6. Моделирование параллельных процессов

Практически любая более или менее сложная система имеет в своем составе компоненты, работающие одновременно, или, как принято говорить на языке техники, параллельно. Параллельно работающие подсистемы могут взаимодействовать самым различным образом, либо вообще работать независимо друг от друга. Способ взаимодействия подсистем определяет вид параллельных процессов, протекающих в системе. В свою очередь, вид моделируемых процессов влияет на выбор метода их имитации.

6.1.Виды параллельных процессов в сложных системах

Асинхронный параллельный процесс — такой процесс, состояние которого не зависит от состояния другого параллельного процесса (ПП).

Пример асинхронных ПП из области вычислительной техники: выполнение вычислений процессором и вывод информации на печать.

Синхронный ПП — такой процесс, состояние которого зависит от состояния взаимодействующих с ним ПП.

Пример синхронного ПП: работа торговой организации и доставка товара со склада (нет товара — нет торговли).

Один и тот же процесс может быть синхронным по отношению к одному из активных ПП и асинхронным по отношению к другому. Так, при работе вычислительной сети по технологии «клиент-сервер» каждый из узлов сети синхронизирует свою работу с работой сервера, но не зависит от работы других узлов.

Подчиненный ПП — создается и управляется другим процессом (более высокого уровня). Весьма характерным примером таких процессов является ведение боевых действий подчиненными подразделениями.

Независимый ПП— не является подчиненным ни для одного из процессов. Скажем, после запуска неуправляемой зенитной ракеты ее полет можно рассматривать как независимый процесс, одновременно с которым самолет ведет боевые действия другими средствами.

Способ организации параллельных процессов в системе зависит от физической сущности этой системы.

Остановимся несколько подробнее на особенностях реализации параллельных процессов в вычислительных системах (ВС). Это обусловлено следующей причиной.

Разработка и использование любой ИМ предполагает ее программную реализацию и исследование с применением ВС. Поэтому для реализации моделей, имитирующих параллельные процессы, в некоторых случаях применимы механизмы, используемые при выполнении параллельных вычислений.

Вместе с тем, реализация параллельных процессов в ВС имеет свои особенности:

• на уровне задач вычислительные процессы могут быть истинно параллельными только в многопроцессорных ВС или вычислительных сетях;

• многие ПП используют одни и те же ресурсы, поэтому даже асинхронные ПП в пределах одной ВС вынуждены согласовывать свои действия при обращении к общим ресурсам;

• в ВС дополнительно используется еще два вида ПП: родительский и дочерний ПП; особенность их состоит в том, что процесс-родитель не может быть завершен, пока не завершатся все его дочерние процессы.

В силу перечисленных особенностей для организации взаимодействия параллельных процессов в ВС используются три основных подхода:

• на основе «взаимного исключения»;

• на основе синхронизации посредством сигналов;

• на основе обмена информацией (сообщениями).

«Взаимное исключение» предполагает запрет доступа к общим ресурсам (общим данным) для всех ПП, кроме одного, на время его работы с этими ресурсами (данными).

Синхронизация подразумевает обмен сигналами между двумя или более процессами по установленному протоколу. Такой «сигнал» рассматривается как некоторое событие, вызывающее у получившего его процесса соответствующие действия.

Часто возникает необходимость передавать от одного ПП другому более подробную информацию, чем просто «сигнал-событие». В этом случае процессы согласуют свою работу на основе обмена сообщениями.

Перечисленные механизмы реализуются в ВС на двух уровнях — системном и прикладном.

Механизм взаимодействия между ПП на системном уровне определяется еще на этапе разработки ВС и реализуется в основном средствами операционной системы (частично — с использованием аппаратных средств).

На прикладном уровне взаимодействие между ПП реализуется программистом средствами языка, на котором разрабатывается программное обеспечение.

Наибольшими возможностями в этом отношении обладают так называемые языки реального времени (ЯРВ) и языки моделирования.

Языки реального времени — это языки, предназначенные для создания программного обеспечения, работающего в реальном масштабе времени, например для разработки различных автоматизированных систем управления (предприятием, воздушным движением и т. д.). К ним, в частности, относятся: язык *Ада,* язык *Модула* и практически единственный отечественный язык реального времени — *Эль-76* (использовавшийся в многопроцессорных вычислительных комплексах семейства «Эльбрус»).

6.2. Методы описания параллельных процессов в системах и языках моделирования

Языки моделирования по сравнению с языками реального времени требуют от разработчика значительно менее высокого уровня подготовки в области программирования, что обусловлено двумя обстоятельствами:

• во-первых, средства моделирования изначально ориентированы на квазипараллельную обработку параллельных процессов;

• во-вторых, механизмы реализации ПП относятся, как правило, к внутренней организации системы (языка) моделирования и их работа скрыта от программиста.

В практике имитационного моделирования одинаково широко используются как процессно-ориентированные языки (системы) моделирования, например *SIMULA*, так и языки, ориентированные на обработку транзактов (например, язык *GPSS).* В тех и других используются аналогичные методы реализации квазипараллелизма, основанные на ведении списков событий. В процессно-ориентированных системах используются списки событий следования, а в транзактных системах — списки событий изменения состояний.

Современные языки и системы моделирования, ориентированные на использование в среде многозадачных операционных систем типа Windows, частично используют их механизмы управления процессами, что делает их применение еще более эффективным. В пакете MATLAB также имеется собственный язык моделирования, и к нему в полной мере можно отнести сказанное выше. Тем не менее во многих случаях оказывается полезным знание общего механизма реализации ПП в языках моделирования.

Рассмотрим его применительно к моделированию на основе тразактов.

В этом случае под событием понимается любое перемещение транзакта по системе, а также изменение его состояния (обслуживается, заблокирован и т. д.).

Событие, связанное с данным транзактом, может храниться в одном из следующих списков.

Список текущих событий. В этом списке находятся события, время наступления которых меньше или равно текущему модельному времени. События с «меньшим» временем связаны с перемещением тех транзактов, которые должны были начать двигаться, но были заблокированы.

Список будущих событий. Этот список содержит события, время наступления которых больше текущего модельного времени, то есть события, которые должны произойти в будущем (условия наступления которых уже определены — например, известно, что транзакт будет обслуживаться некоторым устройством 10 единиц времени).

Список прерываний. Данный список содержит события, связанные с возобновлением обработки прерванных транзактов. События из этого списка выбираются в том случае, если сняты условия прерывания.

В списке текущих событий транзакты расположены в порядке убывания приоритета соответствующих событий; при равных приоритетах — в порядке поступления в список.

Каждое событие (транзакт) в списке текущих событий может находиться либо в активном состоянии, либо в состоянии задержки. Если событие активно, то соответствующий транзакт может быть продвинут по системе; если продвижение невозможно (например, из-за занятости устройства), то событие (и транзакт) переводится в состояние задержки.

Как только завершается обработка (продвижение) очередного активного транзакта, просматривается список задержанных транзактов, и ряд из них пе-

реводится в активное состояние. Процедура повторяется до тех пор, пока в списке текущих событий не будут обработаны все активные события. После этого просматривается список будущих событий. Модельному времени присваивается значение, равное времени наступления ближайшего из этих событий. Данное событие заносится в список текущих событий. Затем просматриваются остальные события списка. Те из них, время которых равно текущему модельному времени, также переписываются в список текущих событий. Просмотр заканчивается, когда в списке остаются события, времена которых больше текущего модельного времени.

В качестве иллюстрации к изложенному рассмотрим небольшой пример.

►*Пусть в систему поступают транзакты трех типов, каждый из которых обслуживается отдельным устройством. Известны законы поступления транзактов в систему и длительность их обслуживания. Таким образом, в системе существуют три параллельных независимых процесса (PI, P2, P3).*

Временная диаграмма работы системы при обслуживании одного транзакта каждого типа показана на рис.2.7.

На рисунке события, относящиеся к процессу P1, обозначены как $C1_i$, относящиеся к P2 и к P3 — соответственно как $C2_i$. и $C3_i$. Моменты времени $t_{вх}$ и $t_{вых}$ соответствуют началу и окончанию обслуживания транзакта.

