Теория вычислительных процессов

Process calculus

Сети Петри

ИВТ и ПМ
ЗабГУ

2021

Outline

Дискретные системы

- *Дискретная динамическая система* – одно из наиболее общих понятий в теоретическом программировании.
- Примеры: компьютеры, их элементы и устройства, компьютерные сети;
  компьютерные программы и операционные системы;
  системы сбора и автоматической обработки цифровой информации;
  системы автоматического управления объектами и процессами;
  производственные системы дискретного характера (сборочные линии);

  социально-экономические и другие.

- Приведенные примеры — сложные системы, имеющие сложную внутреннюю структуру.
- Эти системы дискретные:
  - в их внутренней структуре можно выделить счетное число состояний (в которых они могут пребывать в некоторые моменты времени)
  - а также переходить из состояния в состояние в некоторые моменты времени.
- множество может быть очень и очень большим, но чаще всего, оно конечно и счетно.

- Приведенные примеры — сложные системы, имеющие сложную внутреннюю структуру.
- Эти системы дискретные:
  - в их внутренней структуре можно выделить счетное число состояний (в которых они могут пребывать в некоторые моменты времени)
  - а также переходить из состояния в состояние в некоторые моменты времени.
- множество может быть очень и очень большим, но чаще всего, оно конечно и счетно.
- Многие непрерывные (или аналоговые) системы можно представить дискретными, вводя некоторые границы дискретизации. Например: звук.

- Автомат – математическая абстракция последовательной дискретной системы[1].
- Постоянно появляется необходимость моделировать новые динамические дискретные системы.
- Например параллельные системы с недетерминированным поведением, в которых отдельные компоненты функционируют, в основном, независимо, взаимодействуя друг с другом время от времени.
  Примеры: многопроцессорные вычислительные системы;
  параллельные программы;
  многозадачные операционные системы;
  асинхронные электронные схемы и т.д.

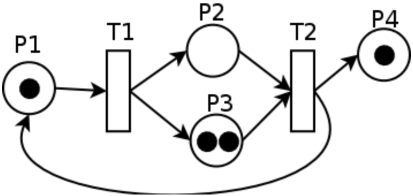
[1] см. также автоматное программирование

- Системы с параллельно функционирующими и асинхронно (т.е. в произвольные моменты времени) взаимодействующими компонентами не описываются адекватно в терминах классической теории автоматов.
- Среди многих существующих методов описания и анализа дискретных параллельных систем выделился подход, который основан на сетевых моделях специального вида – сети Петри.

Outline

Сети Петри

Сети Петри – это инструмент для математического моделирования и исследования сложных систем.



Сети Петри

- Сеть Петри описывает *структуру* и *поведение* динамической системы.
- Идея Сети Петри впервые использована немецким математиком и информатиком Карлом Адамом Петри для описания химических процессов в первой половине XX века
- Формальное описание сети опубликовано в 1962 году
- Сети Петри позволили в том числе развить идеи из области параллельных и распределённых вычислений
- Business Process Model and Notation, диаграмма деятельности, событийная цепочка процессов – графический аналоги сети Петри

Сеть Петри

Определим сеть Петри как четвёрку:

N = (P, T, I, O)

- P = {p1, p2, ..., pn } – конечное множество **позиций**;
- T = {t1, t2, ..., tm } – конечное множество **переходов**;
- $I : T \rightarrow P$ – входная функция, сопоставляющая переходу T мультимножество[2] его входных позиций P;
- $O : T \rightarrow P$ – выходная функция, сопоставляющая переходу мультимножество его выходных позиций P.

[2] множество, которое может содержать несколько экземпляров одного и того же объекта

Сеть Петри

Пример

$P = \{p_1, p_2, p_3\}$
$T = \{t_1, t_2\}$
$I(t_1) = \{p_1, p_1, p_2\}; O(t_1) = \{p_3\}$
$I(t_2) = \{p_1, p_2, p_2\}; O(t_2) = \{p3\}$

Сеть Петри

Наглядное представление сети Петри – двудольный, ориентированный мультиграф.

Сеть Петри

Наглядное представление сети Петри – двудольный, ориентированный мультиграф.

