Севастопольский государственный университет
Институт информационных технологий

Методы и системы искусственного интеллекта

Бондарев Владимир Николаевич

Семантические сети

Семантическая сеть представляет направленный граф, *вершины* которого соответствуют *объектам* (понятиям, сущностям) предметной области, а *дуги* – *отношениям* (связям) между ними. И узлы, и дуги, как правило, имеют метки (имена). Имена вершин и дуг обычно совпадают с именами соответствующих объектов и отношений предметной области.

Объекты предметной области можно условно разделить на три группы: обобщенные, индивидные (конкретные) и агрегатные объекты.

Обобщенный объект соответствует некоторой собирательной абстракции реально существующих объектов, процессов или явлений. Например, “изделие”, “предприятие”, “сотрудник” и т.д. Обобщенные объекты фактически представляют определенные классы предметной области.

Семантические сети

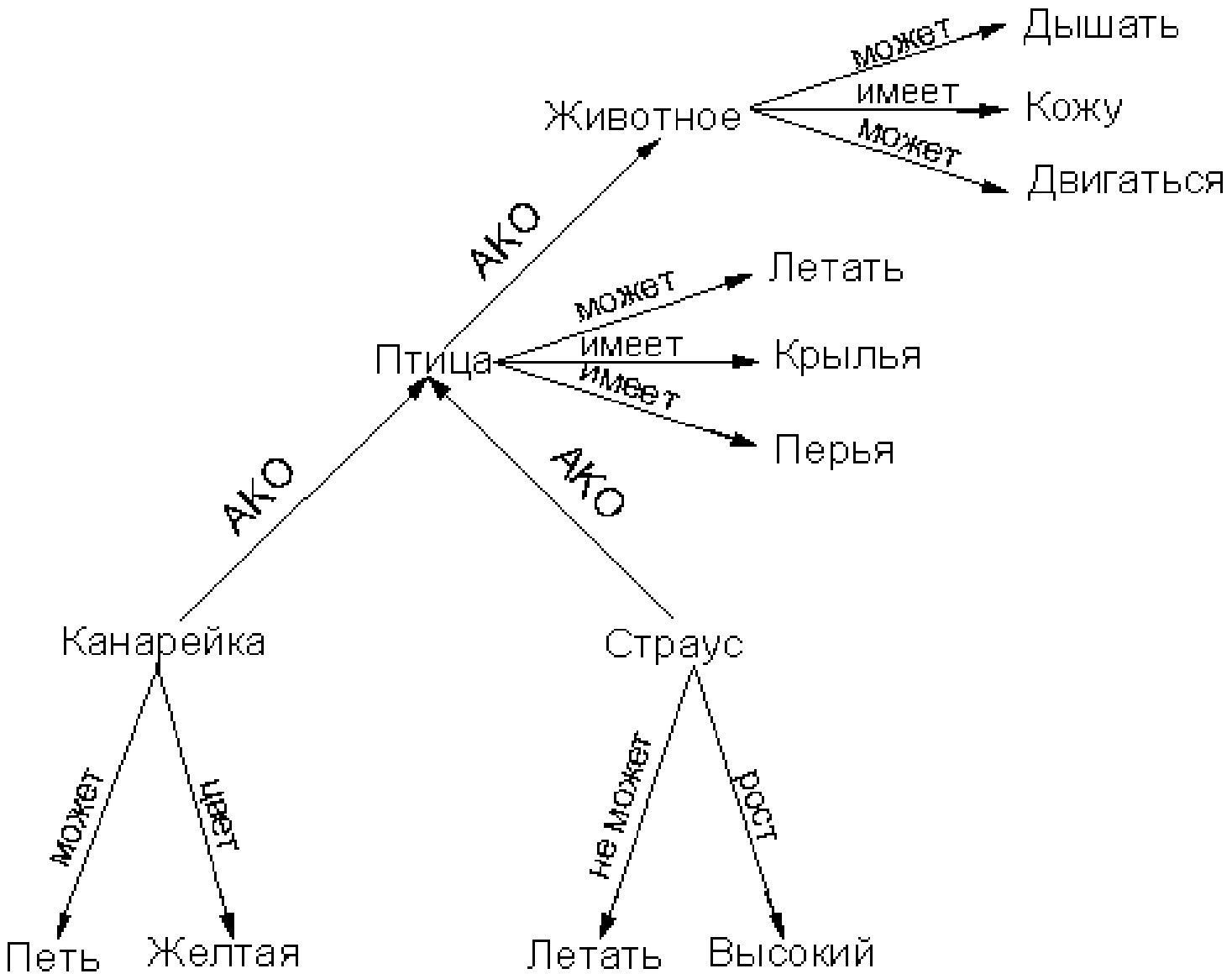
Индивидный объект – это каким-то образом выделенный единичный представитель (экземпляр) класса. Например, “сотрудник Петров И.Н.”

Агрегатным называется составной объект, образованный из других объектов, которые рассматриваются как его составные части.

Типы связей между объектами семантических сетей могут быть любыми. Но чаще всего применяются следующие основные связи (**отношения**): “род-вид”, “является представителем”, “является частью”.

Для формализации связи **“род-вид”** будем использовать отношение **ako** (от англ. **a-kind-of** – разновидность), для связи **“является представителем”** — отношение **is_a** (от англ. **is a member of the class** – быть представителем класса), для связи **“является частью”** — **part_of** или **has.**

Семантическая сеть Куиллиана



В.Бондарев

Семантические сети

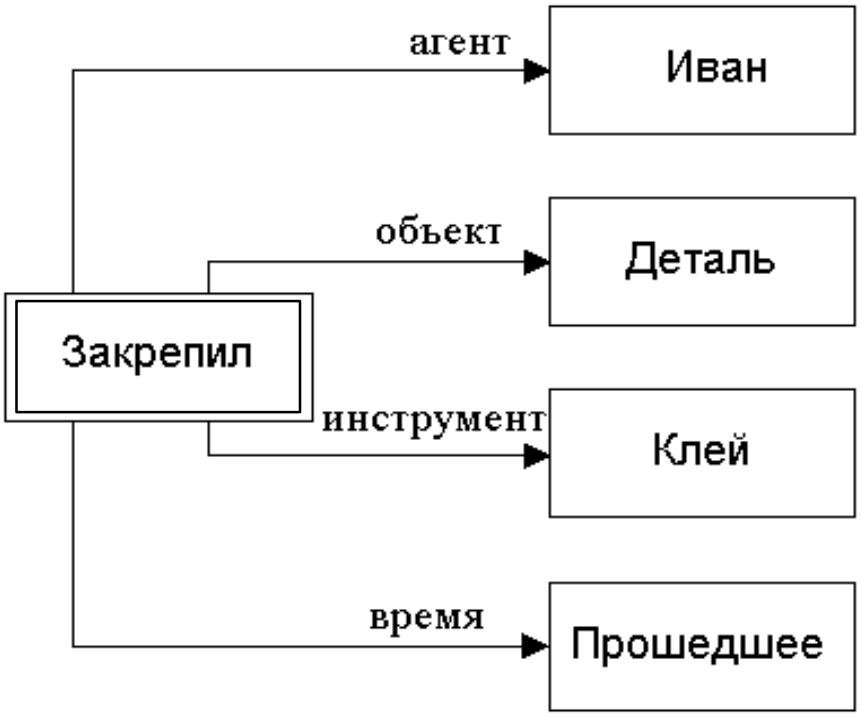
Предложение можно представить вершиной-глаголом и различными падежными связками (отношениями). Такую структуру называют **падежной рамкой.**

