличения описательных возможностей языка в алгебру регулярных СП введены операции сцепления и замещения, средства описания межуровневых связей, ингибиторных и взвешенных дуг, а также расширены возможности операции присоединения.

Определение данных операций дается с помощью вспомогательной операции *слияния позиций*. Пусть $X = \{x_1, x_2, ..., x_n\}$ и $Y = \{y_1, y_2, ..., y_m\}$ – два

сливаемых множества позиций. Если одно из множеств пустое, то результат слияния представляет собой исходную сеть. Операция слияния sl множеств X и Y происходит в два этапа. Сначала каждая позиция x_i, принадлежащая множеству X, копируется в m экземплярах, а каждая позиция y_j, принадлежащая множеству Y, копируется в n экземплярах. Позиции копируются вместе с их разметкой и инцидентными дугами. Затем каждая пара (x_i, y_j) замещается новой позицией z с разметкой $m(z) = \min(m(x_i), m(y_j))$ и соответствующими инцидентными дугами. Рассмотрим определения операций алгебры регулярных СП.

Операция наложения "**,**". Операция объединения двух сетей в одну выполняется путем наложения одноименных переходов:

$$N = (N1, N2) = (P1 \cup P2, T1 \cup T2, I1 \cup I2, O1 \cup o2, M),$$

$$\textit{где} \quad M(p) = M_i(p), \ \textit{если} \ p \in Pi, \textit{где} \ i = 1,2 \ \ u \quad p \in P1 \cap P2;$$

$$M(p) = \min(M1(p),, M2(p)) \ \ \textit{если} \quad p \notin P1 \cap P2.$$

Данная операция представляет собой теоретико-множественное объединение графов.

Операция итерации "".* Унарная операция, которая сливает множества головных и хвостовых позиций сети N:

$$N' = *(N) = sl(pre(N), post(N)).$$

Операция присоединения " **;** ". Данная операция соединяет две сети в одну путем слияния множества хвостовых позиций первой сети $N1$ со множеством головных позиций второй сети $N2$:

$$N = (N1; N2) = sl(post(N1), pre(N2)).$$

Операция исключения "$ *".* Эта операция объединяет две сети в одну путем слияния соответствующих множеств головных и хвостовых позиций данных сетей:

$$N = (N1\$N2) = sl(pre(N1), pre(N2))\$sl(post(N1), post(N2)).$$

Операция разметки " **>** *".* Унарная операция, помещает n меток в каждую позицию, принадлежащую множеству головных позиций сети N. Сеть N' совпадает как граф с сетью N, но имеет другую начальную разметку. Примеры использования описанных операций приведены на рис. 37.

Операция сцепления " + ". Ввод данной операции обусловлен следующим. Ввиду того, что алгебраическое представление СП не включает идентификаторы позиций, при выполнении операции наложения учитываются лишь одноименные переходы. Это приводит к тому, что при наложении СП, представленных формулами, в некоторых случаях появляются непредусмотренные позиции, которые искажают результирующую сеть. Для устранения данного недостатка вводится операция сцепления, которая осуществляет наложение одноименных переходов сетей и производит сцепление позиций с определенными свойствами. Свойства объединяемых позиций *x* и *y* при сцеплении фрагментов СП A1 и A2, содержащих одноименный переход *t*, определяются таблицами 3 и 4.

Таблица 3

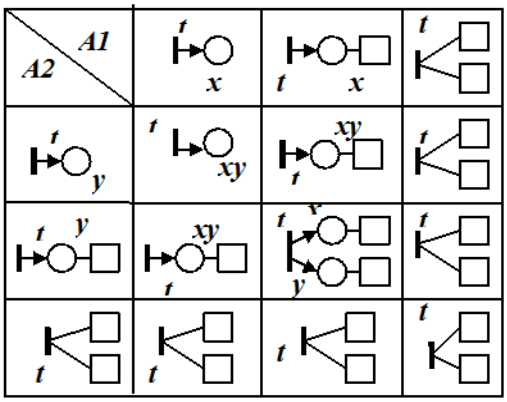
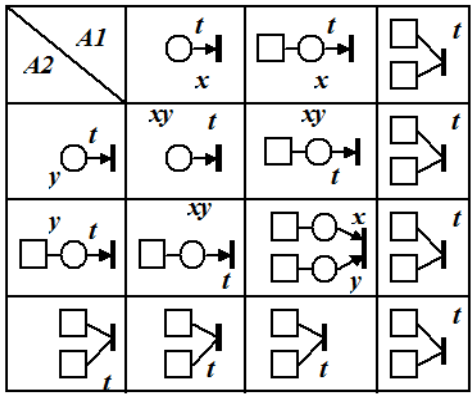


Таблица 4



Пример использования операции сцепления представлен на рис.36,ж).