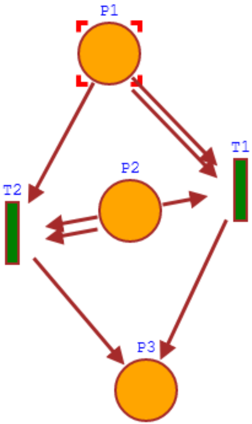
Двудольный граф (биграф) – граф, множество вершин которого можно разбить на две части таким образом, что каждое ребро графа соединяет какую-то вершину из одной части с какой-то вершиной другой части.
Т.е. то есть не существует рёбер между вершинами одной и той же части.

Мультиграф – граф, в котором разрешается присутствие кратных рёбер.

Сеть Петри

Пример

$P = \{p_1, p_2, p_3\}$
$T = \{t_1, t_2\}$
$I(t_1) = \{p_1, p_1, p_2\}; O(t_1) = \{p_3\}$
$I(t_2) = \{p_1, p_2, p_2\}; O(t_2) = \{p3\}$



Симуляция и задание сетей Петри: petri.hp102.ru/pnet.html

Сеть Петри
Пример

Задание в виде продукционных правил:
$t_1 : \{p_3, p_1\} \longrightarrow \{p_1, p_2, p_3\}$
$t_2 : \{p_1\} \longrightarrow \{p_1, p_2\}$

аналогично
$P = \{p_1, p_2, p_3\}$
$T = \{t_1, t_2\}$
$I(t_1) = \{p_3, p_1\}; \quad O(t_1) = \{p_1, p_2, p_3\}$
$I(t_2) = \{p_1\}; \quad O(t_2) = \{p_1, p_2\}$

Маркировка

- Маркировка – это размещение по позициям сети Петри фишек, изображаемых на графе сети Петри точками.
- Фишки используются для определения выполнения сети Петри. Количество фишек в позиции при выполнении сети Петри может изменяться от 0 до бесконечности.

Маркировка сетей Петри

Выполнение сети Петри. Пример.
http://petri.hp102.ru/pnet.html

Для выполнения сети Петри на бумаге можно использовать монеты

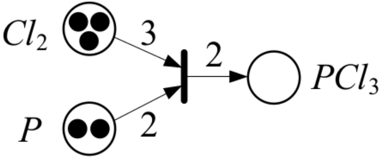
Правила выполнения Сети Петри

- Сеть Петри выполняется посредством *запусков переходов*.
- **Запуск перехода** управляется фишками в его входных позициях и сопровождается удалением фишек из этих позиций и добавлением новых фишек в его выходные позиции.
- Переход может запускаться только в том случае, когда он *разрешен*.
- Переход называется **разрешенным**, если каждая из его входных позиций содержит число фишек, не меньшее, чем число дуг, ведущих из этой позиции в переход (или кратности входной дуги).
- Разрешённые переходы можно запускать в произвольном порядке

Outline

Химическая реакция

$2P + 3Cl_2 = 2PCl_3$



Примеры моделирования сетями Петри

- Простой процесс с двумя состояниями
- Запуск параллельных процессов
- Запуск одного из параллельных процессов
- || процесс с общим ресурсом
- deadlock

Пример

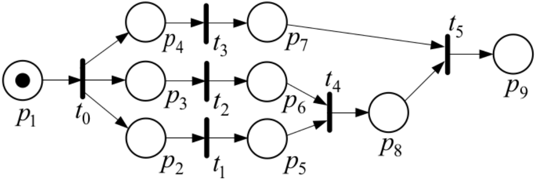
Как можно распараллелить вычисление арифметического выражения?

$$z = (a + b) \cdot (c - d) - \frac{e}{f}$$

Пример

Как можно распараллелить вычисление арифметического выражения?

$$z = (a + b) \cdot (c - d) - \frac{e}{f}$$



Вычисление арифметического выражения

Примеры моделирования сетями Петри

Последовательная обработка запросов сервером
Опишем обработку как набор состояний. Сами состояния будем обозначать позициями

- s1 – сервер ждёт
- s2 – запрос поступил и ждёт
- s3 – запрос обрабатывается
- s4 – запрос обработан

Смену состояний будем называть событиями. События обозначим переходами

- t1 – поступил запрос
- t2 – сервер начал обработку
- t3 – сервер закончил обработку
- t3 – результат отправлен клиенту