Среди **падежных отношений** выделяют следующие: ***агент*** – отношение между событием и тем кто (что) его совершает; ***объект*** – отношение между событием и тем, над чем выполняется действие; ***инструмент*** – объект, с помощью которого совершается действие; ***место*** – место совершения события; ***время*** – время совершения события.

Падежная рамка фиксирует знания падежной (лингвистической) структуры естественных языков в виде сетевого формализма. Рассмотрим пример: "Иван закрепил деталь клеем"

Семантические сети

Падежная рамка



Семантические сети

В большинстве случаев **многообразие объектов** сети можно подразделить на три группы :

1) **объекты-понятия** – сведения о физических и абстрактных объектах, предметной области;
2) **объекты-события** – абстрактные или конкретные действия, которые могут привести к изменению состояния предметной области;
3) **объекты-свойства** – уточняют понятия и события, например, указывают характеристики понятий (цвет, форму, размеры и т.п.), фиксируют параметры событий (место, время, продолжительность).

Семантические сети

Многообразие отношений:

1) **лингвистические отношения**: падежные (агент, объект, инструмент, время, место); глагольные (наклонение, время, вид, число, залог); атрибутивные (цвет, размер, форма и т.п.);

2) **логические отношения** (конъюнкция, дизъюнкция, отрицание, импликация);

3) **теоретико-множественные отношения**:“род-вид”, “класс-подкласс”, “целое-часть”, “элемент множества” и др.

4) **квантифицированные отношения**: кванторы общности и существования, нечеткие кванторы (много, несколько, часто т.д.).

Семантические сети

Рассмотренные ранее отношения (“род-вид”, “быть представителем”, “быть частью”, падежные отношения) далеко не исчерпывают всего набора отношений, применяемых в семантических сетях. Но они образуют хорошую основу для построения прикладных баз знаний.

Особое место при этом занимают теоретико-множественные отношения, обладающие транзитивными свойствами. Отношение R называется *транзитивным*, если для любых объектов α, β, γ таких, что α находится в отношении R с β ($\alpha R \beta$) и β– в отношении R с γ ($\beta R \gamma$), следует справедливость утверждения “α находится в отношении R с γ ”, т.е.

$$(\alpha R \beta \wedge \beta R \gamma) \Rightarrow \alpha R \gamma.$$

Транзитивность обеспечивает возможность наследования свойств. Механизм наследования в обеспечивает проведение дедуктивных заключений типа: “Все люди смертны. Сократ – человек. Следовательно, Сократ смертен”

Семантические сети

В базах знаний выделяют **интенсиональные и экстенсиональные** знания.

Если имеется конечное множество атрибутов $A=\{A_1,A_2,\ldots,A_n\}$ и конечное множество отношений $R=\{R_1,R_2,\ldots,R_m\}$, то *схемой* или *интенсионалом* отношения $R_i(i=1,2,\ldots,m)$ называется набор пар вида:

$$INT(R_i)=\{\ldots,[A_j,DOM(A_j)],\ldots\}$$

где $DOM(A_j)$ – домен атрибута A_j, т.е. множество возможных значений атрибута.

Семантические сети

Экстенсионал отношения R_i – это множество фактов

$$EXT(R_i) = \{F_1, F_2, ..., F_p\},$$

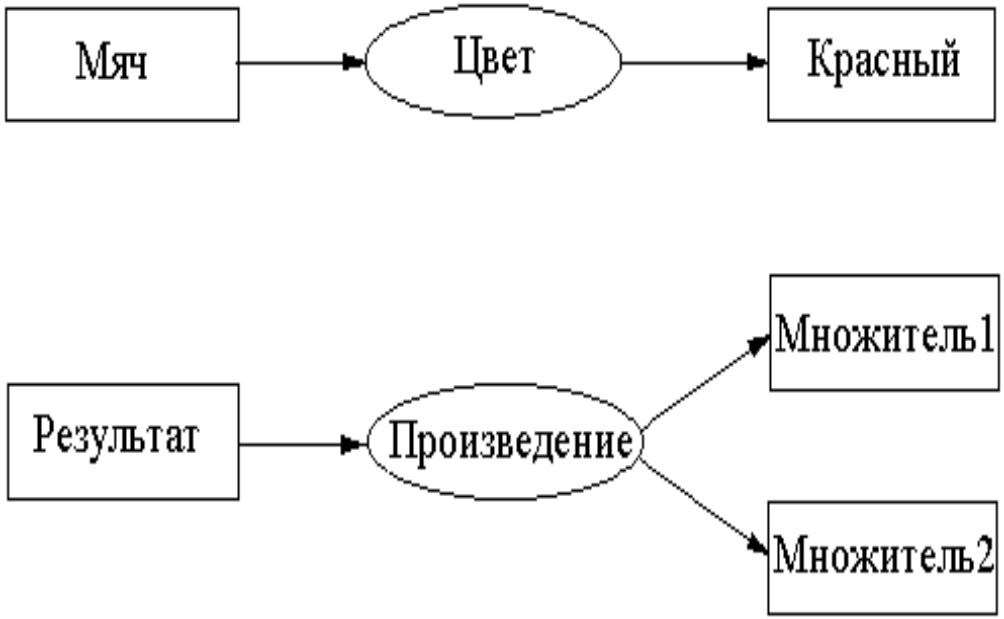
где $F_1, F_2, \ldots, F_p$ – факты отношения R_i, обычно задаваемые в виде совокупности пар “атрибут-значение”. Факт конкретизирует определенное отношение и представляет собой подграф, имеющий звездообразную структуру. Корнем подграфа является вершина предикатного типа с меткой, соответствующей имени отношения. Ребра подграфа отмечены именами атрибутов.

Интенсиональная семантическая сеть представляет моделируемую предметную область на обобщенном, концептуальном уровне, а экстенсиональная – наполняет ее конкретными, фактическими данными. Т.о., семантическую базу знаний можно рассматривать как совокупность объектов и отношений, часть из которых определена интенсионально, а часть – экстенсионально.

