Теория вычислительных процессов и структур

Лекция №8. Сети Петри и программирование

Содержание лекции

Алгебра регулярных сетей

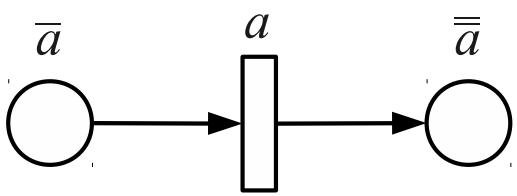
Сети Петри и семантика структур управления

Введение I

- Среди приложений теории сетей Петри к задачам моделирования дискретных систем наибольшее развитие получили работы, связанные с попытками использовать аппарат сетей Петри, их модификации и обобщения, *для описания и изучения структурной динамики параллельных программ*.
- На этой лекции мы познакомимся с подходом к применению сетей Петри для решения задач параллельного программирования, опирающийся на *алгебру регулярных сетей*, элементы которых интерпретируются специальным образом элементами параллельных программ.

Алгебра регулярных сетей I

- *Алгебра* регулярных сетей строится с помощью *операций* над сетями и *класса элементарных сетей*.
- *Элементарная сеть* — это сеть вида



  где a — символ-переход, $\overline{a}$ — *головное место* элементарной сети, $\overline{\overline{a}}$ — ее *хвостовое место*.
- *В формульном представлении* элементарная сеть обозначается символом перехода a.
- Пусть $h(N)$ обозначает множество головных мест сети N, а $l(N)$ — множество ее хвостовых мест.
- Определим операции над сетями.

Операция наложения. Символ “,” I

- Суть этой операции — обычное теоретико-множественное объединение графов, дополненное правилом формирования разметки; с помощью этой операции одна сеть накладывается на другую сеть.
- Если $N_1 = (P_1, T_1, F_1, M_{0_1})$ и $N_2 = (P_2, T_2, F_2, M_{0_2})$, то

$$N = (N1, N2) = (P_1 \cup P_2, T_1 \cup T_2, F_1 \cup F_2, M_0),$$

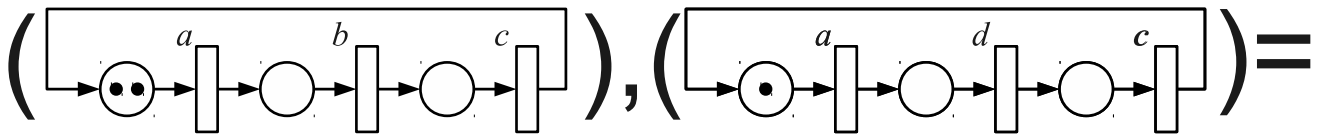
где

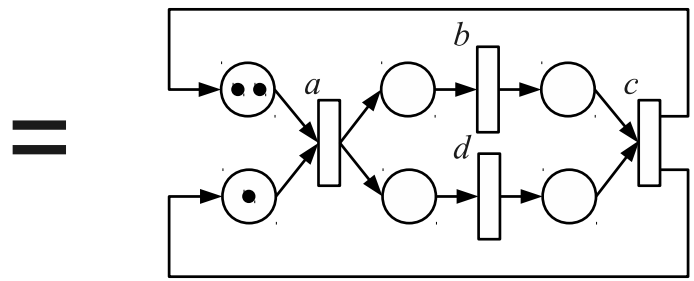
$$M_0 = \begin{cases} M_{0_i}(p), \text{ если } p \in P_i, \text{ где } i = 1, 2, \text{ и } p \notin P_1 \cap P_2, \\ min(M_{0_1}(p), M_{0_2}(p)), \text{ если } p \in P_1 \cap P_2. \end{cases}$$

- По определению операции:
$h(N) = h(N_1) \cup h(N_2), I(N) = I(N_1) \cup I(N_2)$.

