Федеральное государственное автономное образовательное учреждение высшего образования
Санкт-Петербургский национальный исследовательский университет
информационных технологий, механики и оптики

ИНТЕЛЛЕКТУАЛЬНЫЕ СИСТЕМЫ И ТЕХНОЛОГИИ

ЛЕКЦИЯ 3. СИСТЕМЫ, ОСНОВАННЫЕ НА ПРАВИЛАХ

к.т.н., Кашевник Алексей Михайлович,
alexey@iias.spb.su
к.т.н., Пономарев Андрей Васильевич
ponomarev@iias.spb.su

ИНТЕЛЛЕКТУАЛЬНЫЕ СИСТЕМЫ

Интеллектуальные технологии (методы) – это попытки заставить машину осуществлять интеллектуальную деятельность, свойственную человеку.

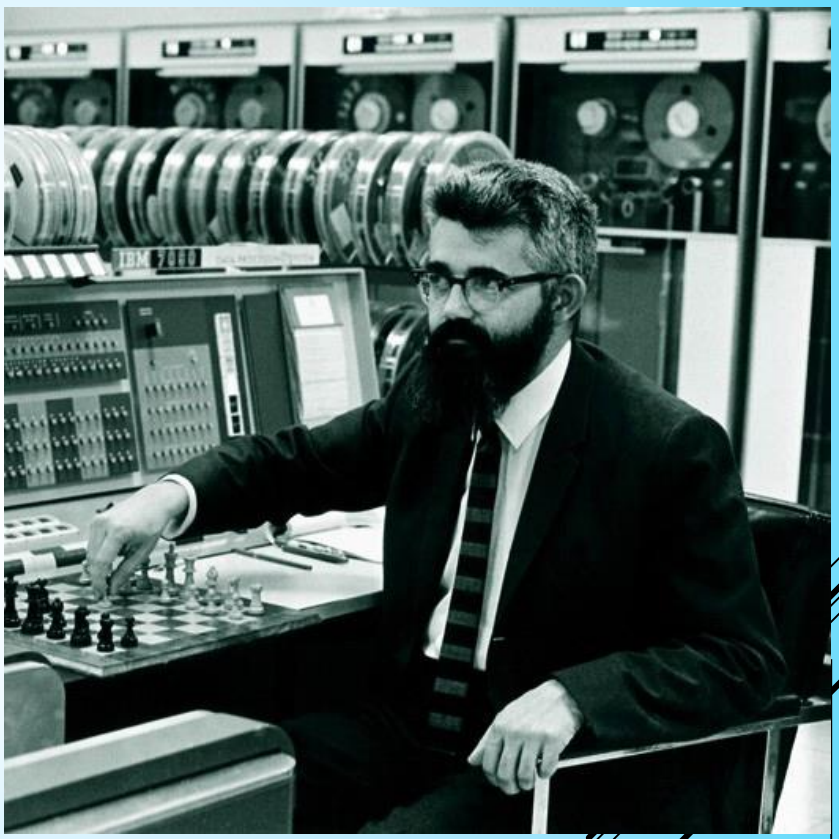
А как человек осуществляет такую деятельность?

- Логические модели
- Фреймовые модели
- Семантические модели
- Нейронные сети
- И др.

ЛОГИКА КАК СРЕДСТВО ПОСТРОЕНИЯ ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМ

Важнейшие вехи:

- Дж. Маккарти (1956) предложил использовать логику первого порядка для создания систем искусственного интеллекта
- А. Робинсон (1965) разработал метод резолюции – полную процедуру вывода для логики первого порядка



АГЕНТЫ, ОСНОВАННЫЕ НА ЗНАНИЯХ

База знаний – множество высказываний на языке представления знаний.

Интерфейс:

Tell – пополнение базы знаний

Ask – запрос к базе знаний

В *логических агентах* ответ на запрос, переданный с помощью Ask, *должен следовать* из того, что было сообщено базе знаний посредством Tell.

СХЕМА АГЕНТА, ОСНОВАННОГО НА ЗНАНИЯХ

```
function KB-Agent(percept) returns действие action
  static: KB, база знаний
          t, счетчик, обозначающий время,
             первоначально равный 0
    Tell(KB, Make-Percept-Sentence(percept, t))
    action <— Ask(KB, Make-Action-Query(t))
    Tell(KB, Make-Action-Sentence(action, t))
    t <- t + 1
    return action
```

ТЕРМИНОЛОГИЯ

$\alpha \vDash \beta$ – “высказывание α влечет за собой высказывание β”

Тогда и только тогда, когда в любой модели, в которой высказывание α является истинным, β также истинно.