Способы описания семантических сетей

Наиболее часто для этих целей используют **концептуальные графы** Дж. Соува и **блочные структуры** Г.Хендрикса.

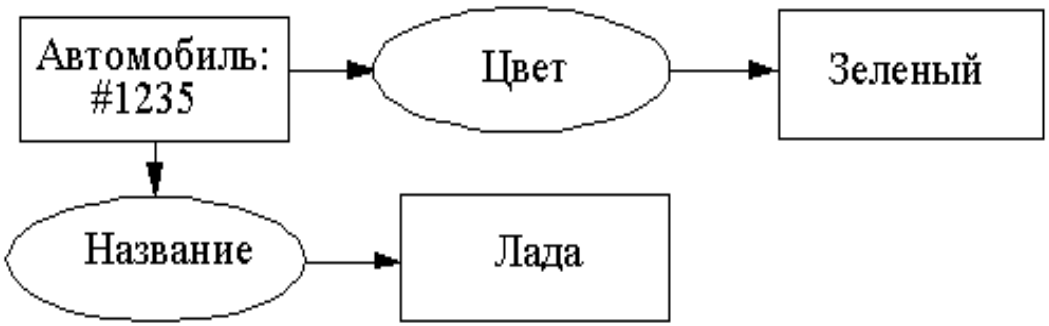
Вершинами **концептуального графа** являются либо объекты (понятия, сущности) предметной области, либо концептуальные отношения. Ребра концептуального графа связывают между собой вершины-понятия и вершины-отношения.



Способы описания семантических сетей

Обычно каждый концептуальный граф фиксирует одно предложение. Тогда **база знаний** будет представляться в виде совокупности таких графов.

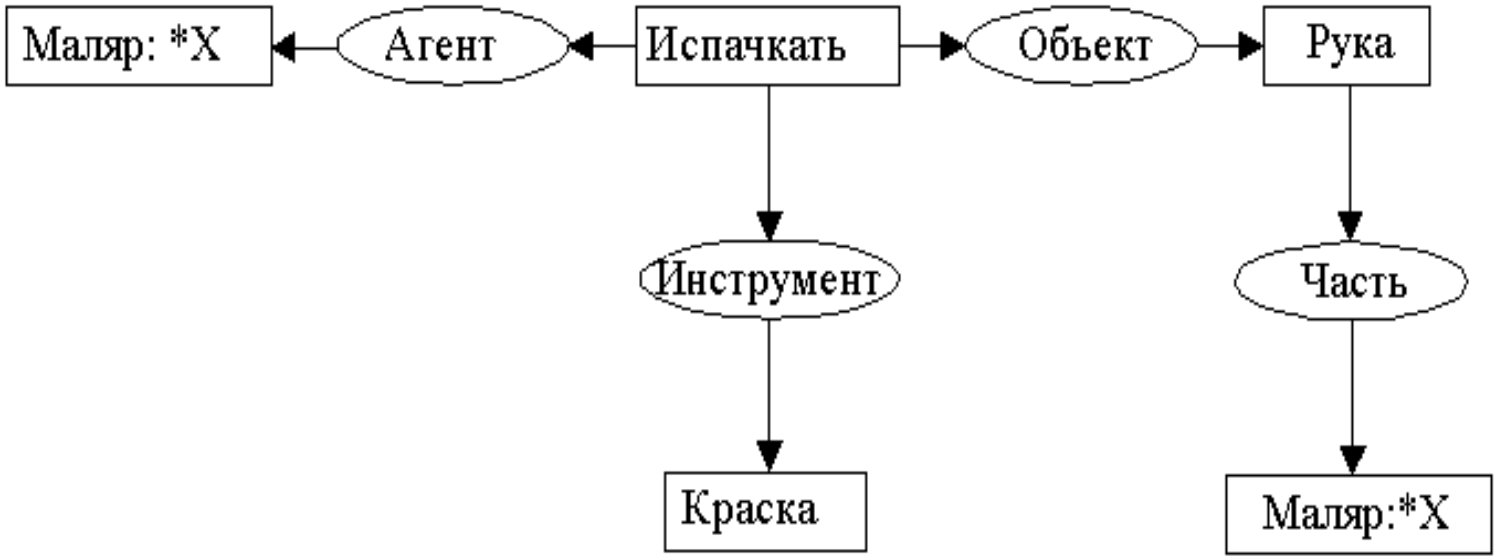
Вершина-понятие концептуального графа может иметь *метку типа*. Тип обозначает класс принадлежности вершины. Метку типа вершины отделяют двоеточием от конкретного имени вершины. На концептуальных графах можно также вводить **индивидные вершины-понятия**. Чтобы различать такие вершины используется маркер **#число**.



Способы описания семантических сетей

На концептуальных графах могут применяться **обобщенные маркеры**, обозначаемые знаком *. В сочетании с переменной, записываемой после этого знака, обобщенный маркер оказывается полезным в ситуациях, когда две различные вершины графа представляют один и тот же объект.

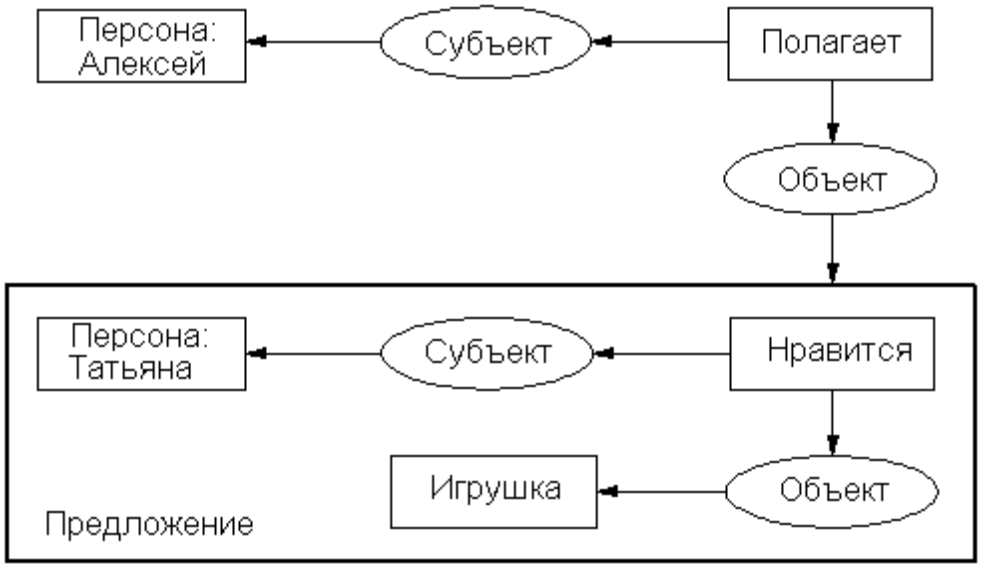
"Маляр испачкал свою руку краской".