Операция наложения. Символ “,” II

- Пример операции показан на следующем рисунке



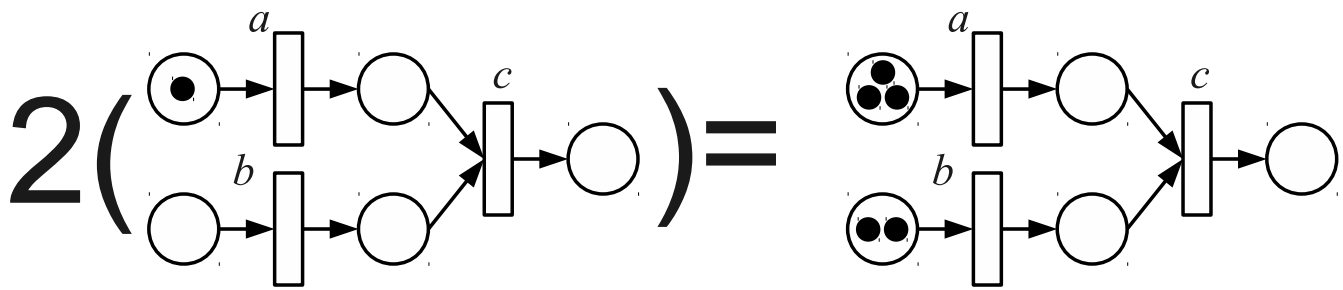


Операция разметки. Символ “$n()$” I

- Сеть $N' = n(N)$ совпадает как граф с сетью N, но имеет другую начальную разметку M'_0, а именно:

$$M'_0(p) = \begin{cases} n + M_0(p), \text{ если } p \in h(N), \\ M_0(p) \text{ в противном случае} \end{cases}$$

- n — целое неотрицательное число фишек, которые надо добавить к головным местам. n может быть специальным симолом ω.
- Пример сети показан на следующем рисунке