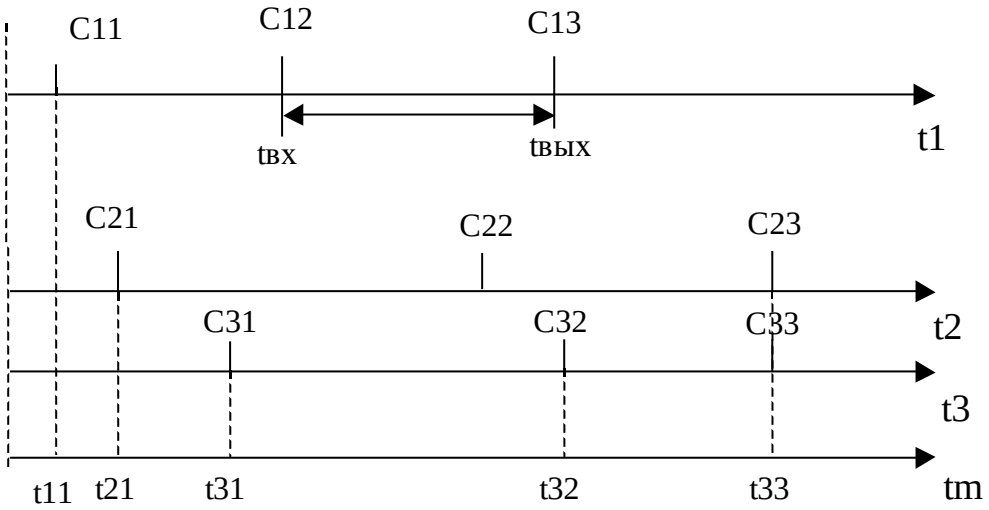


Рис. 5.6

Для каждого процесса строится своя цепь событий, однако списки событий являются общими для всей модели. Формирование списков начинается с заполнения списка будущих событий. Как было отмечено выше, в этот список помещаются события, время наступления которых превышает текущее значение модельного времени. Очевидно, что на момент заполнения списка время наступления прогнозируемых событий должно быть известно. На первом шаге tм=0, и в список будущих событий заносятся события C11, C21, C31. Затем событие с наименьшим временем наступления — C11 — переносится в список текущих событий; если одновременных с ним событий нет, то оно обрабатывается и исключается из списка текущих событий. После этого вновь корректируется список будущих событий и т.д., пока не истечет заданный интервал моделирования.

Динамика изменения списков текущих и будущих событий для рассмотренного примера отражена в приведенной ниже таблице.

t	***Список текущих событий***	***Список будущих событий***
0	*0*	*C11, C21,C31*

t11	*C11*	*C21, C31, C12*
t21	*C21*	*C31, C12, C22*
t31	*C31*	*C12, C22, C32*
t12	*C12*	*C22, C32, C13*
t22	*C22*	*C32, C13, C23*
t32	*C32*	*C13, C23, C33*
t13	*C13*	*C23, C33*
t23	*C23, C33*	

◄

Многие авторы книг по имитационному моделированию считают, что знание механизма ведения списков событий просто необходимо разработчику модели; умение проследить в динамике цепь происходящих в модели событий, во-первых, повышает уверенность создателя модели в том, что она работает правильно и, во вторых, существенно облегчает процесс отладки и модификации модели.

6.3. Применение сетевых моделей для описания параллельных процессов

Этапу программной реализации модели (т. е. ее описанию на одном из языков программирования) должен предшествовать так называемый этап алгоритмизации. Другими словами, прежде чем превратить имитационную модель в работающую программу, ее создатель должен воспользоваться каким-то менее формальным и более наглядным средством описания логики работы будущей программы. Это требование не является обязательным, т.к. при наличии достаточного опыта программа не очень сложной модели может быть написана сразу. Однако при моделировании более сложных систем даже опытные разработчики бывают вынуждены немного «притормозить» на этапе алгоритмизации. Для описания логики работы модели могут быть использованы различные средства: либо русский язык (устный или письменный), либо традиционные схемы алгоритмов, либо какие-то другие «подручные» средства. Первые два варианта являются, как правило, наиболее знакомыми и наиболее часто используемыми. Однако такие схемы совершенно не приспособлены для описания параллельных процессов.

Одним из наиболее элегантных и весьма распространенных средств описания параллельных процессов — описание ***сетями Петри.*** Рассмотрим те основные сведения, которые необходимы с точки зрения реализации технологии имитационного моделирования параллельных процессов.

Сети Петри

При помощи аппарата сетей Петри можно представить динамику функционирования процессов, отражать свойства недетерминированности, асинхронности и параллелизма процессов.

Одно из основных достоинств аппарата сетей Петри заключается в том, что они могут быть представлены как в графической форме (что обеспечивает наглядность), так и в аналитической (что позволяет автоматизировать процесс их анализа).

Карл Адам Петри (1926-2010)--

Петри сформулировал парадигму, которая называется «теория сетей Петри». Его подход можно проиллюстрировать на примере рекурсивных функций. Известно, что общей рекурсивной функции f и элемента n количество необходимых ресурсов для вычисления f(n) не может быть оценено заранее. Вычисления с имеющимся набором ресурсов могут привести к тому, что ресурсов недостаточно. Согласно классической компьютерной архитектуре в этом случае нужно запросить дополнительные ресурсы и начать вычисления заново. Возникает вопрос, существует ли некоторая «растяжимая» архитектура, которая позволяет решить эту проблему? Петри доказал, что это может быть достигнуто: надо, чтобы каждый элемент системы функционировал автономно и система должна работать в асинхронном режиме. Он считал, что теория обработки информации должна базироваться не на последовательных моделях, а на асинхронных моделях.

Петри сформулировал три основных требований для моделирования асинхронных распределенных систем:

1. Модель должна отвечать законам физики. Нет смысла рассматривать модели, оперирующие глобальными состояниями. Дискретное событие в системе, как правило, не затрагивает все компоненты системы, а только некоторые из них. События не должны происходить в соответствии с идеализированной шкалой времени, а должны отвечать локальным причинно-следственным отношениям.
2. Элементарное дискретное действие должно подчинятся законам сохранения и быть обратимым. Т.е. переход из состояния S в состояние S’ предполагает, что в результате вычислений можно перейти обратно в S. Например, оператор x:=x+1 обратим, а оператор x=1 не обратим.
3. Техника моделирования должна быть удобной для пользователей вычислительных систем.

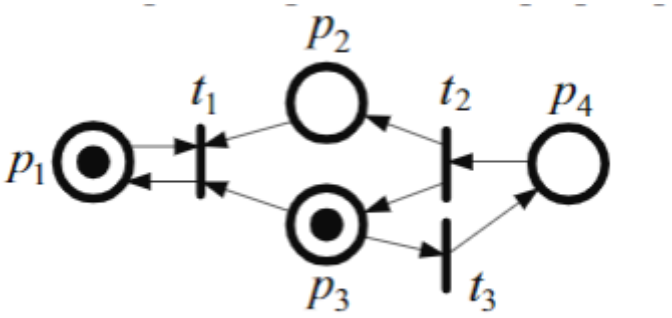
Сеть Петри – это двудольный ориентированный граф с двумя типами вершин (позиции и переходы)

При графической интерпретации сеть Петри представляет двудольный ориентированный граф, состоящий из вершин двух типов — *позиций* и *переходов,* соединенных ориентированными дугами, причем каждая дуга может связывать лишь разнотипные вершины (позицию с переходом или переход с позицией). Вершины-позиции обозначаются кружками, вершины-переходы — черточками. С содержательной точки зрения, переходы соответствуют событиям, присущим исследуемой системе, а позиции — условиям их возникновения. Направленные дуги могут соединять позиции только с переходами, а переходы

только с позициями. Каждая позиция может быть входной или (и) выходной для одного или нескольких переходов.

Таким образом, совокупность переходов, позиций и дуг позволяет описать причинно-следственные связи, присущие системе, но в статике.

Чтобы сеть Петри «ожила», вводят еще один вид объектов сети — так называемые *фишки,* или метки позиций. В общем случае позиция может содержать несколько фишек. Расположение фишек в позициях сети называется ***разметкой сети*** (пример перемещения фишек по сети приведен на рис.5.6).