Способы описания семантических сетей

Концептуальные графы могут содержать вершины-предложениями. Такие вершины изображаются в виде прямоугольного блока, содержащего подграф, соответствующий предложению. Например:

“Алексей полагает, что Татьяне нравится игрушка”.



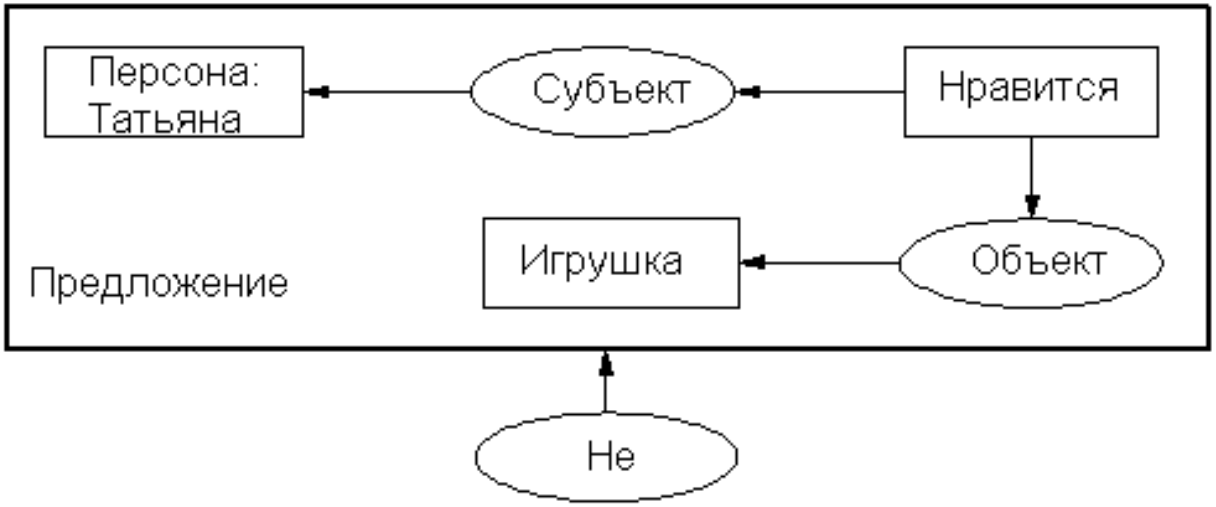
Введение отдельных блоков для обозначения предложений позволяет фиксировать отношения между предложениями. В приведенном примере глагол “полагает” выражает ощущения Алексея. Утверждения, содержащие подобные лексические единицы (доверять, верить, намериваться и т.п.) являются объектом исследований ***модальной*** **логики.**

Способы описания семантических сетей

Концептуальный граф позволяет естественным образом выражать конъюнкцию или дизъюнкцию. Сложнее с отрицанием и кванторами.

Для представления отрицания на концептуальных графах применяется унарный логический оператор “не”.

“Нет игрушек, которые нравятся Татьяне”.



Способы описания семантических сетей

Обобщенные объекты-понятия, отображаемые на концептуальных графах, связаны квантором существования. Например граф, изображенный ранее, описать формулой

$$\exists x \exists y (мяч(x) \wedge цвет(x, y) \wedge красный(y)).$$

Квантор общности получается при отрицании предложений, связанных квантором существования. Например:

$$\forall x \forall y \neg (мяч(x) \wedge цвет(x, y) \wedge красный(y)).$$

По своим выразительным возможностям концептуальные графы эквивалентны логике исчисления предикатов. Существует возможность взаимного перехода от описаний предметной области на языке исчисления предикатов к концептуальным графам, и наоборот. Выбор тех или иных средств определяется характером решаемых задач .

Методы вывода на семантических сетях

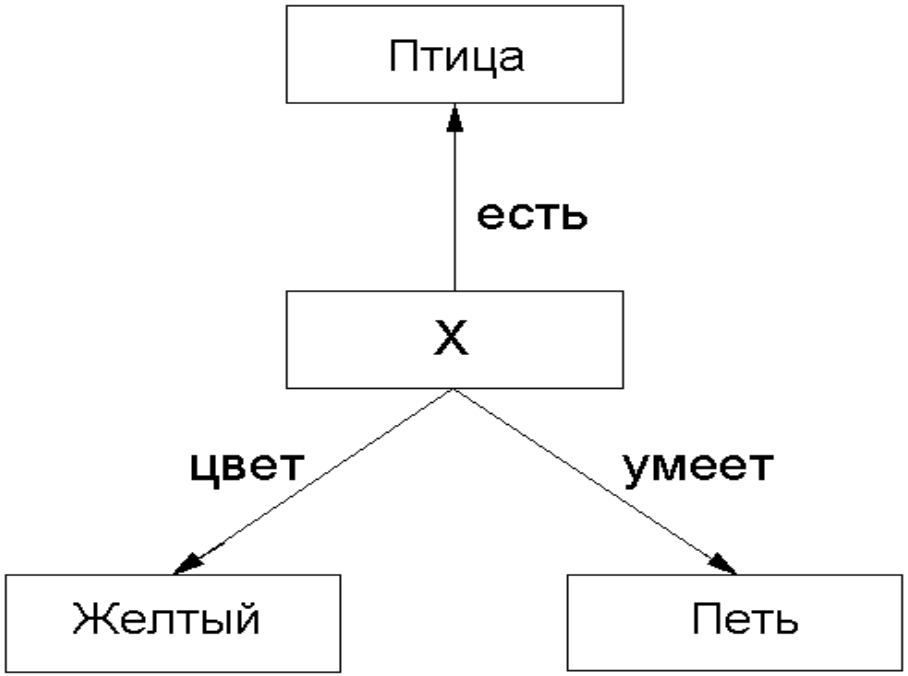
Методы вывода на семантических сетях используют ассоциативные и сопоставляющие алгоритмы, которые сводятся к нахождению путей на графе, построению транзитивных замыканий, выделению подграфов с определенными свойствами.

Одним из механизмов вывода является **сопоставление с образцом.** В данном случае происходит сопоставление отдельных фрагментов семантической сети. При этом запрос к базе знаний представляется в виде автономного подграфа, который строится по тем же правилам, что и семантическая сеть. Поиск ответа на запрос реализуется сопоставлением подграфа запроса с фрагментами семантической сети. Для этого осуществляют наложение подграфа запроса на соответствующий фрагмент сети.

Методы вывода на семантических сетях

“Существует ли такая птица X, которая умеет петь и имеет желтый цвет?”

Подграф запроса



Изоморфное вложение подграфа запроса в семантическую сеть, изображенную выше, позволяет дать ответ **X=“канарейка”.**

Фреймы.

