

Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение высшего образования  
«Российский химико-технологический университет имени Д. И. Менделеева»

Программа дополнительного профессионального образования  
«Прикладные методы, средства и технологии искусственного интеллекта»

Дисциплина «Методы и системы искусственного интеллекта в организациях химической  
отрасли»

## Лекция 4. Сетевые структурно-лингвистические модели представления знаний

Семантические сети, нечеткие сети Петри, онтологии.  
Примеры разработки семантических сетей и онтологий

**Ведущий преподаватель:** кандидат технических наук,  
доцент **Михайлова Павла Геннадьевна**

# Классификация моделей представления знаний

Модель знаний определяет способ формального описания знаний в базе знаний (БЗ)



MDA (Model Driven Architecture) консорциума Object Management Group (OMG)

ODP (Model of Open Distributed Processing), зафиксированная в стандарте ISO/ITU

# Сетевые модели знаний

Наиболее общий способ представления знаний, при котором ПрО рассматривается как совокупность объектов и связывающих их отношений, реализован в **сетевой модели знаний**.

К сетевым моделям относятся: сети Петри, семантические сети, онтологии и др.

В качестве носителя знаний в этой модели выступает **семантическая сеть** – это **ориентированный граф, состоящий из** вершин, которые соответствуют объектам (понятиям) ПрО, и дуг — отношениям между понятиями.

Другими словами, структурно семантическую сеть можно представить в виде

$$\{V, E, R\},$$

где  $V$  — множество вершин;  $E$  — множество ребер;  $R$  — семейство отношений, ставящее каждому элементу  $E$  пару элементов  $V$

Как следует из названия **семантическая сеть** - это **метод представления знаний** позволяет описывать объекты, явления и понятия предметной области с помощью сетевых структур, основанных на **теории графов**.

**Семантика** — это наука, устанавливающая отношения между символами и объектами, которые они обозначают, или наука, определяющая смысл знаков.

Семантические сети являются машинным языком представления знаний и обладают определенной грамматикой. Это делает возможным трансляцию семантических сетей во внутренние структуры данных в памяти компьютера, на базе которых выполняется вывод знаний.

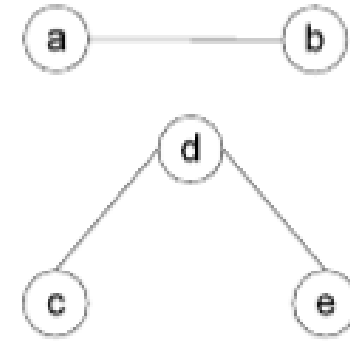
# Семантические сети. Основные понятия.

Для построения графа используются два основных вида примитивов: **узлы** и **связи**, которые могут быть дополнительно промаркированы.

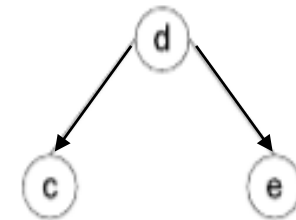
Связи могут быть **направленными** или **ненаправленными**.

Узлы являются начальными и конечными точками связей.

Узлы принято называть **вершинами**, а связи — **ребрами**, или **дугами**.



Несвязный  
неориентированный граф  
с маркированными  
вершинами



Связный ориентированный  
граф с маркированными  
вершинами

# Классификация семантических сетей

- Простые - сети, включающие вершины, не имеющие внутренней структуры.
- Иерархические – это сети, которые содержат вершины, обладающие некоторой структурой, например, в виде сети.
- Динамические (сценарии) - сети с событиями.
- По количеству типов отношений:
  - Однородные обладают только одним типом отношений,
  - Неоднородные - количество типов отношений больше двух ;
- По арности:
  - бинарные – сети с бинарными отношениями (связывающими ровно два понятия). Бинарные отношения очень просты и удобно изображаются на графе в виде стрелки между двумя объектами
  - n-арные – это сети, имеющие отношения, связывающие более двух понятий.

# Некоторые разновидности семантических сетей

Среди моделей семантических сетей, разработанных в России в последние десятилетия, наибольшую известность получили следующие:

- расширенные семантические сети И.П. Кузнецова;
- неоднородные семантические сети Г.С. Осипова;
- нечеткие семантические сети И.А. Перминова;
- обобщенная модель представления знаний о ПрО А.И. Башмакова.
- функциональные И. Н. Дорохова, В. В. Меньшикова.

**Функциональные семантические сети:** вершины – исходные данные или вычисляемые величины, дуги – отношения математического типа (формулы, операторы, сложные математические конструкции), с помощью которых одни данные преобразуются в другие.

# Семантические сети

Понятия в семантической сети связаны различными отношениями. Все они представлены в ЕЯ и обладают разными свойствами как с математической, так и с лингвистической точек зрения. Разработка универсального представления семантической сети при таком количестве типов отношений является сложной и трудоемкой задачей. К тому же оперировать подобной универсальной моделью было бы затруднительно.

При решении конкретных задач значимыми являются не все типы отношений, а их подмножество. Поэтому модели семантических сетей обычно разрабатываются как проблемно-ориентированные. В таких моделях используют ограниченное число типов отношений.

Отношения между понятиями в семантических сетях имеют следующие значения :

1) Отношение принадлежности (**является**), абстрактное - конкретное.

**IA (is a...).**

2) Отношение "**свойство**".

**HP (has property).**

3) Отношение "**является видом**".

**AKO (a kind of...).**

4) Отношение "**является частью**"/"**имеет часть**".

**PO (part of...)/ HPt (has part).**

5) Отношение "**имеет значение**"

**HU (has unit).**

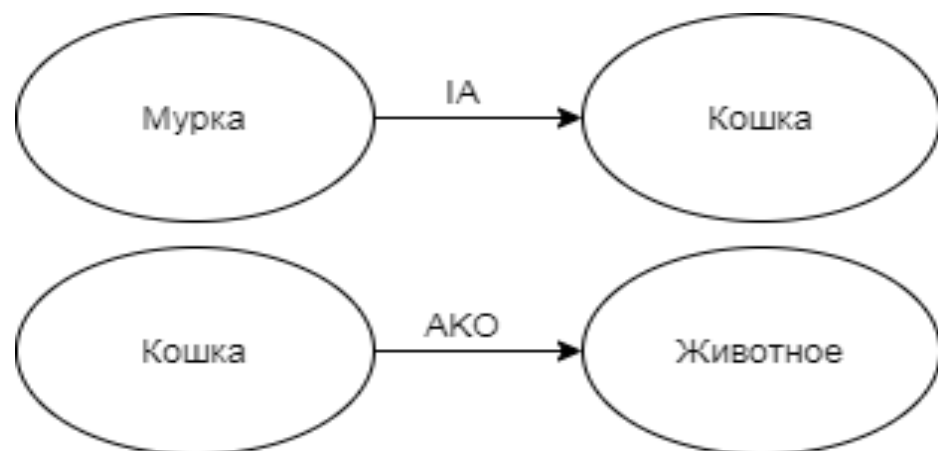
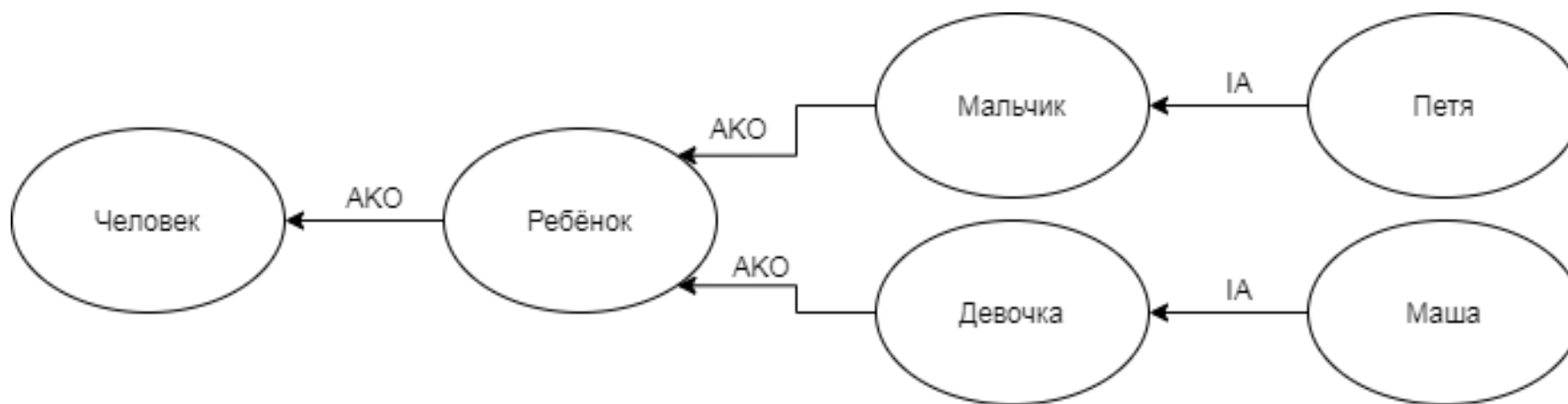
# Вспомогательные типы отношений в семантических сетях