Outline

Переходы

- $\#\hat{}: T \times P \to \mathbb{N}_0$
- $\#\hat{}(t, p)$ – кратность дуги из t в p
- $\hat{}\# : P \times T \to \mathbb{N}_0$
- $\hat{}\#\ (p, t)$ – кратность дуги из p в t
- $\mathbb{N}_0$ – множество натуральных чисел и 0
- $\mu(p)$ – число фишек в позиции p

Расширенные входные и выходные функции

- Расширенная входная функция: $I : P \longrightarrow T^*$
- Расширенная входная функция: $O : P \longrightarrow T^*$
- T^* – мультимножество

Переходы

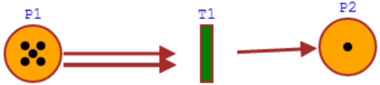
- Переход t **разрешён** если $\forall p \in I(t)$ справедливо $\mu(p) \geq {}^{\wedge}\#\,(p,t)$
- **Запуск** перехода

$$\mu'(p) = \mu(p) - {}^{\wedge}\#\,(p,t) + \#^{\wedge}(t,p)$$

- Сеть можно запускать до тех пор, пока в ней есть разрешённые переходы
- Порядок запуска переходов не определён

Переходы

- $\mu \quad = < 5, 1 >$
- $\mu' \quad = < 3, 2 >$
- Запуск $\mu \to \mu'$

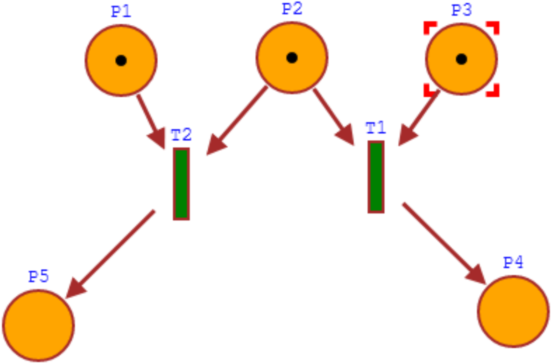


  - Фишки обычно означает: в буфере есть запись, переменная доступна
  - место – состояние, переменная или буфер
  - переход – функция или устройство

Одновременность и конфликт

- Сети Петри – асинхронны
- Не измеряем время
- Переход – примитивное событие, не занимающее времени
- Но различаем порядок событий
- В один и тот же момент времени может быть запущен только один переход

Конфликт



При данной маркировке разрешены оба перехода, но запуск любого из них деактивует другой переход

Outline

Диаграммы маркировок

Диаграмма маркировок – орграф, вершинами которого являются маркировки из множества M достижимых маркировок, а дуги направлены из маркировки μ^a в маркировку μ^b, если $\mu^a, \mu^b \in M$ и существует непосредственный переход $\mu^a \to \mu^b$

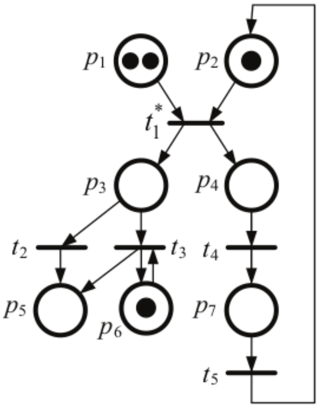
Каждая дуга – запуск перехода.

$\mu^a \to \mu^a$ – петля в вершине диаграммы маркировок

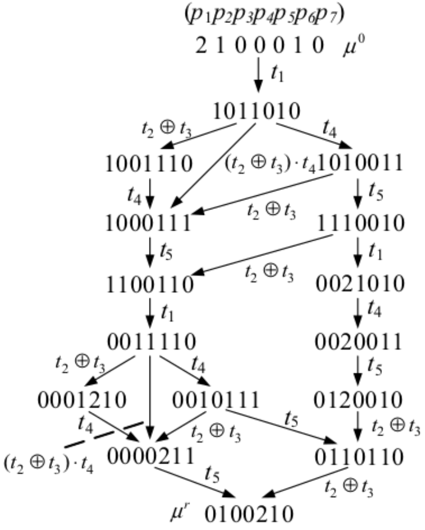
Пометка дуг вида $(t_i \oplus t_k) \cdot t_k$ – срабатывание переходов t_i, t_k или $t_{,t_k}$