Динамика в сети Петри описывается перемещением маркеров по сети в соответствии с соглашением о правиле срабатывания перехода, носящим локальный характер. Переход считается активным (событие может произойти), если в каждой его входной позиции есть хотя бы одна фишка. Через некоторое время (заранее не заданное) активный переход срабатывает. Срабатывание перехода считается неделимой операцией, в результате которой из каждой входной позиции изымается по одной фишке и добавляется по одной фишке в каждую выходную позицию.

Поведение сети характеризует диаграмма маркировок, которая представляет собой ориентированный граф, вершинами которого являются маркировки, а дуги помечены наименованиями переходов срабатывающих переходов.

В аналитической форме сеть Петри может быть Переход считается активным (событие может произойти), если в каждой его входной позиции есть хотя бы одна фишка. в представлена следующим образом:

$$P=(P,T,E,\mu^0),$$

где $\boldsymbol{P} = \{p_i\}$ — конечное непустое множество позиций;

$\boldsymbol{T} = \{t_i\}$ — конечное непустое множество переходов;

$\boldsymbol{E} = \boldsymbol{P}\times\boldsymbol{T} \vee \boldsymbol{T}\times\boldsymbol{P}$ – конечное множество дуг

$\boldsymbol{P}\times\boldsymbol{T}$— входная функция (прямая функция инцидентности), которая для каждого перехода задает множество его входных позиций;

$\boldsymbol{T}\times\boldsymbol{P} \to 0,1$ — выходная функция (обратная функция инцидентности), которая для каждого перехода задает множество его входных позиций;

μ^0—начальная маркировка, $\mu^0 : P \to N$, где $N = \{0,1,2, \ldots\}$

Множества входных и выходных позиций перехода $t_i \in T$ обозначается $I(t_j)$ и $O(t_j)$ соответственно; $I(p_i)$ и $O(p_i)$ – обозначения переходов, являющишся соответственно входами и выходами позиции p_i. Тогда сеть Петри можно задать в виде:

<P,T,I,O, $\boldsymbol{\mu^0}$ >

M — функция разметки сети, ***M : B*** -> 0, 1, 2,... — ставит каждой позиции сети в соответствие неотрицательное целое число.

С учетом введенных обозначений необходимое условие срабатывания перехода *tj* может быть записано следующим образом:

$$\forall p_i \in I(t_j) \{M(p_i) \geq 1\}$$

(*для всех входных позиций разметка должна быть >1*).

Срабатывание перехода d_j изменяет разметку сети *M(p)* на разметку *M'(p)* по следующему правилу:

$$M'(P)=M(P)-I(d_j)+O(d_j),$$

то есть переход d_j изымает по одной метке из каждой своей входной позиции и добавляет по одной метке в каждую из выходных позиций. Смену разметки обозначают так:

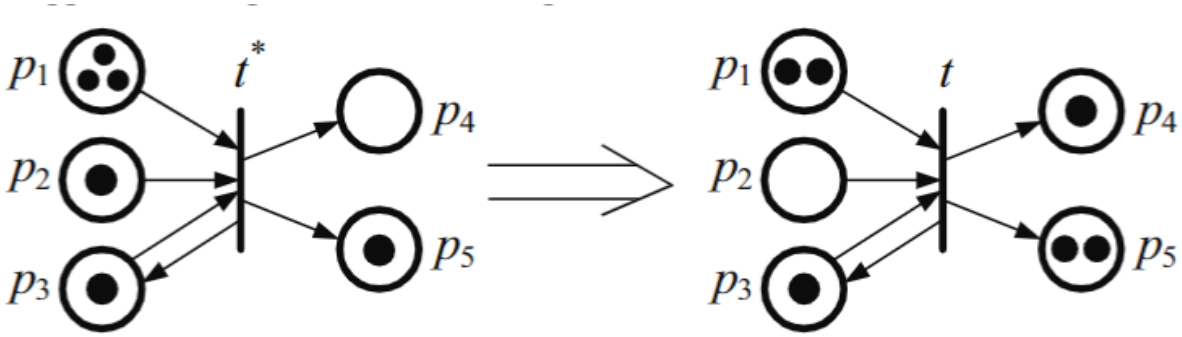
$$M_o \xrightarrow{t_j} M'$$

Срабатывание перехода считается неделимым актом, т.е. предполагается, что изъятие маркеров из входных позиций и помещение их в выходную позицию происходит мгновенно, однако по соглашению неделимость этой операции может быть нарушена.

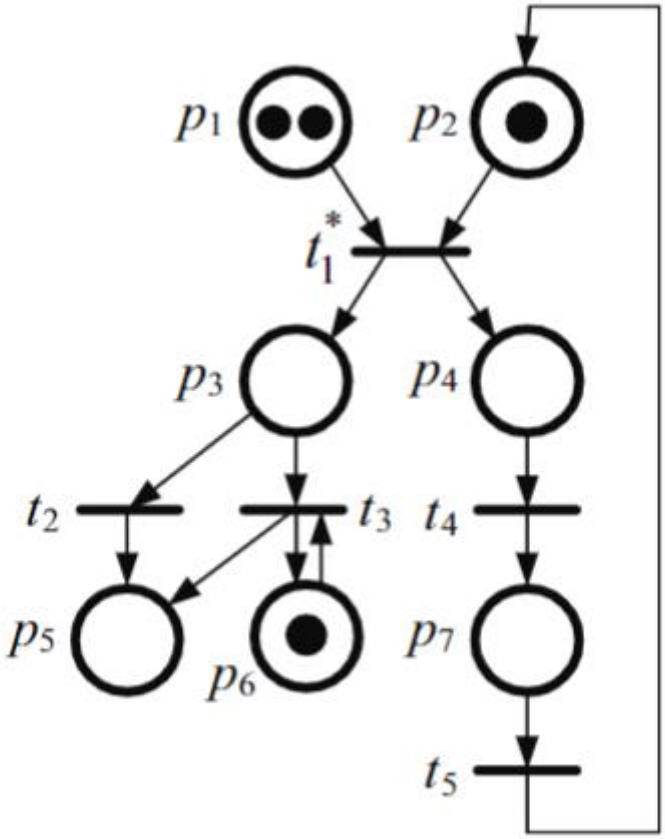
Входная и выходная функции сети Петри (***I*** и ***O***) позволяют описать любую сеть с помощью двух матриц размера *m x n* (матриц входных и выходных позиций), имеющих следующую структуру:

	t_1	t_2	...	t_j	...	t_n
p_1	0	1	...	0	...	0
p_2	1	1	...	0	...	1
...	...	...	...	...	...	...
p_j	0	1	...	0	...	1
...	...	...	...	...	...	...
p_m	1	0	...	1	...	0

Соглашение о правиле срабатывания переходов отражает концепцию причинно-следственной связи: при наличии необходимых условий событие может наступить. После наступления события возникают новые условия, которые повлекут следующие события. Первоначальные условия могут либо исчезнуть, либо сохраняться. Результат срабатывания переходов:



Рассмотрим пример. Сеть Петри



Для этой сети начальная маркировка имеет вид

M^0=(2100010).

Активен переход t_1. После срабатывания перехода сеть будет иметь маркировку M^1=(1011010).