В семантических сетях часто используются также следующие отношения:

- функциональные связи (определяемые обычно глаголами «производит», «влияет»...);
- количественные (больше, меньше, равно...);
- пространственные (далеко от, близко от, за, под, над...);
- временные (раньше, позже, в течение...);
- логические (И, ИЛИ, НЕ) и др.



# Примеры



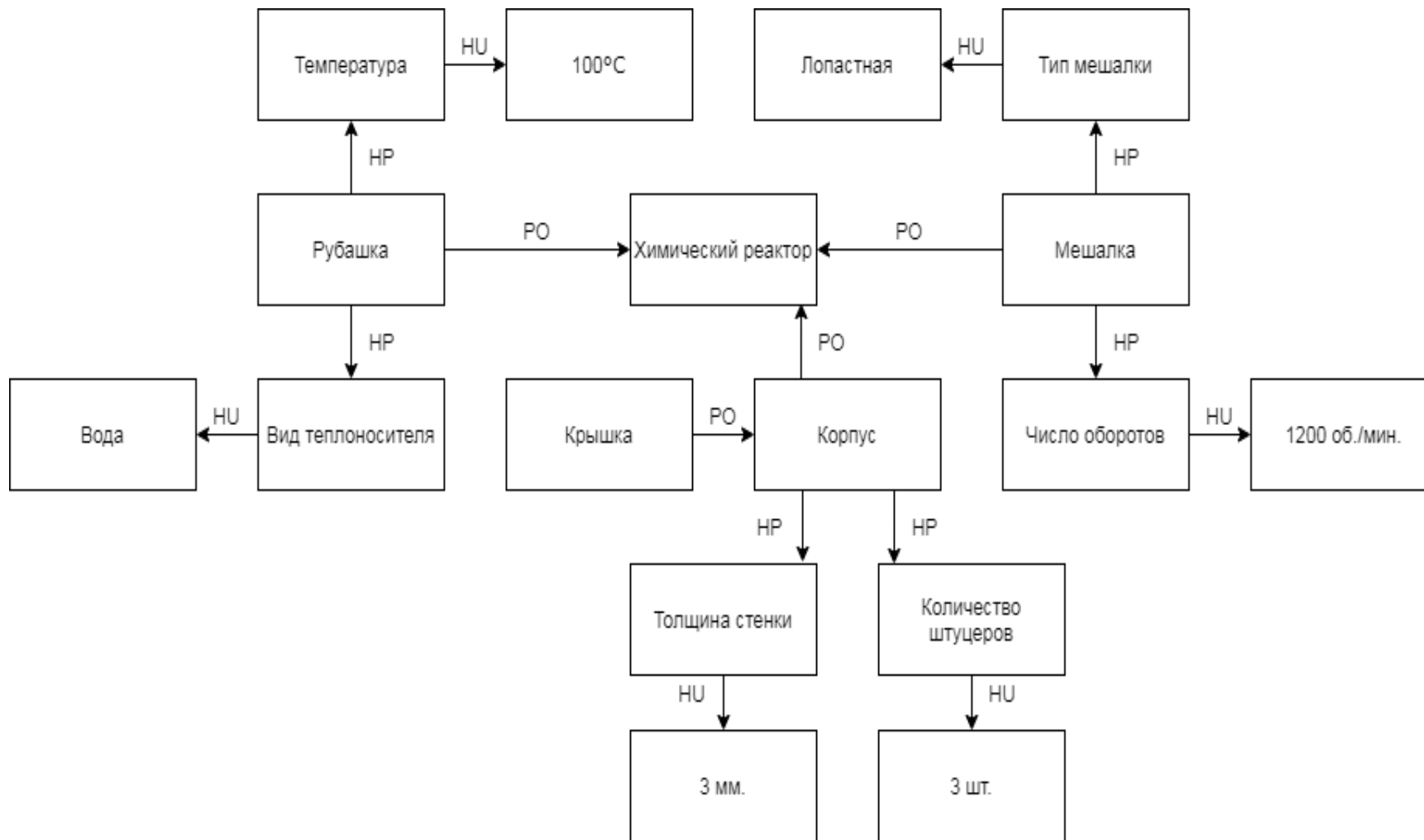
Отношение принадлежности (является),  
абстрактное - конкретное.

**IA (is a...).**

Отношение "является видом".

**АКО (a kind of...)**

# Семантическая сеть "Химический реактор"



- 1) Отношение принадлежности (**является**), абстрактное - конкретное.  
**IA (is a...).**
- 2) Отношение "**свойство**".  
**HP (has property).**
- 3) Отношение "**является видом**".  
**АКО (a kind of...).**
- 4) Отношение "**является частью**" / "**имеет часть**".  
**PO (part of...)/ HPt (has part).**
- 5) Отношение "**имеет значение**".  
**HU (has unit).**

Отличие баз знаний от баз данных состоит в том, что базы знаний содержат связи между объектами в явной форме.

# Логический вывод в семантических сетях

Основной механизм вывода: движение по связям между узлами.

Поиск пересечений -- поиск в сети фрагмента, совпадающего с фрагментом-запросом. При этом надо запоминать уже пройденные узлы.

Связи вида «**является видом**» (**АКО**) или «**иметь частью**» (**РО**) **транзитивны**, что позволяет говорить об установлении с помощью этой связи свойств иерархии наследования в сети. Это означает, что элементы более низкого уровня в сети могут наследовать свойства элементов более высокого уровня.

Наследование обеспечивает рассуждения по умолчанию.