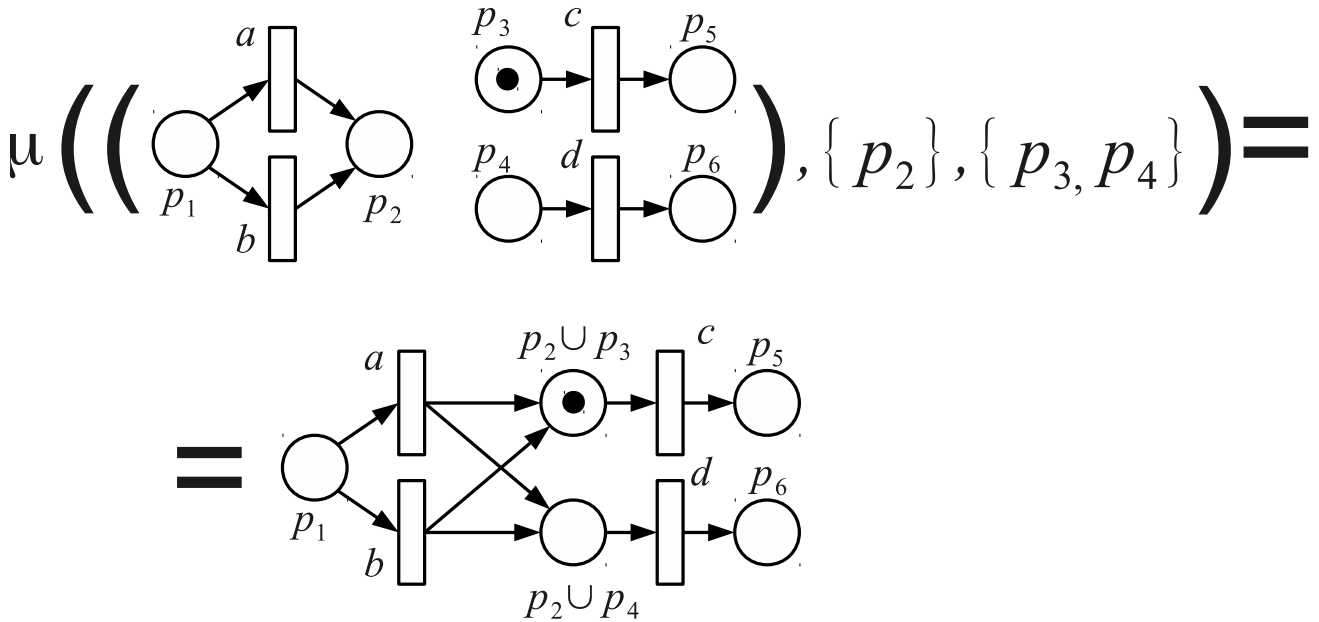


Операция слияния мест. Символ "μ" I

- Пусть дано два сливаемых множества мест $X = (x_1, \ldots, x_n)$ и $Y = (y_1, \ldots, y_m)$ сети $N = (P, T, F, M_0)$.
- Если X или Y — пустое множество, то результат слияния — исходная сеть N.
- Операция слияния μ множества мест сети X и Y сети N строит новую сеть $N' = \mu(N, X, Y)$ в два этапа:
  1. Снчала каждое место $x_i \in X$ копируется в m экземплярах, где m — число места в Y, а каждое место $y_j \in Y$ копируется в n экземплярах, где n — число мест в X. Места копируются вместе с их разметкой и инцидентными дугами.
  2. Затем каждая пара мест (x_i^j, y_j^i) заменяется новым местом $x = x_i \cup y_j$ с разметкой $M_0(z) = M_0(x_i) + M_0(y_j)$ и соответствующими инцидентыми дугами.

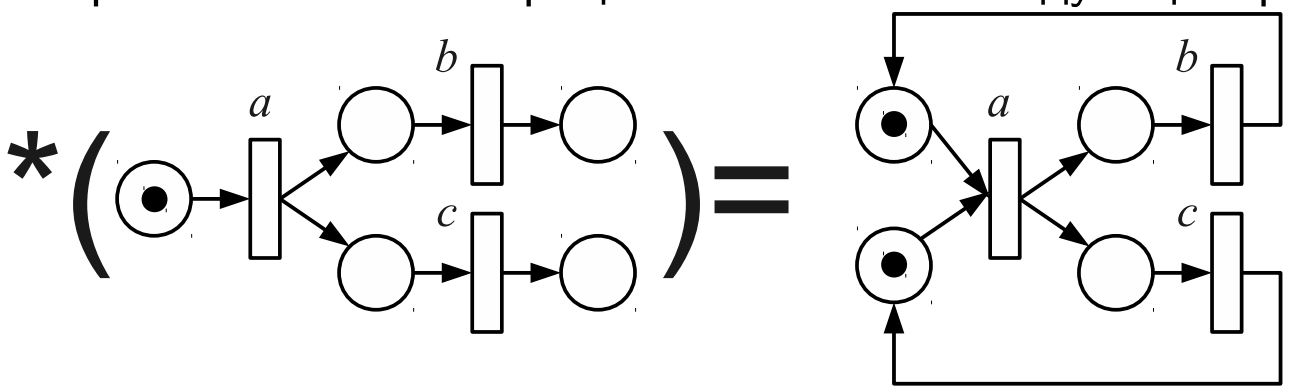
Операция слияния мест. Символ "μ" II

- Пример операции показан на следующем рисунке



Операция итерации. Символ “$*$” I

- Эта операция унарная, она сливает множество головных и множество хвостовых мест сети и применима к сети N, если множество головных и хвостовых мест не пересекаются: $N' = *(N) = \mu(N, h(N), l(N))$, где $h(N) \cap l(N) = \emptyset$
- По определению множества головных и хвостовых мест сети $N' = *(N)$ совпадают друг с другом и с множеством мест, полученных слиянием головных и хвостовых мест исходной сети.
- Пример выполнения операции показан на следующем рисунке.