В этой маркировке активны переход t_4 и переходы t_3 и t_2. Причем, из переходов t_3 и t_2 может сработать только один из них, т.к. срабатывание одного блокирует второй. В любом случае срабатывание этих переходов дает маркировку

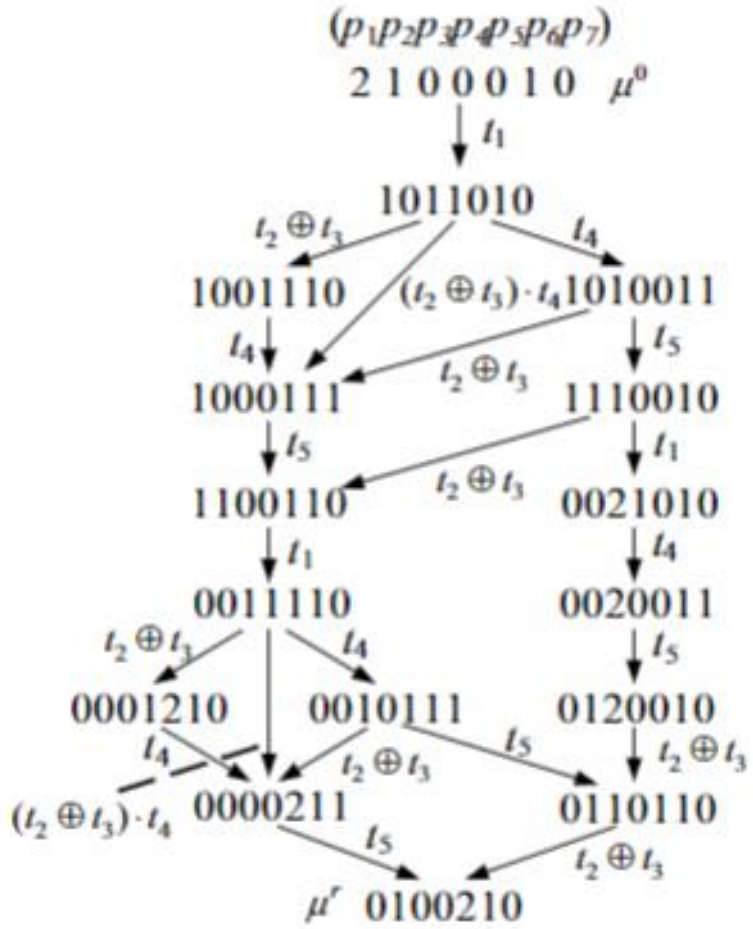
M^2=(1001110).

Срабатывание перехода t_4 дает маркировку M^3=(1010011), а одновременное срабатывание – M^4=(1000111).

Некоторые исследователи считают, что одновременное срабатывание переходов маловероятным.

Диаграмма маркировок.

Диаграмма маркировок представляет собой ориентированный граф, вершинами которого являются маркировки из множества M достижимых маркировок, а дуги направлены из маркировки μ^{α} в маркировку μ^{β}, если существует μ^{α}, $\mu^{\beta} \in M$ и существует непосредственный переход от μ^{α} к μ^{β} ($\mu^{\alpha} \rightarrow \mu^{\beta}$).



На рисунке приведена диаграмма маркировок сети. Обозначение

$$(t_2 \oplus t_3) \cdot t_4$$

Означает, что одновременно сработает одна из пар: (t2, t4) или (t3, t4).

Из диаграммы следует, что из заданной начальной маркировки достижима только одна тупиковая маркировка, из которой процесс не может выйти. Т.е. процесс однозначно завершен.

По диаграмме можно определить последовательность срабатывания переходов, т.е. последовательность событий (траекторий процесса). Из диаграммы видно, что процесс не является детерминированным, т.к. возможны различные пути достижения тупиковой маркировки.

Возможна случаи, когда из начальной маркировки можно получить не одну, а несколько результирующих маркировок. Асинхронность процесса отражается в модели отсутствием указаний на время срабатывания переходов, что гарантирует недетерминизм модели.

Отношения между событиями

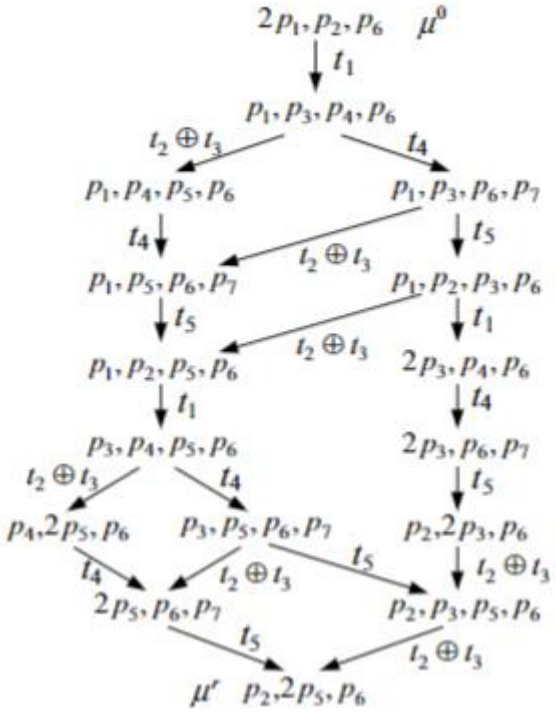
Содержательный смысл перехода в сети Петри – это некоторое событие. Говоря о срабатывании какой-то пары переходов можно говорить об *отношении* этих событий.

1. Переход ti следует за переходом tj (ti→tj) отражает последовательность соответствующих событий, их причинно-следственную связь;
2. Может сработать переход ti или tj, но не оба вместе (исключающее или) – такое отношение отражает свободный выбор, или недетерминизм;
3. Могут сработать как переход ti, так и переход tj в любой последовательности или оба вместе (ti ‖ tj) – такое выражение означает непоследовательность, одновременность или параллелизм.

Представление маркировок сетей Петри не является единственным. Маркировки могут быть представлены в виде списка позиций вершин, число маркеров в которых больше нуля. При таком представлении дуги вида

$$(t_2 \oplus t_3) \cdot t_4$$

Являются избыточными. Пример диаграммы маркировок во вершинам приведен на рисунке:



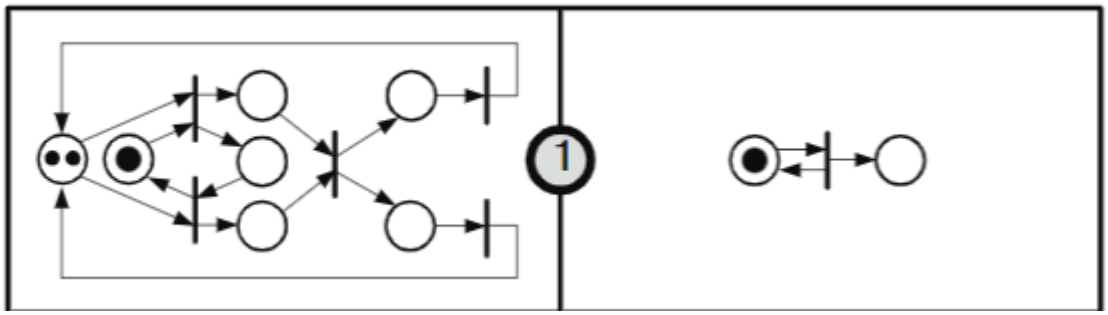
Классификация сетей Петри

Классификация основывается на качественных и количественных ограничениях, накладываемые на допустимые конфигурации (статические ограничения) и типы маркировок (динамические ограничения)

Классификация по динамическим ограничениям

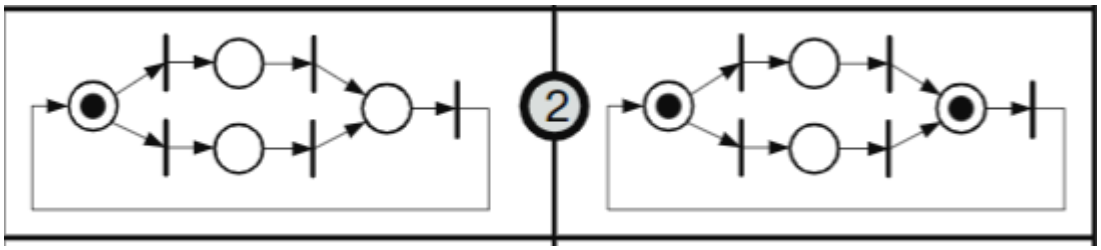
1. Сеть Петри называется *k-ограниченной*, если на множестве ее достижимых состояний не найдется ни одной позиции piЄP, для которой $\mu(p_i) \geq (k+1)$

Да нет



2. Сеть Петри называется *безопасной,* если она 1-ограничена.

