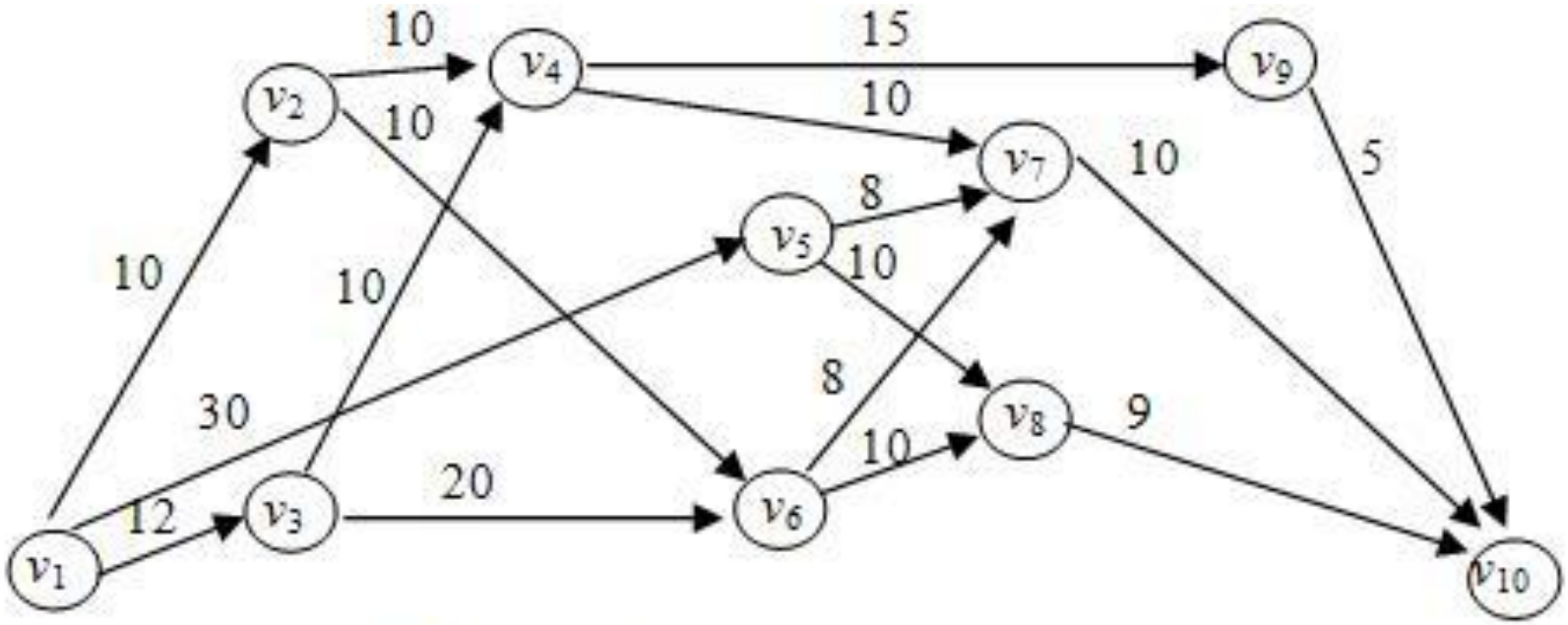
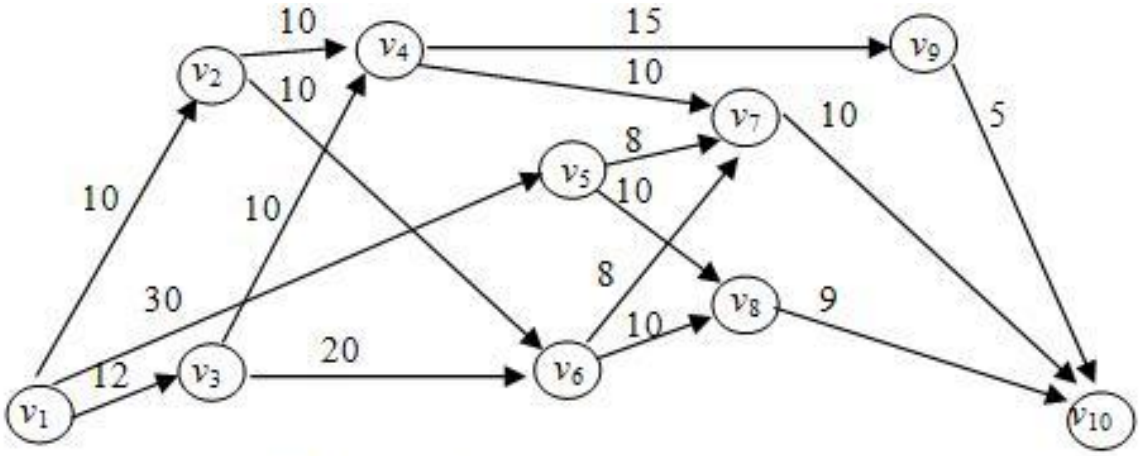
N-схема моделирования

N-схема (от англ. Net) применяется для описания сложно структурированных объектов.



Понятие графа



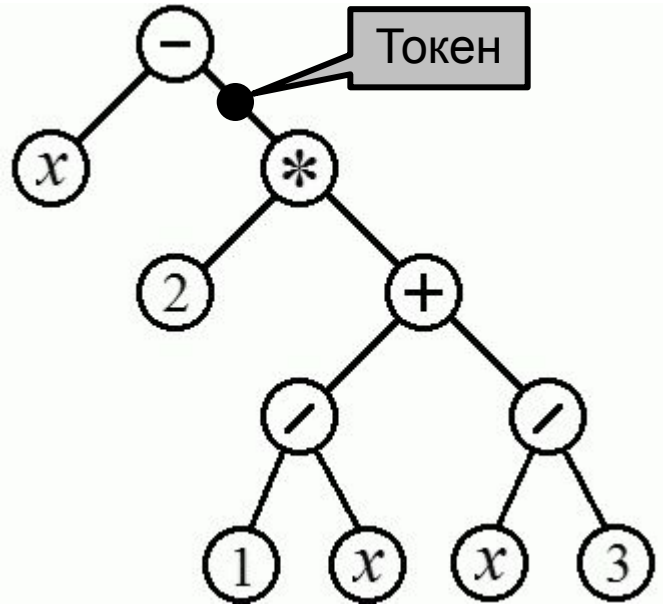
Граф – это
G={V,U}, где
V – множество вершин;
U ∈ V x V – множество ребер.

При применении для моделирования узлы и рёбра могут помечаться. Например, веса узлов и ребер, а также различные информационные конструкции.

Также узлы могут и дуги могут сопоставляться на различными объектами и явлениями.

Арифметический граф

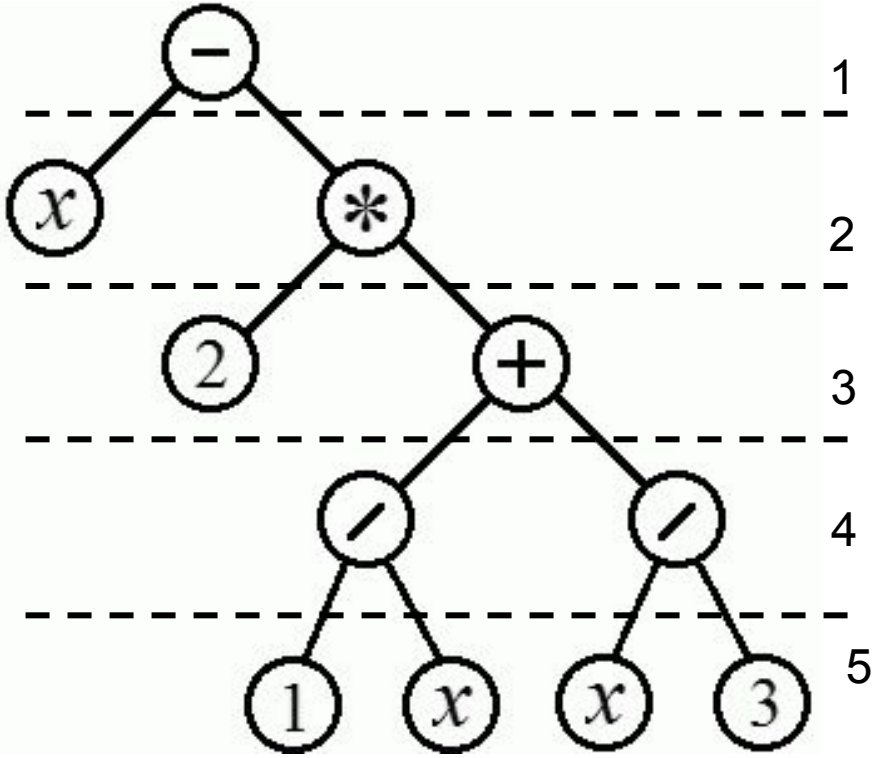
В арифметическом графе узлы обозначают арифметические операции, а дуги – передаваемые между операциями операнды. Такой граф можно использовать для моделирования вычислительного процесса без циклом, где применяются исключительно арифметические операции.



Токен – операнд, передаваемый между операциями (узлами гарафа)

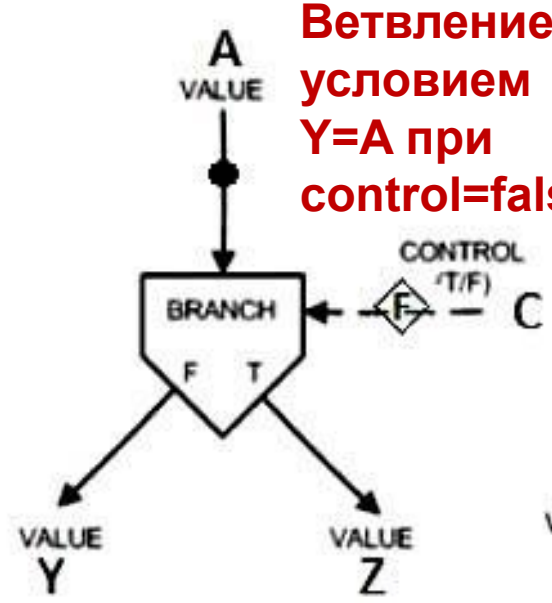
Ярусизированный (ярусный) арифметический граф

Ярус узла – это самый длинный путь от основания графа (т.е. вершин, из которых имеются только исходящие дуги) до этого узла. Операции, ассоциированные с узлами на одном ярусе могут выполняться параллельно, т.к. они незывисимы по данным.

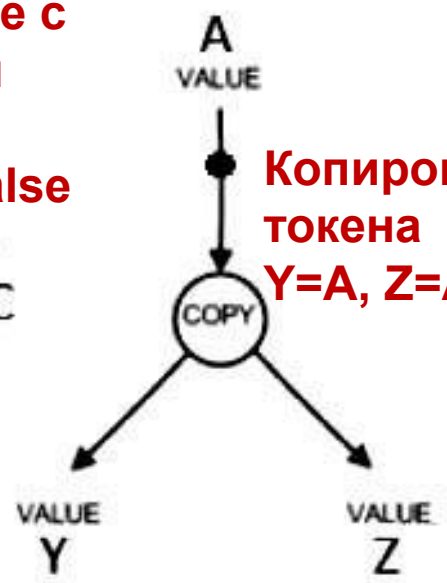


Число ярусов – это число тактов, за которое происходит выполнение алгоритма. Максимальное число узлов на ярусе – количество необходимых вычислительных узлов.

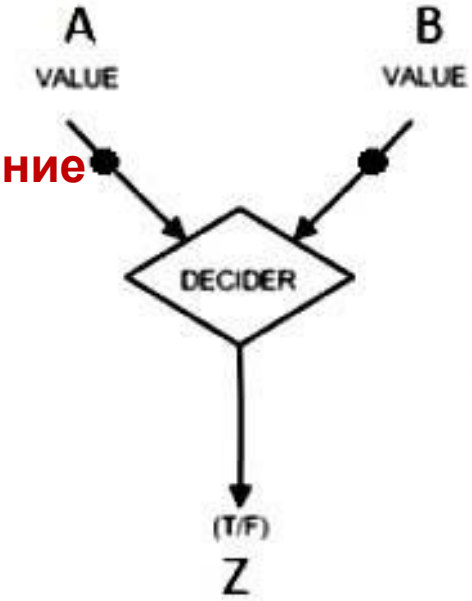
Потоковый граф (операции)



Ветвление с условием
Y=A при control=false

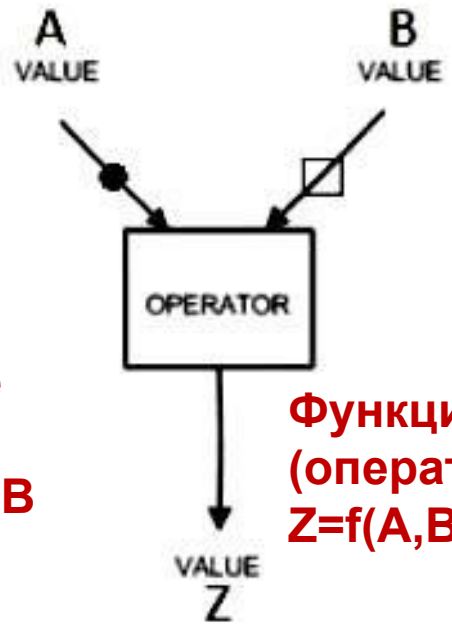


Копирование токена
Y=A, Z=A

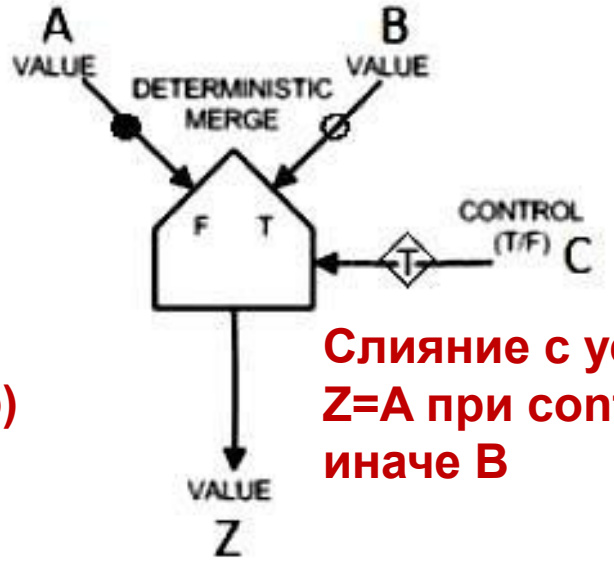


«Решатель»
Z=true при приходе токена A, иначе false

Слияние токенов
Z=A или B

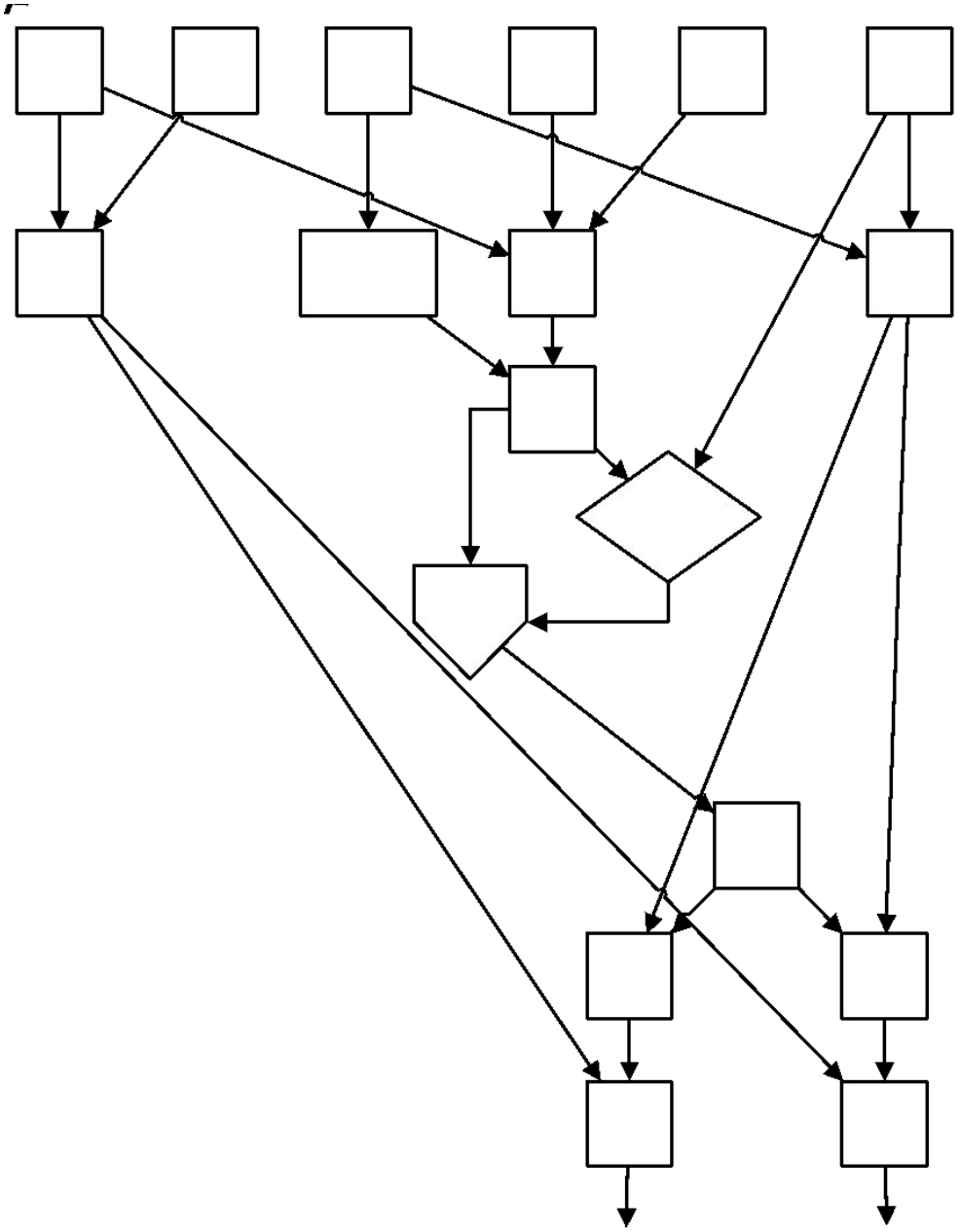


Функция (оператор)
Z=f(A,B)