Операция присоединения. Символ “;” I

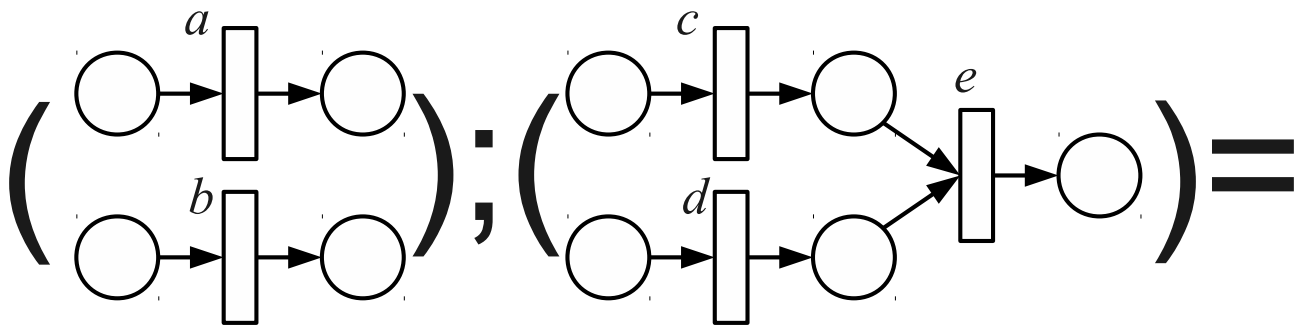
- Эта операция соединяет две сети в одну сливая множество хвостовых мест первой сети со множеством головных мест второй сети:

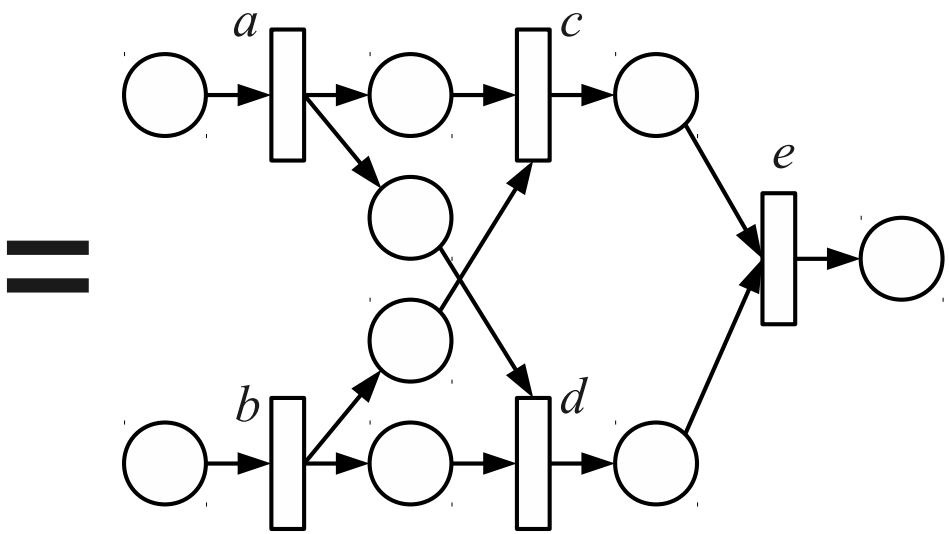
$$N = (N_1; N_2) = \mu((N_1, N_2), I(N_1), h(N_2)).$$

- По определению головными местами новой сети являются “бывшие” головные места сети N_1 и те места, в образовании которых участвовали места из $h(N_1)$; аналогично назначаются хвостовые места новой сети N.