Диаграммы маркировок

Построим диаграмму маркировок для сети



Диаграммы маркировок



Диаграммы маркировок

- В примере выше получилась всего одна конечная маркировка: `0100210`
- В общем случае одной начальной маркировке могут соответствовать несколько результирующих маркировок.
- Выполнение модели *асинхронно*
- Порядок срабатывания переходов неопределён

Outline

Классификация сетей Петри

- Различные СП могут отличаться друг от друга не только структурами графов, но и диаграммами смены маркировок.
- Как правило одной модели соответствует несколько одинаковых сетей Петри с разными маркировками
- Классифицируем сети Петри по динамическим (зависящих от маркировок) и по статическим (зависящим только от конфигурации сети, но не от маркировки)

Outline

Классификация по динамическим ограничениям

- **k-ограниченной** ($k \geq 1$ – целое число), если на множестве ее достижимых состояний не найдется ни одной позиции $p_i \in P$, для которой $\mu(p_i) > k$ (в которой при функционировании сети Петри появилось бы более k маркеров);
- **безопасной** , если она 1-ограничена (ни в одной ее позиции не может появиться более одного маркера);
- **ограниченной**, если найдется такое k, для которого она k-ограничена;

Свойство ограниченности СП отражает конечность информации, хранимой в «памяти» СП

Классификация по динамическим ограничениям

- **1-консервативной**, если в процессе функционирования сети Петри общее число маркеров в ней остается постоянным, т.е. для любого $t_r \in T$ имеет место

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

- **Консервативной** если существует положительная целочисленная функция $f : P\text{ß}N$ такая, что для любого перехода $t_r \in T$ имеет место

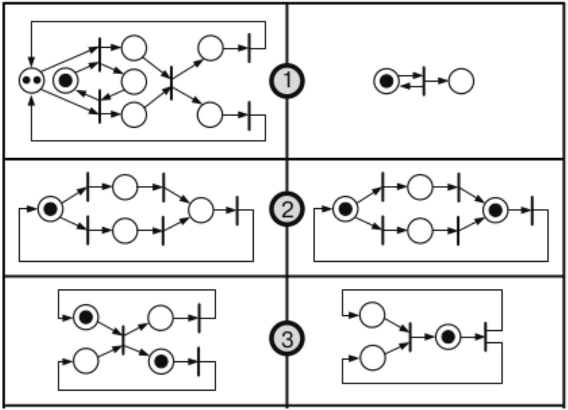
$$\sum_{p_i \in I(t_r)} f(p_i) = \sum_{p_j \in I(t_r)} f(p_j)$$

Классификация по динамическим ограничениям

- Консервативность СП может, например, показывать, что в моделируемой системе ограничен объем ресурсов.
- В современных постановках задач часто необходимо моделировать потоки потребляемой энергии (важнейшего ресурса в мобильных системах с ограниченной емкостью батарейки). Моделируя маркерами кванты энергии, выделяемой на выполнение операций (переходов СП), таким образом можно связать ограничения по энергии со свойствами модели системы.

Примеры

Слева приведён пример, справа – контрпример



ограниченные (1)
безопасные (2)
1-консервативные (3)

Классификация по динамическим ограничениям

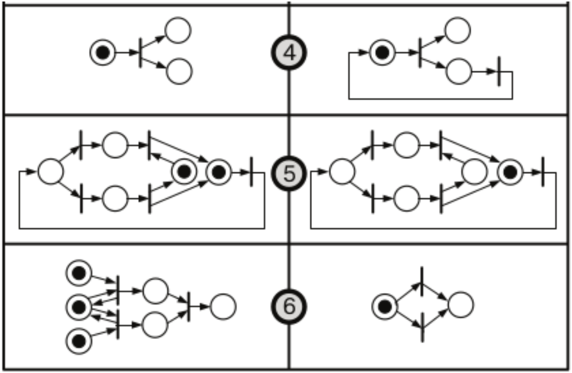
- **живой** (активной), если каждый переход $t \in T$ является потенциально срабатывающим при любой маркировке из M;
- **устойчивой**, если для всех пар $t_i, t_j \in T, i \neq j$ и любой допустимой маркировки, при которой t_i и t_j возбуждены, срабатывание одного из них не может снять возбуждения другого.