# Достоинства семантических сетей как метода представления знаний

Достоинства :

- Наглядность представления знаний и понятность
- Легкость преобразования в логику предикатов 1-го порядка

Недостатки:

- Сложно представлять в виде графа большой объем знаний, соответственно, сложно обозревать большую семантическую сеть
- Недостаточная структурированность
- Нет возможности представлять процедурные знания

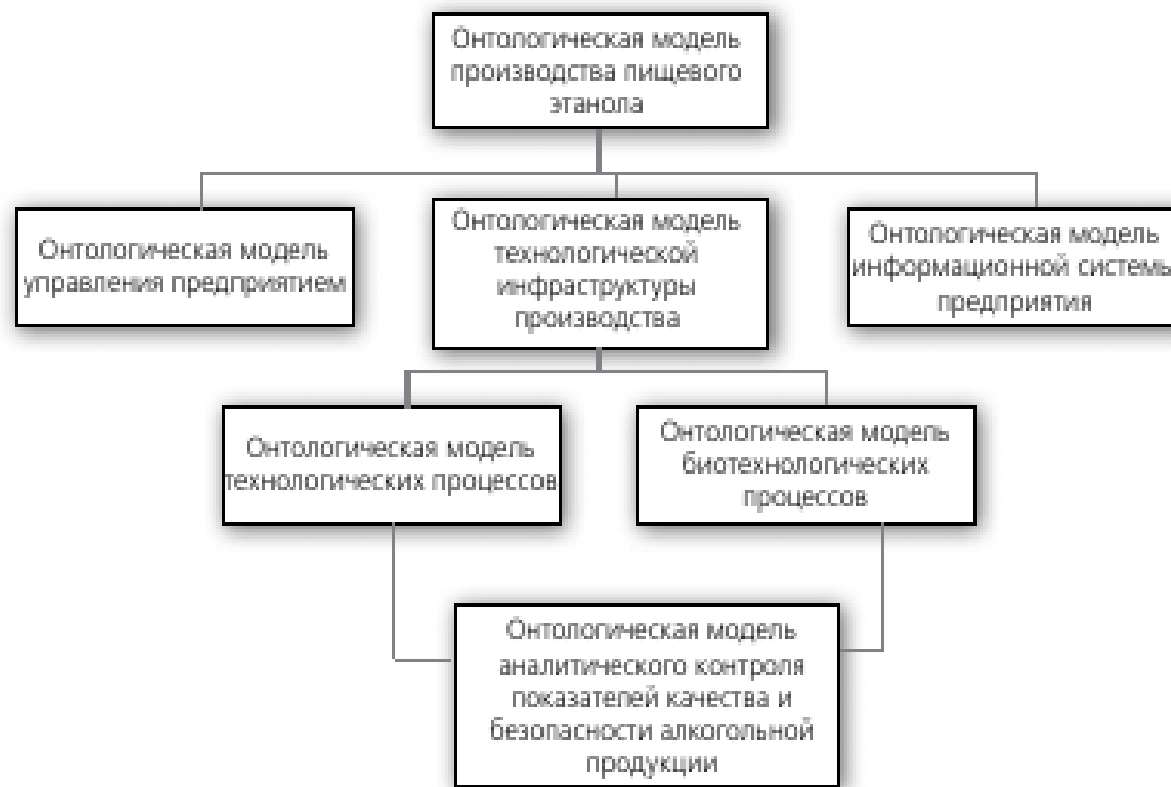
# Онтологии

Под **онтологией** (в наиболее общих чертах) понимают **систему понятий некоторой предметной области**, которая представляется как **набор сущностей, соединенных различными отношениями**.

Онтологии используются для формальной спецификации понятий и отношений, которые характеризуют определенную область знаний. Основным преимуществом онтологий как способа представления знаний является их формальная структура, которая упрощает их компьютерную обработку.

Онтологии используются как источники данных для многих компьютерных приложений (для информационного поиска, анализа текстов, извлечения знаний и т. д.), позволяя более эффективно обрабатывать сложную и разнообразную информацию. Этот способ представления знаний позволяет приложениям распознавать те семантические отличия, которые являются само собой разумеющимися для людей, но не известны компьютеру.

Онтологии получили широкое распространение в решении проблем представления знаний и инженерии знаний, семантической интеграции информационных ресурсов, информационного поиска и т.д. Интеллектуальные системы на основе онтологий показали на практике свою эффективность, однако построение онтологии требует экспертных знаний в исследуемой предметной области и занимает существенный объем времени, поэтому актуальной задачей является автоматизация процесса построения онтологии



Онтологический подход к систематизации знаний в области бродильных производств

# Онтологии

В целом онтология состоит из **иерархии понятий предметной области, связей между ними и законов, которые действуют в рамках этой модели**. Онтология строится как сеть, состоящая из концептов и связей между ними. Связи могут быть различного типа (например, «является», «состоит из», «является исполнителем» и т. п.).

Для выполнения роли общего языка будет включать в себя **словарь** (множество слов/лексических единиц) и **набор определений понятий** (формальных, на языке логической теории, или неформальных), задающих смысл элементов словаря. Связи между словарем (знаками) и семантикой ограничивают множество возможных интерпретаций знаков. Например, после установки связи между словом «Ягуар» и интенциональным определением «хищное животное семейства кошек» мы исключаем из множества возможных интерпретаций этого слова машины и напитки. В результате онтологии позволяют правильно соотносить знаки, используемые людьми и компьютерами, с понятиями (семантическими моделями) и обозначаемыми объектами реального мира.

Штудер и его коллеги дают более развернутое определение онтологии в информатике (computer science) [Studer et al, 1998] — это **формальная спецификация разделяемой концептуальной модели**,

где под «**концептуальной**» **моделью** подразумевается абстрактная модель предметной области, описывающая систему понятий данной области,

под «**разделяемой**» подразумевается согласованное понимание концептуальной модели определенным сообществом (группой людей),

«**спецификация**» подразумевает описание системы понятий в явном виде,

«**формальная**» подразумевает, что концептуальная модель задана на формализованном языке.

# Онтологии

Формально онтология определяется как

$$O = \langle X, P, R, F \rangle,$$

где  $X$  — конечное множество понятий предметной области (классов сущностей),  
 $P$  — конечное множество свойств этих понятий (классов),  
 $R$  — конечное множество отношений между понятиями,  
 $F$  — конечное множество функций интерпретации.

Конечное **множество отношений между понятиями** представляется **иерархией классов** (таксономией).

Конечное **множество функций интерпретаций** представлено **правилами**.

Особо важным характеристическим свойством онтологии является наличие родовидовых отношений между классами объектов — **таксономии классов**.

# ТИПЫ ОТНОШЕНИЙ В ОНТОЛОГИЯХ

Некоторые типы отношений между понятиями (классами) встречаются почти во всех онтологиях и составляют основу онтологического инжиниринга.

## Отношения класс-подкласс и таксономия

Отношение между классами и подклассами понятий может носить разное название в зависимости от терминологических традиций в области использования ресурса: **таксономическое отношение, родовидовое отношение, IS-а отношение, АКО (акроним от A Kind Of — является видом), отношение гипонимии и гиперонимии (в лексических ресурсах).**

**Концептуальная структура (фрагмент онтологии), построенная на основе отношения класс-подкласс называется таксономией.**

Этот термин (от греч. расположение по порядку + закон) в науке и математике обозначает **классификацию и систематизацию сложноорганизованных областей действительности**, имеющих обычно иерархическое строение (органический мир, объекты географии...).