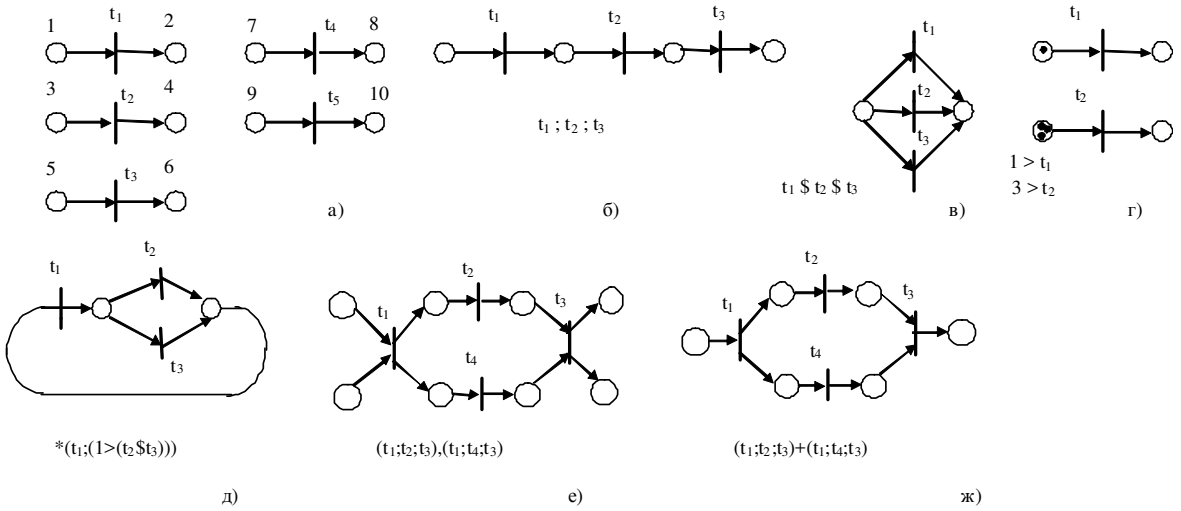


Рис. 37. Примеры выполнения операций алгебры сетей Петри: а) множество исходных элементарных сетей; б) присоединение; в) исключение; г) разметка; д) итерация; е) наложение; ж) сцепление

Операция замещения " - ". На практике часто возникает ситуация, когда необходимо оценить влияние некоторого блока на работу ВС в зависимости от места включения блока. Например, на рис.38, а приведена СП-модель с идентификатором $N100$. Данная СП-модель описывает ВС, которая имеет последовательные и параллельные процессы. Пусть необходимо оценить работу ВС в том случае, когда блок $Q10$ (рис.38,б) включается вместо последовательного процесса (например, вместо t_1, рис.39,а) и когда блок $Q10$ включается вместо параллельного процесса (например, вместо t_4, рис.39,б). Как будет выглядеть алгебраическое описание СП-модели в первом и во втором случаях? Исходная СП-модель ВС и блок $Q10$ имеют следующее описание:

$$N100 : *(1 > t_1 ; (t_4 \$ (t_2 ; t_3)); t_5), \quad (2)$$

$$Q10 : t_{21} \$ t_{22} \$ t_{23} . \quad (3)$$

ВС, в которой блок $Q10$ замещает последовательный процесс (t_1), имеет следующее алгебраическое описание:

$$N100 : *(1 > (t_{21} \$ t_{22} \$ t_{23}); (t_4 \$ (t_2 ; t_3)); t_5),$$

а ВС, в которой $Q10$ замещает параллельный процесс (t_4), имеет следующее описание: $N100 : *(1 > t_1 ; ((t_{21} \$ t_{22} \$ t_{23}) \$ (t_2 ; t_3)); t_5)$.