Слияние с условием
Z=A при control=false, иначе B

Потоковый граф решения квадратного трехчлена



Сетевой граф

Решение сетевой задачи включает следующее:

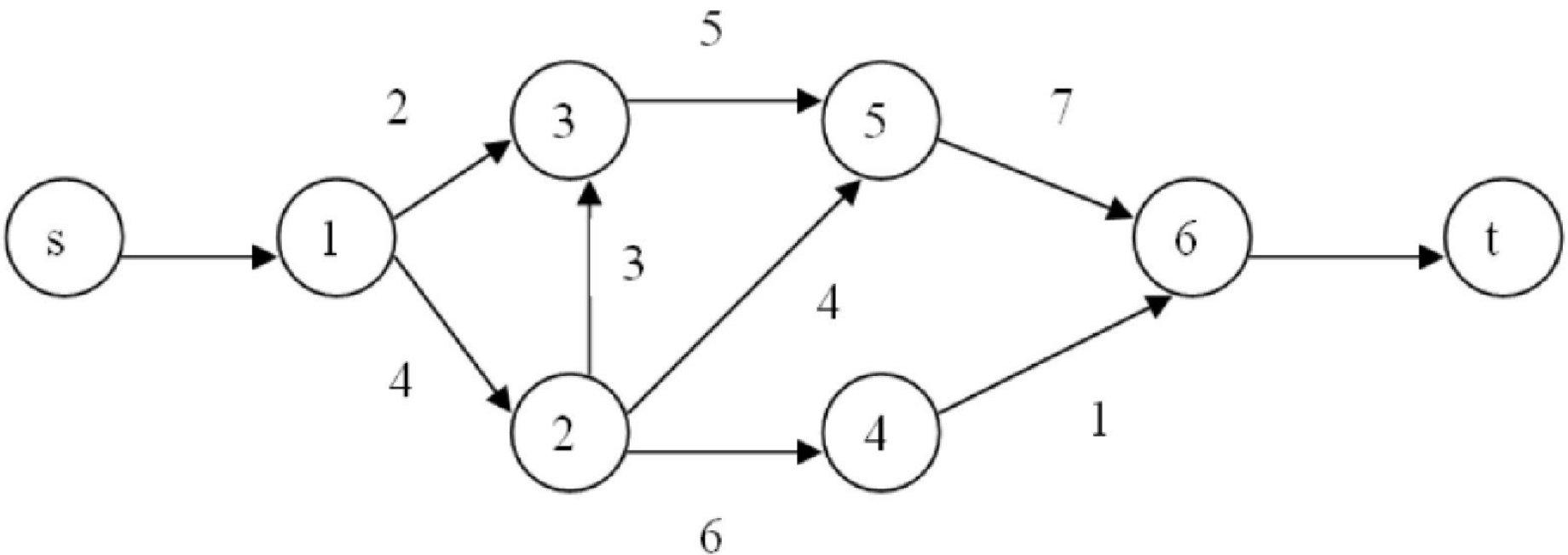
1) определение или оценивание длительности j p выполнения каждой элементарной работы j ,

2) определение критического, т.е. наиболее длинного пути между начальной и конечной вершиной сети.

Наиболее распространены два типа сетевых моделей, называемые

1) действие-на-вершине

2) действие-на-дуге.



Характеристики событий сетевой модели

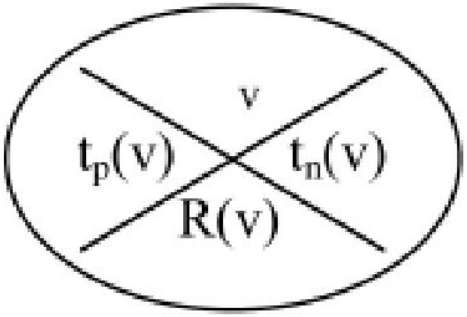
1. Ранний срок свершения события $t_p(s)=0$, $t_p(v)=\max_u\{t_p(u)+t_{uv}\}$ характеризует самый ранний срок завершения всех путей в него входящих. Этот показатель определяется «прямым ходом» по графу модели, начиная с начального события сети.

2. Поздний срок свершения события $t_n(t)=t_p(t)$, $t_n(u)=\min_v\{t_n(v)-t_{uv}\}$ характеризует самый поздний срок, после которого остается ровно столько времени, сколько требуется для завершения всех путей следующих за этим событием. Этот показатель определяется “обратным ходом” по графу модели, начиная с завершающего события сети.

3. Резерв времени события $R(v)=t_p(v)-t_n(v)$ показывает, на какой максимальный срок можно задержать наступление этого события, не вызывая при этом увеличения срока выполнения всего комплекса работ.

Резервы времени для событий на критическом пути равны нулю, $R(v)=0$.

Характеристики событий сетевой модели



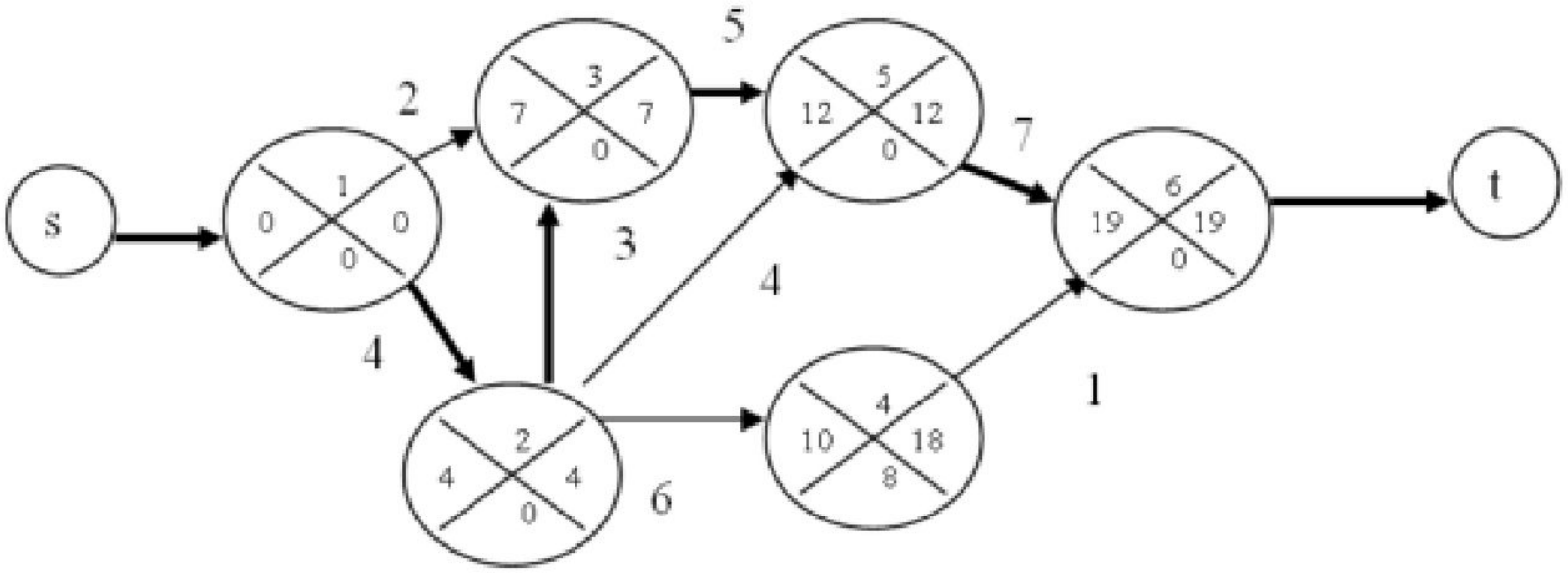
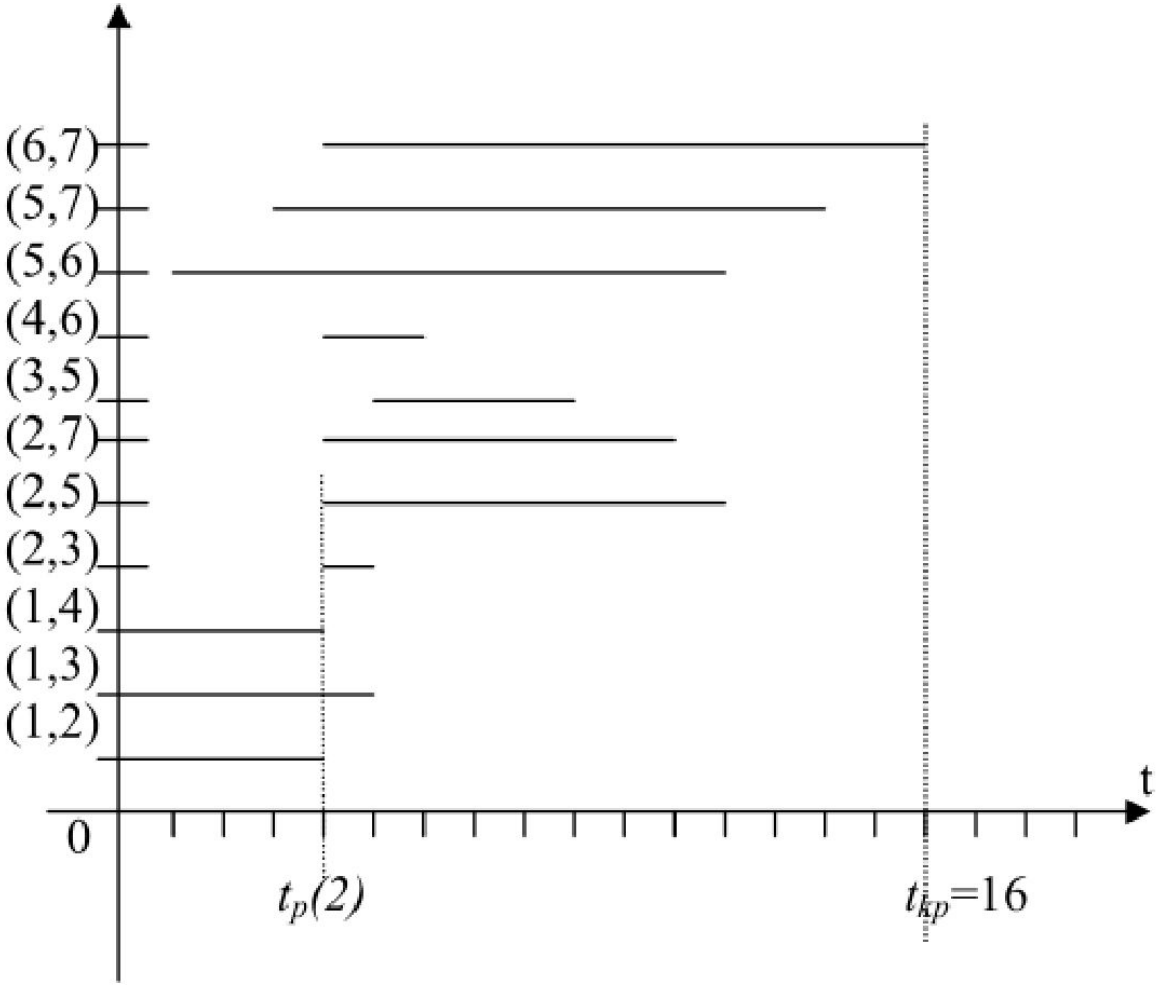
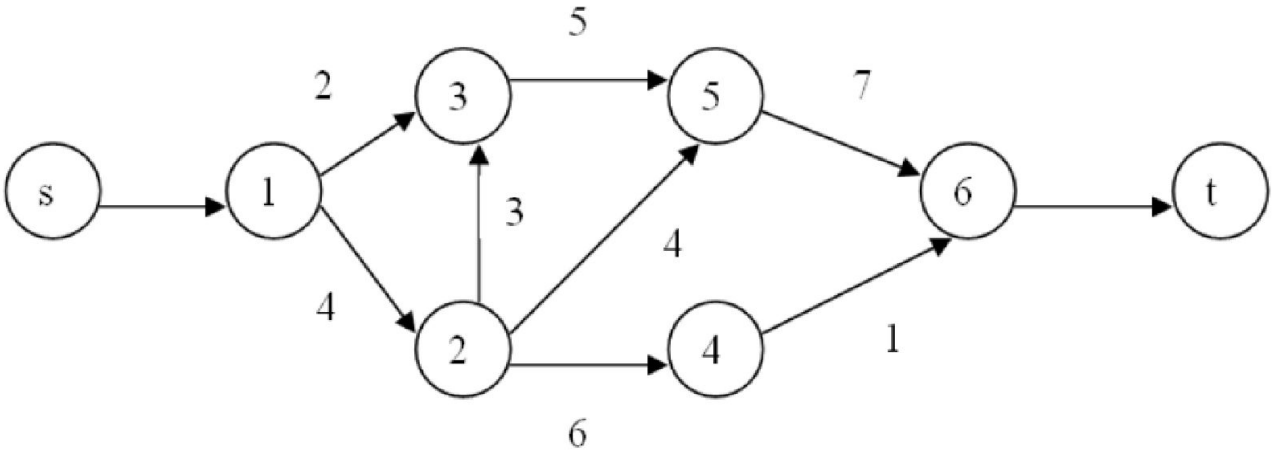


Диаграмма Ганта

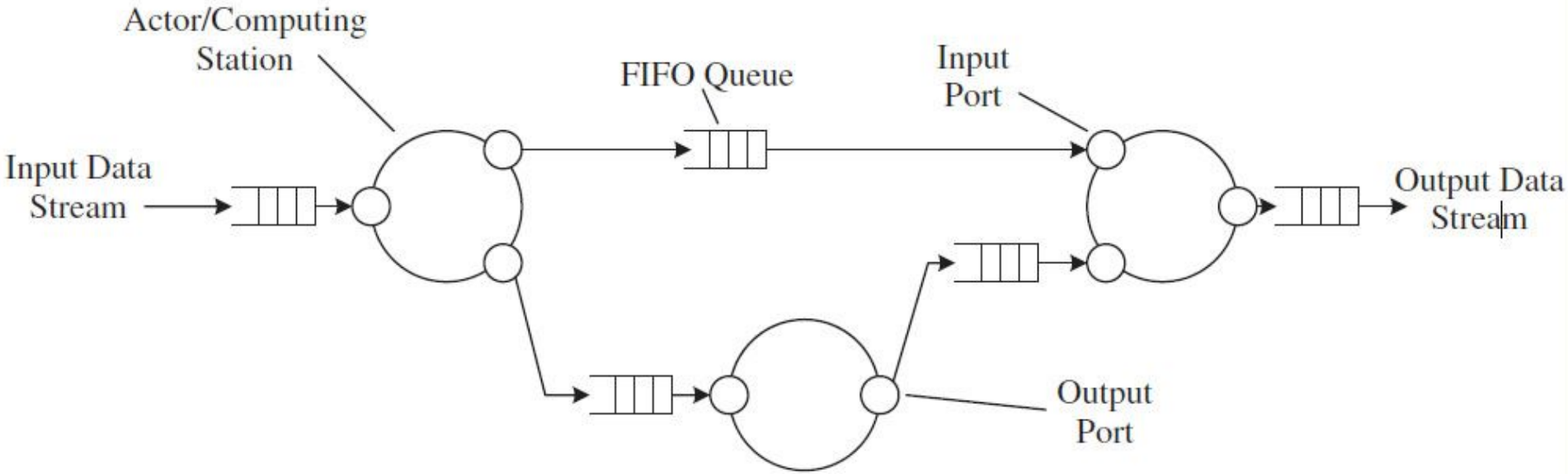




Процессная сеть Кана (KPN)

KPN (Kahn process network) – это совокупность устройств, выполняющих вычисления над приходящими к ним данным, оформленных в виде токеном. Каждое устройство снабжено очередью токенов, куда попадают пришедщие токены, если устройство занято обработкой данных.

KPN идеально подходят для моделирования распределенных вычислительных систем с управлением данными (dataflow), однако не подходят для систем с централизованной памятью.



Сеть Петри

Назначение: моделирование систем с взаимодействующими параллельно функционирующими компонентами. Основная задача моделирования – поиск *тупиковых ситуаций* и *переполнений* в моделируемой системе.

Аппарат сетей Петри был предложен ***Карлом Адамом Петри*** в своей докторской диссертации "Связь автоматов" в 1962 году. Работой заинтересовалась группа под руководством ***Дж. Денниса***, занимающейся разработкой вычислительных систем с управлением данными (dataflow). Группа стала плотно заниматься данной тематикой и «раскрутила» ее, т.к. стала источником множества публикаций.