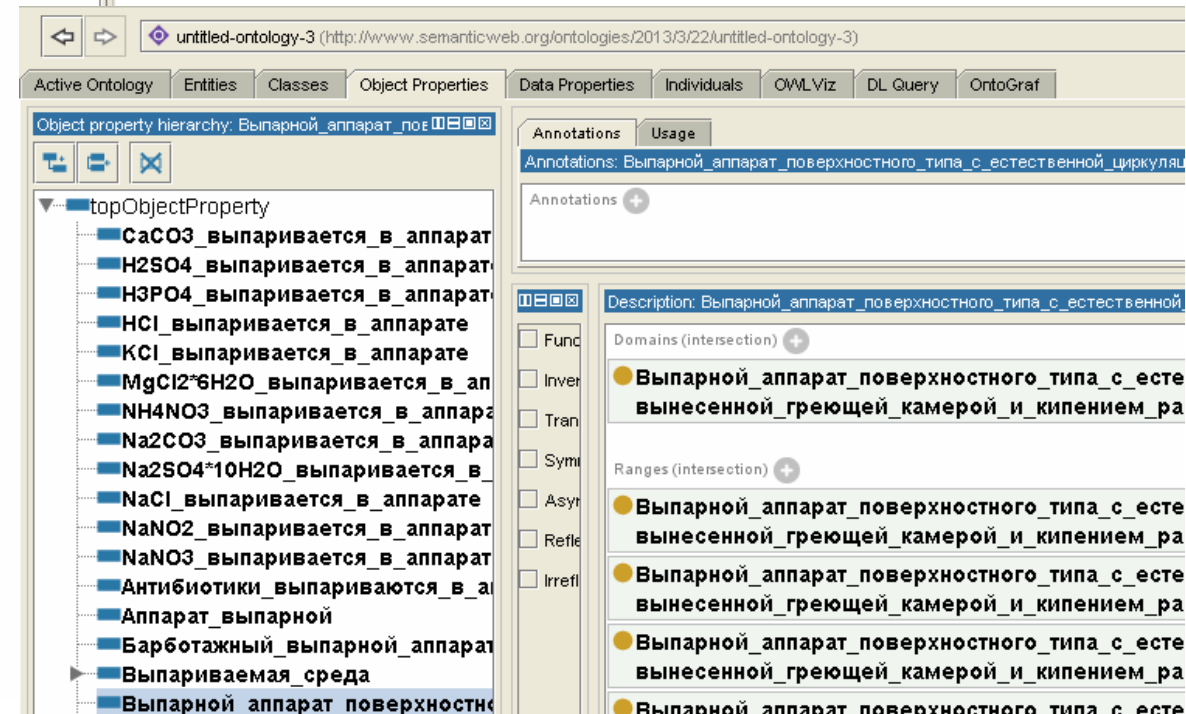
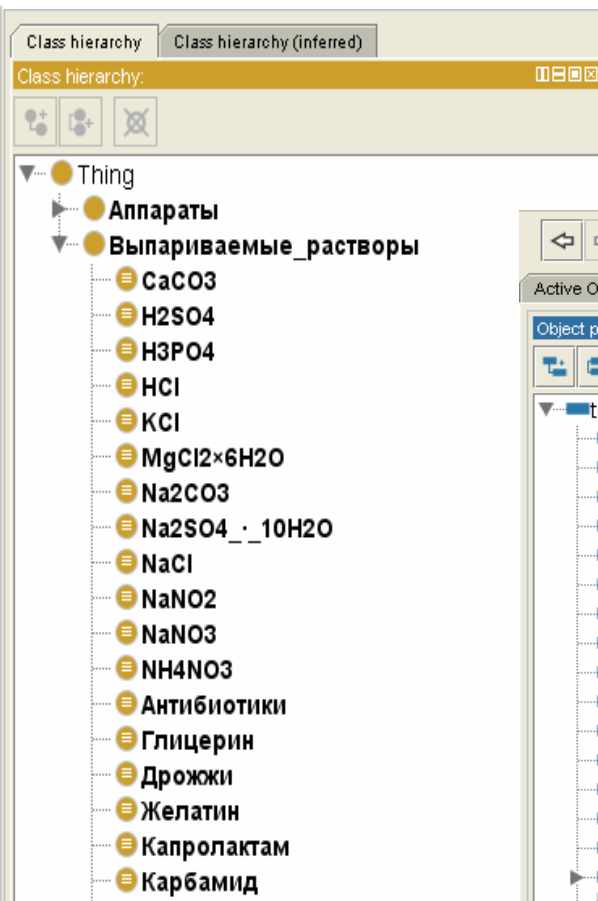
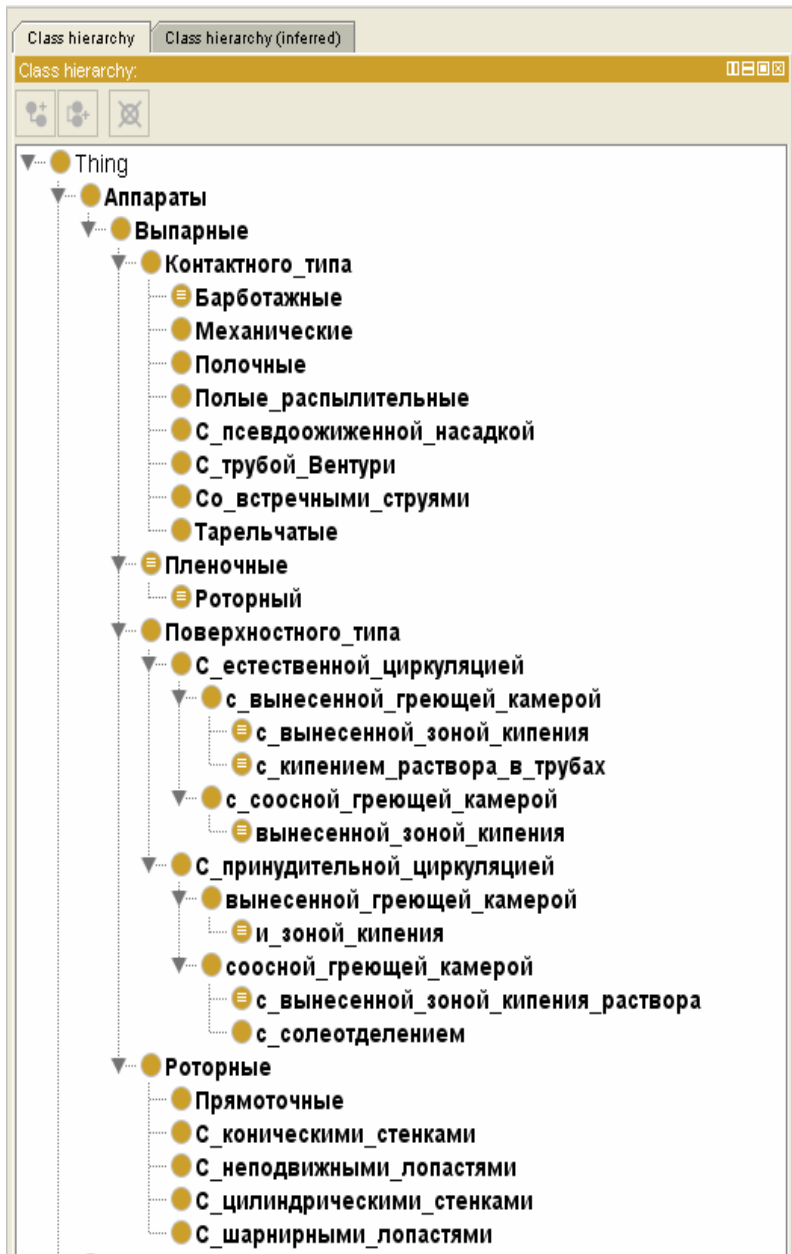
В качестве **синонима** таксономии может **выступать родовидовая классификация понятий**, в которой **вышестоящий** элемент называется **родом**, а его **потомок — видом**.

**Отношения класс-подкласс** обладают такими важными свойствами, как **транзитивность и наследование**, на которых основывается логический вывод во многих компьютерных системах.