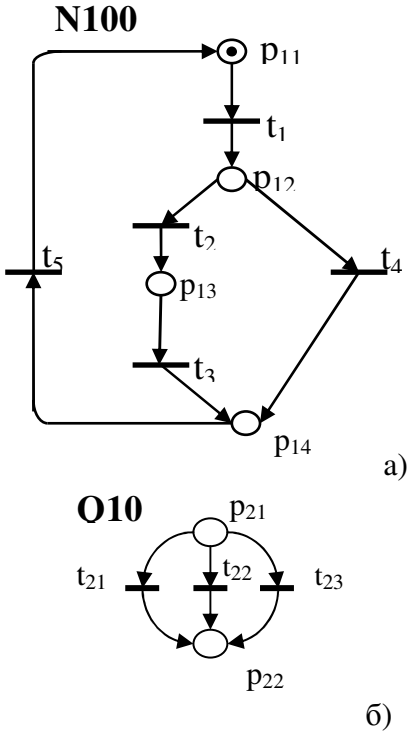


Рис.38. Пример исходной сети Петри (а) и включаемого блока (б)

Очевидно, что подобный подход к алгебраическому представлению СП-моделей приводит к дублированию описания ИПР, а следовательно, к повышению сложности входного описания.

Ликвидировать данный недостаток позволяет ввод дополнительной операции замещения. Правило записи данной операции имеет следующий вид: $\langle П1\rangle:\langle П2\rangle-\langle П3\rangle$, где $П1$ – идентификатор СП-модели, в которой производится замещение; $П2$ – идентификатор замещающего ИП; $П3$ – идентификатор замещаемого перехода.

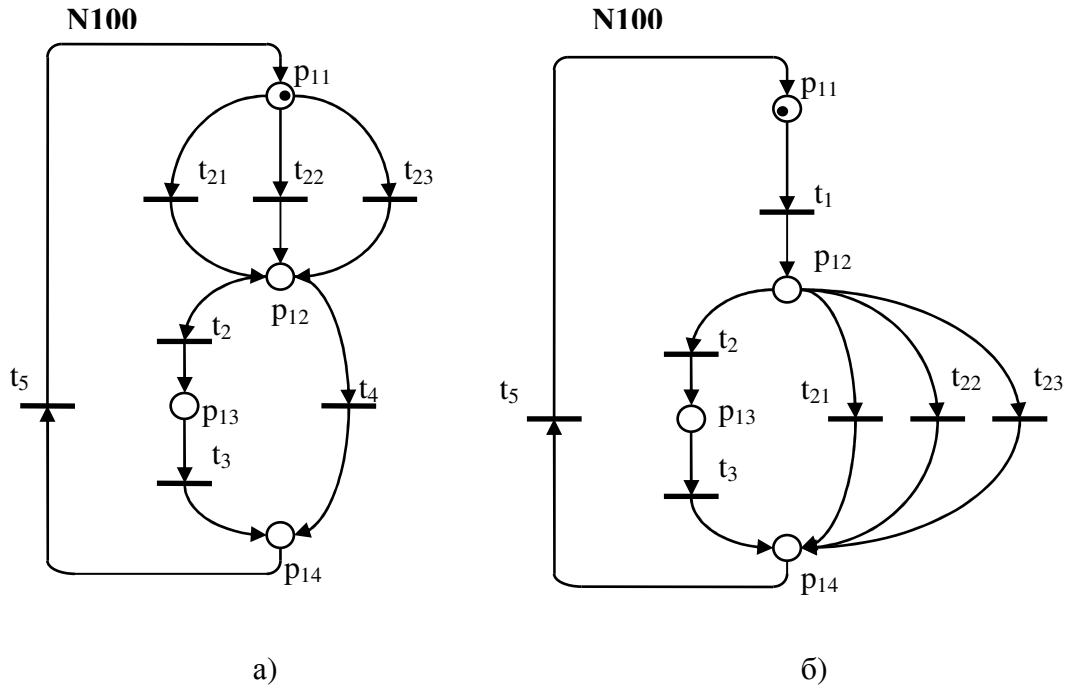
С учетом операции замещения СП-модели, представленные на рис. 39, будут иметь следующие описания:

$N100:Q10-t_1$,

$N100:Q10-t_4$,

где $N100$ и $Q10$ – описаны выражениями (2) и (3).