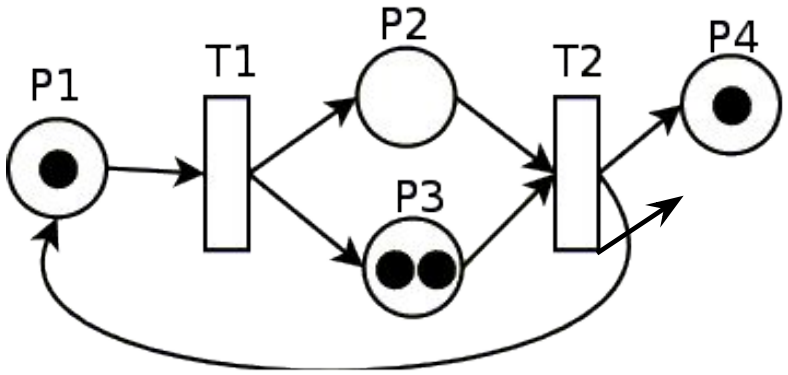
Математическая база сети Петри

Математической основой теории сетей Петри является теория комплектов.
Комплект – это множество, в которое один элемент может входит несколько раз (в классической теории множеств элемент либо не входит во множество, либо входит только один раз).

Комплект также может называться мультимножеством или кортежем (упорядоченное мультимножество).

Функция числа экземпляров обозначается #. Запись #(x,B) значит, что элемент x входит во множество B раз, где $0 \leq B \leq \infty$. Ситуация, когда $0 \leq B \leq 1$, соответствует классической теории множеств.

Сеть Петри (СП)



Сеть Петри - это

двудольный ориентированный мультиграф

A=(P,T,I,O), где

P – множество состояний;

T – множество переходов;

I – множество (картеж) входных дуг;

O – множество (картеж) выходных дуг;

Маркировка СП – расстановка фишек в состояниях СП. Это вектор неотрицательных целых чисел, размерность которого равна числу состояний СП.

Состояния (P):

$P=\{P_1,P_2,P_3,P_4\}$

Переходы (T):

$T=\{t_1,t_2\}$

Маркировка (μ):

$\mu=(1,0,2,1)^T$

Входы (I):

$I(t_1)=\{P_1\}$

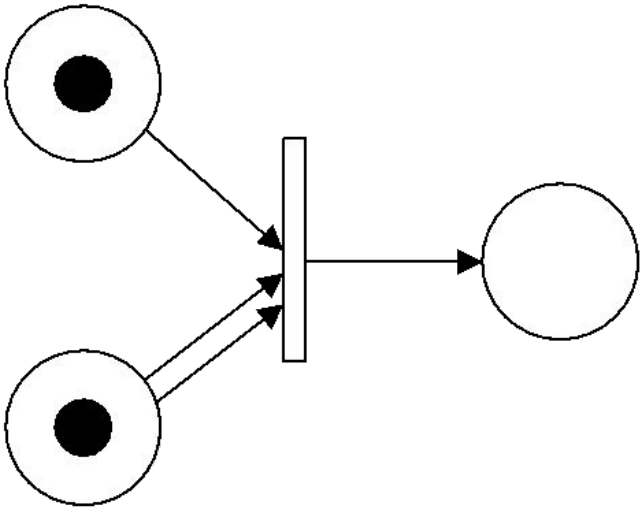
$I(t_2)=\{P_3,P_2\}$

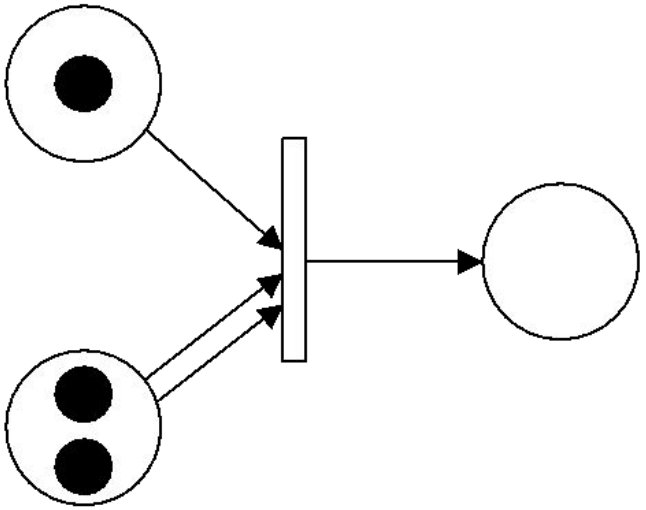
Выходы (O):

$O(t_1)=\{P_2,P_3\}$

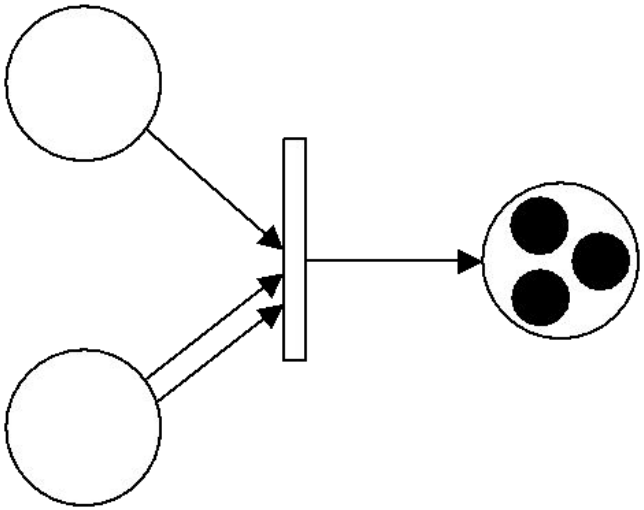
$O(t_2)=\{P_4,P_4\}$

Выполнение сети Петри





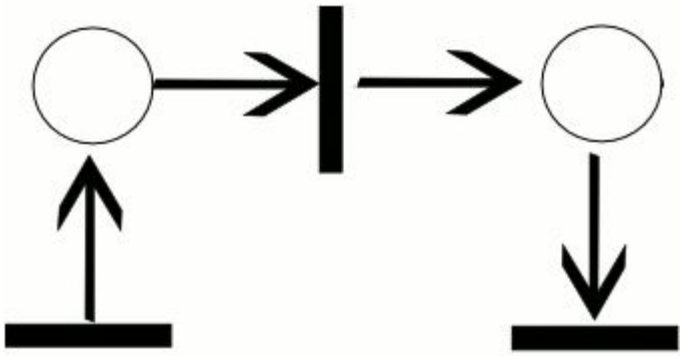
Выполнение сети Петри
S=t_{i1}, t_{i2}… , где t_{ij} – номер активированного перехода.



Переходы могут быть активными и неактивными. У активных переходов каждой выходящей дуге соответствует хотя бы одна фишка

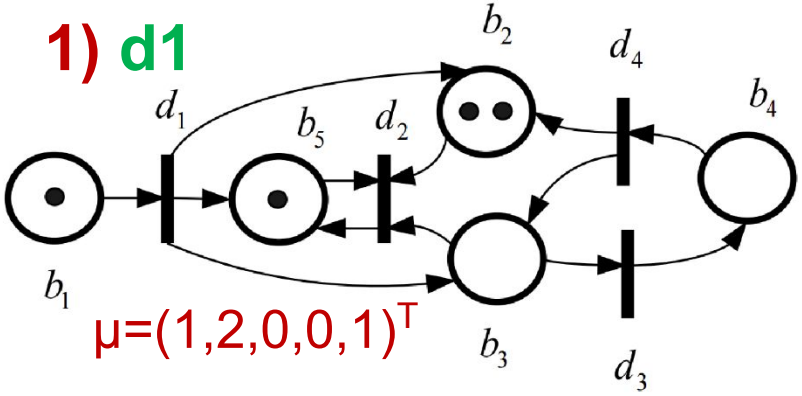
Выполнение сети Петри

Выполнение $S=t_{i1}, t_{i2}\ldots$,, где t – номер активированного перехода. Т.е. выполнение сети Петри – это последовательность активированных переходов.



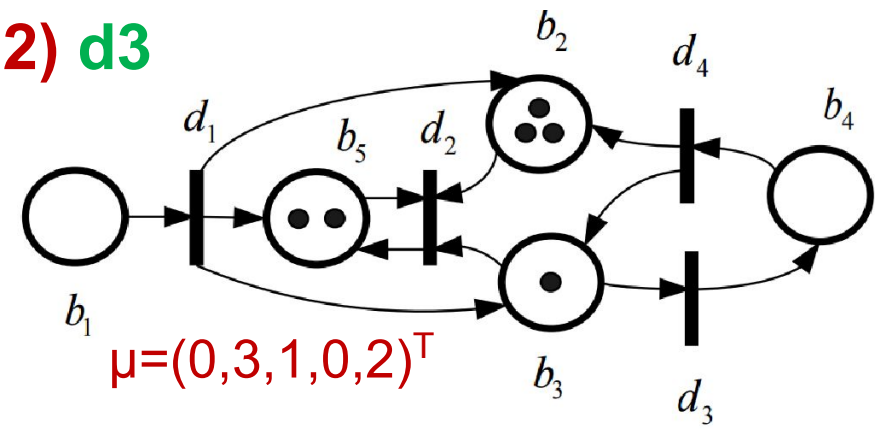
Пример выполнения сети Петри

1) d1



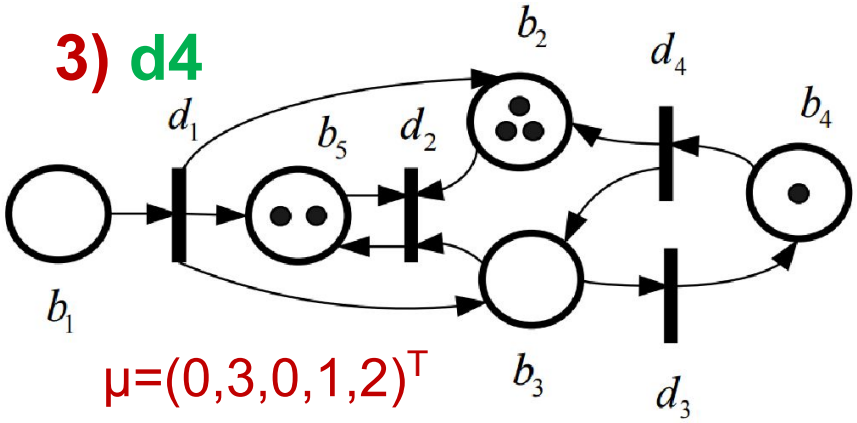
$\mu=(1,2,0,0,1)^T$

2) d3



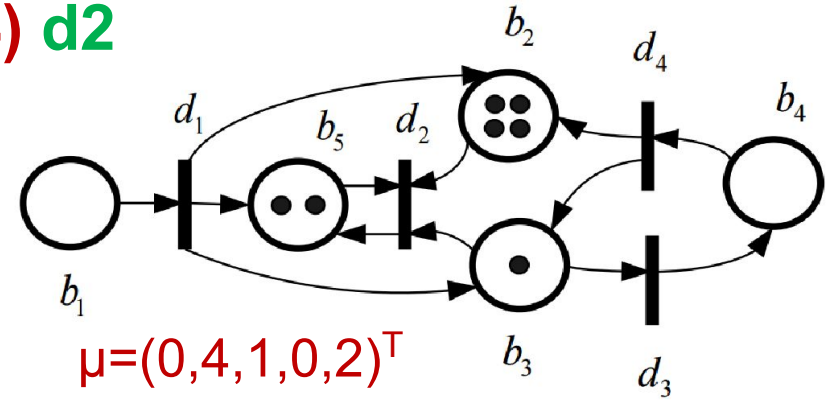
$\mu=(0,3,1,0,2)^T$

3) d4



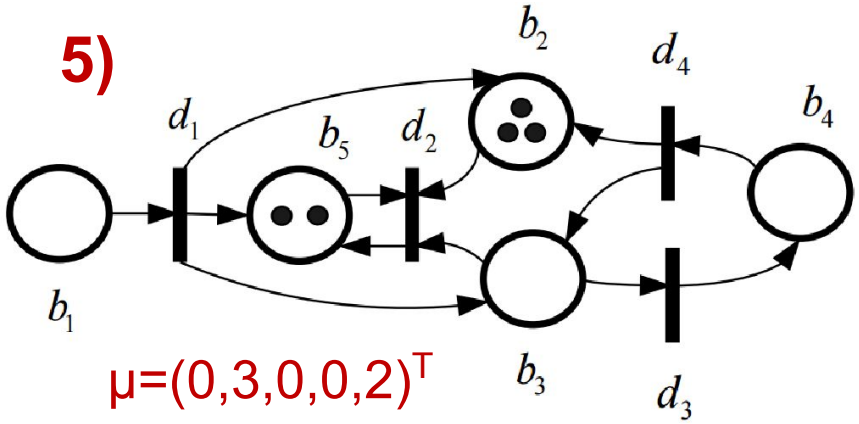
$\mu=(0,3,0,1,2)^T$

4) d2



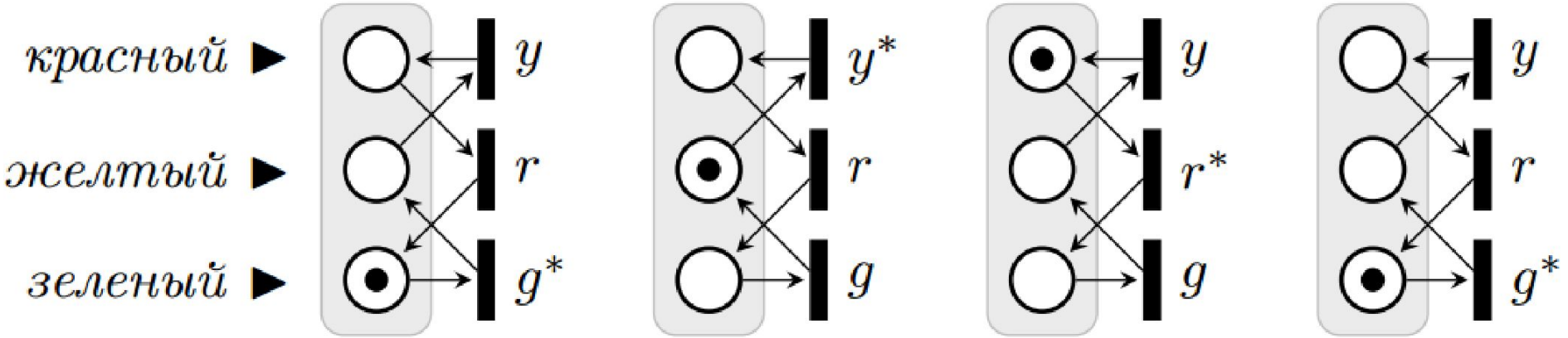
$\mu=(0,4,1,0,2)^T$

5)