Операция присоединения. Символ “;” II

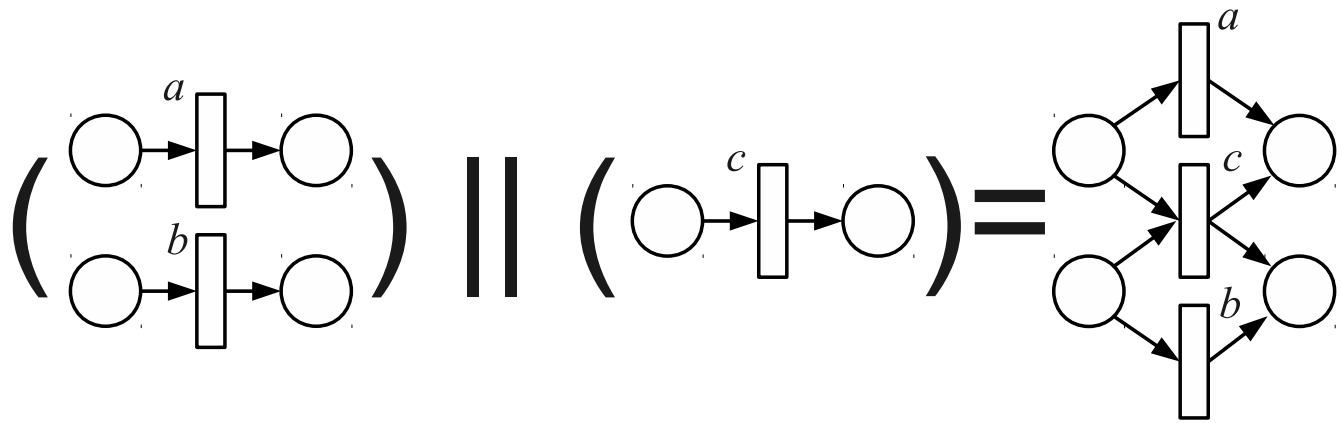
- Пример операции приведен на следующем рисунке





Операция исключения. Символ “||” I

- Эта операция объединяет две сети N_1 и N_2 в одну сеть $N = (N_1||N_2)$, сливая соответственно головные и хвостовые места.
- По определению головные места сети N образованы местами, в которые вошли при слиянии мест головные места сетей N_1 и N_2; аналогично назначаются ховостовые места сети N.
- Пример выполнения операции показан на следующем рисунке



Структура управления I

- Под структурой управления программы понимают совокупность базовых структурных единиц — операторов (а также модулей, подпрограмм и т.п.) — и специальных управляющих примитивов, позволяющих в процессе исполнения программы формировать сложные вычислительные процессы из более простых, задаваемых упомянутыми структурными единицами.
- К числу управляющих примитивов относятся, например, операторы управления в алгоритмических языках, такие как условные операторы, операторы перехода, операторы цикла и т.п.
- Рассмотрим основные типы структур управления в программах и методы их представления с помощью сетей Петри.

Последовательная структура управления I

- Обычно для алгоритмических языков последовательная структура управления представляет собой совокупность операторов различных типов, которые связаные некоторым отношением следования.
- Начальный оператор (единственный в программе) не следует ни за каким-либо другим оператором. После безусловного оператора следует ровно один оператор. После условного оператора следует два оператора (используется также оператор выбора, за которым следует некоторая совокупность операторов). После заключительного оператора не следует никакой другой оператор.
- Исполнение программы с последовательной структурой управления начинается с начального оператора. После заврешения исполнения безусловного оператора начинается исполнение следующего за ним оператора. После завершения исполнения условного оператора начинается исполнение одного из двух следующих за ним операторов в зависимости от значения логического условия, вычисленного в условном

Последовательная структура управления II

оператора. Исполнение программы завершается исполнением заключительного оператора.