Расширение операции присоединения. Операции алгебры регулярных СП позволяют описывать ограниченные структуры СП. А если принять во внимание то, что подобные структуры встречаются в большинстве СП-моделей, то становится очевидным ограниченность операций алгебры регулярных СП при описании и исследовании СП-моделей.



а) б)

Рис.39. Результаты использования операции замещения: а) включение ИПР Q100 в последовательный процесс; б) включение ИПР Q100 в параллельный процесс

Пусть выполняется операция $A1;A2$, где ";" – операция присоединения. Если фрагмент $A1$ имеет хвостовые позиции, т.е. $|post(A1)|>0$, а фрагмент $A2$ имеет головные позиции, т.е. $|pre(A2)|>0$, то в этом случае операция присоединения выполняется, как описано выше.

Рассмотрим случай, когда фрагмент $A2$ не имеет головных позиций ($|pre(A2)|=0$), либо фрагмент $A1$ не имеет хвостовых позиций ($|post(A1)|=0$). Тогда при выполнении операции присоединения для фрагментов $A1$ и $A2$ определяются позиции, которые входят в *петли*. Позиция p входит в петлю, если она принадлежит множеству $pre(pre(p))$ (либо множеству $post(post(p))$.

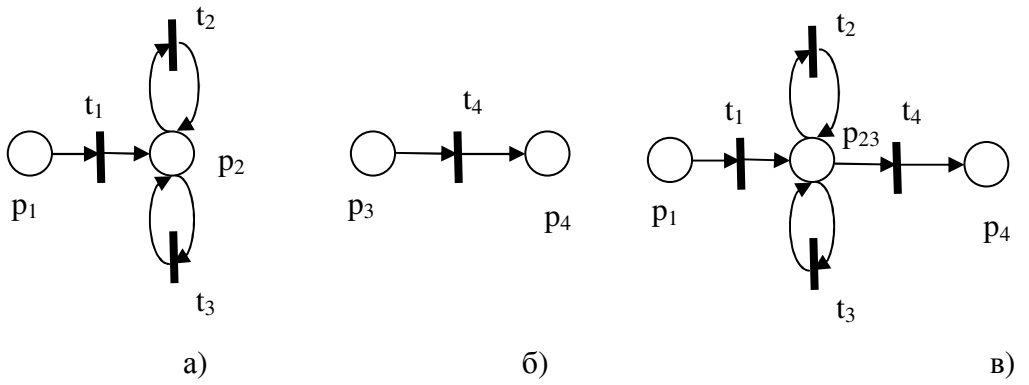


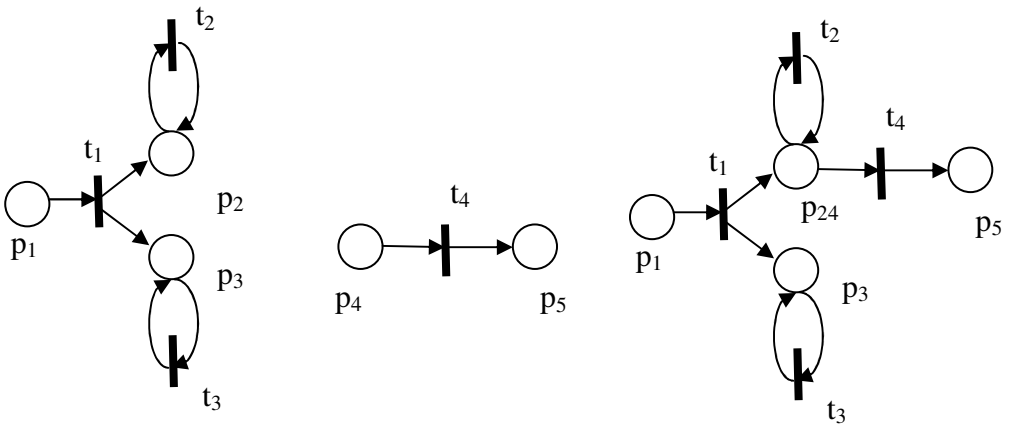
Рис. 40. Пример выполнения расширенной операции присоединения (A1;A2):
а) аргумент A1; б) аргумент A2; в) результат

Рассмотрим возможные варианты.