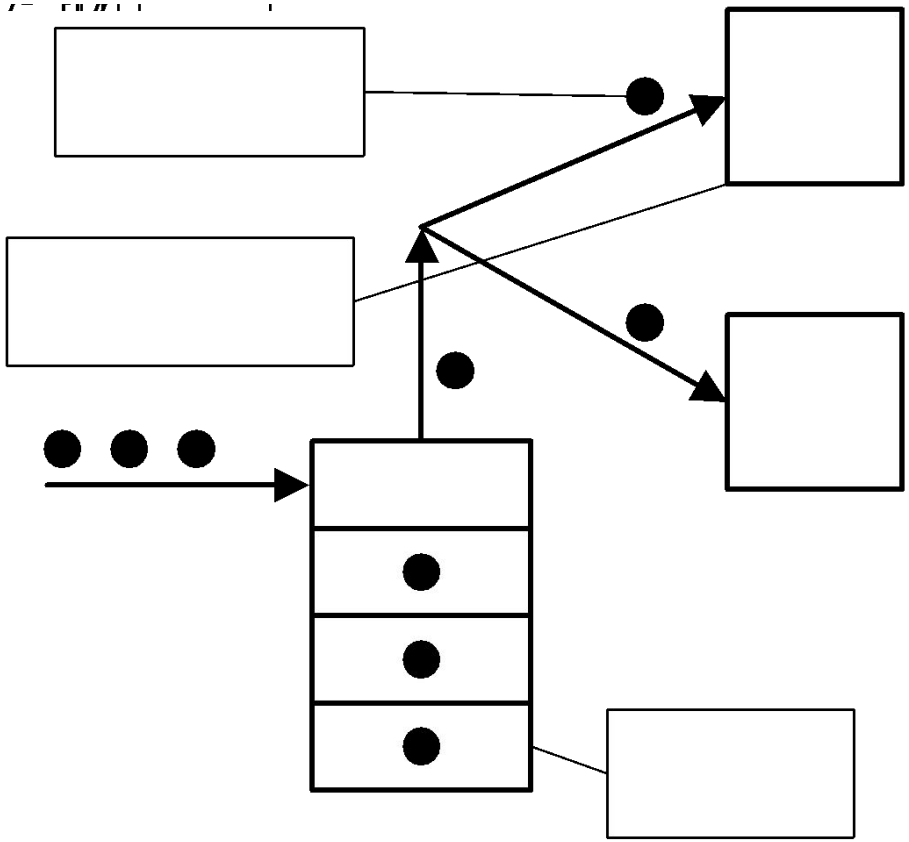
$\mu=(0,3,0,0,2)^T$

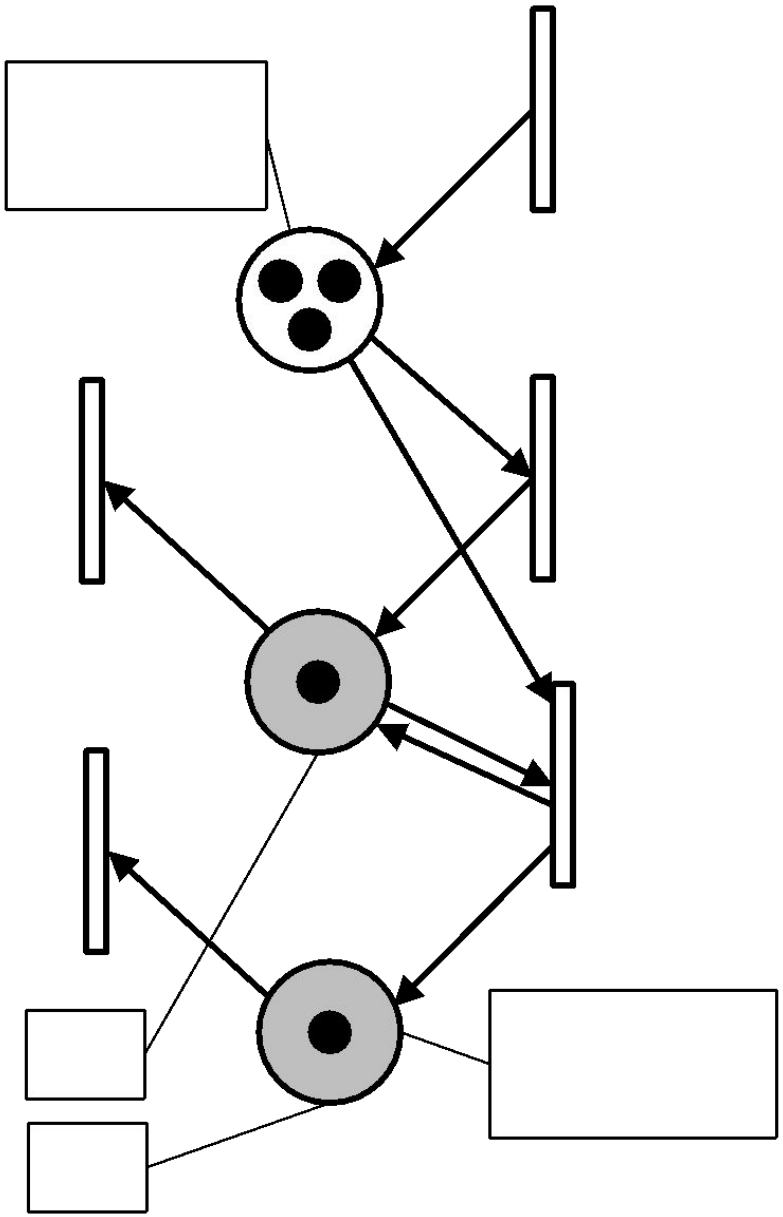
Моделирование работы светофора с помощью сети Петри



Моделирование параллельной вычислительной системы

Вычислительная систем с двумя параллельно работающими обслуживающим устройствами и очередью ожидания заявок



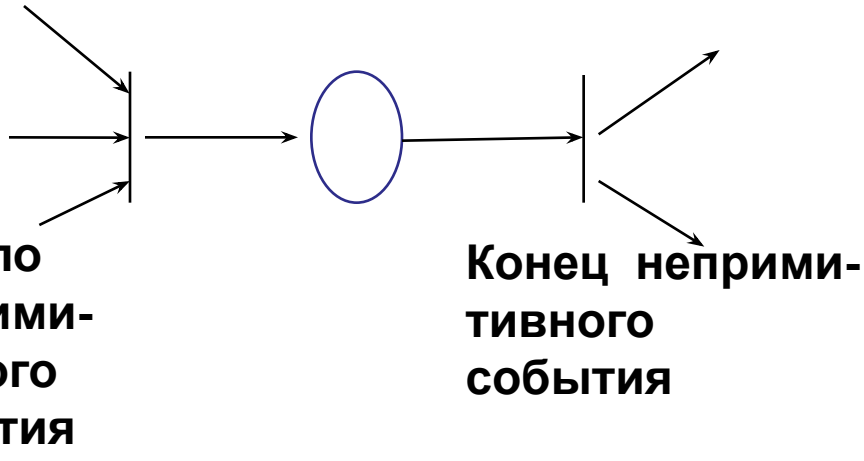


Виды событий в сети Петри

Виды событий:

1. Примитивное (мгновенные). Обозначаются переходами (Т). Такие события всегда неодновременны.
2. Неприметивное (длительные). Обозначаются состояниями. Такие состояния могут быть одновременными.

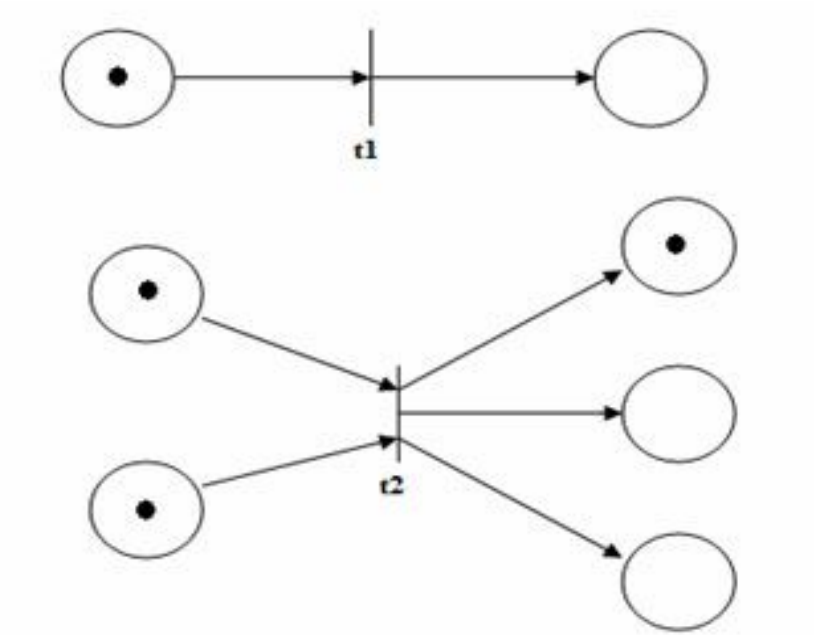
Непримитивное событие можно обозначать с помощью двух переходов (начало и конец непримитивного события) и одного состояния, обозначающего непримитивное событие событие.



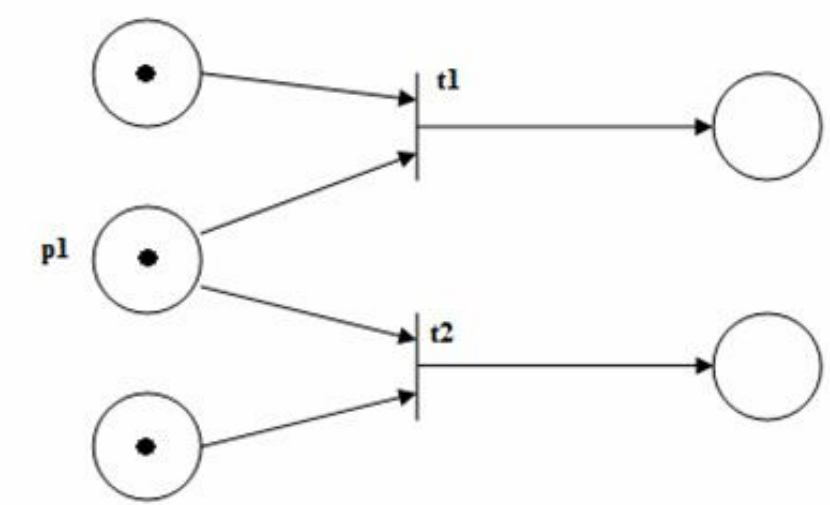
Виды событий в сети Петри

Виды событий:

1. Одновременное событие



2. Конфликт



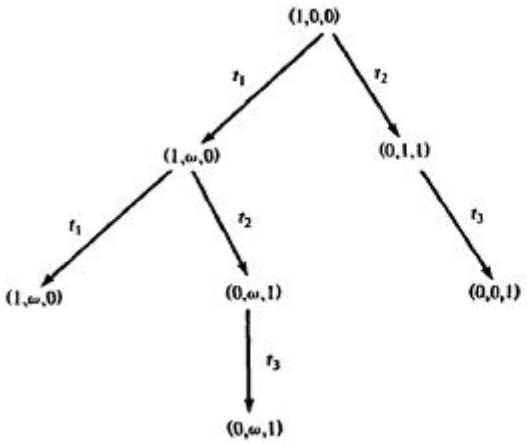
Проверка критериев сети Петри при их моделировании

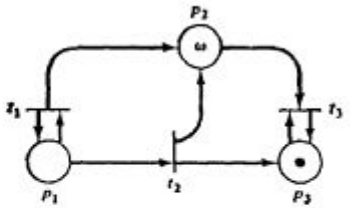
- ***ограниченность*** — число меток в любой позиции сети не может превысить некоторого значения K;
- ***безопасность*** — частный случай ограниченности, K=1;
- ***сохраняемость*** — постоянство загрузки ресурсов, $\sum A_i N_i$, где Ni – число маркеров в i-той позиции; Ai – весовой коэффициент;
- ***достижимость*** — возможность перехода сети из одного заданного состояния (характеризуемого распределением меток) в другое;
- ***живость*** — возможность срабатывания любого перехода при функционировании моделируемого объекта.

Стратегии моделирования сетей Петри

Две стратегии моделирования:

1. Дерево достижимости





2. Матричный метод

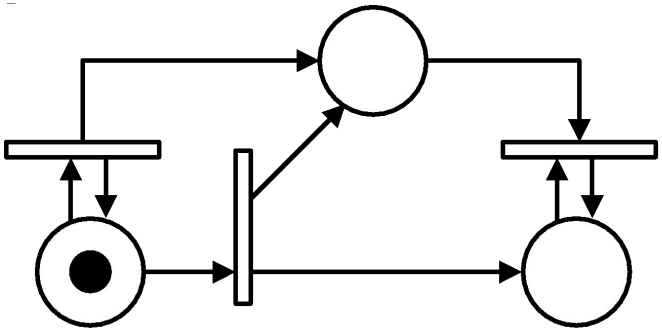
$$W^- = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}, \quad W^+ = \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad W = \begin{bmatrix} -1 & 1 \\ 1 & -1 \\ 1 & -1 \\ 0 & 1 \end{bmatrix}$$

Алгоритм построения дерева достижимости

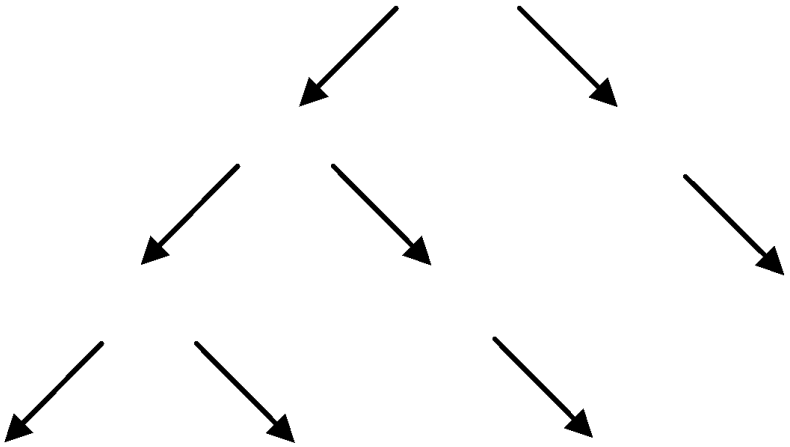
1. Значение $m(x)_i$- = w. Тогда для порожденной вершины z: m (z)i = w.
2. На пути от корневой вершины к вершине x существует вершина *y* с меньшей маркировкой m (y) < m (x), $m(y)_i < m(x)_i$. В этом случае происходит порождение символа бесконечности: $m(x)_i = w$, $m(z)_i = w$. В рассматриваемом примере на след. Слайде такой вершиной, например, является вершина с маркировкой (1, 2, 0) -> (1, w, 0).
3. Если существует разрешенная маркировка, то в порождаемой вершине z заносится эта маркировка в соответствии с правилами срабатывания перехода.

Алгоритм завершает свою работу, когда все вершины являются внутренними, терминальными, дублирующими.

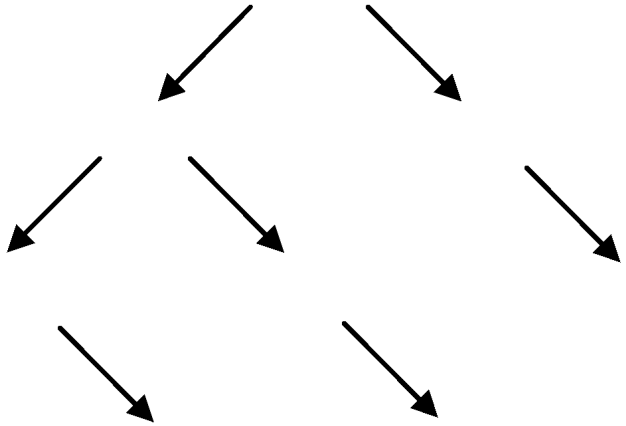
Моделирование сети Петри методом дерева достижимости



Бесконечное дерево достижимости



Конечное дерево достижимости



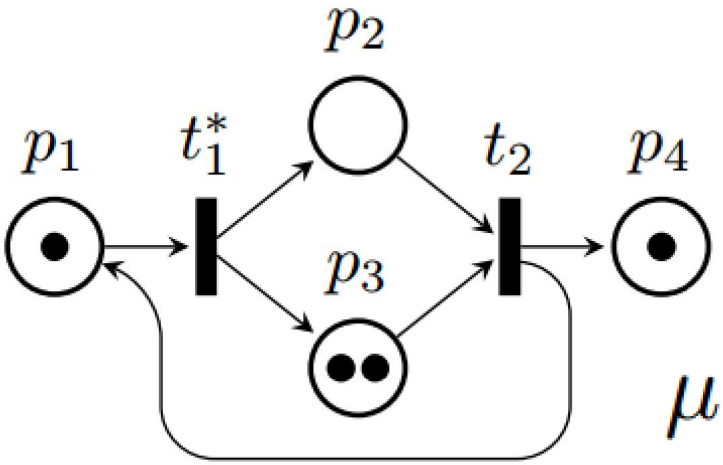
w – обозначение позиции, в которой накапливается бесконечное число фишек

Моделирование сети Петри методом дерева достижимости

Виды вершин дерева достижимости:

- Граничная: вершина, которая рассматривается на текущем шаге моделирования.
- Терминальная: вершина, из которой нет выходящих дуг (т. е. при данной маркировке нет ни одного разрешенного перехода).
- Внутренняя: вершина, которая уже была рассмотрена при построении дерева достижимости.
- Дублирующая: вершина, которая имеет маркировку, которая уже встречалась в дереве.

Матричный подход к моделированию сети Петри



$$\mu = [1, 0, 2, 1]$$

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

Моделирование сети Петри, представленной в матричном виде

Изменение маркировки сети Петри описывается с помощью формулы:

μ‘=μ +W*e

Где v – вектор переходов

μ‘ – новая маркировка

S = (t1, t2, t1) вектор v = [2, 1]

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

Физическая интерпретация сеть Петри при моделировании компьютерных сетей

Позиции – место хранения данных;
Метки – передаваемые данные;
Переходы – обработка данных.

Позиции – обработка данных;
Метки – передаваемые данные;
Переходы – место хранения данных.

Недостатки сетей Петри

Как отмечает профессор Массачусетского технологического института, США, Карл Хьюит (англ. Carl Hewitt), сетям Петри присущи следующие **недостатки**:

- они моделируют управление потоком, но не сам поток данных;
- сложность описания одновременных действий, происходящих во время вычислительного процесса;
- физическая интерпретация перехода в сетях Петри весьма сомнительна.

Виды сетей Петри

Виды сетей Петри

Некоторые виды сетей Петри:

- ***Временна́я*** сеть Петри — переходы обладают весом, определяющим продолжительность срабатывания (задержку).
- ***Стохастическая*** сеть Петри — задержки являются случайными величинами.
- ***Функциональная*** сеть Петри — задержки определяются как функции некоторых аргументов, например, количества меток в каких-либо позициях, состояния некоторых переходов.
- ***Цветная*** сеть Петри — метки могут быть различных типов, обозначаемых цветами, тип метки может быть использован как аргумент в функциональных сетях.
- ***Ингибиторная*** сеть Петри — возможны ингибиторные дуги, запрещающие срабатывания перехода, если во входной позиции, связанной с переходом ингибиторной дугой, находится метка.
- ***Иерархическая*** сеть — содержит не мгновенные переходы, в которые вложены другие, возможно, также иерархические, сети. Срабатывание такого перехода характеризует выполнение полного жизненного цикла вложенной сети.
- ***WF-сети*** Петри - подкласс сетей Петри, называемый также сетями потоков работ. Формализм WF-сетей введён Вил ван дер Аальстом (англ. Wil van der Aalst) для моделирования потоков работ в workflow-системах.