Пусть  $T(X, Y)$  — отношение *класс-подкласс* между понятиями  $X$  и  $Y$ ,  $Y$  является подклассом (видом)  $X$ ,  $R(X, Z)$  — это произвольное отношение между понятиями  $X$  и  $Z$ . Тогда свойства отношения *класс-подкласс* могут быть записаны следующим образом:

- $T(X, Y) \wedge T(Y, Z) \rightarrow T(X, Z)$  — **транзитивность** отношения *класс-подкласс* (Если  $Y$  является подклассом  $X$  и  $Z$  является подклассом  $Y$ , то  $Z$  является подклассом  $X$ )
- $T(X, Y) \wedge R(X, Z) \rightarrow R(Y, Z)$  — **свойство наследования** по отношению *класс-подкласс*: «Если  $Y$  является подклассом  $X$  и  $X$  связан неким отношением с  $Z$ , то  $Y$  тоже связан этим отношением с  $Z$ ».





Дерево иерархии  
(таксономия) растворов

Свойства, связывающие выпарные аппараты и  
растворы, выпариваемые в них

Онтология реализована в среде свободно распространяемой программы  
Protégé

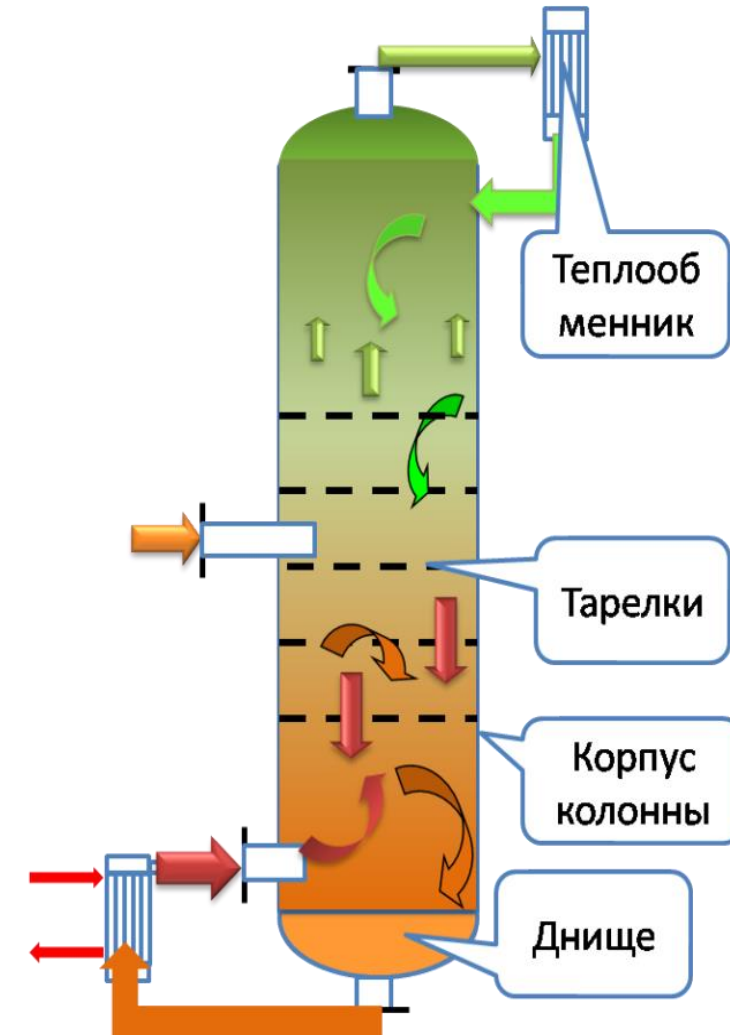
Дерево иерархии  
(таксономия) аппаратов

# ТИПЫ ОТНОШЕНИЙ В ОНТОЛОГИЯХ

## Отношение «часть-целое» и партономия

Отношение «часть-целое» играет существенную роль во многих предметных областях. Для компьютерных приложений особое значение представляет такое свойство отношения «часть-целое», как **транзитивность**, на основе которой может строиться многошаговый логический вывод. Для некоторых типов отношений «часть-целое» может выполняться наследование свойств и операций от части к целому и от целого к части, что также может стать базой для логического вывода.

**Концептуальная структура или фрагмент онтологии, основанный на отношении часть-целое называется партономией.**



# Базовая методология создания онтологий

Предложена Натальей Ной и Деборой МакГиннесс для начинающих инженеров по знаниям [Noy, McGuinness, 2001]. Данная методология состоит из 7 шагов.

Шаг 1. Определение области и масштаба онтологии.

Шаг 2. Рассмотрение вариантов повторного использования существующих онтологий.

Шаг 3. Перечисление важных терминов в онтологии.

Шаг 4. Определение классов и иерархии классов.

Шаг 5. Определение свойств классов (слотов).

Шаг 6. Определение ограничений на значения свойств (слотов).

Шаг 7. Создание экземпляров.

# Базовая методология создания онтологий

## Шаг 1. Определение области и масштаба онтологии

Для этого нужно ответить на несколько основных вопросов:

1. Какую область будет охватывать онтология?
2. Для чего мы собираемся использовать онтологию?
3. На какие типы вопросов должна давать ответы информация в онтологии?
4. Кто будет использовать и поддерживать онтологию?

# Базовая методология создания онтологий

## Шаг 2. Рассмотрение вариантов повторного использования существующих онтологий

Почти всегда стоит учесть, что сделал кто-то еще, и проверить, можем ли мы улучшить и расширить существующие источники для нашей конкретной предметной области и задачи. Повторное использование существующих онтологий может быть необходимым, если нашей системе нужно взаимодействовать с другими приложениями, которые уже вошли в отдельные онтологии или контролируемые словари. Многие онтологии уже доступны в электронном виде и могут быть импортированы в используемую среду проектирования онтологий. В литературе и Интернете существуют библиотеки повторно используемых онтологий.

Например, библиотеки биомедицинских онтологий <https://bioportal.bioontology.org/>, <http://www.obofoundry.org/>.

Существуют инструменты поиска онтологий: <http://watson.kmi.open.ac.uk/WatsonWUI/>.

# Базовая методология создания онтологий

## Шаг 3. Перечисление важных терминов в онтологии

Полезно составить список всех терминов, о которых мы хотели бы сказать что-либо или которые хотели бы объяснить пользователю.

- Какие термины мы бы хотели рассмотреть?
- Какие свойства имеют эти термины?
- Что бы мы хотели сказать об этих терминах?

Вначале важно получить полный список терминов, не беспокоясь о пересечении понятий, которые они представляют, об отношениях между терминами, о возможных свойствах понятий или о том, чем являются термины — классами или свойствами.

# Базовая методология создания онтологий

## Шаг 4. Определение классов и иерархии классов

Существует несколько возможных подходов для разработки иерархии классов:

- процесс **нисходящей разработки** начинается с определения самых общих понятий предметной области с последующей конкретизацией понятий;
- процесс **восходящей разработки** начинается с определения самых конкретных классов, листьев иерархии, с последующей группировкой этих классов в более общие понятия;
- процесс **комбинированной разработки** — это сочетание нисходящего и восходящего подходов: сначала мы определяем более заметные понятия, а затем соответствующим образом обобщаем и ограничиваем их.

## Шаг 5. Определение свойств классов (слотов)

Классы сами по себе не предоставляют достаточно информации для ответа на вопросы проверки компетентности из Шага 1. После определения некоторого количества классов мы должны описать их внутреннюю структуру. Мы уже выбрали классы из списка терминов, который мы создали на Шаге 3. Большинство оставшихся терминов, вероятно, будут свойствами этих классов.