$KB \vdash_i \alpha$ - “высказывание α получено путем логического вывода из базы знаний КВ с помощью алгоритма I” или "алгоритм i позволяет вывести логическим путем высказывание α из базы знаний КВ".

ВАЖНЫЕ СВОЙСТВА АЛГОРИТМА ВЫВОДА

Алгоритм логического вывода, позволяющий получить только такие высказывания, которые действительно следуют из базы знаний, называется **непротиворечивым**, или **сохраняющим истинность**.

Алгоритм логического вывода называется **полным**, если позволяет вывести любое высказывание, которое следует из базы знаний.

НЕОБХОДИМОЕ ДОПУЩЕНИЕ

Если база знаний является истинной в реальном мире, то любое высказывание α, полученное логическим путем из этой базы знаний с помощью непротиворечивой процедуры логического вывода, является также истинным в реальном мире.

ЛОГИКА ПЕРВОГО ПОРЯДКА

a.k.a. **исчисление предикатов первого порядка, FOL – First Order Logic, FOPC – First Order Predicate Calculus**

Структура:

- Объекты
  - John, Mary, Jacob, Jane, …
- Отношения (множество кортежей)
  - Friend = {<John, Jacob>, <Mary, Jane>}
- Функции
  - Mother = {<John, Mary>} или Mother(John) = Mary

СИНТАКСИС

```
Sentence —> AtomicSentence
          | ( Sentence Connective Sentence )
          | Quantifier Variable, ... Sentence
          | ¬Sentence
AtomicSentence —> Predicate (Term, ...) | Term = Term
     Term —> Function (Term, ...)
          | Constant
          | Variable
```

Терм - обозначение объекта (прямое или косвенное). Терм без переменных – базовый терм.

СИНТАКСИС

(продолжение)

```
Connective –> ⇒ | ∧ | ∨ | ⇔
Quantifier –> ∃ | ∀
  Constant –> A | X | John | ...
  Variable –> a | x | s | ...
 Predicate –> Before | HasColor | Raining | ...
  Function –> Mother | LeftLeg | ...
```

Примеры синтаксически корректных высказываний:

$\forall x, y(y = Mother(x)) \Rightarrow Woman(y)$

$LeftLeg(John) = Mary$

СЕМАНТИКА

Модели представляют собой математические абстракции, каждая из которых устанавливает, является ли истинным или ложным каждое высказывание, относящееся к данной модели.

Семантика связывает высказывания с моделями, для того чтобы можно было определить истинность. Чтобы иметь возможность решить такую задачу, требуется **интерпретация**, которая определяет, на какие именно объекты, отношения и функции ссылаются те или иные константные, предикатные и функциональные символы.

Интерпретация примера:
Woman(x) – «быть женщиной»
Mother(x) – «мать объекта x»
LeftLeg(x) – «левая нога объекта x»
John – Mr. John White, садовник
Mary – Mrs. Mary White, жена John White

КВАНТОРЫ

$\forall x$ - квантор всеобщности («Для всех x»)

$\forall x King(x) \Rightarrow Person(x)$ – *«Для всех x, если x – король, то x – человек».*

Высказывание $\forall xP$ истинно в данной модели при данной интерпретации, если выражение *P* истинно при всех возможных **расширенных интерпретациях**, сформированных из данной интерпретации, где каждая расширенная интерпретация задает элемент проблемной области, на которую ссылается объект *x*.

$\exists x$ - квантор существования («Существует x, такой, что …», или «Для некоторого x...»)

Высказывание $\exists xP$ истинно в данной конкретной модели при данной конкретной интерпретации, если выражение *P* истинно по меньшей мере в одной **расширенной интерпретации**, в которой присваивается *x* одному из элементов проблемной области.

Связь между кванторами – правило де Моргана.

ИСПОЛЬЗОВАНИЕ ЛОГИКИ ПЕРВОГО ПОРЯДКА. УТВЕРЖДЕНИЯ И ЗАПРОСЫ

Высказывания вводятся в базу знаний с помощью операции Tell. Такие высказывания называются **утверждениями**. Например, можно ввести утверждения, что Джон — король и что короли — люди:

Tell (KB, $King(John)$)

Tell (KB, $\forall x King(x) \Rightarrow Person(x)$)

Мы можем задавать вопросы о содержимом базы знаний с использованием операции Ask:

Ask(KB, $King(John)$)

Вопросы, заданные с помощью операции Ask, называются **запросами**, или **целями**. На любой запрос, который логически следует из базы знаний, *должен* быть получен утвердительный ответ:
Ask(KB, $Person(John)$) --- True
Ask(KB, $\exists x Person(x)$) --- True (И???...)