1. Если фрагмент $A1$ (рис.40,а) содержит одну позицию, входящую в петлю или петли, то операция присоединения $(A1;A2)$ выполняется путем объединения данной позиции и головной позиции фрагмента $A2$ (рис.40,б). Результат данной операции представлен на рис.40,в.
2. Если фрагмент $A1$ (рис.41,а) содержит несколько позиций, причем часть из них входит в петли, то операция присоединения $(A1;A2)$ выполняется путем объединения головной позиции фрагмента $A2$ с позицией фрагмента $A1$, входящей в одну из петель и имеющей наименьший порядковый номер. Результат данной операции представлен на рис.41,в.
3. В том случае, когда фрагмент $A1$ не имеет хвостовых позиций, а фрагмент $A2$ не имеет головных позиций, $|post(A1)|=0\,\&\,|pre(A2)|=0$, то операция присоединения (A1;A2) невыполнима.

Описание межуровневых связей. Для описания в СП-моделях межуровневых связей введем в алгебру сетей Петри составные имена перехо-

дов. Рассмотрим использование данных имен на примере. Пусть в СП-модели (рис.42) существует связь между переходами, которые принадле-



а) б) в)

Рис. 41. Пример выполнения расширенной операции присоединения(A1;A2):
а) аргумент A1; б) аргумент A2; в) результат

жат различным ИПР. Тогда с использованием составных имен данная связь может быть описана следующим образом:

$$N1:(Q1,Q2)+(tQ1_{12};tQ2_{21}),$$

где $Q11:t_{11}\$t_{12}$ и $Q2:t_{21};t_{22}$.

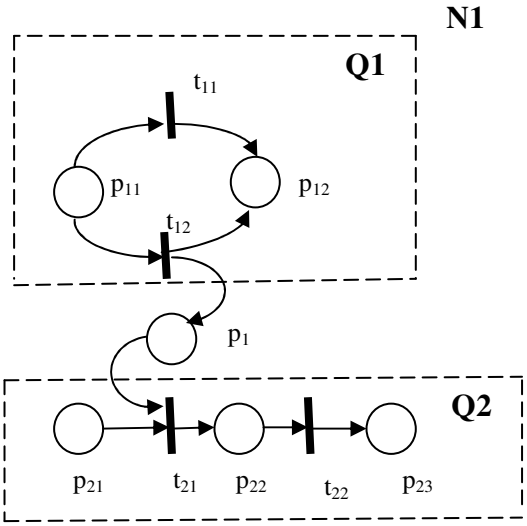
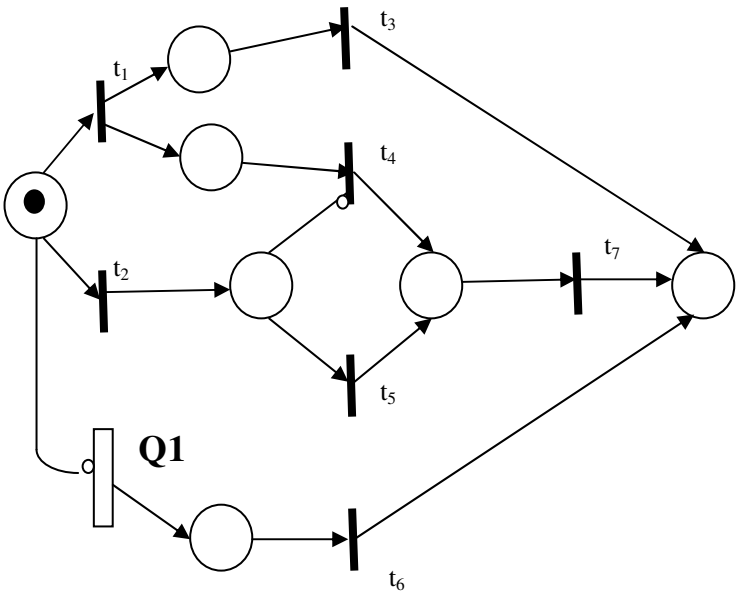