# Базовая методология создания онтологий

## Шаг 6. Определение ограничений на значения свойств (слотов)

**Свойства** (в некоторых системах используется понятие «слоты») могут иметь различные **ограничения, которые описывают тип значения, разрешенные значения, число значений (мощность, кардинальность)** и другие характеристики значений, которые может принимать свойство.

**Мощность (кардинальность)** свойства определяет, сколько значений может иметь свойство. В некоторых системах различаются только единичная мощность (возможно только одно значение) и множественная мощность (возможно любое число значений).

Минимальная мощность  $N$  означает, что слот должен иметь не менее  $N$  значений.

**Тип значения свойства.** Вот список наиболее общих типов значений: Строка, Число, Булевы свойства (простые флаги «да — нет»), Нумерованные свойства (список конкретных разрешенных значений свойства).

**Объектные свойства** (в соответствии с языком представления онтологий **RDFS (Resource Description Framework Schema - Схема структуры описания ресурса)**) позволяют определить отношения между экземплярами классов (индивидами). Для объектных свойств также должен быть определен список разрешенных классов, экземпляры которых можно использовать в качестве значений свойства.

**Домен свойства и диапазон значений свойства.** Любое свойство в онтологии имеет область значений — домен (domain), и область определений — диапазон (range). Домен представляет собой класс, экземпляры которого будут связываться этим свойством, а диапазон — класс, с экземплярами которого будет связывать указанное свойство или тип данных.



# Базовая методология создания онтологий

## Шаг 7. Создание экземпляров

Последний шаг — это создание отдельных экземпляров классов в иерархии.

Для определения отдельного экземпляра класса требуется

1. выбрать класс,
2. создать отдельный экземпляр этого класса,
3. ввести значения свойств.

В случае необходимости получения более формальной онтологии, могут быть добавлены шаги с описанием ограничений классов (в том числе задание формальных определений)

# Сети Петри. Определение

*Сетью Петри* называется *ориентированный граф*:

$$G=(P, T, F, M_0), \quad (1)$$

где  $P = \{P_i / i = \overline{1, I}\}$  – конечное множество вершин, называемых **позициями**;

$T = \{t_j / j = \overline{1, J}\}$  – конечное множество вершин, называемых **переходами**;

$F: P \times T \cup T \times P \rightarrow \{0, 1\}$  – **функция инцидентности**, указывающая наличие дуг, связывающих позиции с переходами и переходы с позициями;

$M_0: P \rightarrow \{0, 1, 2, \dots\}$  – начальная **разметка** (маркирование) сети.

# Графическая интерпретация сети Петри

- **Позиции**

$P = \{P_i / i = \overline{1, I}\}$  - состояние процесса, т. е. технологические операциями, которые осуществляются в аппарате периодического действия

- **Переходы**

$T = \{t_j / j = \overline{1, J}\}$  – события, которые происходят в системе и переводят ее из одного состояния в другое (окончание технологических операций)

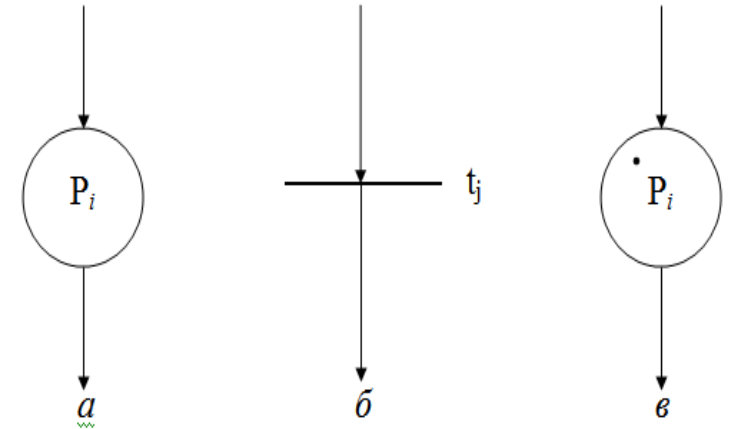


Рис. 1. Графическая интерпретация сети Петри:  
а – позиция; б – переход;  
в –маркирование

# Функция инцидентности

Функция инцидентности сети Петри имеет следующий смысл.

1. Если  $F(P_i, t_j) = 1$ , то **позиция**  $P_i$  - называется **входной** по отношению к переходу  $t_j$ , а **переход**  $t_j$  называется **выходным** по отношению к позиции  $P_i$ .

a)  $*t_j = \{P_i / F(P_i, t_j) = 1\}$  - множество **входных позиций** перехода  $t_j$  (рис. 2)

b)  $P_i^* = \{t_j / F(P_i, t_j) = 1\}$  - множество **выходных переходов** позиции  $P_i$  (рис. 3).

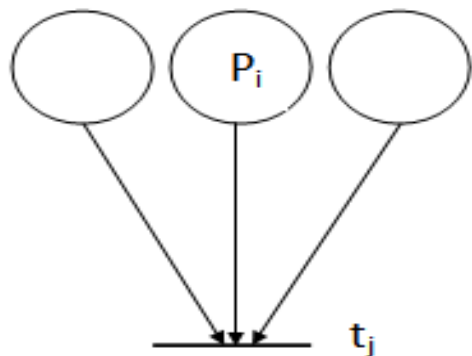


Рис. 2. Входные позиции перехода  $t_j$

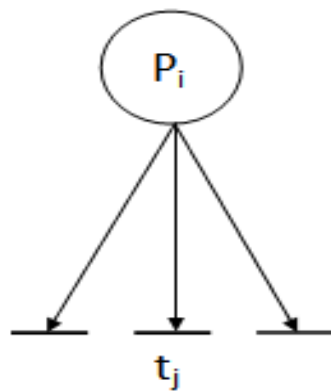


Рис. 3. Выходные переходы позиции  $P_i$

# Функция инцидентности

2. Если  $F(t_j, P_i) = 1$ , то **переход**  $t_j$  называется **входным** по отношению к позиции  $P_i$ , а **позиция**  $P_i$  называется **выходной** по отношению к переходу  $t_j$ .

a)  $*P_i = \{t_j / F(t_j, P_i) = 1\}$  - множество **входных переходов** позиции  $P_i$  (рис. 4)

b)  $t_j^* = \{P_i / F(t_j, P_i) = 1\}$  - множество **выходных позиций** перехода  $t_j$  (рис. 5).

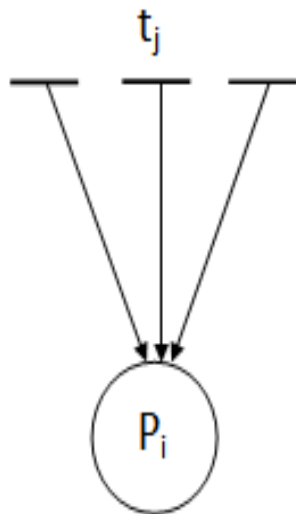


Рис. 4. Входные переходы позиции  $P_i$

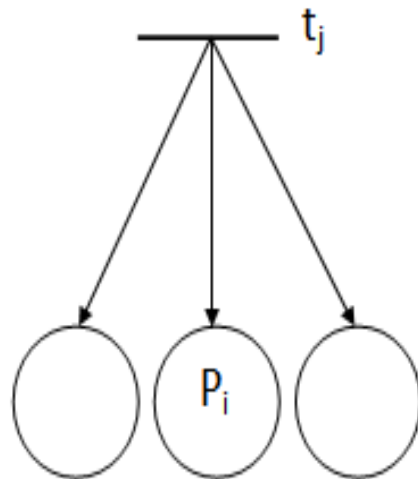


Рис. 5. Выходные позиции перехода  $t_j$

# Условия срабатывания перехода

Переход  $t_j$  может сработать, когда:

$$(\forall P_i \in {}^*t_j) [M(P_i) \geq F(P_i, t_j)], \quad (2)$$

т. е. когда число меток в каждой входной позиции этого перехода не меньше числа кратных дуг, соединяющих эту позицию с данным переходом. Переход, удовлетворяющий этому условию, называется **активным** или **возбужденным**.