Классификация по динамическим ограничениям

- если мы хотим, чтобы все органы в моделируемой системе оставлись активными в течение всего цикла ее работы, соответствующая сеть Петри должна быть живой.
- В системе с нарушением устойчивости в ее модели возможны поведенческие конфликты

Примеры

Слева приведён пример, справа – контрпример



консервативные (4)
живые (5)
устойчивые (6)

Outline

Классификация по статическим ограничениям

Смысл следующей классификации связан с возможностью представления в поведении параллельных вычислительных процессов определенных черт: ветвлений, обусловленных выбором, и параллельных потоков, связанных с независимыми действиям на разных ресурсах.

Классификация по статическим ограничениям

- **сеть свободного выбора**, если для любых $t_j \in T$ и $p_i \in I(t_j)$ позиция p_i является либо единственной входной позицией перехода t_j, т.е. $|O(p_i)| = 1$, либо этот переход имеет единственную входную позицию, т.е. $|I(t_j)| = 1$

другими словами, если два перехода имеют общую входную позицию, то эта позиция единственна для каждого перехода

Классификация по статическим ограничениям

- **маркированный граф**, если каждая позиция имеет в точности по одному входному и выходному переходу, т.е. если $|O(pi)| = |I(pi)| = 1$

Марикированные графы – это класс СП, в которых невозможен выбор по условиям. В них может лишь присутствовать параллельность. Полезны при моделировании не меняющихся (от данных) последовательностями инструкций, например, в конвейерах.

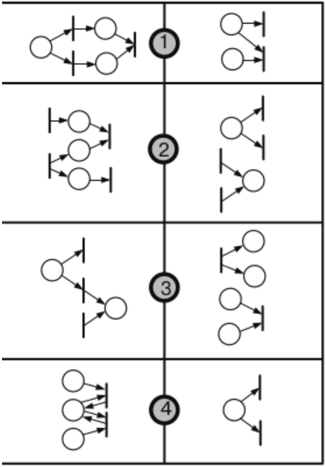
Классификация по статическим ограничениям

- **автоматной**, если каждый переход t j имеет не более одного входа и не более одного выхода, т.е. если $|O(tj)| = |I(tj)| = 1$

В автоматной СП, напротив, не может быть параллельности, зато может быть выбор. Эти классы в принципе дуальны, и дополняют друг друга, что отражено в сетях свободного выбора, которые покрывают оба этих класса по своим возможностям

Примеры

Слева допустимые фрагменты, справа недопустимые



СП свободного выбора (1)
СП - маркированный граф (2)
автоматная СП (3)
бесконфликтая СП (4)

Классификация по статическим ограничениям

- **бесконфликтной** , если либо для каждой ее позиции $p_i \in P$ существует не более одной исходящей дуги, т.е. $|O(pi)| \leq 1$, либо для всех $t_j \in O(p_i)$ выполняется $t_j \in I(p_i)$ (любая позиция, являющаяся входной для более, чем одного перехода, является одновременно и выходной для каждого такого перехода).

Бесконфликтные СП устойчивы, хотя обратное справедливо не всегда.

Outline

Дедлок

Взаимная блокировка (deadlock) — ситуация в многозадачной среде или СУБД, при которой несколько процессов находятся в состоянии ожидания ресурсов, занятых друг другом, и ни один из них не может продолжать свое выполнение.

Шаг	Процесс 1	Процесс 2
0	Хочет захватить A и B, начинает с A	Хочет захватить A и B, начинает с B
1	Захватывает ресурс A	Захватывает ресурс B
2	Ожидает освобождения ресурса B	Ожидает освобождения ресурса A
3	Взаимная блокировка	