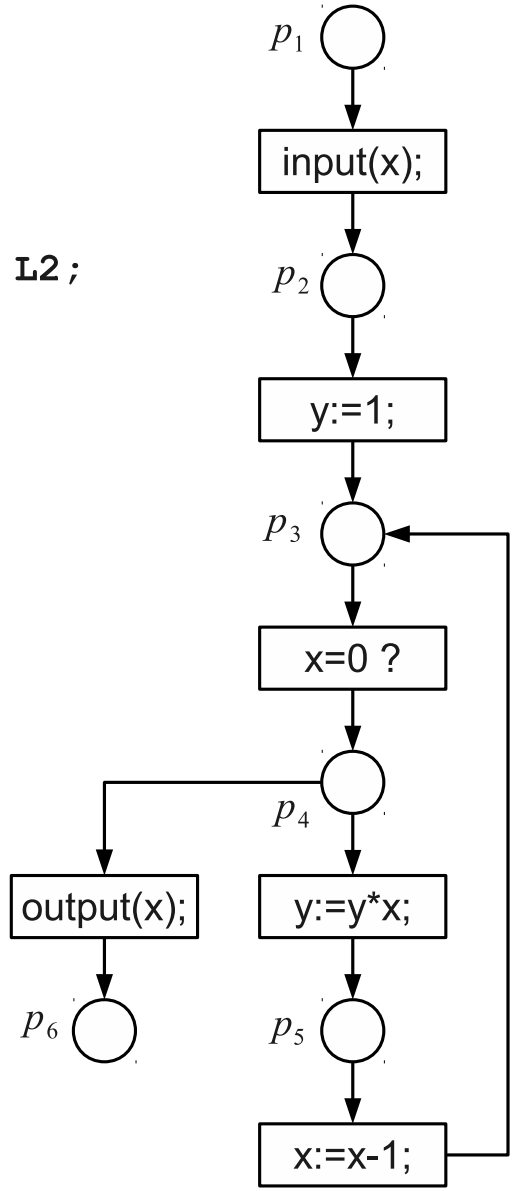
- *Последовательная структура управления* может быть представлена как сеть Петри с определенным топологическим ограничением, элементы которой интерпретируются специальным образом:
  - Такая сеть имеет единственное головное место.
  - Переход интерпретируются как “безусловный” или “условный” оператор программы.
  - Моделирование условия состоит в выделении определенного места в сети, которое бы отражало факт вычисления этого условия с последующим разветвлением структуры управления на соответствующие действия.
  - Если в последовательной программе оператор a' следует за оператором a, то в сети соответствующие переходы связаны дугами с общим местом, которое является выходным для a и выходным для a.

Последовательная структура управления III

- Пример программы (на алгоритмическом языке) и ее структуры управления показан на следующем рисунке.

Последовательная структура управления IV

```
Begin
     input(x);
     y:=1;
L1:  if x=0 then L2;
     y:=y*x;
     x:=x-1;
     goto L1;
L2:  output(y);
End
```