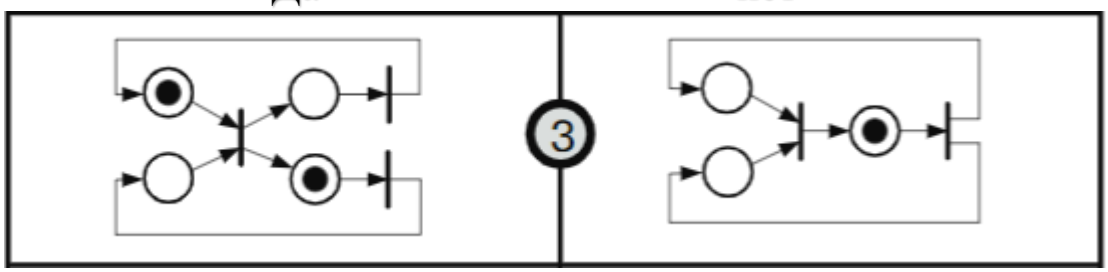
Да нет



3. Сеть Петри называется *ограниченной,* если найдется k, для которого она k-ограничена.
4. Сеть Петри называется *1-консервативной*, если в процессе функционирования общее число маркеров остается постоянным, т.е. для любого момента времени выполняется условие:

$$\sum_{p_i \in I} \mu(p_i) = \sum_{p_j \in O} \mu(p_j)$$

Да нет



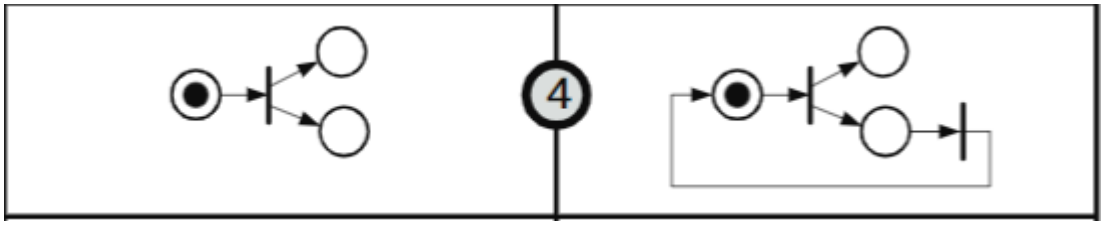
5. Сеть Петри называется *консервативной,* если существует положительная целочисленная функция f:P→N такая, что для любого перехода имеет место

$$\sum_{p_i \in I} f(p_i) = \sum_{p_j \in O} f(p_j),$$

или, другими словами

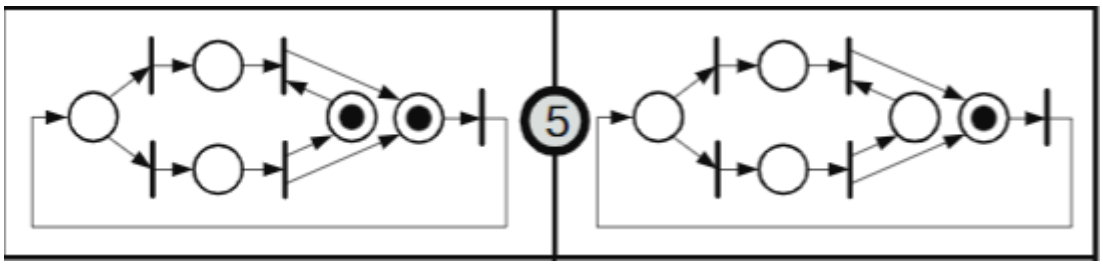
$$\sum_{p_i \in I} a_i \mu(p_i) = \sum_{p_j \in O} a_j \mu(p_j).$$

Да нет



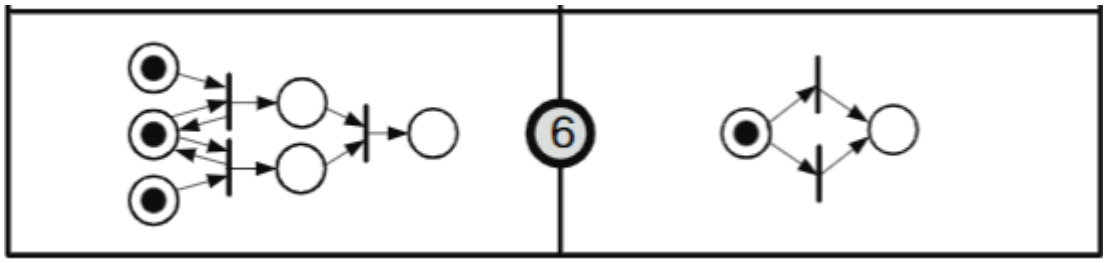
6. Сеть Петри называется *живой (активной*), если каждый переход является потенциально срабатывающим при любой маркировке .

Да нет



7. Сеть Петри называется *устойчивой,* если для всех пар ti, tj (i≠j) и любой допустимой маркировке, при которой ti, tj возбуждены, срабатывание одного из них не может снять возбуждения другого.

Да нет



Практический смысл определений заключается в следующем.

Свойство ограниченности отражает конечность информации, хранимой в памяти сети.

Свойство безопасности соответствует нахождению прямого соответствия между Сетью Петри и некоторой логической схемой, число состояний которой равно 2^n, где n- число позиций сети Петри.

Консервативность показывает ограниченность ресурсов в моделируемом объекте.

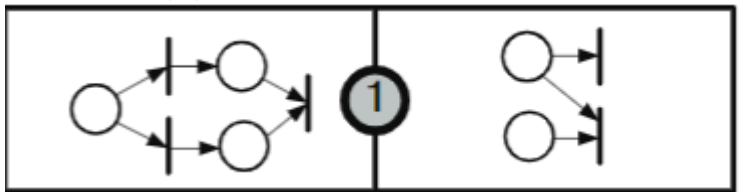
Свойство живости сети используют для выявления неактивных элементов моделируемой системы. Для того, чтобы все элементы моделируемой системы были активны, надо, чтобы соответствующая сеть Петри была живой.

Свойство устойчивости может использоваться для выявления поведенческих конфликтов.

Классификация по статическим ограничениям

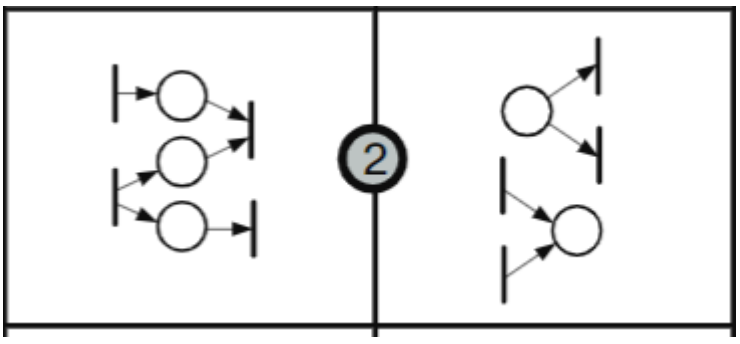
1. Сеть Петри называется *сетью свободного выбора*, если для любых $t_i \in T$ и $p_i \in I(t_i)$ позиция p_i является либо единственной входной позицией перехода t_i, либо этот переход имеет единственную входную позицию. (если два перехода имеют общую входную позицию, то эта позиция единственна для каждого перехода).

Да нет



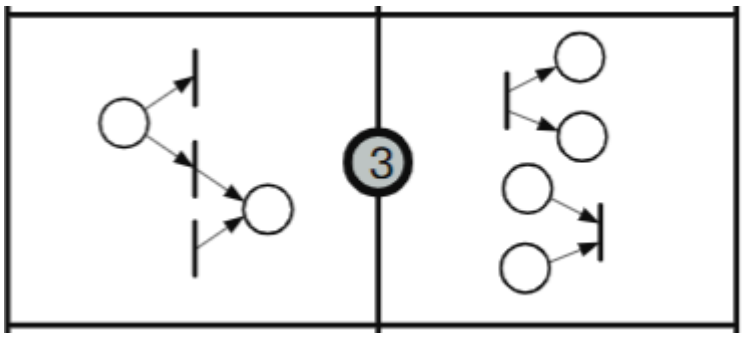
2. Сеть Петри называется *маркированным графом*, если каждая позиция имеет в точности по одному входному и одному выходному переходу.

Да нет



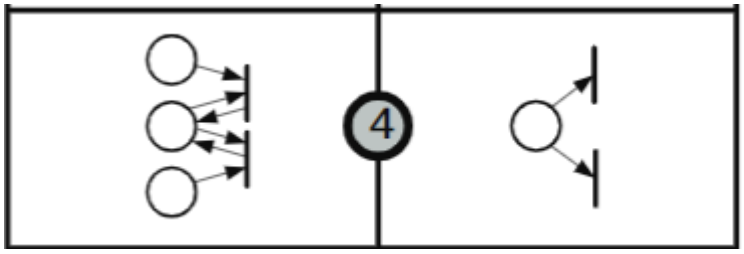
3. Сеть Петри называется *автоматной,*, если каждый переход t_i, имеет не более одного входа и не более одного выхода.