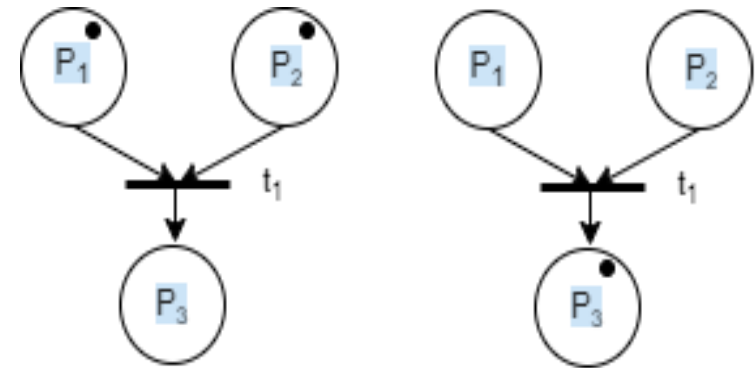
**Переход срабатывает, когда все его входные позиции содержат маркеры.**

При срабатывании активного перехода  $t_j$  маркирование  $M'$  сети Петри заменяется маркирование  $M''$  в соответствии со следующим алгоритмом:

$$(\forall P_i \in {}^*t_j) [M''(P_i) = M'(P_i) - F(P_i, t_j)]; \quad (3)$$

$$(\forall P_i \in t_j^*) [M''(P_i) = M'(P_i) + F(t_j, P_i)], \quad (4)$$

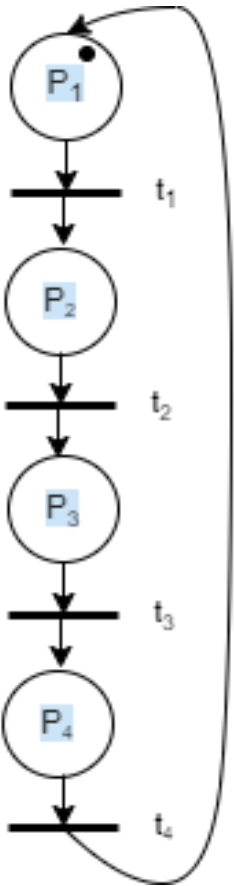
т. е. при срабатывании каждого перехода из каждой его входной позиции удаляется столько меток, сколько дуг ведет из этой позиции к данному переходу, а в каждую его выходную позицию добавляется столько меток, сколько дуг ведет в эту позицию из данного перехода.



# Пример

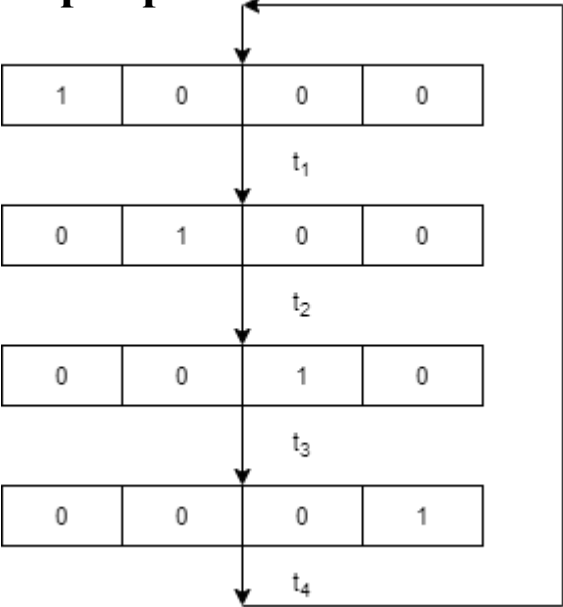
Пусть имеем технологический аппарат периодического действия (реактор). Пусть есть партия сырья. Когда основной технологический аппарат свободен, то начинается технологический процесс, по окончании которого партия выгружается из реактора.

№ п/п	Технологические операции	Позиции	События, происходящие в системе	Переходы
1	АПД свободен, готов к технологическому процессу	P <sub>1</sub>	Открытие клапана на линии подачи сырья	t <sub>1</sub>
2	Загрузка сырья	P <sub>2</sub>	Загрузка сырья в АПД окончена	t <sub>2</sub>
3	Протекание технологического процесса	P <sub>3</sub>	Технологический процесс окончен	t <sub>3</sub>
4	Выгрузка продукта	P <sub>4</sub>	Выгрузка окончена	t <sub>4</sub>



Сеть Петри, моделирующая работу аппарата периодического действия

Для того, чтобы показать переход меток по сети Петри, используется **граф достижимых маркировок**



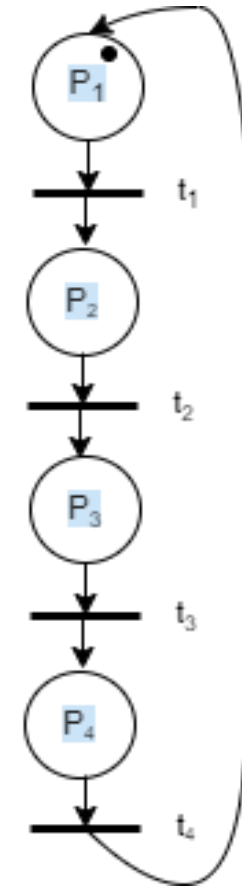
# Матрицы инцидентности

Позиция-переход (P-t)

	$t_1$	$t_2$	$t_3$	$t_4$
$P_1$	1	0	0	0
$P_2$	0	1	0	0
$P_3$	0	0	1	0
$P_4$	0	0	0	1

Переход-позиция (t-P)

	$P_1$	$P_2$	$P_3$	$P_4$
$t_1$	0	1	0	0
$t_2$	0	0	1	0
$t_3$	0	0	0	1
$t_4$	1	0	0	0





# Нечеткие сети Петри

Рассмотренные выше классы моделей сетей Петри позволяют представить структуру и динамику функционирования моделируемых систем в условиях отсутствия влияния тех или иных факторов неопределенности. Указанное предположение о детерминированном характере структурных взаимосвязей и динамики функционирования сетей Петри существенно ограничивает возможности практического использования моделей данных классов и не отражает адекватным образом отдельные аспекты знаний о предметной области.