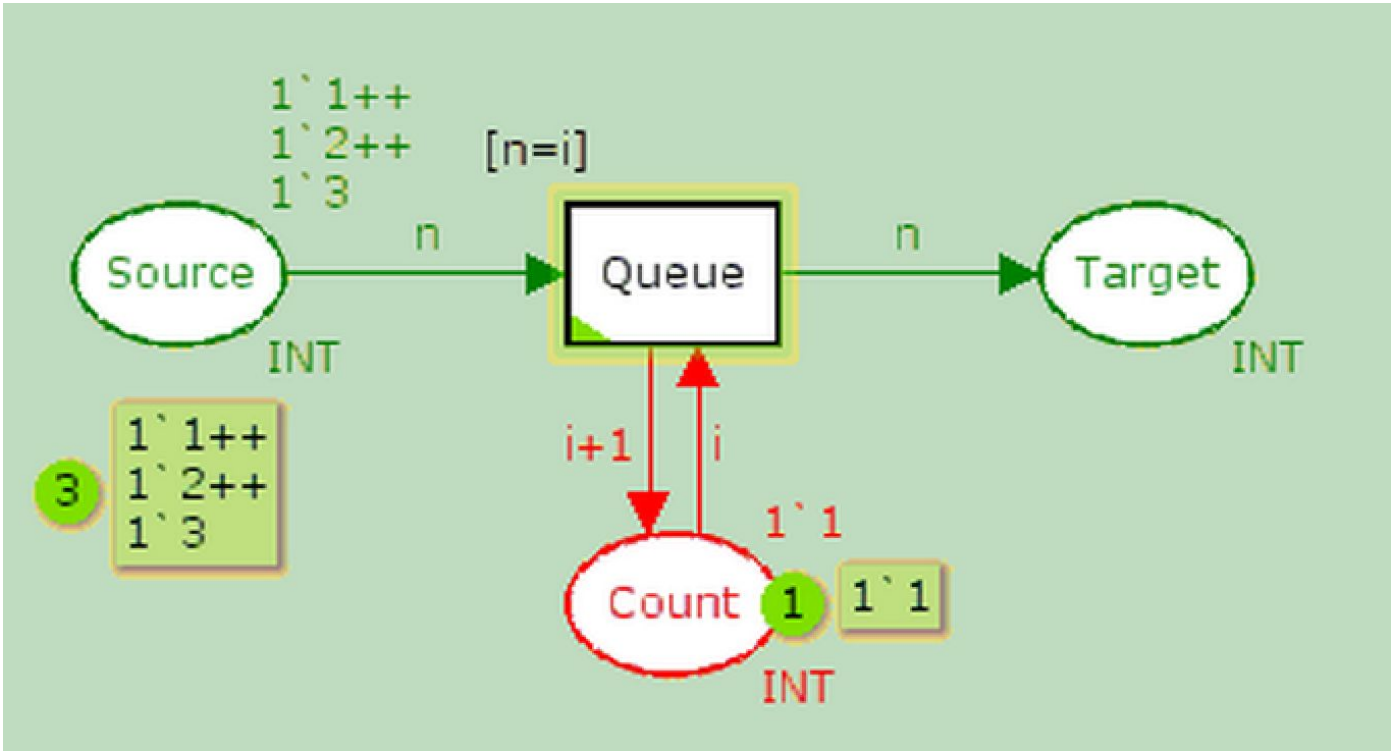
В 1974 году Тилак Аджервала показал, что ингибиторная сеть Петри является универсальной алгоритмической системой.

Цветные (раскрашенные) сети Петри (Coloured Petri Net)

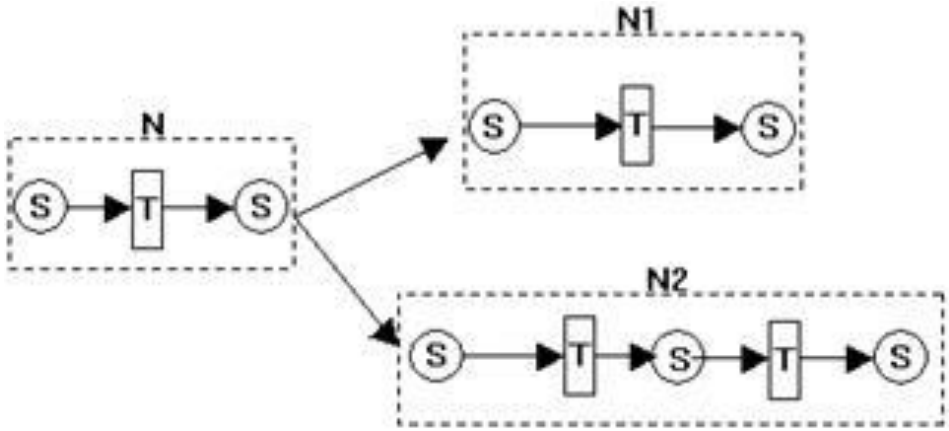
N=(P,T,Pre,Post,C,cd), где

C – множество цветов,

cd : P ∪ T → C – отображение множества C на вершины и узлы сети Петри



Иерархическая сеть Петри



В иерархической сети Петри каждой позиции и каждому переходу может быть приписана сеть Петри

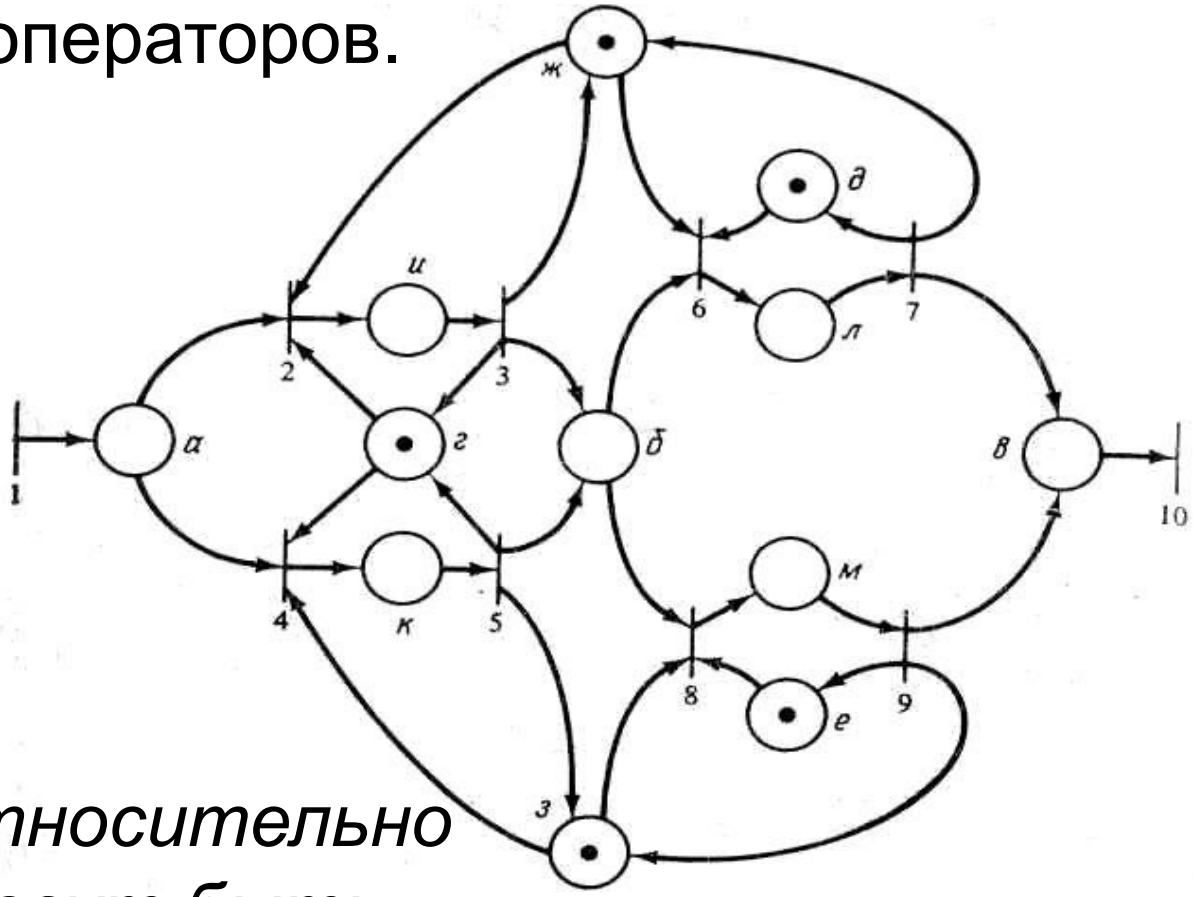
Преимущества: декомпозиция задачи, следовательно, упрощение модели

Недостатки сетей Петри, выделенные Карлом Хьюитом (англ. Carl Hewitt)

- сети Петри имеют следующее ограничение: они моделируют управление потоком, но не сам поток данных;
- сложность описания одновременных действий, происходящих во время вычислительного процесса;
- физическая интерпретация перехода в сетях Петри весьма сомнительна.

Громоздкость сети Петри для моделирования параллельных процессов

Модель автомата-продавца, состоящего из трех автоматов и двух операторов.



Т.е. модели даже относительно простых систем могут быть достаточно громоздкими

Графовая грамматика (graph grammar)
Граф-трансформирующая система (graph rewriting system)

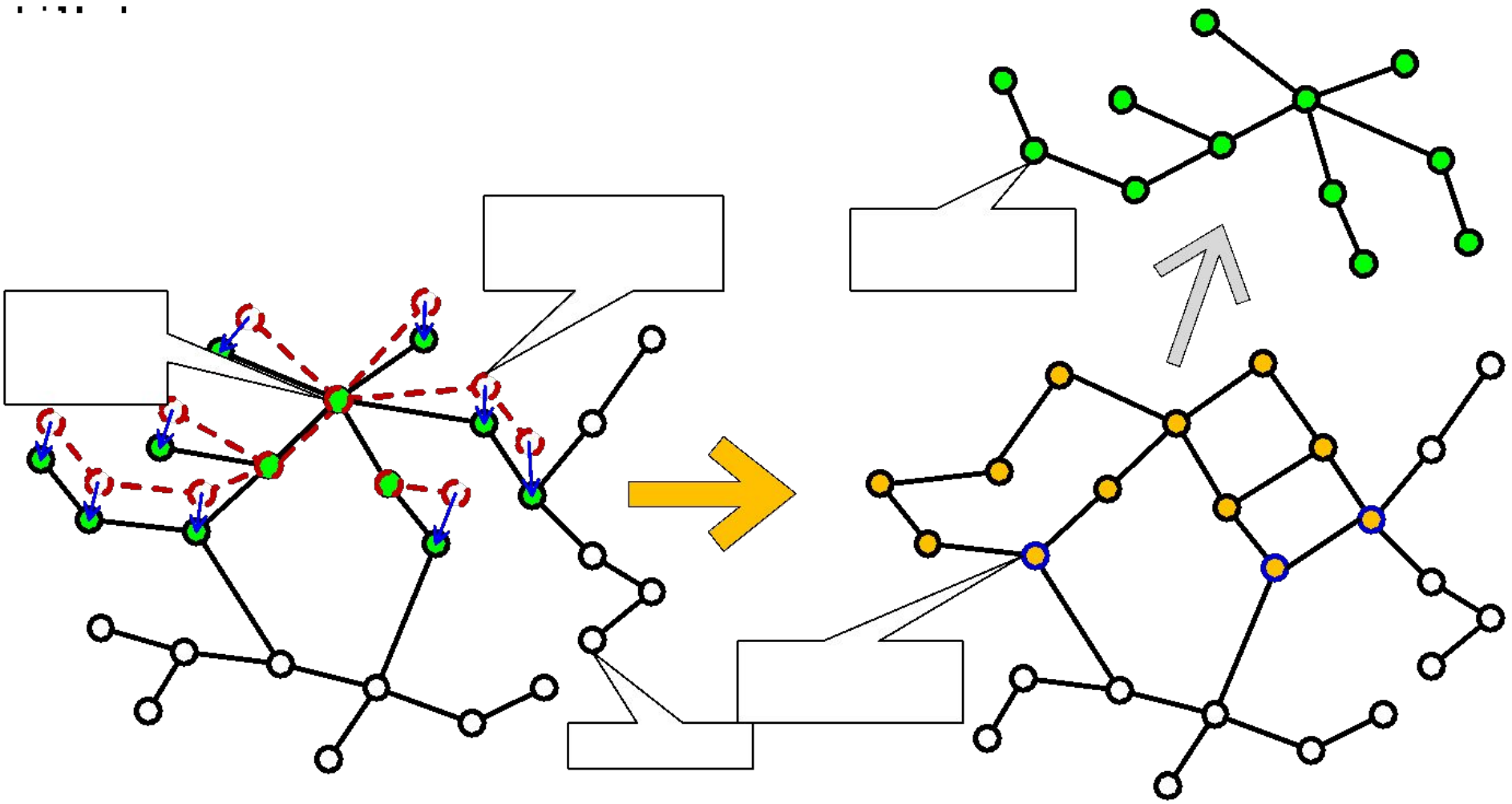
GT={M, D, E}, где

M и D – «материнский» («mother») и «дочерний» («daughter») подграфы,

E – механизм встраивания дочернего подграфа в трансформируемый граф H («host»).

Продукция – это описание элементарного преобразования графа.

Операция трансформации графовой грамматики (продукция)



Существующие граф-трасформирующие системы (графовые грамматики)

Графовые грамматики

Tree Grammar

Transformation Grammar

Head-driven Phrase Structure Grammar (HPSG)

Node Label Controlled (NLC)

Neighbourhood Controlled Embedding (NCE)

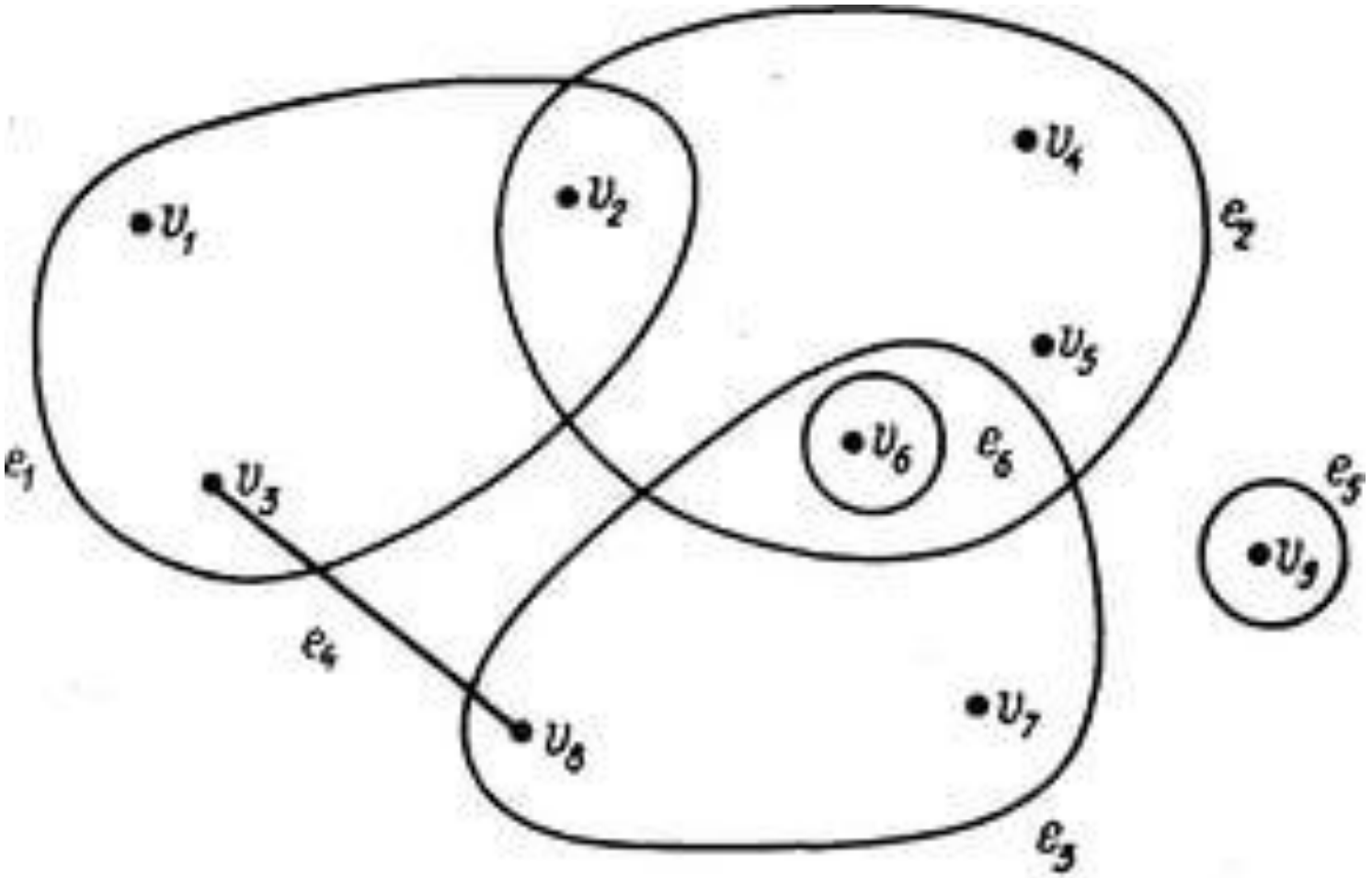
Гиперграфовые грамматики

Synchronous Hyperedge Replacement Grammar (SHRG)

Adaptive Synchronous Hyperedge Replacement Grammar (ASHRG)

Граф и гиперграф

Гиперграф – граф, где ребро может объединять более, чем две вершины графа. Такое ребро называется гиперребром.