Структура фрейма

Фреймы впервые были предложены для представления знаний в 1975г. М. Минским. Согласно его определению ***фреймы*** – это структуры данных, предназначенные для представления стереотипных ситуаций.

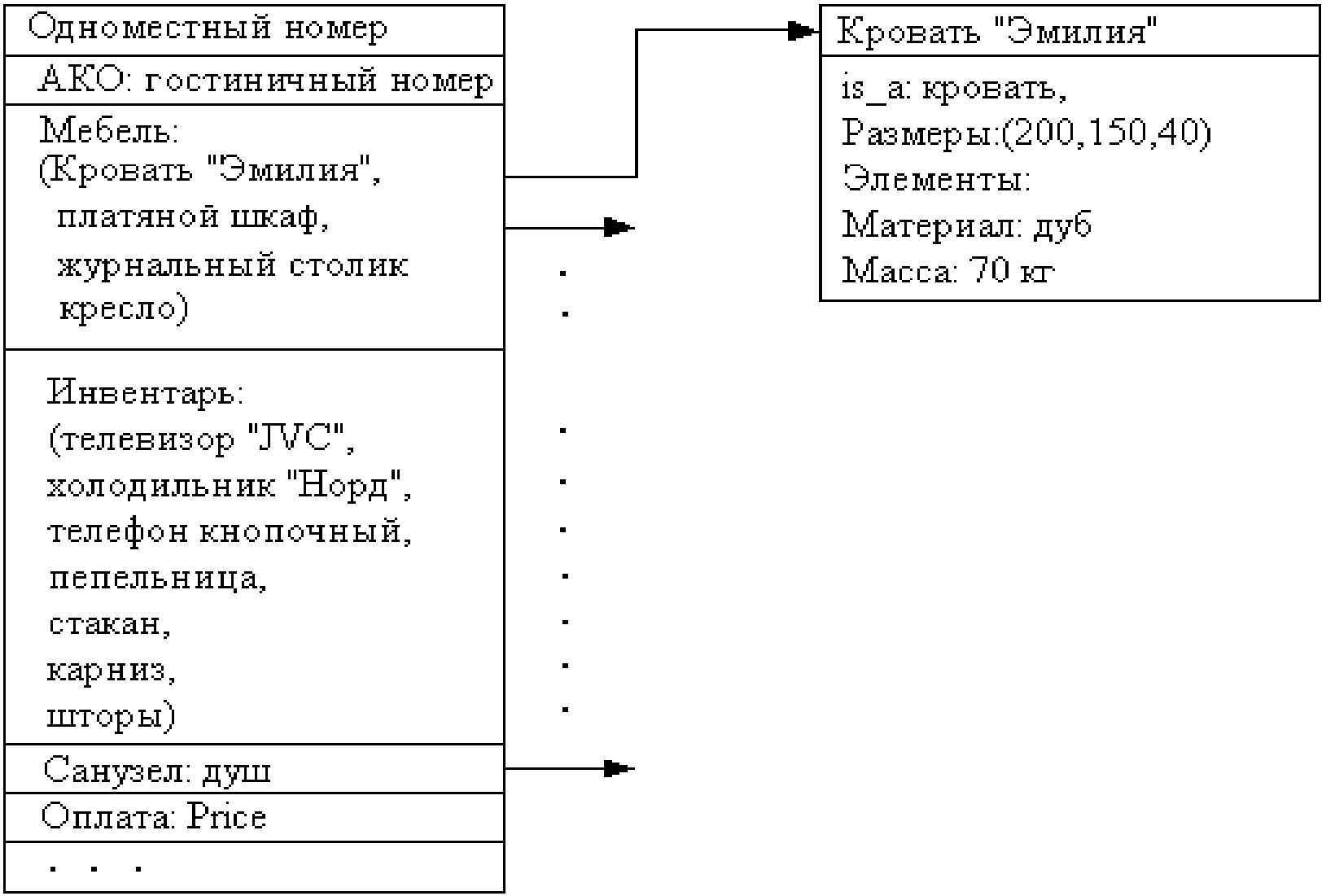
Фрейм состоит из слотов. ***Слоты*** – это некоторые незаполненные подструктуры фрейма. В простейшем случае под фреймом понимают следующую структуру

$$\{n,(ns_1,vs_1,ps_1),(ns_2,vs_2,ps_2),\ldots,(ns_k,vs_k,ps_k)\},$$

где n – имя фрейма, ns_1 – имя слота, vs_1 – значение слота, ps_1 – имя присоединенной процедуры.

После заполнения слотов конкретными данными, фрейм будет представлять ту или иную ситуацию, явление или объект предметной области. В качестве значений слотов могут выступать имена других фреймов, что обеспечивает построение сети фреймов.

Фрейм - гостиничный номер



Структура фрейма

Во фреймовых системах часть фреймов представляет индивидные объекты предметной области. Такие фреймы называют ***экземплярами фреймов*** или ***фреймами-примерами.***

Другие фреймы, представляющие обобщенные объекты, называют ***классами*** или ***фреймами-прототипами.***

Фрейм может содержать **специальный слот**, с помощью которого задается отношение данного фрейма с другими фреймам сети. Специальный слот, называемый **is_a** слотом, фиксирует отношение “**экземпляр класса**”. Слот **ako** указывает, что данный фрейм является прямым ***подклассом*** вышестоящего по иерархии фрейма. Вышестоящий фрейм называют родительским фреймом или ***суперклассом.***

Структура фрейма

Важной характеристикой фреймов является возможность включения в слоты **присоединенных процедур.** Выделяют два вида процедур: процедуры-демоны и процедуры-слуги.

Процедуры-демоны активизируются автоматически при каждой попытке добавления или удаления данных из слота. Чаще всего используются три типа процедур-демонов: “**если-добавлено**”, “**если-удалено**” и “**если-нужно**”. Эти процедуры активизируются при выполнении соответствующих условий.

Процедуры-слуги активизируются только по запросу при обращении к слоту. Например, при обращении к слоту “оплата” будет активизирована процедура “Price”, которая определит стоимость проживания в номере за сутки.

Структура фрейма

Фреймовые системы представляют собой иерархически организованные структуры, реализующие **принцип наследования информации.** В ряде фреймовых систем в состав слотов включают различные **указатели наследования.**

Рассмотрим в качестве примера структуру фрейма системы **FMS**. Здесь выделяют следующие типы указателей наследования: **unique, same, range, override и др.**

Unique (уникальный) означает, что слоты с одними и теми же именами во фреймах, находящихся на разных уровнях иерархии, могут иметь различные значения.

Same (такой же) показывает, что слоты должны иметь одинаковые значения.

Range (диапазон) устанавливает некоторые границы, в которых может находится значение слота фрейма нижнего уровня.