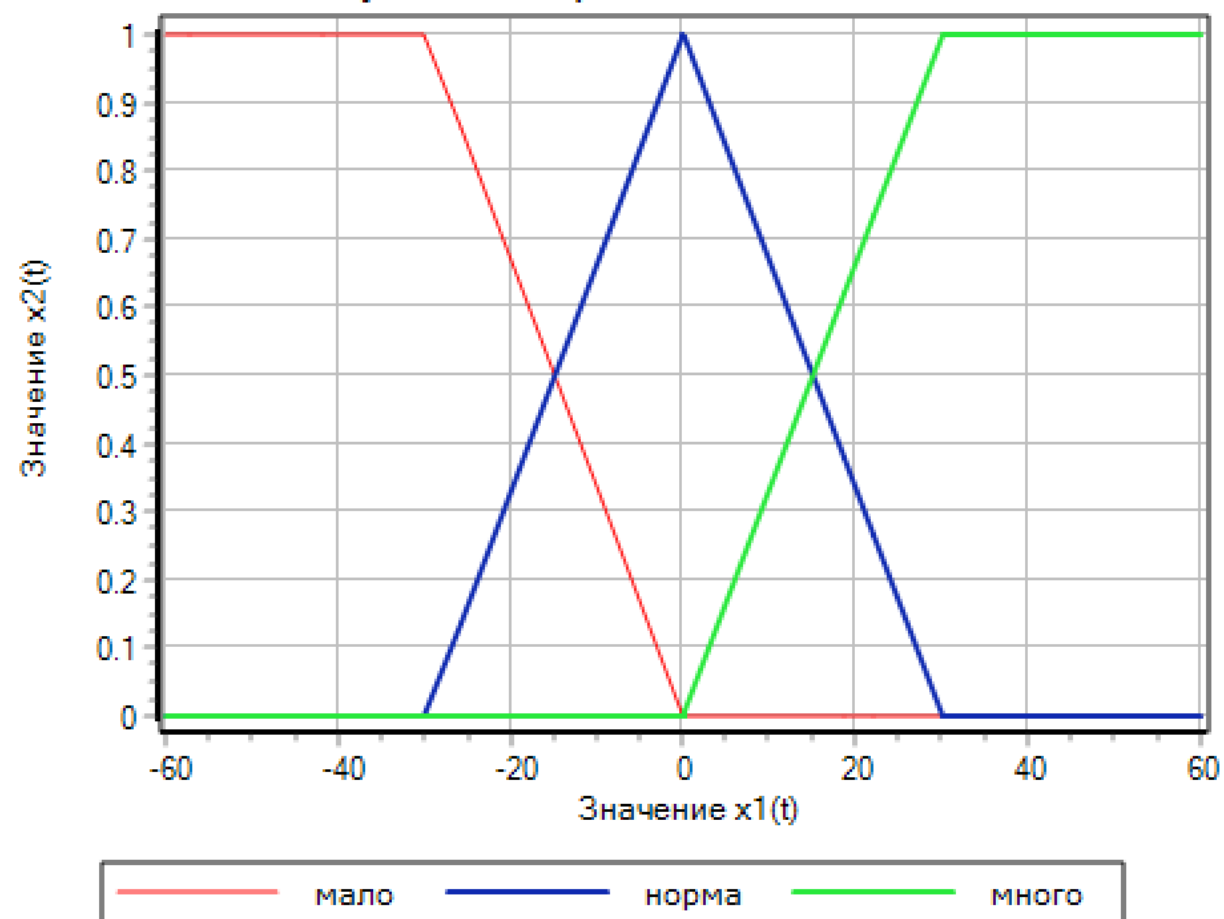
Для учета неопределенностей различной природы используется **подход нечеткой логики и нечетких множеств**.

Нечеткая логика, как следует из названия, предполагает неточные, приблизительные, примерные оценки. Она предполагает, что ситуации оцениваются приблизительно, а не точно. Необходимость такого подхода вызвана тем, что:

- в некоторых ситуациях невозможно или не нужно точное определение параметров;
- по мере роста сложности систем постепенно падает наша способность делать точные и в то же время значащие утверждения относительно ее поведения, пока не будет достигнут порог, за которым точность и значимость становятся почти взаимоисключающими характеристиками.

Таким образом, можно **вводить неопределенность в маркировку, структуру сети Петри, правила функционирования, время срабатывания переходов** и т.д. Рассмотрим подкласс нечетких сетей Петри  $C_f$  с нечеткой маркировкой.

## Функции принадлежности



# Нечеткие сети Петри

$$V_f = \langle N, \hat{M}_0, \mathcal{R} \rangle,$$

где  $N = (P, T, I, O)$  – структура НСП  $V_f$ :  $P = \{p_1, \dots, p_n\}$  – множество позиций;  $T = \{t_1, \dots, t_u\}$  – множество переходов;  $I$  – входная функция переходов,  $I: P \times T \rightarrow \mathcal{N}_0$ ;  $O$  – выходная функция переходов,  $O: T \times P \rightarrow \mathcal{N}_0$ ;  $\hat{M}_0$  – матрица начальной маркировки размерности  $(n \times (d + 1))$ . Каждый ее элемент  $m_{ij}^0 \in [0, 1]$  ( $\forall i \in \{1, 2, \dots, n\}, \forall j \in J$ ) равен значению функции принадлежности наличия  $(j - 1)$  числа маркеров в позиции  $p_i$  на момент начала запуска НСП,  $J = \{1, 2, \dots, d, d + 1\} \subset \mathcal{N}_0$ . При этом число столбцов матрицы  $\hat{M}_0$  определяется максимальным числом маркеров  $d$  в позициях НСП;  $\mathcal{R}$  – множество правил, модификация которых отражает специфику введенной нечеткости в начальную маркировку.

# Нечеткие сети Петри

**$\mathcal{R}_1$ : Правило определения текущего состояния:** Любое состояние НСП определяется некоторой его маркировкой, т. е. матрицей  $M$  размерности  $(n \times (d + 1))$ , каждый элемент которой  $m_{ij} \in [0, 1]$ , и интерпретируется как значение функции принадлежности наличия  $\{0, 1, 2, \dots, d\}$  маркеров в соответствующих позициях  $p_i \in P$ . Начальное состояние НСП определяется матрицей начальной маркировки  $M_0$ .

**$\mathcal{R}_2$ : Правило активности переходов.** Переход  $t_k \in T$  НСП называется активным при некоторой текущей маркировке  $M$ , если выполнено условие во всех его входных позициях имеется нечеткое количество маркеров, большее или равное числу дуг, соединяющих соответствующие входные позиции с рассматриваемым переходом нечеткой сети Петри.

# Нечеткие сети Петри

$\mathcal{R}_5$ : **Правило срабатывания переходов.** Если некоторый переход  $t_k \in T$  НСП активен при некоторой текущей маркировке  $M_l$ , то срабатывание данного перехода, осуществляемое мгновенно, приводит к новой маркировке сети определяются следующим образом.

- для каждой из входных позиций  $p_i \in P$ , для которых  $I(p_i, t_k) > 0$

$$\begin{aligned} m_{i1}^{l+1} &= \max_{(j \in \{1, 2, \dots, I(p_i, t_k)+1\})} \{m_{ij}\}, \\ m_{ij}^{l+1} &= m_{i, j+I(p_i, t_k)}, \quad j = 2, \dots, d+1. \end{aligned} \tag{1}$$

- для каждой из выходных позиций  $p_i \in P$ , для которых  $O(t_k, p_i) > 0$

$$\begin{aligned} m_{ij}^{l+1} &= \min\{m_{ij}, 1 - q_k\}, \quad j = 1, 2, \dots, O(t_k, p_i), \\ m_{ij}^{l+1} &= \max\{\min\{m_{ij}, 1 - q_k\}, \min\{m_{i, j-O(t_k, p_i)}, q_k\}, \\ &\quad (\forall j \in J) \wedge (j > O(t_k)), p_i\}. \end{aligned} \tag{2}$$

# Нечеткие сети Петри

где  $q_k$  – степень принадлежности, характеризующая возможность нечеткого срабатывания перехода  $t_k$ , рассчитываемая следующим образом:

$$q_k = \min_{(i \in \{1, 2, \dots, n\})} \{ \max_{(j \in J) \wedge (j > I(p_i, t_k))} \{m_{ij}\} \}, (\forall t_k \in T).$$

Если некоторые из позиций  $P_i$  являются одновременно входными и выходными для разрешенного перехода  $t_k$ , то для них координаты вектора новой маркировки рассчитываются последовательно, сначала по формуле (1), а затем по формуле (2).