ПОДСТАНОВКА (СПИСОК СВЯЗЫВАНИЯ)

Подстановка, или **список связывания** – множество пар "переменная—терм".

То есть:
Ask(KB, $\exists x Person(x)$) --- **{x/John}**

ПРИМЕР: ПРОБЛЕМНАЯ ОБЛАСТЬ РОДСТВА

Объекты: люди.

Предикаты: (унарные) Male, Female;

(бинарные) Parent, Sibling, Brother, Sister, Child, Daughter, Son, Spouse, Wife, Husband, Grandparent, Grandchild, Cousin, Aunt, Uncle.

Функции: Father, Mother.

ПРИМЕР: ПРОБЛЕМНАЯ ОБЛАСТЬ РОДСТВА

(продолжение)

Мать — это родитель женского пола:

$$\forall x, y\ Mother(x) = y \Leftrightarrow Female(y) \wedge Parent(y, x)$$

Муж — это супруг мужского пола:

$$\forall w, h\ Husband(h, w) \Leftrightarrow Male(h) \wedge Spouse(h, w)$$

Мужчины и женщины — непересекающиеся категории людей:

$$\forall x\ Male(x) \Leftrightarrow \neg Female(x)$$

Отношения между родителями и детьми являются взаимно противоположными:

$$\forall p, c\ Parent(p, c) \Leftrightarrow Child(c, p)$$

Дедушка или бабушка — это родитель родителя:

$$\forall g, c\ Grandparent(g, c) \Leftrightarrow \exists p\ Parent(g, p) \wedge Parent(p, c)$$

И т.д.

АКСИОМЫ И ТЕОРЕМЫ

Каждое из этих высказываний может рассматриваться как одна из **аксиом** в проблемной области родства. Аксиомы предоставляют основную фактическую информацию, на основании которой могут быть получены логическим путем полезные заключения.

Теоремы – высказывания, которые следуют из аксиом.

С логической точки зрения в базе знаний должны содержаться только аксиомы, но не теоремы, поскольку теоремы не увеличивают множество заключений, которые следуют из базы знаний. Но с практической точки зрения важным свойством теорем является то, что они *уменьшают вычислительные издержки* на логический вывод новых высказываний.

ИНЖЕНЕРИЯ ЗНАНИЙ

1. Идентификация задания.
2. Сбор относящихся к делу знаний.
3. Определение словаря предикатов, функций и констант.
4. Регистрация общих знаний о проблемной области. Составление аксиом для всех терминов словаря.
5. Составление описания данного конкретного экземпляра задачи.
6. Передача запросов процедуре логического вывода и получение ответов.
7. Отладка базы знаний.

ВЫВОД В ЛОГИКЕ ПЕРВОГО ПОРЯДКА

3 основных семейства алгоритмов:

1. Прямой логический вывод (дедуктивные базы знаний, продукционные системы).
2. Обратный логический вывод (системы логического программирования).
3. Системы доказательства теорем на основе резолюций.

ХОРНОВСКИЕ ВЫРАЖЕНИЯ

Хорновское выражение представляет собой дизъюнкцию литералов, среди которых положительным является не больше чем один. Например:

$\mathrm{P} \vee \neg Q \vee \neg R$ – хорновское

$\neg Q \vee \neg R$ – хорновское

$\mathrm{P} \vee Q$ – *не* хорновское

Зачем они нужны:

1) Может быть записано как импликация вида:

Определенное выражение

$$Q \vee R \Rightarrow P$$

тело

голова

2) Логический вывод может осуществляться с помощью алгоритма прямого логического вывода и обратного логического вывода.

3) Получение логических следствий может осуществляться за время, **линейно зависящее от размера БЗ** (для пропозициональной логики).