Node Label Controlled (NLC) grammar

NLC работает с графом, узлы которого ассоциируются с символьными метками. Метки делятся на терминальные и нетерминальные. Узлы с нетерминальными символами могут подвергаться модификации.

Формальное описание NLC

NCL – это пятерка G=(Σ,Δ,P,C,S), где
Σ - алфавит терминальных символов/меток,
Δ - алфавит нетерминальных символов/меток,
P – конечный набор продукций,
C – набор правил соединения дочернего подграфа к host-графу,
S – начальный host-граф.

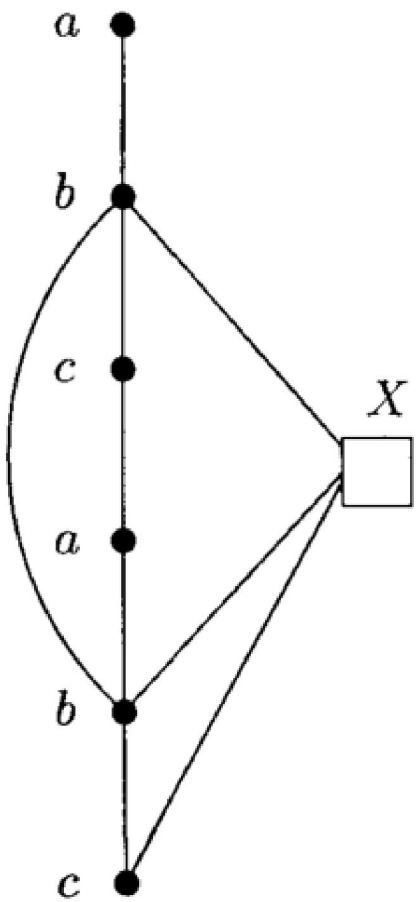
Продукция:

X→D, где X – нетерминальная метка,D – дочерний граф, на который заменяется узел графа с меткой X. . D – это ненаправленный граф, метки вершин которого могут быть как терминальными, так и нетерминальными.

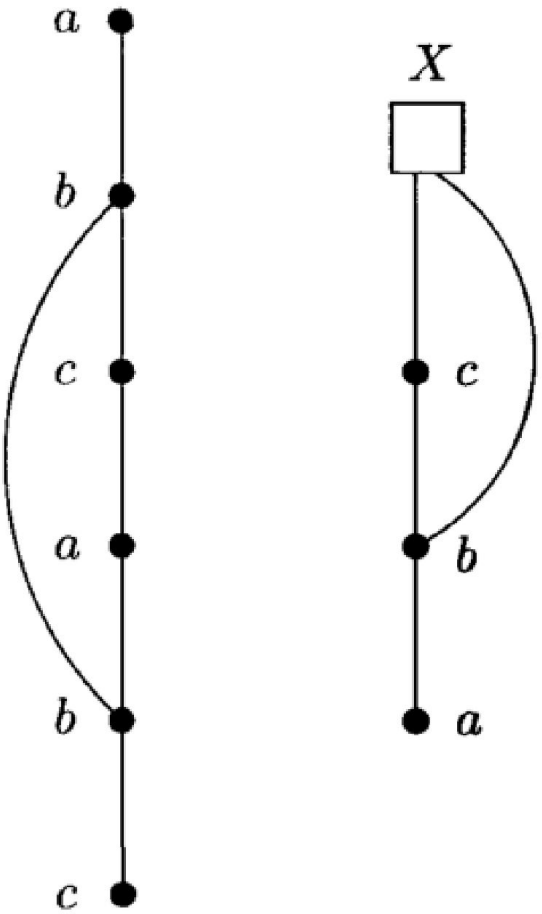
Инструкция соединения – это правило, с помощью которого происходит соединение дочернего и host-графов. Это двойка (μ,δ), где μ и δ являются терминальными или нетерминальными символами, причем μ -метка узла в H, δ метка узла в D. Такая инструкция значит, что метки μ и δ должна быть соединены с помощью ребра.

Если в H нет метки μ, или в D не присутствует метка δ, то продукция считается запрещенной и не может быть выполнена.

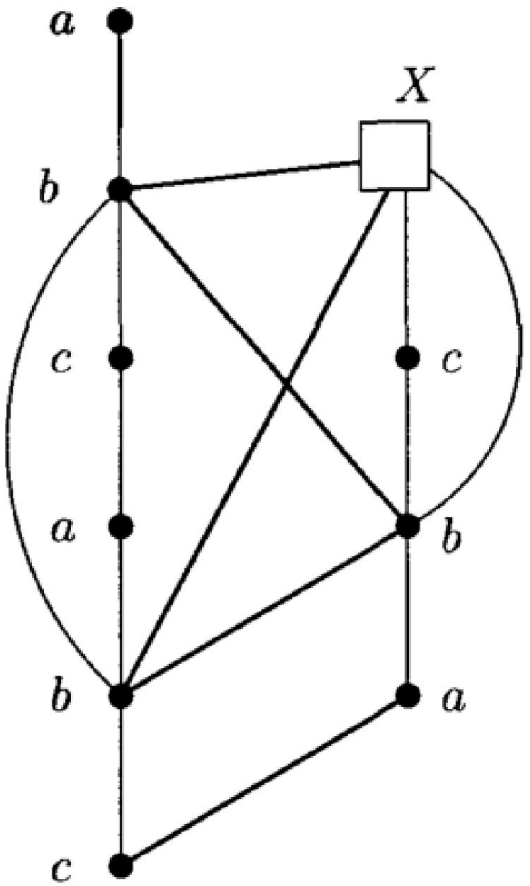
Пример использования NLC



Исходный граф



Граф с удаленной вершиной и дочерний граф



Результирующий граф

Инструкции соединения: (c,a), (b,b)
X – нетерминальная вершина

Атрибутная грамматика Н. Хомского

Формальная грамматика – это G = (T, N, P, S), где T, N, P, Z, соответственно, множество терминальных и нетерминальных символов, множество правил вывода и начальный символ.

В атрибутной грамматике каждому символу x приписывается множество атрибутов A(x), а с каждой продукцией ассоциируется правило модификации атрибутов, приписанных к символу.

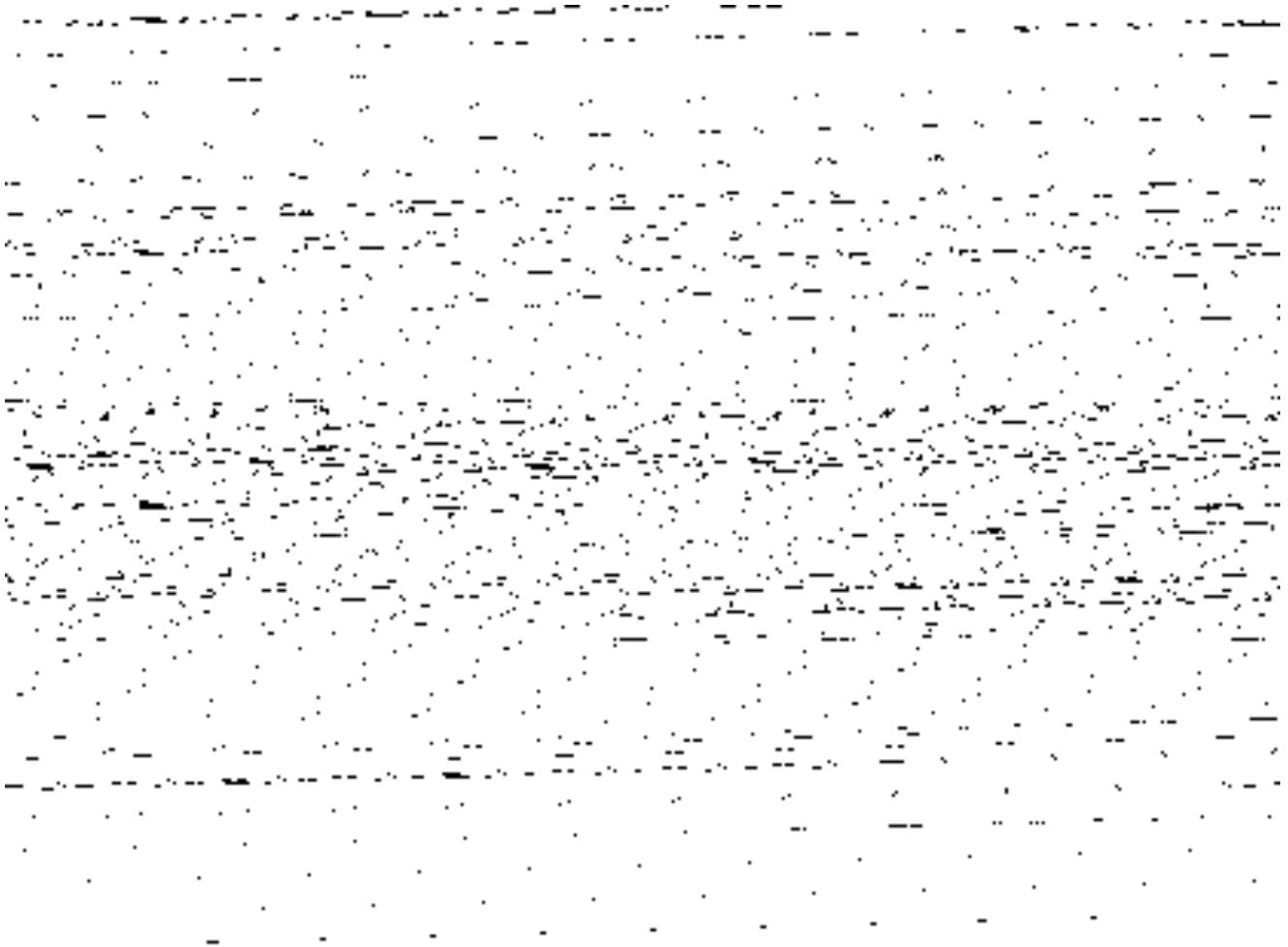
$a_0(i_0) = fpa_0(i_0)(a_1(i_1), \ldots , a_j(i_j))$, где (7)

$a(i_0) \in A(i_0)$ – один из атрибутов, приписанных к символу i;

fpa_0 – функция, зависящая от значения всех других атрибутов в синтезированной строке терминалов и нетерминалов.

Атрибутная грамматика может быть использована и для синтеза древовидной структуры, если считать синтезированную цепочку символов узлами-потомками той вершины графа, которая помечена нетерминальным символом. Недостатком такой грамматики является то, что на может работать только с графами топологии тип дерево.

Атрибутная грамматика Н. Хомского (пример)



ОА-грамматика

OAG={A,L,P,I,G},

где

A – множество атрибутов ИП ($A \subseteq N$);

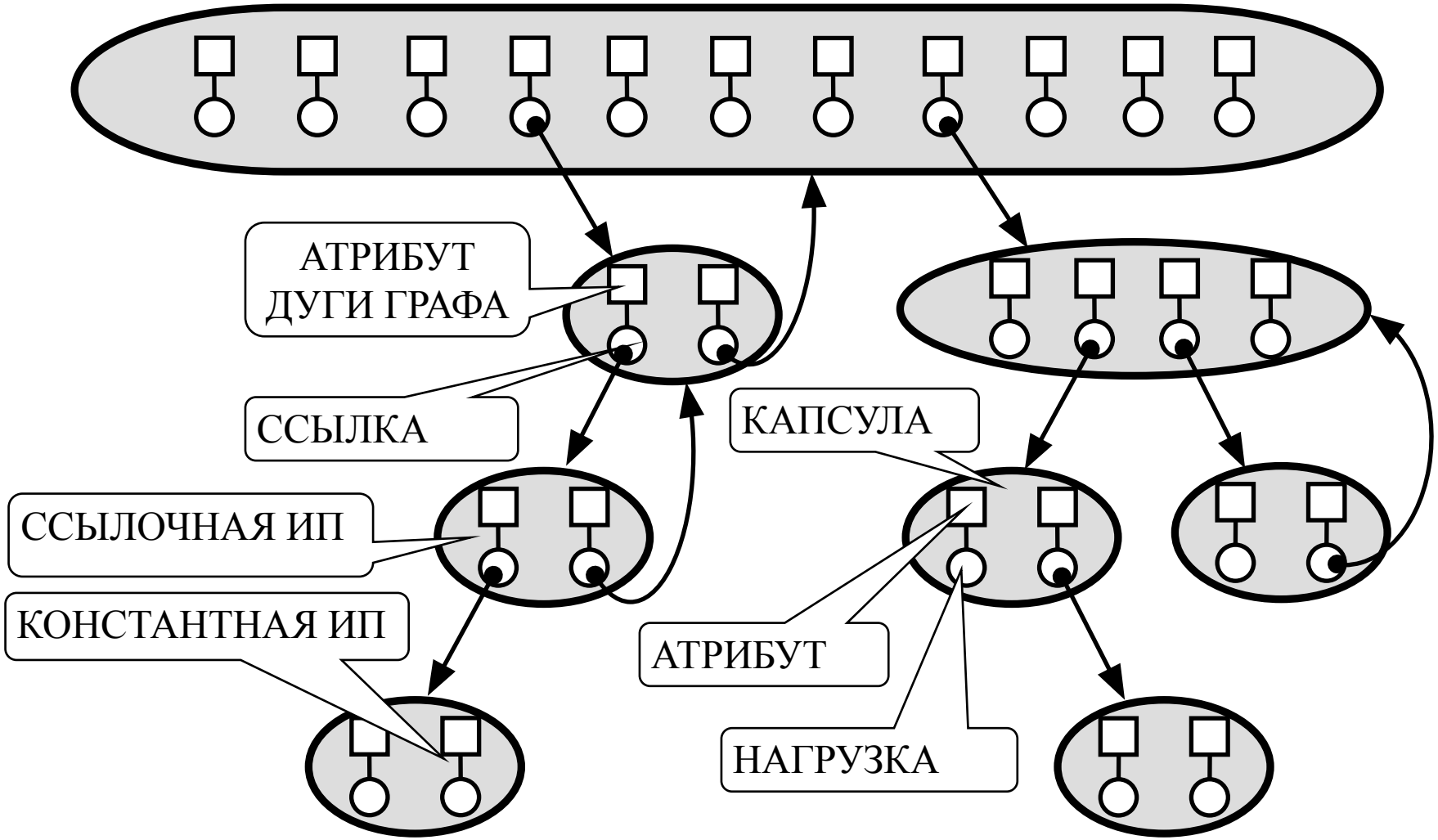
L – множество нагрузок ($L=\{const \cup \Omega\}$, где const – множество констант), Ω - множество индексов ИК (каждая ИК, в том числе синтезированная во время трансформации графа, получает свой уникальный индекс);

G – исходный граф, который будет подвергаться трансформации;

P – совокупность правил преобразования ОА-графа.

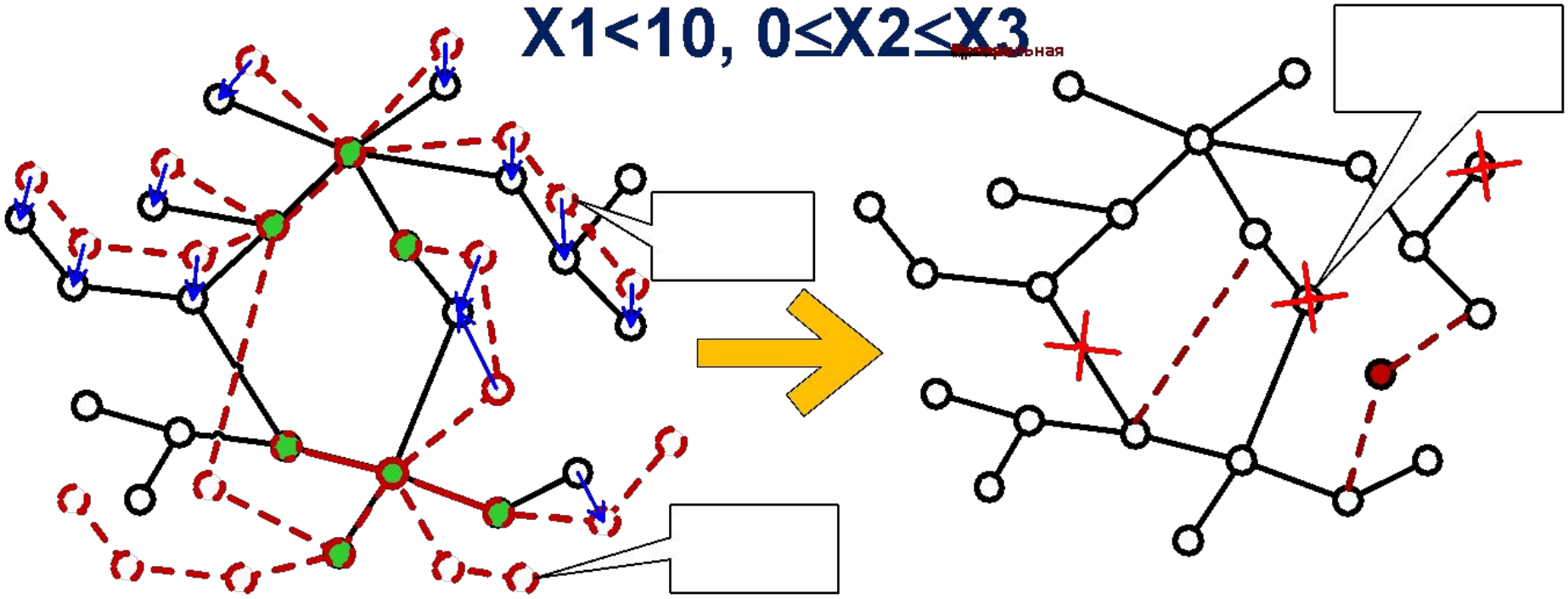
I – множество индексов правил ОА-грамматики ($I \subseteq N$).