Override комбинирует свойства указателей **unique** и **same**.

Структура фрейма

Типы данных слотов соответствуют обычным типам данных, применяемых в языках программирования. Однако применяются также и другие типы данных, повышающие эффективность работы с системой. Например, TEXT (текстовый), TABLE (табличный), LIST (списочный), LISP (присоединенная процедура), FRAME (указатель на другой фрейм) и др. Т.о., фрейм в системе **FMS**:

<фрейм>::=<имя фрейма> <ссылка на суперкласс> <слот> {<слот>}
<слот>::=<имя> <указатель наследования> <тип данных>
<значение> [<процедуры-демоны>]
<указатель наследования>::= unique| same| range| override

<тип данных>::= INTEGER| REAL| BOOL| STRING| TEXT| TABLE|
LISP| LIST| FRAME
<процедуры-демоны>::=<если-добавлено>| <если-удалено>|
<если-нужно>|

Структура фрейма

Другим примером фреймовой системы является язык **FRL** (**F**rame **R**epresentation **L**anguage), представляющий расширение языка Лисп. Слот состоит из нескольких “фацет” (англ. facet – аспект, грань, сторона), характеризуемых значением. Например, “***Лошадь серого цвета***” можно представить на языке FRL в виде:

frame *Лошадь*, slot *цвет*, facet *$value*, value *серый*.

В состав слота входит шесть фацет:

$value – текущее значение слота;

$require – допустимые значения фацета **value**;

$default – значение по умолчанию для фацета **value**;

$if-added – процедура-демон “**если-добавлено**”;

$if-removed – процедура-демон “**если-удалено**”;

$if-needed – процедура-демон “**если-нужно**”.

Каждый фрейм, создаваемый на языке **FRL**, содержит предварительно определенный слот **ako**, позволяющий наследовать значения слотов в соответствии с иерархией фреймов.

Управление выводом

Во фреймовых системах используется **три способа управления выводом**: с помощью механизма наследования; с помощью процедур-демонов; с помощью присоединенных процедур.

Механизм наследования является основным встроенным средством вывода, которым оснащаются фреймовые системы. В общем случае порядок наследования определяется с помощью ***списка предшествований***. В случае линейной иерархической схемы этот список формируется в соответствии с направлением **is_a** и **ako** связей.

Более сложная ситуация возникает, если фрейм имеет несколько is_a или ako связей. В этом случае говорят **о множественном наследовании.**

Управление выводом

Иная возможность вывода во фреймовых системах основана на использовании процедур-демонов: “**если-нужно**”, “**если-добавлено**”, “**если-удалено**”. Процедура “**если-нужно**” вызывается, когда поступает запрос, требующий установления значений соответствующего слота. Процедуры “**если-добавлено**” и “**если-удалено**” активизируются при записи и удалении значений из слота.

Третья возможность вывода во фреймовых системах основана на применении **присоединенных (служебных) процедур**. Имя присоединенной процедуры выступает в качестве значения слота. Присоединенная процедура запускается по сообщению, переданному из другого фрейма. Для этого может использоваться специальная функция **MSG**, формат вызова которой имеет вид:

MSG(<имя фрейма>,<имя слота>,<параметры>).

Механизм наследования

Простейшая иерархия (фрейм имеет только один суперкласс).

Рисунок 1

Список предшествований

Каждый подкласс или экземпляр класса наследует слоты своего суперкласса. Если подкласс (экземпляр класса) и суперкласс имеют слоты, с совпадающими именами, то определения значений слотов, сделанные внутри подкласса (экземпляра класса), перекрывают определения суперкласса.

В общем случае порядок наследования определяется с помощью ***списка предшествований.*** В случае линейной иерархической схемы этот список формируется в соответствии с направлением **is_a** и **ako** связей. Для рассматриваемого случая его можно записать в виде:

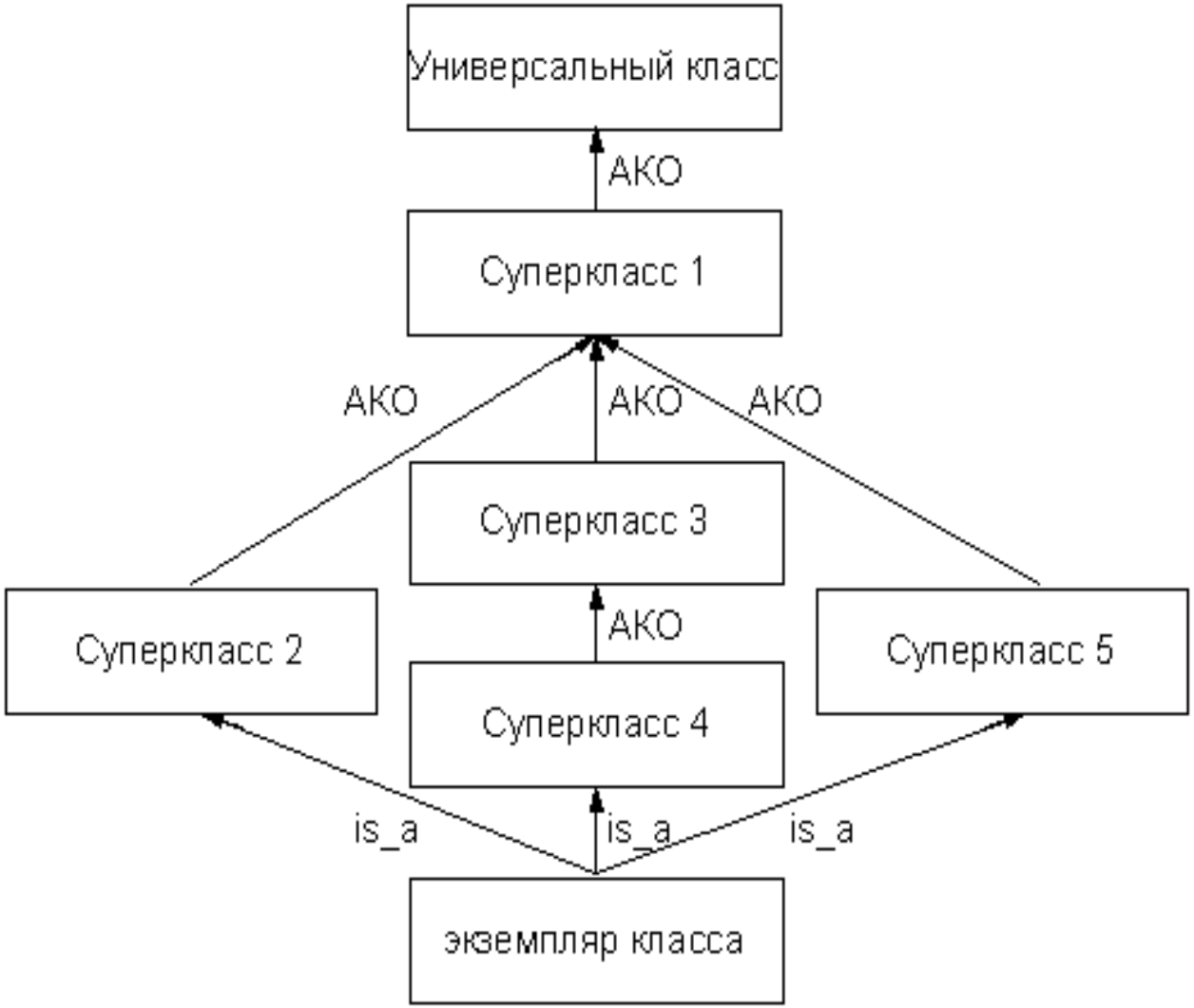
(“пингвин Чилли-Вилли”, “пингвины”, “птицы”).