ХОРНОВСКИЕ ВЫРАЖЕНИЯ

(продолжение)

$P \vee \neg Q \vee \neg R$	$Q \wedge R \Rightarrow P$		**Определенное выражение**
P	$True \Rightarrow P$	Факт	
$\neg Q \vee \neg R$	$Q \wedge R \Rightarrow False$	Ограничение	

В логике первого порядка:

$\text{King(x)} \wedge \text{Greedy(x)} \Rightarrow \text{Evil(x)}$ соответствует $\forall x\ \text{King(x)} \wedge \text{Greedy(x)} \Rightarrow \text{Evil(x)}$

ОБОБЩЕННОЕ MODUS PONENS

Для атомарных высказываний p_i, p_i' и q, если существует подстановка θ, такая, что Subst(θ, p_i') = Subst(θ, p_i), то для всех *i* имеет место следующее:

$$\frac{p_1', p_2', \ldots, p_n', (p_1 \wedge p_2 \wedge \cdots \wedge p_n \Rightarrow q)}{Subst(\theta, q)}$$

Например:

$King(x) \wedge Greedy(x) \Rightarrow Evil(x)$

$Greedy(m)$

$King(John)$

Тогда:

p_1' - $King(John)$; p_1 - $King(x)$; p_2' - $Greedy(m)$; p_2 - $Greedy(x)$; q - $Evil(x)$

θ - $\{x/John, m/John\}$; $Subst(\theta, q)$ - $Evil(John)$

УНИФИКАЦИЯ

Применение обобщенного М.Р. связано с поиском подстановок, в результате которых различные логические выражения становятся идентичными. Этот процесс называется **унификацией** и является ключевым компонентом любых алгоритмов вывода в логике первого порядка.

Алгоритм Unify принимает на входе два высказывания и возвращает для них **унификатор**, если таковой существует:
$Unify(p, q) = \theta$, где $Subst(\theta, p) = Subst(\theta, q)$

Например:
$Unify(Knows(John, x), Knows(John, Jane)) = \{x/Jane\}$
$Unify(Knows(John, x), Knows(y, Bill)) = \{x/Bill\,, y/John\}$
$Unify(Knows(John, x), Knows(y, Mother(y))) = \{y/John\,, x/Mother(John)\}$
$Unify(Knows(John, x), Knows(y, z)) = \{y/John\,, x/z\}$

Не $\{y/John\,, x/John\,, z/John\}$!
Наиболее общий унификатор

АЛГОРИТМ УНИФИКАЦИИ

Идея: рекурсивно исследовать два выражения одновременно, "бок о бок", наряду с этим формируя унификатор, но создавать ситуацию неудачного завершения, если две соответствующие точки в полученных таким образом структурах не совпадают.

Особый случай: если переменная согласуется со сложным термом, необходимо провести проверку того, встречается ли сама эта переменная внутри терма; в случае положительного ответа на данный вопрос согласование оканчивается неудачей, поскольку невозможно сформировать какой-либо совместимый унификатор.

Детально: Рассел С., Норвиг П. Искусственный интеллект. Современный подход, 2-е изд.

ПРОСТОЙ АЛГОРИТМ ПРЯМОГО ЛОГИЧЕСКОГО ВЫВОДА

Идея: В каждой итерации добавлять к базе знаний КВ все атомарные высказывания, которые могут быть выведены за один этап из импликационных высказываний и атомарных высказываний, которые уже находятся в базе знаний.

N.B.: Для хорновских баз знаний!

ПРОСТОЙ АЛГОРИТМ ПРЯМОГО ЛОГИЧЕСКОГО ВЫВОДА

function FOL-FC-Ask(KB, α) **returns** подстановка или значение false

  inputs: KB, база знаний - множество определенных выражений первого порядка

  α, запрос - атомарное высказывание

  local variables: new, новые высказывания, выводимые в каждой итерации

  repeat until множество new не пусто

    new <— {}

    for each высказывание r in KB **do**

      $(p_1 \wedge p_2 \wedge \cdots \wedge p_n \Rightarrow q)$ <- Standardize-Apart(r)

      for each подстановка θ, такая что

        Subst(θ, $p_1 \wedge p_2 \wedge \cdots \wedge p_n$) = Subst($\theta$, $p'_1 \wedge p'_2 \wedge \cdots \wedge p'_n$)

        для некоторых $p'_1, p'_2, \ldots, p'_n$ в базе знаний KB

        // (на следующем слайде)

      добавить множество new к базе знаний KB

    return false

ПРОСТОЙ АЛГОРИТМ ПРЯМОГО ЛОГИЧЕСКОГО ВЫВОДА

θ – исследуемая подстановка
q – голова текущего определенного выражения
new – новые высказывания, выводимые в каждой итерации
α – запрос - атомарное высказывание

q' <- Subst(θ, q)
if выражение q' не является переименованием некоторого высказывания,
которое уже находится в КВ, или рассматривается как элемент
множества new **then do**
добавить q' к множеству new
ϕ <- Unify(q', α)
if значение ϕ не представляет собой *fail*
then return ϕ

СВОЙСТВА АЛГОРИТМА FOL-FC-ASK

1. **Непротиворечив,** поскольку каждый этап представляет собой применение обобщенного М.Р., которое само по себе непротиворечиво.
2. **Полон**, применительно к базам знаний с *определенными выражениями*, то есть, способен ответить на любой запрос, ответы на который следуют из БЗ.