Рис. 42. Пример ИСП с межуровневыми связями

Описание ингибиторных дуг. Возможны ингибиторные дуги, запрещающие срабатывания перехода, если во входной позиции, связанной с переходом ингибиторной дугой, находится метка. Для алгебраического описания ингибиторных СП введем операцию задания ингибиторных дуг (^). Действие данной операции распространяется на один аргумент (унарная операция), в качестве которого может быть либо простой, либо иерар-

хический переходы. Пример описания ингибиторной иерархической сети приведен на рис.43.



$$1 > (((t_1;t_3)\$(t_2;(\hat{}\,t_4\$t_5);t_7)\$(\hat{}\,Q1;t_6)) + (t_1;t_4))$$

Рис. 43. Пример описания сети Петри с ингибиторными дугами

Описание взвешенных дуг. Для того чтобы иметь возможность описывать кратные дуги, встречающиеся в СП-моделях, вводятся операции задания весов входных и выходных дуг к переходам. Входная дуга к переходу t_i кратности r описывается с помощью оператора $g(rg(t_i))$, выходная - с помощью оператора $h(rh(t_i))$. Например, дуги СП, представленной на рис.44, могут быть описаны следующим образом: $2gt_1,3ht_1$ или $2g,3ht_1$.

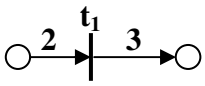


Рис. 44. Сеть Петри с взвешенными дугами

Язык алгебраического описания иерархических сетей Петри

Возможности алгебраического описания представляются входным языком СП, в основу которого положена алгебра СП. Предложения языка СП включают только идентификаторы переходов и выполняемых над ними операций. Идентификаторы позиций генерируются в процессе интерпретации предложений языка СП и являются величинами переменными.

Алфавит языка включает следующие классы символов:

буквы: N,T,Q;

специальные знаки: ";",":",",","$","+","*","-",">",".","#", "(", ")", "g", "h", "^";

цифры: 0,1,2,3,4,5,6,7,8,9;

пробел.

Грамматика языка СП представлена на рис.44. Длина идентификаторов сети, ИПР и простых переходов (ПП) не должна превышать шести позиций. Причем каждый идентификатор должен начинаться с буквы. Каждый ИПР в терминах языка СП может быть описан либо как ИПР Q-типа, либо как ИПР N-типа. Использование того или иного типа ИПР позволяет влиять на степень детализации ИСП при ее анализе. В качестве примера опишем на языке СП СП-модель системы вычислительных устройств, которая представлена на рис.31 и рис.32. Одно из возможных описаний СП-модели можно представить следующим образом:

$$N0 : *t_0 \tag{4}$$

$$Q11 : (t_{11}; t_{12}) + (t_{11}; t_{21}; t_{12}) \tag{5}$$

$$Q2 : (t_{22}; t_{23}) \tag{6}$$

$$Q12 : (t_{13}; t_{14}) + (t_{13}; Q2; t_{14}) \tag{7}$$

$$Q13 : (t_{15}; t_{16}) \tag{8}$$

$$Q31 : (t_{24}; t_{25}) + (t_{24}; Q13; t_{25}) \tag{9}$$

$$Q100 : *(1 > Q11; (*N12); t_{17}) + (Q11; (*N13); t_{17}; t_{26}) \tag{10}$$

$$Q200 : *(1 > t_{21}; (*N2); t_{26}) + (t_{21}; (*N31); t_{26}) \tag{11}$$

$$Q100 : Q12 - N12 \tag{12}$$

$$Q100 : Q13 - N13 \tag{13}$$

$$Q200 : Q2 - N2 \tag{14}$$

$$Q200 : Q31 - N31 \tag{15}$$

$$N1 : Q100 + Q200\# \tag{16}$$

Поясним данное описание. Выражение (4) не несет физического смысла и присутствует в описании СП-модели только из-за требований грамматики языка. Ввиду того, что ИПР N1 состоит из одного перехода t_{21}, то в алгебраическом описании данный переход замещен простым переходом t_{21}.