Объектно-атрибутный граф



Нотация продукции ОА-грамматики

.. .



{Atr=10} {Atr2=0} : x+y>(x-y)*2→ {SemProp={Atr=x}*{Atr=y}}

Нотация продукции ОА-грамматики (общие обозначения в продукции)

→ – знак продукции

→̲ – замена подграфа, совпавшего с дочерним графом;

: – условие (ограничение на переменные)

= информационная пара;

Point{…} – информационная капсула (Point – указатель на ИК)

Atr=Const – константная ИП;

Atr=Point – ссылочная ИП;

Atr=Point{…} – указатель на ИК в нагрузке ИП;

~ – элементы не входят в множество.

{Atr=10} {Atr2=0} : x+y>(x-y)*2→ {SemProp={Atr=x}*{Atr=y}}

Нотация продукции ОА-грамматики (обозначения в левой части продукции)

{…} – неупорядоченное множество ИП;

(…) – упорядоченное множество ИП;

[…] - необязательный элемент множества;

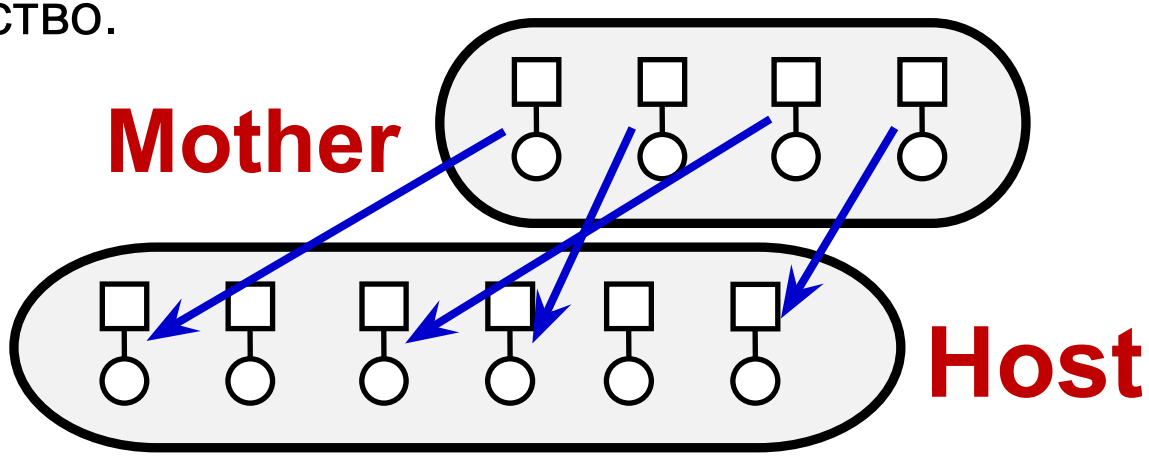
<…> - недопустимый элемент множества;

⊕ - множество исключающее ИЛИ – в ИК присутствует только один элемент из перечисленных;

|| - множество ИЛИ – множество включает один или несколько элементов из перечисленных;

=( ) – ограничение на константу в нагрузке ИП;

S^{NF} – неполное множество.



Нотация продукции ОА-грамматики (обозначения в центральной части продукции)

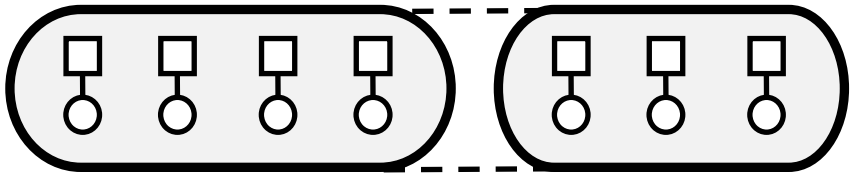
+,-,*,/ и т.д. – арифметические операции для определение ограничений на переменные;

=, <>, <=, &&, ||, ! и т.д. – логические операции для определения ограничений на переменные

{Atr=10} {Atr2=0} : x+y>(x-y)*2→ {SemProp={Atr=x}*{Atr=y}}

Нотация продукции ОА-грамматики (обозначения в правой части продукции)

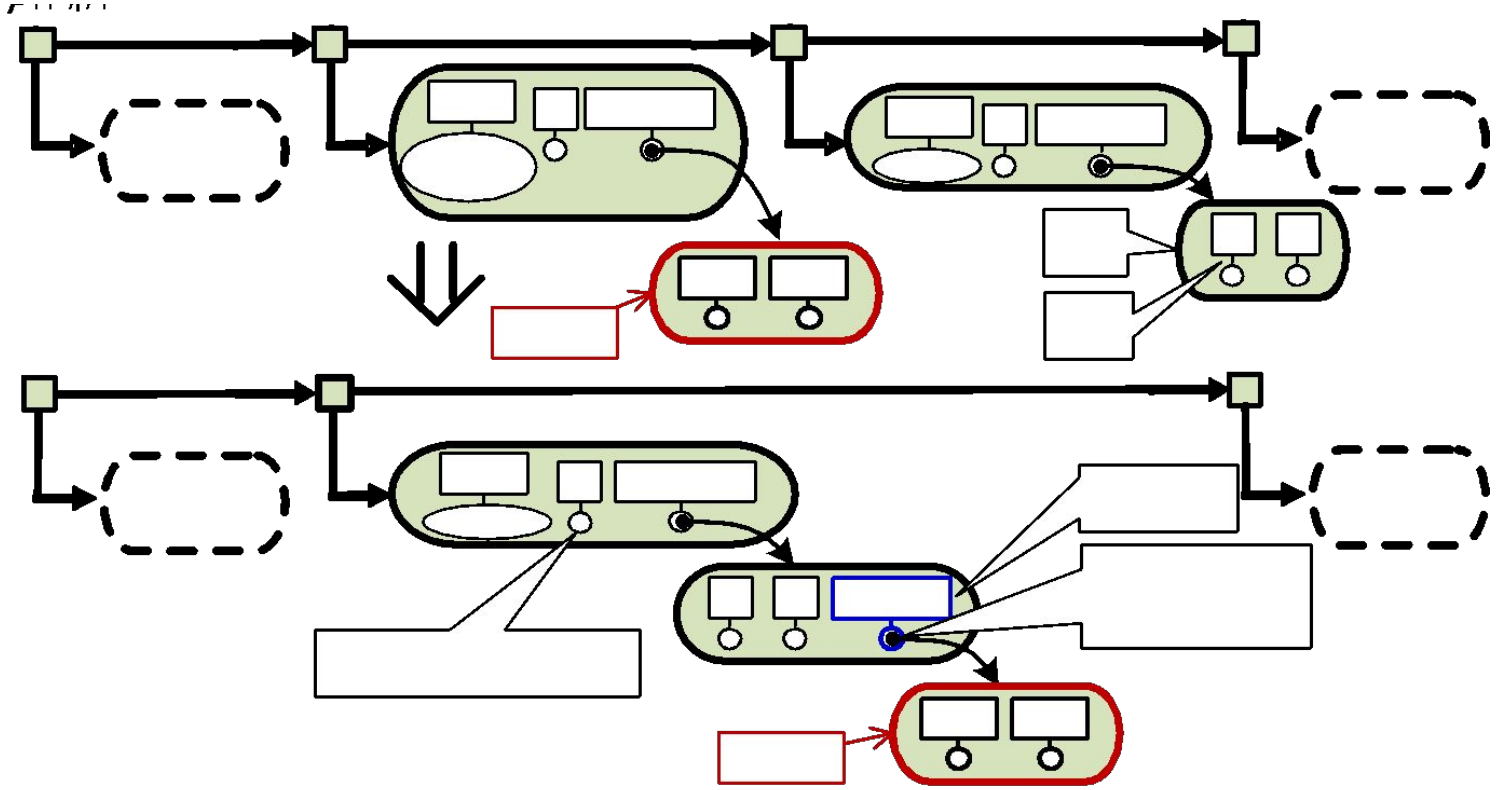
* – Конкатенация (сцепление) двух ИК;



Atr=*{…} – Конкатенация к ИК по адресу из нагрузки ИП;
#Point - копирование ИК, на который указывает Point;
S ^{F} - первый элемент в множестве ИП;
S ^{L} - последний элемент в множестве;
’S - удаление первого элемента множества;
S ’ - удаление последнего элемента множества;
Variable(…) - значение переменной по умолчанию.

{Atr=10} {Atr2=0} : x+y>(x-y)*2→ {SemProp={Atr=x}*{Atr=y}}

Пример продукций ОА-грамматики (1)



{Elem={POS=ADVERB_DEGREE SemProp=temp}
Elem={POS= ADJECT}} →
{Elem={POS=ADJECT SemProp=*{DEGREE=temp}}}

Продукции для разбора числительных английского языка

1. {LangConstr=Numaral DigitPalce=1 SemProp=tmp1 Ordinal=0} {LangConstr=Numaral Mul=100 SemProp=tmp2 Ordinal=**x**} →

{ LangConstr=Numaral DigitPalce=100 SemProp=(tmp1*tmp2) Ordinal=**x** }

2. ({LangConstr=Numaral DigitPalce=100 SemProp=tmp1 Ordinal=x1 } ||

((["and"] { -"- DigitPalce=10 SemProp=tmp2 Ordinal=x2}) || (["and"]{ -"- DigitPlace=1 SemProp=tmp3 Ordinal=x3}))

⊕ (["and"] { -"- DigitPalce=19 SemProp=tmp3 Ordinal=x2}) :

$(((x1,x2,x3)^{L}=0),(x1+x2+x3=0))$ || $((x1,x2,x3)^{L}=1),(x1+x2+x3=1))$ →

{ LangConstr=Num SemProp=(tmp1(0)+tmp2(0) +tmp3(0)) }

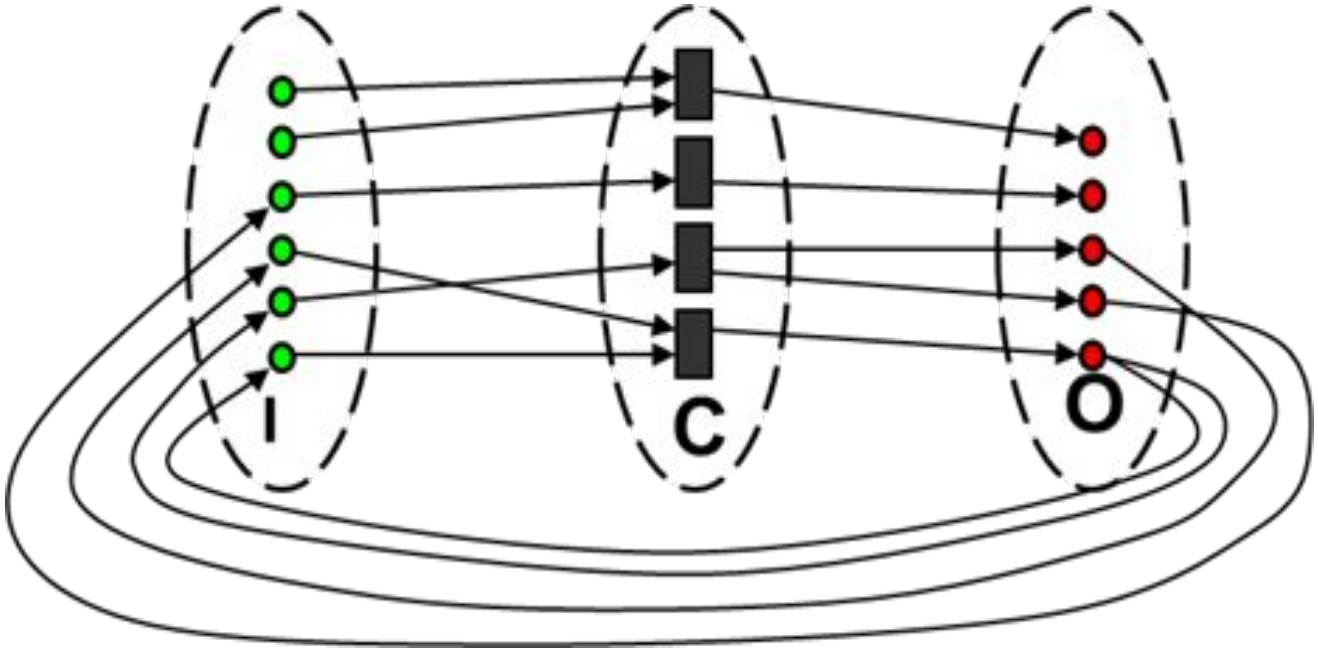
3.({LangConstr=Num SemProp=(**TmpPrev**>tmpMul) Ordinal=0} {LangConstr=Num SemProp=(tmpNum<1000)} } {LangConstr=Numaral Mul=(tmp1> **tmpMul**) Ordinal=x }) ⊕

({LangConstr=Num SemProp=(TmpPrev>tmpMul) Ordinal=0} {LangConstr=Num SemProp=(tmpNum<1000)} Ordinal=x}) ⊕

({LangConstr=Num SemProp=(tmpNum<1000) Ordinal=0} } {LangConstr=Numaral Mul=tmpMul} Ordinal=x)