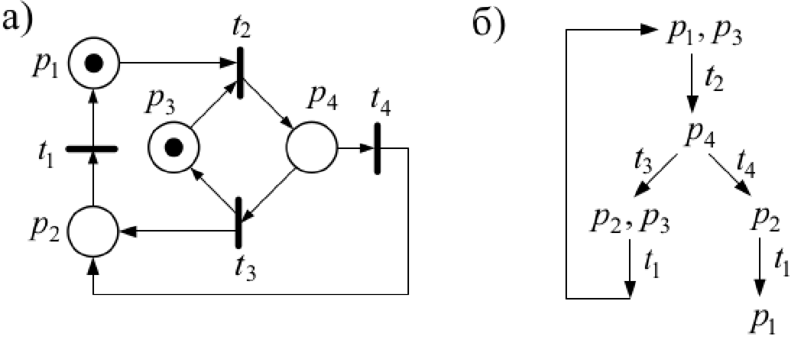
Дедлок

Дедлоком в сети Петри – подмножество позиций D, обусловливающих попадание сети в тупиковое состояние (маркировку), в котором не возбуждается ни один переход.

$D \subseteq P$ такое, что $I(D) \subseteq O(P)$.

Дедлок – это не обязательно тупиковая маркировка, скорее это потенциальный тупик, т.е. маркировка, обещающая стать тупиковой вследствие возможного дефицита маркеров

Дедлок



Какие позиции образуют дедлок?

Дедлок

$D = \{p_3, p_4\}$.

Дедлок

$D = \{p_3, p_4\}$.

Докажем это
$I(p_3) = \{t_3\}$, $I(p_4) = \{t_2\}$,
Тогда
$I(p_3, p_4) = I(p_3) \cup I(p_4) = \{t_2, t_3\}$,

$O(p_3) = t_2, O(p_4) = \{t_3, t_4\}$,
$O(\{p_3, p_4\}) = O(p_3) \cup O(p_4) = \{t_2, t_3, t_4\}$,
откуда

$I(p_3, p_4) \subset O(p_3, p_4)$, так что позиции p_3 и p_4 образуют дедлок.

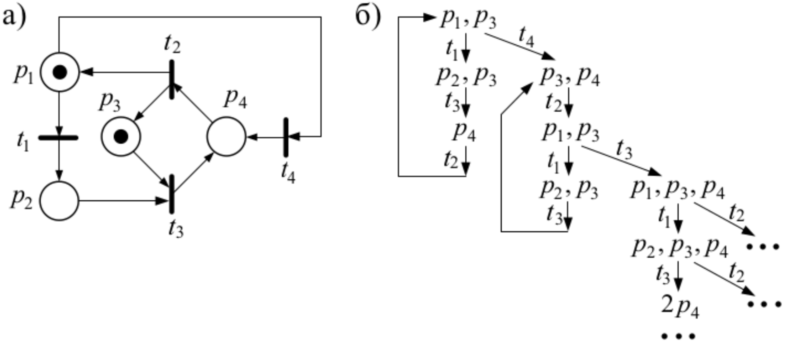
Ловушка

Ловушкой в сети Петри называется такое множество позиций L, что каждый переход, входом для которого является одна из позиций множества, имеет выходом другую позицию этого же множества.

Ловушка – цикл в сети Петри, из которого нет выхода.

$L \subseteq P$ такое, что $O(L) \subseteq I(L)$.

Ловушка



$$L = p_3, p_4$$

Outline

Анализ диаграммы переходов: матричный подход

С ростом размерности сети сложность выявления фундаментальных свойств (ограниченности, живости и др.) путём построения диаграммы маркировок становится слишком большой.

Анализ диаграммы переходов: матричный подход

Будем задавать сеть Петри так: $N =< P, T, \alpha, \beta >$

- P – непустое множество позиций
- T – непустое множество переходов
- $\alpha : P \times T \rightarrow \mathbb{N}_0$ – функция инцидентности по входу
- $\beta : T \times P \rightarrow \mathbb{N}_0$ – функция инцидентности по выходу

Анализ диаграммы переходов: матричный подход

Тогда сеть N можно описать матрицей инцидентности, где

$$c_{ij} = \begin{cases} \beta(t_j, p_i), & \text{если } \beta(t_j, p_i) \neq 0 \\ -\alpha(p_i, t_j), & \text{если } \alpha(p_i, t_j) \neq 0 \\ 0, & \text{в противном случае} \end{cases}$$