Да нет



4. Сеть Петри называется *бесконфликтной*, если либо для каждой ее позиции $p_i \in P$ существует не более одной исходящей дуги, либо любая позиция, являющейся входной для более чем одного перехода, является входной каждого такого перехода. Бесконфликтные сети устойчивы, обратное не всегда верно.

Да нет



Задачи и алгоритмы анализа сетей Петри

Классическими задачами анализа сетей Петри являются:

- *Достижимость* – возможно ли достижение некоторой исходно заданной маркировки из начальной, т.е. содержится ли заданная маркировка в диаграмме маркировок.
- *Живость* – возможно ли в принципе срабатывание любого перехода данной сети
- *Одновременность* – возможно ли параллельное срабатывание нескольких переходов
- *Ограниченность* – может ли в процессе функционирования сети в какой-либо ее позиции наблюдаться число маркеров, не превышающее заданное; в неограниченной сети может накапливаться бесконечное число маркеров.
- *Безопасность* – частный случай ограниченности, когда элементы векторов достижимых маркировок, соответствующие позициям, содержат не более одного маркера (булевы векторы)
- *Устойчивость* – срабатывание одного перехода не может вызвать снятие возбуждения другого.

Методы анализа сетей Петри можно разбить на три группы.

Первая группа: *к этой группе относятся задачи, сводимые к задаче достижимости или живости*

Проблема достижимости: в сети Петри с начальной разметкой M_0 требуется определить, достижима ли принципиально некоторая разметка μ^* из μ^0.
С точки зрения исследования моделируемой системы, эта проблема интерпретируется как проблема достижимости (реализуемости) некоторого состояния системы.

Постановка задачи: заданы сеть Петри <P,T,E, μ^0> и некоторая целевая маркировка μ* (целочисленный вектор той же длины, что и начальная маркировка). Требуется определить, принадлежит ли требуемая маркировка множеству M достижимых из начальной.

Эта задача имеет модификации:

- Требуемая маркировка является пустой;
- Построение графа маркировок;
- Определение множества достижимых маркировок;
- Поиск траектории, приводящий к требуемой маркировке;
- Определение всех траекторий, приводящей к достижимой и т.д.

Очевидно, что диаграмма маркировок с дугами, нагруженными переходами, содержит всю информацию для решения задач достижимости. Однако, возможны случаи, когда такую диаграмму построить невозможно, например потому, что число достижимых маркировок может быть неограниченным.

Например, сеть вида

Порождает бесконечную диаграмму маркировок $10 \rightarrow 11 \rightarrow 12 \rightarrow 13 \rightarrow \ldots \rightarrow 1\infty$

Алгоритм построения диаграммы маркировок

Множество переходов, активных при текущей маркировке μ^i называется селектором и обозначается $S(\mu^i)$

Маркировка μ^d, при которой нет активных переходов $S(\mu^d) = \emptyset$, называется тупиковой.

Маркировка μ^i покрывает маркировку μ^j (или $\mu^i \geq \mu^j$), если для каждого p_k выполняется условие: $\mu^i(p_k) \geq \mu^j(p_k)$ (или $S(\mu^i) \geq S(\mu^j)$).
Если в процессе функционирования сети Петри достигнута строго покрывающая ее маркировка и при этом переход от μ^i к μ^j связан с добавлением некоторого числа

маркеров в определенные позиции (т.е. наблюдается накопление маркеров в позициях при циклическом функционировании сети), то это накопление может быть задано символом ω или ∞.

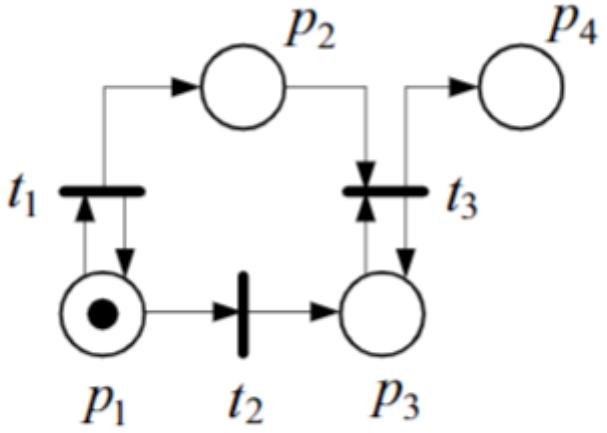
В процессе построения диаграммы каждая вершина может быть отнесена к одному из следующих видов:

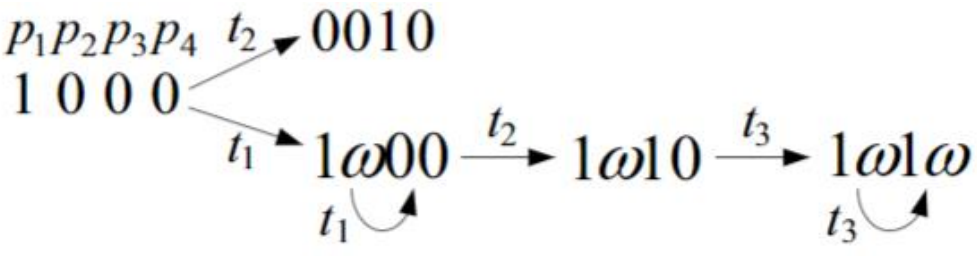
- Тупиковая -- $S(\mu^d) = \emptyset$
- Внутренняя – включенная в диаграмму
- Граничная – еще не обработанная.

Алгоритм.

0. Начальная маркировка объявляется граничной вершиной X и обрабатывается следующим образом.
1. Если в диаграмме уже имеется некоторая обработанная вершина Y и с ней связана маркировка $\mu^y = \mu^x$, то вершина X исключается из рассмотрения, а все дуги, ведущие в X заводятся в Y.
2. Если $S(\mu^x) = \emptyset$, то вершина x -- тупиковая.
3. Для каждого перехода $t_i \in S(\mu^x)$ создается новая вершина Z с маркировкой μ^z, которая определяется по следующим правилам:
   - Если $\mu^x(p_k) = \infty$, то $\mu^z(p_k) = \infty$;
   - Если на пути от вершины, соответствующей начальной маркировке, к X найдется такая вершина Y такая, что $\mu^y < \mu^r$, где μ^r удовлетворяет $\mu^x \rightarrow \mu^r$, и $\mu^y(p_i) < \mu^r(p_i)$, то $\mu^z(p_i) = \infty$;
   - В противном случае $\mu^y(p_i) = \mu^r(p_i)$.
4. Если множество граничных вершин не пусто, то выбрать любую граничную вершину и перейти к п.1. Иначе завершить процедуру.

Пример:





Свернутые диаграммы маркировок, содержащие символ ω теряют часть информации о поведении сети. Например, не ясно, как накапливаются маркеры, есть ли уменьшение числа маркеров с последующим возрастанием и т.п.

Несмотря на указанный недостаток диаграмма маркировок позволяет установить свойства сети.

- Сеть Петри принадлежит *к классу ограниченных* тогда и только тогда, когда символ ω отсутствует в диаграмме маркировок.
- Верхняя граница $\mu(p_i)$ указывает, для какого k сеть *является k-ограниченной*.
- *Сеть является безопасной*, если в любой маркировке μ диаграмм $\mu(p_i) \leq 1$ для всех μ.
- *Свойство 1-консервативности* проверяется путем суммирования компонент каждой маркировки μ по числу позиций $|\mu| = \sum_{i=1}^{n} \mu(pi)$. Если $|\mu| = const$ для всех маркировок, то сеть Петри обладает этим свойством (число маркеров в сети постоянно).

Задача живости. Под живостью перехода понимают возможность срабатывания любого перехода в данной сети.

Задача живости может быть поставлена в двух модификациях.

1. Для данной сети $<P,T,E, \mu^0>$ определить, все ли ее переходы $t \epsilon T$ могут сработать (живы) – *общая задача живости.*
2. Для данной сети определить, является ли живым некоторый ее переход $t_i \epsilon T$ – задача живости одного перехода.