< иерархическая сеть >:: = < идентификатор сети > : < описание сети>#
< описание сети > :: = < выражение > < список ИПР > | < выражение >
| < список ИПР >
< список ИПР > :: = < список ИПР > | < ИПР >
< ИПР > :: = < идентификатор ИПР> : < описание сети >
< выражение > :: = < терм > < операция > < выражение > | < операция >
< выражение > | < терм >
< терм > :: = (< выражение >) | < идентификатор сети > |
< идентификатор ИПР > | < идентификатор перехода >
< операция > :: = <число > > | , | ; | $ | + | * | - | ^ |<число>q |<число>h
< идентификатор сети > :: = N < число >
< идентификатор ИПР> :: = Q < число >
< идентификатор перехода > :: = T < указатель перехода >
< указатель перехода > :: = < число > | N < число> . < число > | Q
< число > . < число >
< число > :: = < цифра > < число > | < цифра >
< цифра > :: = 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Рис. 45. Грамматика языка сетей Петри

Иерархические переходы $N2$ и $N13$ описаны выражениями (6) и (8) и представлены как ИПР Q-типа. Данный подход выбран с целью наиболее полного раскрытия моделирующей СП. Выражения (5), (7), (9) описывают взаимодействие ВУ1 и ВУ2 на этапах *BEGIN* и *INIT*. Данные выражения описывают отдельные подсистемы и соответствуют структурному подходу к проектированию сложных систем. Выражения (10) и (11) описывают иерархические СП, моделирующие работу ВУ1 и ВУ2. Выражения (12)-(15) путем замены ИПР N-типа на ИПР Q-типа повышают уровень детализации СП-модели и делают ее более подробной. Полное описание моделирующей сети $N1$, в которой не содержится ни одного ИПР, получается в результате выполнения операции (16).

Следует отметить, что рассмотренное описание ИСП $N1$ не является оптимальным с точки зрения времени выполнения. Действительно, в выражении (10) ИПР $Q11$ включен дважды, поэтому при выполнении операции сцепления будут просматриваться и объединяться все переходы, составляющие ИПР $Q11$, что может занимать лишнее время в случае сложного $Q11$. Точно такого же результата, но значительно быстрее, можно добиться, если некоторые выражения представить в следующем виде:

$$N11:(t_{11};t_{12})+(t_{11};t_{21};t_{12}) \quad (5)$$

$$Q100:*(1>N11;(*N12);t_{17})+(N11;(*N13);t_{17};t_{26}) \quad (10)$$

$$Q100:N11-Q11 \quad (17)$$

$$N1:Q100+Q200\# \quad (12)$$

В этом случае при выполнении операции сцепления в выражении (10) ИПР $N11$ не раскрывается и выступает в роли простого перехода. Поэтому ИПР $Q100$, описанный выражением (10), строится значительно быстрее, так как содержит меньшее количество переходов. После обработки выражения (17) в ИПР $Q100$ происходит замена ИПР $N11$ на ИПР $Q11$, в результате чего получается результат, аналогичный предыдущему описанию.

Литература:

1. Кулагин В.П. Моделирование структур параллельных вычислительных систем на основе сетевых моделей: Учебное пособие. - Москва: Московский государственный институт электроники и математики (технический университет), 1998. – 102 с.
2. Проститенко, О.В. Моделирование дискретных систем на основе сетей Петри: учебное пособие / О.В. Проститенко, В.И. Халимон, А.Ю. Рогов. – СПб.: СПбГТИ(ТУ), 2017. - 69 с.
3. Механизмы синхронизации процессов [Электронный ресурс] – URL: https://infopedia.su/10x5c9b.html
4. Основы информационных технологий [Электронный ресурс] – URL: https://studfiles.net/preview/5269953/page:42/
5.Сети Петри [Электронный ресурс] – URL: http://it.kgsu.ru/SetiPetri/oglav.html
6.Сети Петри [Электронный ресурс] – URL: http://5fan.ru/wievjob.php?id=99658

Электронное учебное издание

Неля Николаевна **Короткова**

МАТЕМАТИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ПРОГРАММНЫХ СИСТЕМ

Учебное пособие

Электронное издание сетевого распространения

Редактор Матвеева Н.И.

Темплан 2019 г. Поз. № 17.
Подписано к использованию 06.05.2019. Формат 60x84 1/16.
Гарнитура Times. Усл. печ. л. 4,06.

Волгоградский государственный технический университет.
400005, г. Волгоград, пр. Ленина, 28, корп. 1.

ВПИ (филиал) ВолгГТУ.
404121, г. Волжский, ул. Энгельса, 42а.