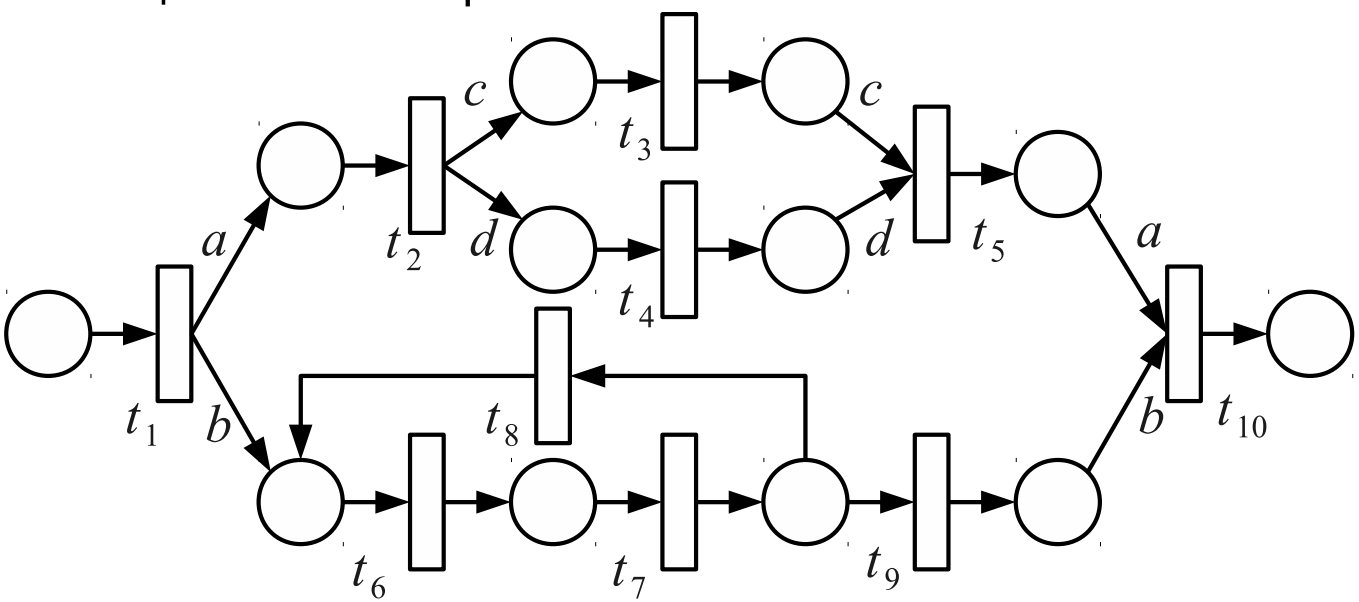


Последовательно-параллельная структура управления I

- *Последовательно-параллельная структура управления* представляет собой обощение последовательной структуры управления за счет включения в набор управляющих примитивов специальных операторов или указателей, которые выделяют *параллельные ветви* программы, исполняемые независимо друго от друга.
- Ветви имеют общее начало — точку расхождения ветвей — и конец — точку схождения ветвей.
- Совокупность ветвей с общим началом и концом образует *сегмент* параллельной структуры.
- При исполнении программы процесс вычислений продвигается до начала первого сегмента, после чего “расщепляется” на столько копий, сколько ветвей содержит сегмент.
- Каждый из параллельных процессов вычислений ветвей протекает независимо от других, и достигая конца ветви, останавливается, ожидая, пока все остальные процессы в сегменте не достигнут конца сегмента.

Последовательно-параллельная структура управления II

- В конце сегмента все процессы “сливаются” в один.
- Параллельные ветви сегментов могут, в свою очередь, содержать дургие сегменты.
- В последовательно-параллельных языках программирования для выделения параллельных ветвей используются специальные программные примитивы: `parbegin...parend`, `cobegin...coend`, `fork...join`.
- На следующем рисунке показан пример последовательно-параллельной структуры, изображенной с помощью сети Петри.



Последовательно-параллельная структура управления III

- Отличительной особенностью последовательно-параллельных структур управления является отсутствие каких-либо взаимодействий между параллельными ветвями. Это существенный недостаток — малая гибкость управления.
- Чтобы преодолеть этот недостаток введем описание взаимодействий в последовательно-параллельную структуру управления.

Взаимодействие в последовательно-параллельной структуре I

- Для организации взаимодействия нужны дополнительные программные средства, обеспечивающие саму организацию взаимодействия и правильность их реализации в процессе исполнения параллельных ветвей.
- В большинстве случаев нужны средства, которые позволяли бы приостанавливать процесс исполнения ветви, пока не прибудут данные от другой ветви или пока другие ветви не освободят общий ресурс.
- Тот отрезок ветви, который требует “захвата” общего ресурса, называют *критической секцией (или интервалом)* ветви.
- Дейкстра предложил для этих целей механизм семафоров, включающий специальные переменные нового типа (*семафоры*) и две операции P и V, аргументами которых могут быть только переменные типа семафоров.
- Область значений семафора — целые неотрицательные числа.

Взаимодействие в последовательно-параллельной структуре II

- Если область значений сужена до двух — 0,1, то семафор называют *бинарным* (или *мьютексом*).
- Операция V изменяет значение s семафора на $s + 1$.
- Действие операции P определяется следующим образом:
  - если $s \neq 0$, то P уменьшает значение s на 1,
  - если $s = 0$, то P не уменьшает значение s и не завершается до тех пор, пока некоторая другая ветвь не изменит значение s с помощью операции V.
- Существенным является тот факт, что операции P и V считаются “неделимыми”. По отношению к V это означает следующее. Операция V состоит из трех фаз: считывание значения семафора из памяти, увеличение значения семафора, помещение нового значения в память. Неделимость $V(s)$ заключается в том, что с самого начала выполнения этой операции до ее завершения доступ к переменной-семафору s запрещен для всех других операций.
- Аналогично дело обстоит с операцией P.

Взаимодействие в последовательно-параллельной структуре III

- Обычно P предшествует критическом интервалу ветви, а V его завершает.
- В каждый момент времени, когда значение семафора s изменяется с 0 на 1, только одна из операций может завершиться и разрешить вход в критический интервал только одному процессу исполнения ветви.
- Рассмотрим пример программы на алгоподобном языке программирования. Параллельный сегмент выделен командами `parbegin...parend`.