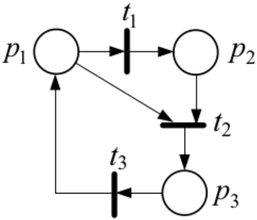
Тогда запуск перехода t_j можно записать так:

$$\mu' = \mu + C_j$$

Анализ диаграммы переходов: матричный подход

- Матрица входной функции $D^{-} = ||d_{ij}||$, где $d_{ij} = \alpha(p_j, t_j)$
- Матрица выходной функции $D^{+} = ||d_{ij}||$, где $d_{ij} = \beta(p_j, t_j)$

Анализ диаграммы переходов: матричный подход



	p_1	p_2	p_3
t_1	–1	1	0
t_2	–1	–1	1
t_3	1	0	–1

Сеть Петри и её матрица инцидентности

Анализ диаграммы переходов: матричный подход

Введём дополнительные обозначения:

- $\sigma = t_{j_1} t_{j_2} \ldots t_{j_k}$ – последовательность срабатывания k переходов, приводящая к маркировке μ^k

  Для этой последовательности можно записать вектор
- $\bar{\sigma} = (\sigma_1, \ldots, \sigma_1)$,
  каждая компонента равна числу вхождений t_j в σ

Тогда

$$\mu^k = \mu^0 + \sigma C$$

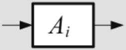
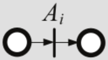
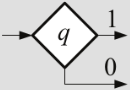
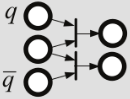
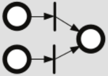
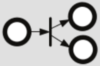
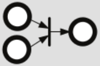
фундаментальное уравнение СП

Анализ диаграммы переходов: матричный подход

Пример выполнения сети

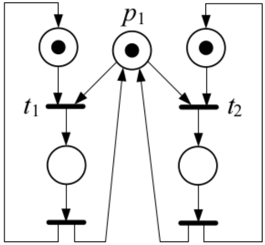
Outline

|| программирование и сети Петри

Номер рис.	Тип вершины в ПАБС	Название вершины	Фрагмент сети Петри
а)		Оператор	
б)		Условный переход	
в)		Сборка	
г)		Бифуркатор	
д)		Синхронизатор	

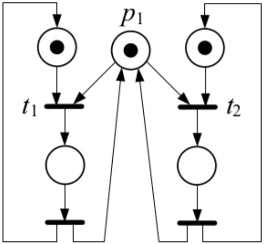
ПАБС – параллельная асинхронная блок-схема

|| программирование и сети Петри



p_1 – общий для двух процессов ресурс

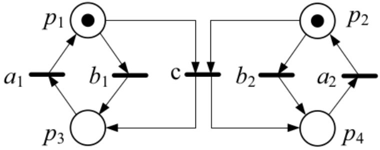
|| программирование и сети Петри



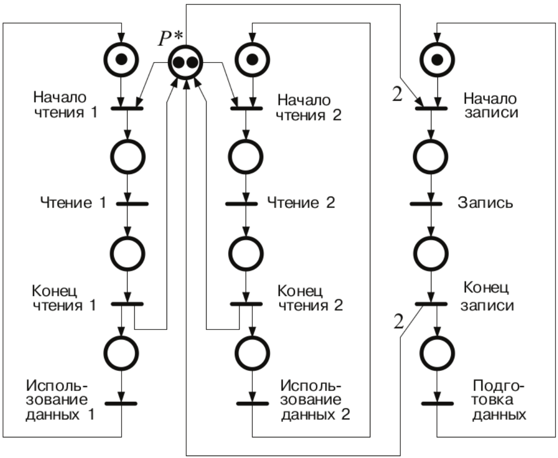
p_1 – общий для двух процессов ресурс

Будет ли выполнение || алгоритма эффектным, если один из процессов выполняется быстрее?

|| программирование и сети Петри



|| программирование и сети Петри



Outline

Ссылки

- Моделирование параллельных процессов. Сети Петри. Мараховский В. Б., Розенблюм Л. Я., Яковлев А. В. — СПб.: Профессиональная литература, 2014. – 400 с
- Теория сетей Петри и моделирования систем, Питерсон Дж. 1984
- petri.hp102.ru/pnet.html – создание и запуск сетей Петри
- apo.adrian-jagusch.de – создание и анализ сетей Петри

Ссылки

Материалы дисциплины

github.com/ivtipm/ProcessCalculus