Множественное наследование

Просмотр иерархии фреймов выполняется сначала в глубину, а затем слева – направо. Список предшествований будет иметь вид: **(“пингвин Чилли-Вилли”, “пингвины”, “птицы”, “персонаж мультфильма”).**

Множественное наследование

В реальных ситуациях взаимосвязь фреймов оказывается значительно сложнее. Например



Множественное наследование

Наряду с поиском в глубину и слева направо, применяется **принцип исключения** из списка предшествований повторяющихся элементов. В списке всегда остается только та повторяющаяся вершина, которая встретилась последней. Например, для рисунка список предшествований можно представить в виде:

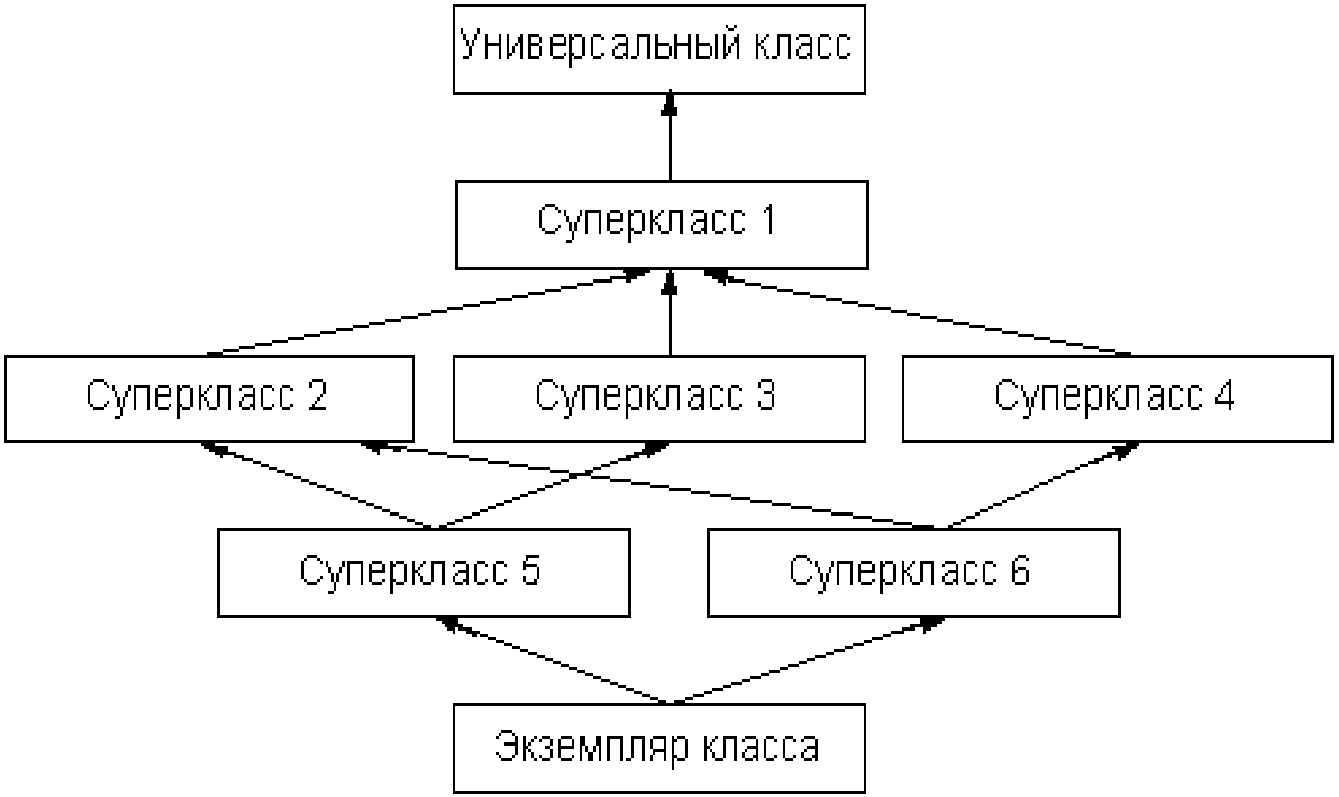
(экземпляр, суперкласс 2, суперкласс 4, суперкласс 3, суперкласс 5, суперкласс 1, универсальный класс).

Данный принцип позволяет реализовать требование, заключающееся в том, что *любой класс должен появляться в списке предшествований раньше, чем его непосредственный суперкласс.* Реализация этого требования обеспечивает передачу по умолчанию наиболее специфичных данных, что соответствует нашему интуитивному представлению о наследовании свойств.

Множественное наследование

Применение рассмотренной процедуры наследования к иерархии фреймов на рисунке приведет к проблеме:

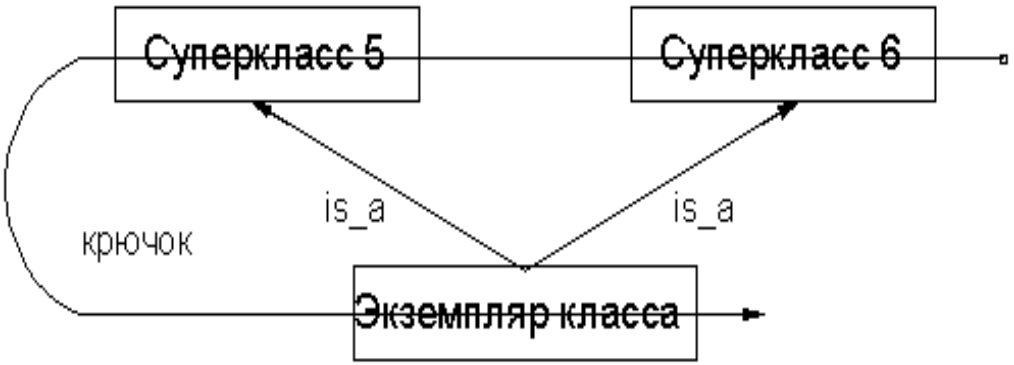
(экземпляр, суперкласс 5, суперкласс 3, суперкласс 6, суперкласс 2, суперкласс 4, суперкласс 1).