→ { LangConstr=Num SemProp=(TmpPrev(0)+ tmpNum* tmpMul(1)) Ordinal=x }

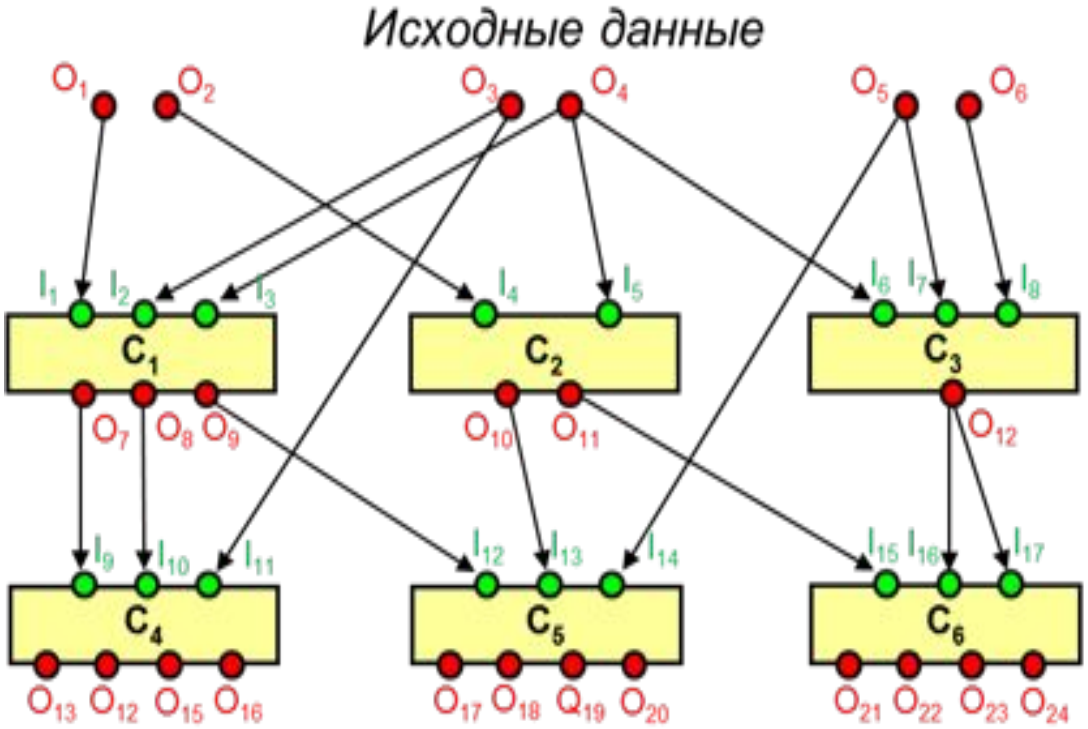
A-сеть

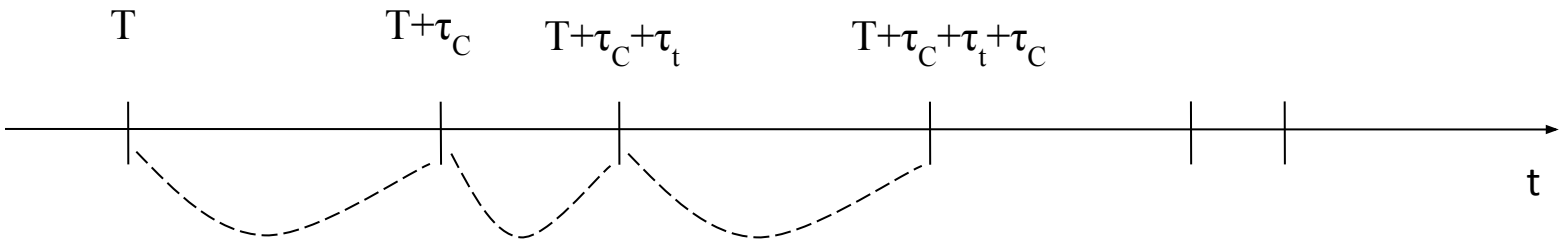


$A=\{I,C,O,EC,CO,OI,IM,OM\}$,

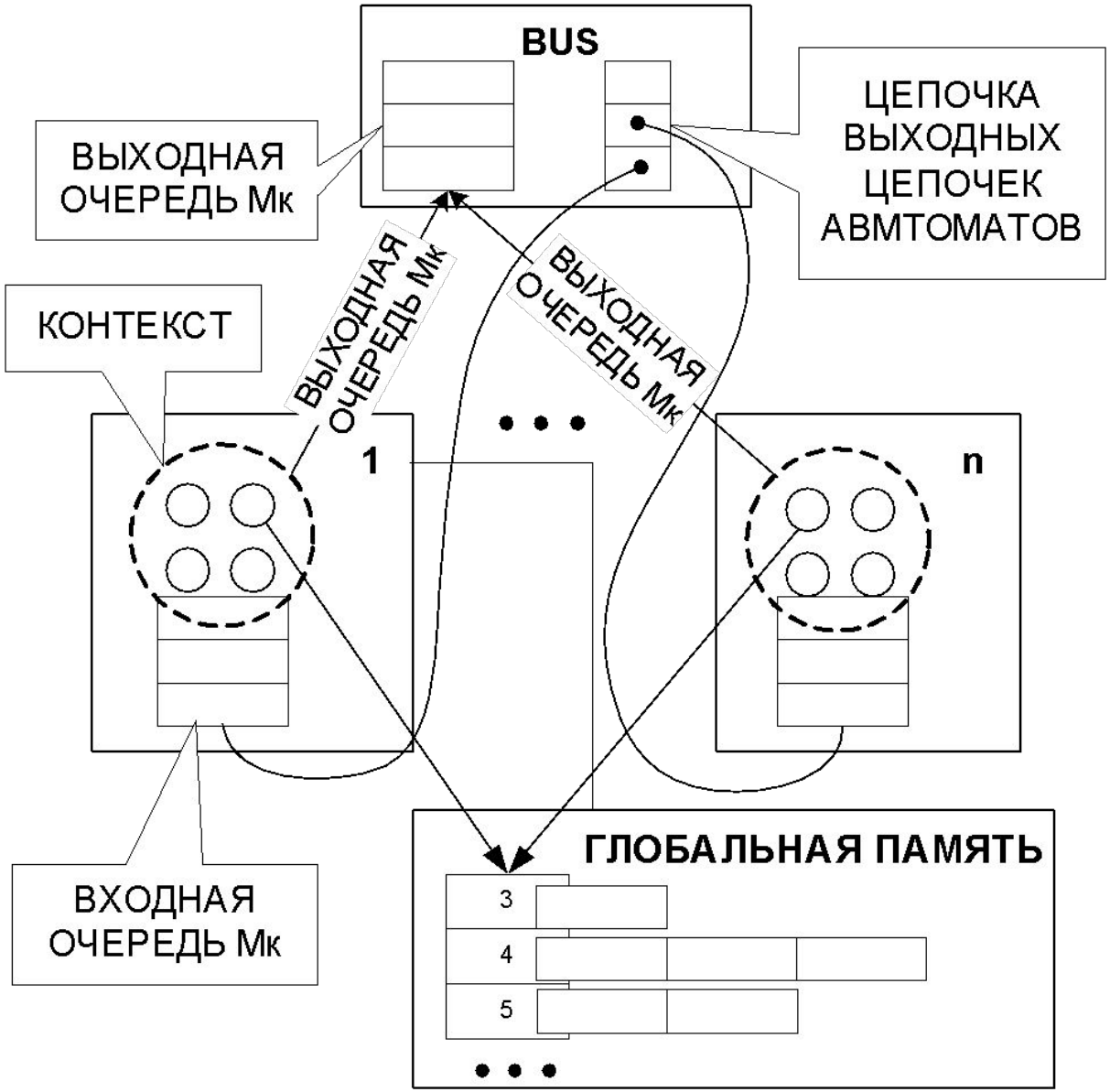
Где I – множество входных узлов;
C – множество вычислительных узлов;
O – множество выходных узлов;
IC: I → C – множество дуг из входных вершин в вычислительные узлы;
CO: C →O – множество дуг из вычислительных узлов в выходные узлы;
OI: O → I – множество дуг, соединяющих выходные вершины со входными;
IM – вектор маркировок входных узлов;
OM – вектор маркировок выходных узлов.

A-сеть


Исходные данные
O1 O2 O3 O4 O5 O6
I1 I2 I3 I4 I5 I6 I7 I8
C1 C2 C3
O7 O8 O9 O10 O11 O12
I9 I10 I11 I12 I13 I14 I15 I16 I17
C4 C5 C6
O13 O12 O15 O16 O17 O18 Q19 O20 O21 O22 O23 O24


T
T+τC
T+τC+τt
T+τC+τt+τC
t

А-автоматная сеть



ОА-автомат

$$AF=\{A, L, K, MkIn, MkOut, W, F\},$$

где

A и L – это множество атрибутов милликоманд и множество данных в нагрузке ИП (милликоманда – это двойка Mk=(a,l), где), $a \in A$, $l \in L$;

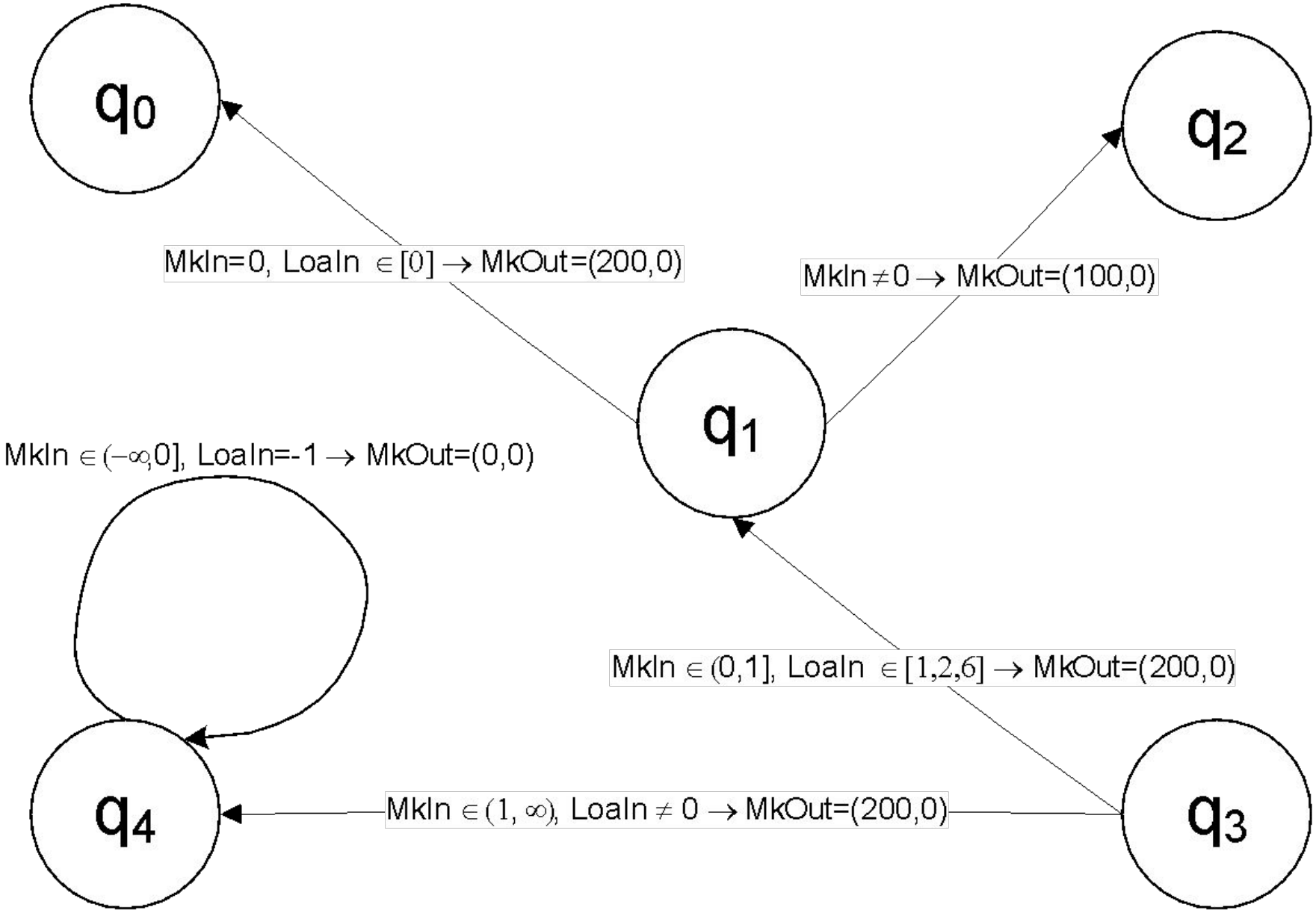
W – общая память;

MkIn и MkOut – очереди входных и выходных милликоманд, соответственно;

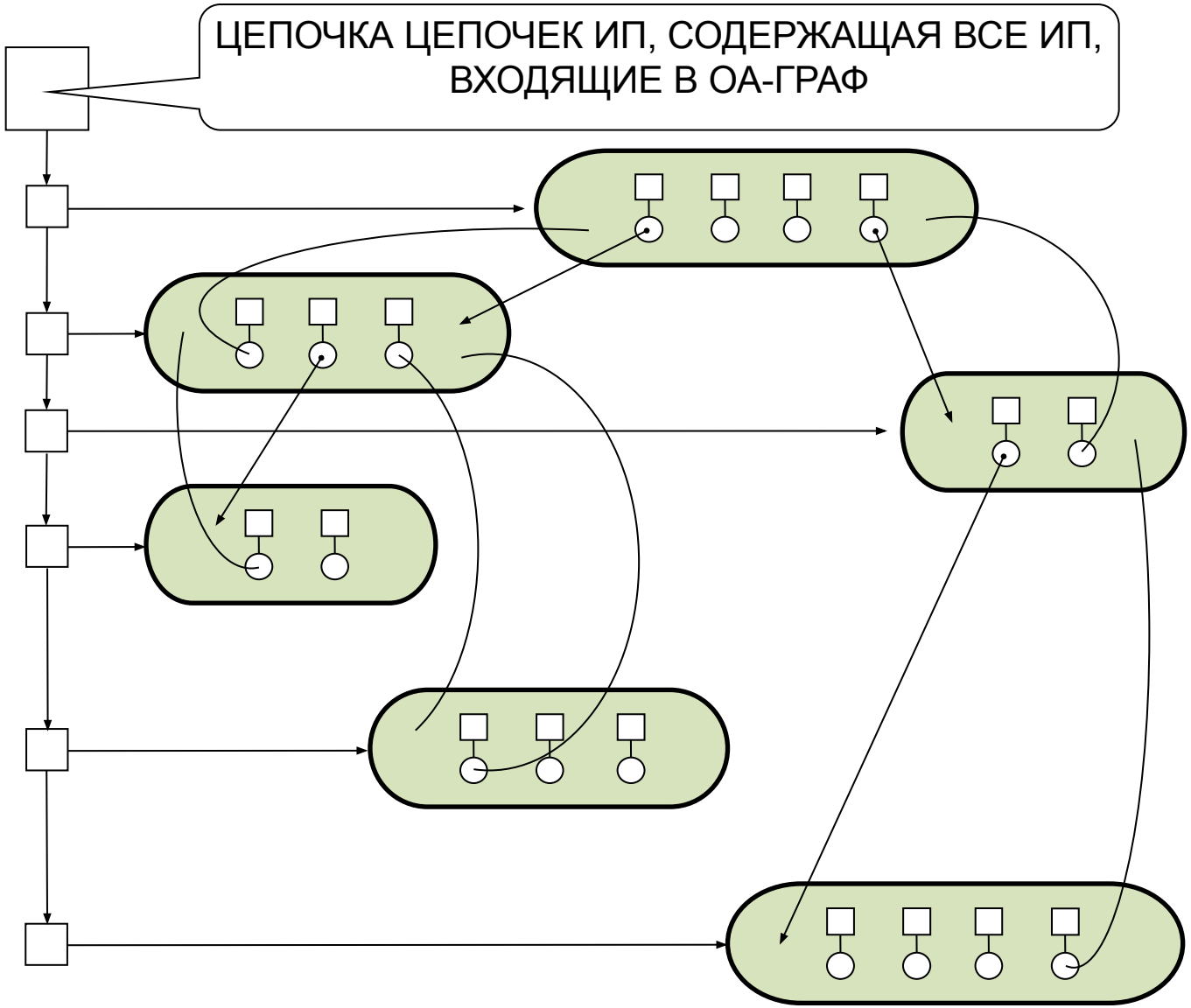
K – контекст ОА-автомата;

F – функция изменения контекста.

Пример ОА-автомата



Память А-автоматной сети



Выполнение ОА-автоматной сети

Выполнение автоматной сети – это последовательность изменения состояний ОА-автоматов, входящих в ОА-сеть. Каждое изменение состояния ОА-автомата, происходит при обработки милликоманды из входной очереди милликоманд.

Начальная конфигурация ОА-автоматной сети – совокупность милликоманд, находящихся во входных очередях милликоманд всех устройств в начальный момент модельного времени.

Моделирование заканчивается в тот момент, когда все очереди милликоманд оказываются пустыми и ни одно и все ОА-автоматы заканчивают обработку данных.

Спасибо за внимание!

Программные продукты для моделирования сетей Петри

CPNTools (http://www.daimi.au.dk/CPNTools/) – цветные сети Петри (университете г.Орхуса (Дания))

Список литературы

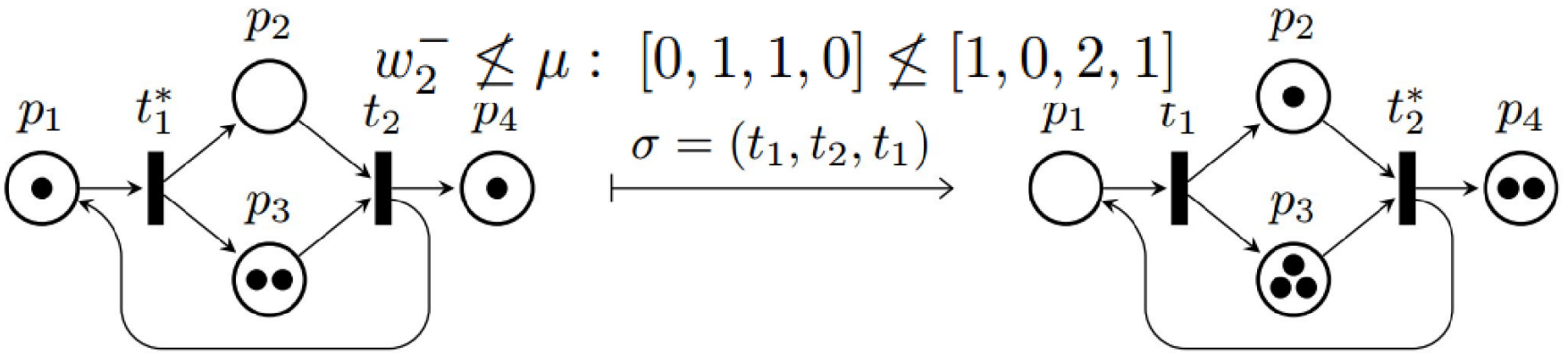
1. Есипов Б.А. Методы оптимизации и исследование операций. Конспект лекций: учеб. пособие / Б.А.Есипов,. –Самара: Изд-во Самар. гос. аэрокосм. ун-та, 2007. – 90 с.
2. Гладких Б. А. Методы оптимизации и исследование операций для бакалавров информатики. Ч. I. Введение в исследование операций. Линейное программирование: Учебное пособие. — Томск: Изд-во НТЛ, 2009. — 200 с.
3. Вентцель Е.С. Исследование операций. Задачи, принципы, методология. М.: Высш. шк.,2001.
4. Зарубин В.С. Математическое моделирование в технике: Учеб. для вузов / Под ред. В.С. Зарубина, А.П. Крищенко. – 2-е изд., стереотип. – М.: Изд-во МГТУ им. Баумана, 2003. – 496 с. (Сер. Математика в техническом университете; Вып. XXI, заключительный).

Дополнения

Реактивные системы – программно-аппаратные комплексы, где аппаратная составляющая используется для согласования управляющей (программной) логики с реальной средой (механизмы, датчики и т.п.). Для них часто используется параллелизм для ускорения работы системы.

Моделирование сети Петри, представленной в матричном виде

$$\mu = [1, 0, 2, 1]$$



$$w_1^- \leq \mu : \ [1, 0, 0, 0] \leq [1, 0, 2, 1]$$