Задача живости может быть сведена к задаче достижимости.

Анализ модели на свойство живости позволяет выявить невозможные состояния в моделируемой системе (например, неисполняемые ветви в программе).

Вторая группа методов анализа

Вторая группа методов анализа касается задач установления принадлежности сети Петри классам сети свободного выбора, бесконфликтным сетям, маркированным графам и автономным сетям. Эти задачи решаются методами теории графов и их рассматривать не будем.

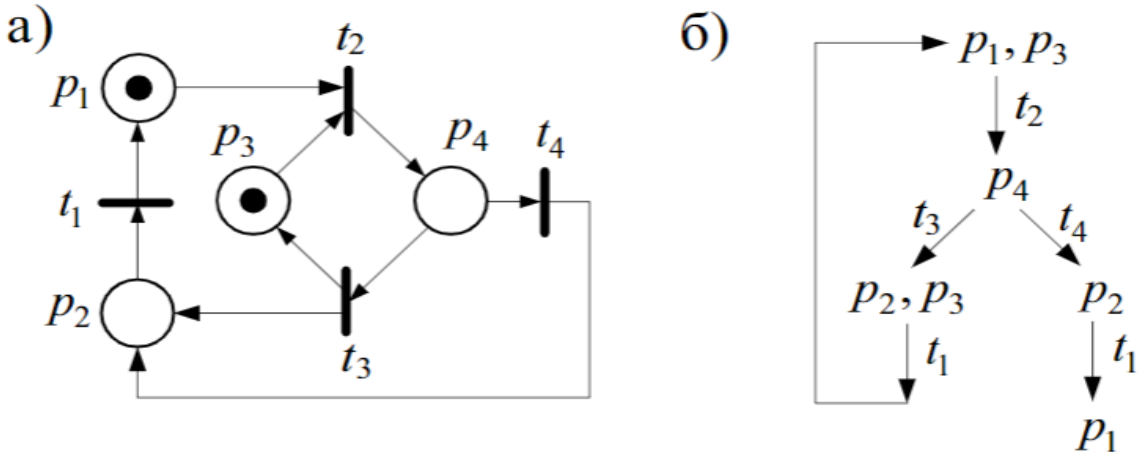
Третья группа методов анализа

Третья группа методов анализа ориентирована на выявление специфических конфигураций сети, в которых при функционировании возможен дефицит маркеров или их накопление. Методы анализа этой группы основаны на использовании структурного подхода и не связаны с построением диаграмм маркировок.

Дедлоком в сети Петри называют подмножество позиций сети, обуславливающих попадание сети в тупиковое состояние (маркировку) в которой не возбуждается ни один переход.

Формальное определение: *дедлоком* в сети Петри <P,T,E, μ^0> называют подмножество ее позиций D⊂P такое, что I(D)⊂O(P).

Это означает, что каждый переход, имеющий принадлежащую дедлоку выходную позицию, должен иметь своими входными позициями только те из них, которые принадлежат дедлоку. Дедлок не обязательно тупиковая маркировка, но она может стать такой из-за дефицита маркеров.



Дедлок образуют позиции p_3 и p_4. Из диаграммы маркировки видно, что из начальной маркировки возможны две траектории. Одна возвращает сеть в исходное состояние, а другая приводит в тупиковое состояние.

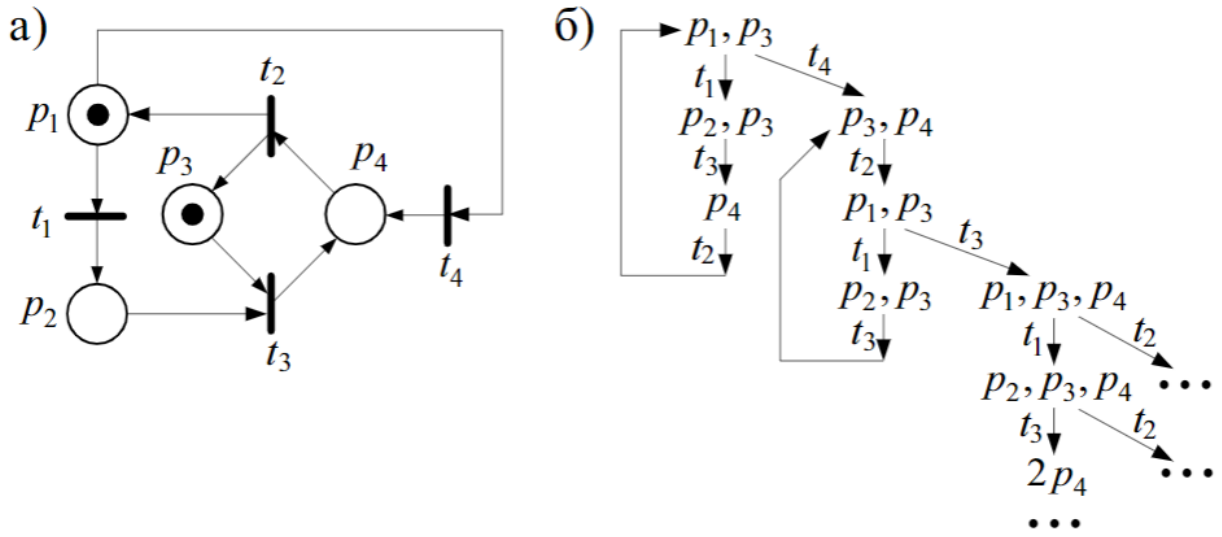
Ловушкой в сети Петри называется такое множество позиций, что каждый переход, входом которого является одна из позиций множества, имеет выходом другую позицию того же множества. Другими словами, ловушка представляет собой цикл в сети Петри, из которого нет выхода.

Формальное определение: ловушкой в сети Петри называется подмножество L⊂P, такое, что O(L)⊂I(L).

Если из ловушки забирается маркер, то сеть обязательно забрасывает в нее по крайней мере один маркер. Если в ловушку попал хотя бы один маркер, то никогда все позиции ловушки не окажутся пустыми.

Из приведенного примера видно, что при функционировании сети Петри число маркеров в ловушке может расти. Дедлоки и ловушки не исключают друг друга.

Сеть свободного выбора *жива* тогда и только тогда, когда каждый ее дедлок содержит непустую (маркированную) ловушку – Теорема Коммонера.



Итак, достоинства сетей Петри заключаются в том, что они:

1) позволяют моделировать ПП всех возможных типов с учетом вероятных конфликтов между ними;

2) обладают наглядностью и обеспечивают возможность автоматизированного анализа;

3) позволяют переходить от одного уровня детализации описания системы к другому (за счет раскрытия/закрытия переходов).

Вместе с тем, сети Петри имеют ряд недостатков, ограничивающих их возможности. Основной из них — время срабатывания перехода считается равным 0, что не позволяет исследовать с помощью сетей Петри временные характеристики моделируемых систем.

В результате развития аппарата сетей Петри был разработан ряд расширений. Одно из наиболее мощных — так называемые .Е-сети (evaluation — «вычисления», «оценка») — «оценочные сети».

В отличие от сетей Петри, в Е-сетях:

1) имеются несколько типов вершин-позиций: простые позиции, позиции-очереди, разрешающие позиции;

2) фишки (метки) могут снабжаться набором признаков (атрибутов);

3) с каждым переходом может быть связана ненулевая задержка и функция преобразования атрибутов фишек;

4) введены дополнительные виды вершин-переходов.

5) в любую позицию может входить не более одной дуги и выходить также не более одной.

В связи с этим любой переход может быть описан тройкой параметров:

$$d_j=(S,t(d_j),\rho(dj)),$$

где S — тип перехода,

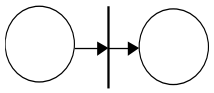
$t(d_j)$, — функция задержки,

$\rho(d_j)$ — функция преобразования атрибутов.

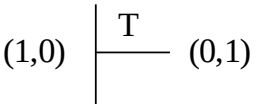
Базовые переходы Е-сети описаны ниже.

Т-переход («исполнение», «простой переход»).