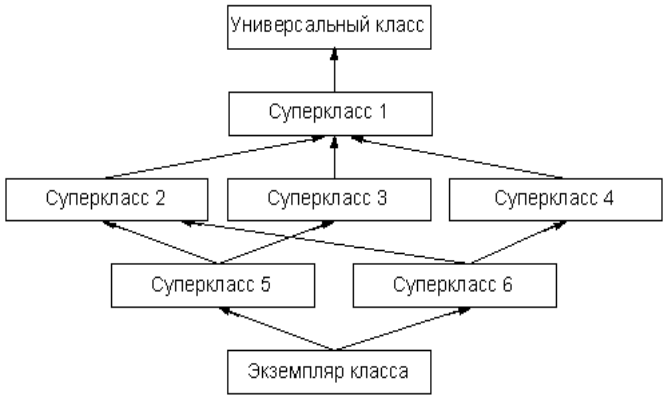


Топологическая сортировка

На первом шаге составляется список экземпляров и суперклассов, которые (для рассматриваемого фрейма) достижимы посредством **is_a** и **ako** связей. Для рассматриваемого примера получим: **(экземпляр, суперкласс 5, суперкласс 2, суперкласс 1, суперкласс 3, суперкласс 6, суперкласс 4, унив. класс).**

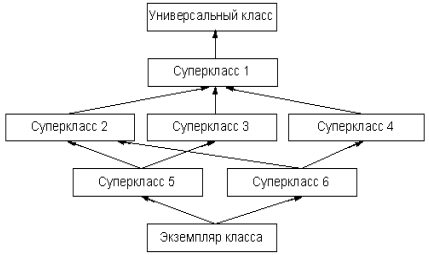
На втором шаге для каждого элемента полученного списка формируется список пар в соответствии с принципом, **принципом “рыболовецкого крючка “**





Топологическая сортировка

Следуя указанному принципу, для каждого элемента списка получим список соответствующих пар



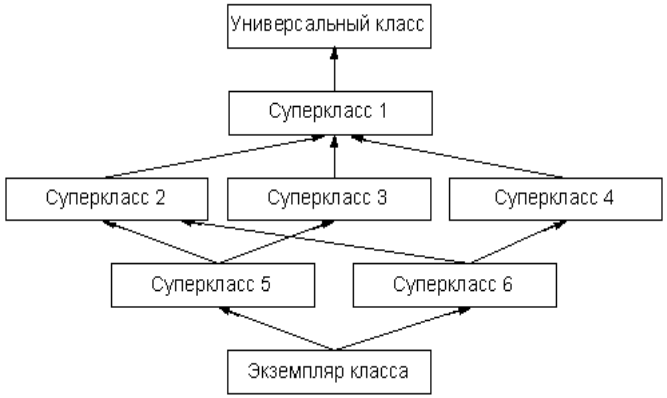
Элемент	Пары
экземпляр	**экземпляр** – суперкласс 5, суперкласс 5 – суперкласс 6
суперкласс 5	суперкласс 5 – суперкласс 2, суперкласс 2 – суперкласс 3
суперкласс 2	суперкласс 2 – суперкласс 1
суперкласс 1	суперкласс 1 – универсальный класс
суперкласс 3	суперкласс 3 – суперкласс 1
суперкласс 6	суперкласс 6 – суперкласс 2, суперкласс 2 – суперкласс 4
суперкласс 4	суперкласс 4 – суперкласс 1
универсальный класс	универсальный класс

На третьем шаге просматривают элементы таблицы и выделяют **текущий элемент**, который встречается только с левой стороны пары и никогда не встречается с правой стороны. Таким элементом является “**экземпляр**”.

Топологическая сортировка

Текущий элемент добавляется в конец списка предшествований, и все пары, в которых он встречается, вычеркиваются. В рассматриваемом случае это пара “экземпляр–суперкласс 5”.
Затем действия повторяются. Текущим элементом становится суперкласс 5, и вычеркиваются пары “суперкласс 5 – суперкласс 6” и “суперкласс 5 – суперкласс 2” и т.д. В итоге формируется следующий список предшествований: **(экземпляр, суперкласс 5, суперкласс 6, суперкласс 2, суперкласс 4, суперкласс 3, суперкласс 1, унив. класс).**

Если на роль текущего элемента претендуют сразу несколько суперклассов, то **конфликт разрешается** в пользу того суперкласса, который является непосредственным суперклассом для самого правого суперкласса, уже находящегося в списке предшествований.



Характер вывода

В рассматриваемом примере конфликт возникает между суперклассом 3 и суперклассом 4 после того, как в список предшествований будет помещен суперкласс 2. Предпочтение отдается суперклассу 4, так как он является суперклассом самого правого суперкласса, уже находящегося в списке предшествований, т.е. суперкласса 6.

Вывод значений слотов на основе наследования является, по своей сути, **немонотонным**, так как перекрытие значений в процессе наследования может менять истинность ранее установленных фактов.

Например, из рисунка 1 следует, что пингвин Чилли-Вилли покрыт перьями. Однако если мы добавим слот “**покров**” со значением ”***пух***” во фрейм “**пингвины**”, то исходное утверждение уже будет неправильным.

Характер вывода

Также немонотонный характер вывода обеспечивается процедурами-демонами. Например, процедура “**если-нужно**” вызывается, когда поступает запрос, требующий установления значений соответствующего слота. При этом ***процедура не меняет значение слота, а каждый раз вычисляет его заново, учитывая определенные условия.***

В частности, для слота “**рост**” экземпляра фрейма “**пингвин**” (рисунок1) процедура может устанавливать рост пингвина в зависимости от значения слота “**вид**”. Например, если значением слота “**вид**” является “***императорский пингвин***”, то вернуть значение **100 – 120см**, иначе вернуть значение **40 – 50см**.

Характер вывода

Процедуры “**если-добавлено**” и “**если-удалено**” активизируются при записи и удалении значений из слота. Например, процедура “**если-добавлено**” может активизироваться, когда слот “**вид**” получит значение (рисунок 1). Если новое значение – “***галапагосский пингвин***”, то записать в слот “**красная книга**” значение “***да***”. Очевидно, что данная процедура устанавливает **ограничивающие отношения** между значениями слотов “**вид**” и “**красная книга**”.

Можно сказать, что процедуры “**если-добавлено**” и “**если-удалено**” устанавливают зависимость значений одного слота от значений другого и играют роль **хранителя отношений ограничения** во фреймовых системах.