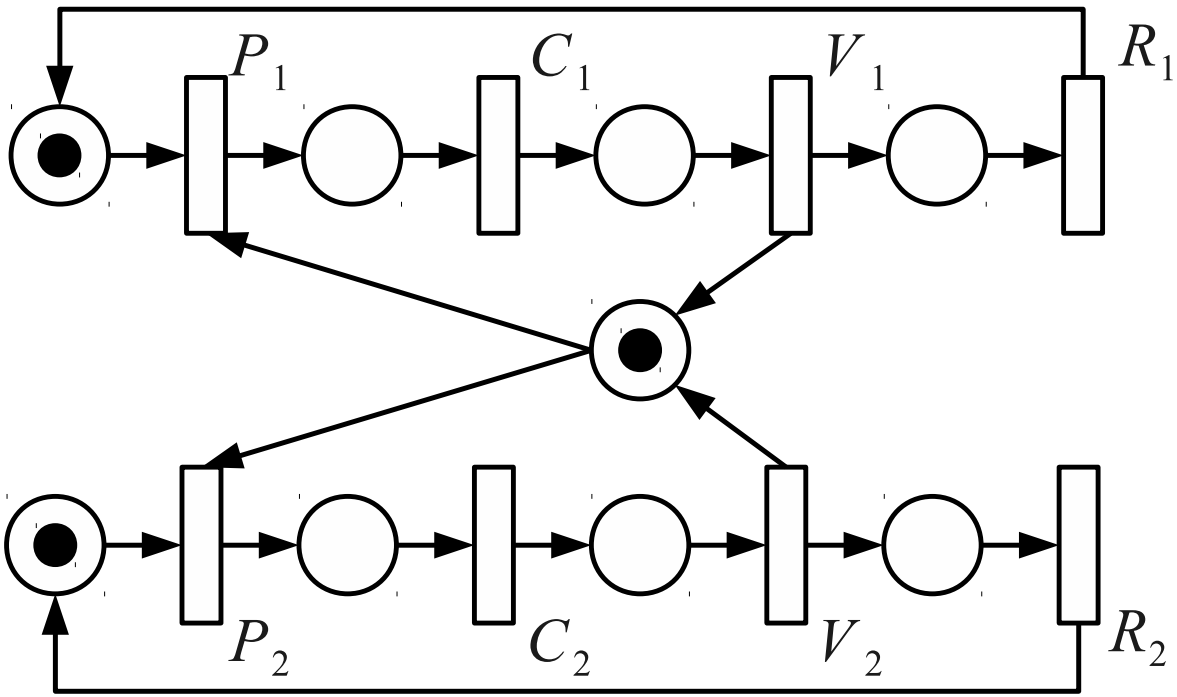
Взаимодействие в последовательно-параллельной структуре IV

```
Begin
     semaphore mutex;
     mutex:=1;
Parbegin
  Begin process 1
L1: P(mutex);
     Крит. Интер. 1;
     V(mutex);
     Остаток цикла 1;
     goto L1;
  End,
  Begin process 2
L2: P(mutex);
     Крит. Интер. 2;
     V(mutex);
     Остаток цикла 2;
     goto L2;
  End
ParEnd
End
```

Взаимодействие в последовательно-параллельной структуре V

- В этом примере решается задача взаимного исключения исполнения критических интервалов двух циклических процессов параллельных ветвей.
- Последовательно-параллельная структура управления, дополненная семафорным механизмом, наглядно изображается с помощью сетей Петри.
- На следующем рисунке показана сеть, моделирующая структуру программы с взаимноисключающими критическими интервалами двух циклических ветвей.

Взаимодействие в последовательно-параллельной структуре VI



- Переход P_i соответствует операции P в ветви i $(i = 1, 2)$; переход C_i — критическом интервалу ветви i; V_i — операции V ветви i; R_i — остатку цикла в ветви i. Место p_s соответствует семафору s.