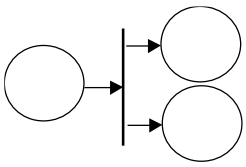
Его графическое представление аналогично представлению вершины-перехода сети Петри:



Переход срабатывает при наличии метки во входной позиции и отсутствии ее в выходной позиции. Формально это можно записать так:

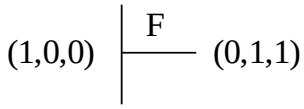


Т-переход позволяет отразить в модели занятость некоторого устройства (подсистемы) в течение некоторого времени, определяемого параметром *t(d)*



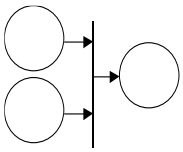
F-переход («разветвление») имеет графическое представление:

и срабатывает при тех же условиях, что и Т-переход:

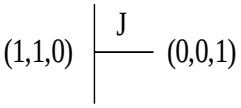


С содержательной точки зрения F-переход отображает разветвление потока информации (транзактов) в системе.

J-переход («объединение»). Графическое представление:

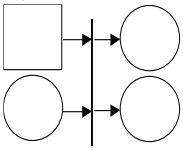


Переход срабатывает при наличии меток в обеих входных позициях:

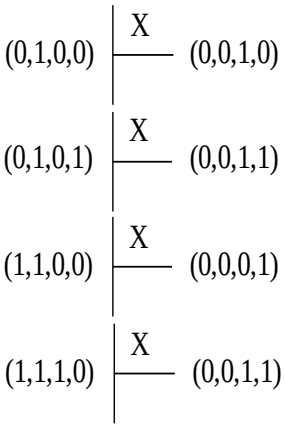


Он моделирует объединение потоков или наличие нескольких условий, определяющих некоторое событие.

Х-переход («переключатель»). По сравнению с тремя предыдущими типами переходов он содержит дополнительную управляющую («разрешающую») позицию, которая графически обозначается обычно либо квадратиком, либо шестиугольником:

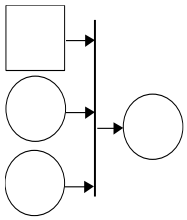


Логика его срабатывания задается следующими соотношениями:



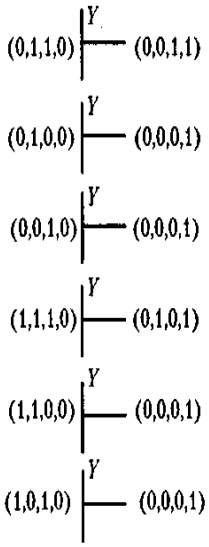
X-переход изменяет направление потока информации (транзактов). В общем случае разрешающая процедура может быть сколь угодно сложной, но сущность ее работы заключается в проверке выполнения условий разветвления потока (с точки зрения программиста разрешающая позиция аналогична условному оператору типа *if).*

Y-переход («выбор», «приоритетный выбор»). Этот переход также содержит разрашающую позицию:



Логика срабатывания Y-перехода:

Y-переход отражает приоритетность одних потоков информации (транзактов) по сравнению с другими. При этом разрешающая процедура может быть определена различным образом: как операция сравнения фиксированных приоритетов меток; как функция от атрибутов меток (например, чем меньше время обслуживания, тем выше приоритет).

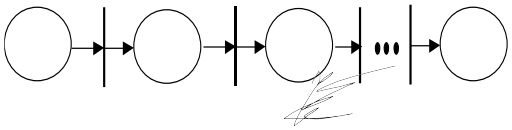


В некотором смысле он работает аналогично оператору выбора типа *case.*

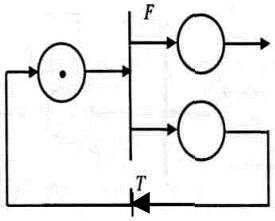
В Е-сети все переходы обладают свойством безопасности. Это означает, что в выходных позициях (которые в свою очередь могут быть входными для следующего перехода) никогда не может быть более одной метки.

Вместе с тем, в Е-сетях существуют понятия *макроперехода* и *макропозиции,* которые позволяют отображать в модели процессы накопления обслуживаемых транзактов в тех или иных узлах системы.

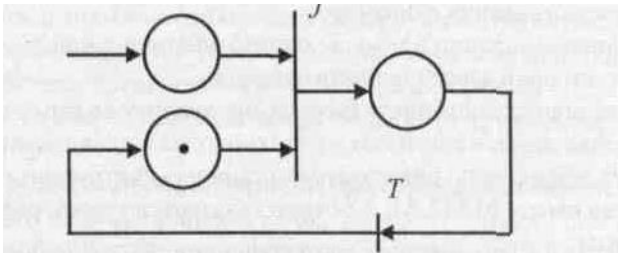
Макропозиция *очередь* представляет собой линейную композицию вершин-позиций и Т-переходов; количество вершин-позиций определяет «емкость» очереди:



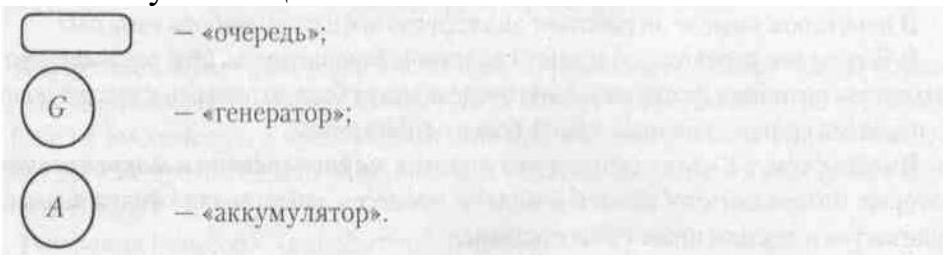
Макропозиция *генератор* позволяет представлять в сети источник меток (тразактов):



Если необходимо задать закон формирования меток, то «генератор» может быть дополнен разрешающей позицией. Поскольку в Е-сети нельзя «накапливать» метки, то вводится **макропозиция *поглощения***

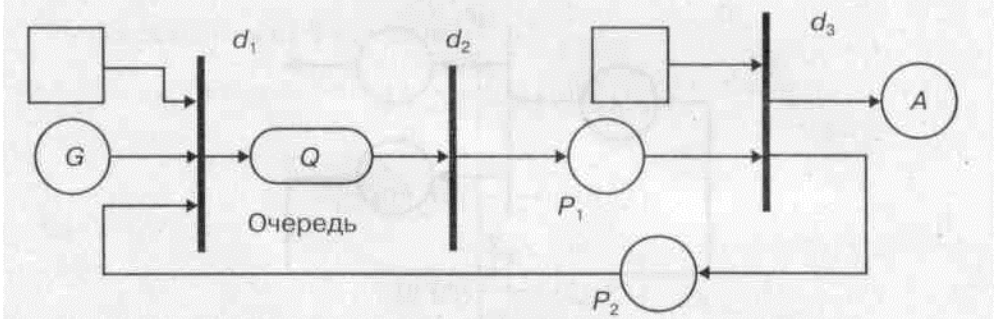


В целях повышения компактности и наглядности Е-сети для обозначения макропозиций используют специальные символы:



Аналогичным образом путем композиции N однотипных переходов могут быть получены макропереходы всех типов: X_N, Y_N, J_N.

Рассмотренные особенности Е-сетей существенно расширяют их возможности для моделирования дискретных систем вообще и параллельных процессов в частности. Ниже приведен пример описания в виде Е-сети мультипрограммной вычислительной системы. Обработка поступающих заданий организована в ней по принципу квантования времени: каждому заданию выделяется равный отрезок (квант) процессорного времени; если задание выполнено, то оно покидает систему, если же времени оказалось недостаточно, то задание встает в очередь и ждет повторного



На рисунке переходы, соответствующие определенным событиям в системе (d_i), имеют следующие обозначения:

d_1 — постановка задания в очередь;

d_2 — выполнение задания в течение одного кванта времени;

d_3 — анализ степени завершенности задания.

Помимо очевидных достоинств Е-сетей, проявленное к ним в первой части книги внимание объясняется еще и тем, что технология моделирования систем в виде Е-сетей может быть реализован с помощью инструмента SIMULINK, входящего в состав пакета M ATLAB.

Особенности Е-сетей существенно расширяют их возможности для моделирования дискретных систем вообще и параллельных процессов в частности. Технология моделирования систем в виде Е-сетей может быть реализована с помощью инструмента SIMULINK, входящего в состав пакета M ATLAB.