Однако в приведенной реализации:

- возможна генерация бесконечного числа фактов;
- внутренний цикл связан с поиском всех возможных унификаторов, что может быть дорогостоящей операцией;
- повторная проверка каждого правила в каждой итерации для определения того, выполняются ли его предпосылки;
- может вырабатывать много фактов, не имеющих отношения к текущей цели.

ОБРАТНЫЙ ЛОГИЧЕСКИЙ ВЫВОД

Идея: Алгоритмы обратного логического вывода действуют в обратном направлении, от цели, проходя по цепочке от одного правила к другому, чтобы найти известные факты, которые поддерживают доказательство.

ПРОСТОЙ АЛГОРИТМ ОБРАТНОГО ЛОГИЧЕСКОГО ВЫВОДА

function FOL-BC-Ask(KB, goals, θ) **returns** множество подстановок

  inputs: KB, база знаний

  goals, список конъюнктов, образующих запрос (подстановка θ уже применена)

  θ, текущая подстановка, первоначально пустая подстановка {}

  local variables: answers, ответы - множество подстановок, первоначально пустое

  if список goals пуст **then return** {θ}

  q' <- Subst(θ, First(goals))

  for each высказывание r in KB, где Standardize-Apart(r) = $(p_1 \wedge p_2 \wedge \cdots \wedge p_n \Rightarrow q)$ и θ <- Unify(q, q') является выполнимым

    new_goals <- [$p_1, p_2, \ldots, p_n$ | Rest(goals)]

    answers <- FOL-BC-Ask(KB, new_goals, Compose(θ', θ)) ∪ answers

  return answers

ЛОГИЧЕСКОЕ ПРОГРАММИРОВАНИЕ

Обратный логический вывод находит широкое применение, например, в системах логического программирования, одним из ярких представителей которого является язык **Prolog**.

Выполнение программ Prolog осуществляется по принципу обратного логического вывода, при котором попытка применения выражений выполняется в том порядке, в каком они записаны в базу знаний. Но некоторые описанные ниже особенности языка Prolog выходят за рамки стандартного логического вывода:

- встроенные функции для выполнения арифметических операций;
- встроенные предикаты, вызывающие побочные эффекты (ввод-вывод, модификация БЗ);
- допускается определенная форма отрицания (как невозможность доказательства).

МЕТОД РЕЗОЛЮЦИИ

Полная процедура вывода для **логики первого порядка**.

Не обязательно хорновские правила!

Общая схема:

1) Приведение БЗ к конъюнктивной нормальной форме (конъюнкции выражений, каждое из которых представляет собой дизъюнкцию литералов):

$$\forall x A(x) \wedge W(x) \wedge S(x,y,z) \Rightarrow C(x) \text{ --- } \neg A(x) \vee \neg W(x) \vee \neg S(x,y,z) \vee C(x)$$

2) Использование правила резолюции:

$$\frac{l_1 \vee \cdots \vee l_k, m_1 \vee \cdots \vee m_n}{Subst(\theta, l_1 \vee \cdots \vee l_{i-1} \vee l_{i+1} \vee \cdots \vee l_k \vee m_1 \vee \cdots \vee m_{j-1} \vee m_{j+1} \vee \cdots \vee m_n)}$$

где $Unify(l_i, \neg m_j) = \theta$

Применение: системы автоматического доказательства теорем.

ЕЩЁ О ПРЕДСТАВЛЕНИИ

Онтологический вклад в описание действительности – представление о характере действительности. Например, в пропозициональной логике предполагается, что существуют лишь *факты*, которые относятся или не относятся к данному миру. В логике первого порядка приняты более широкие предположения, а именно, что мир состоит из *объектов*, между которыми могут быть или не быть некоторые *отношения*. Логики специального назначения могут иметь больший онтологический вклад: временн*ая* логика, логика высокого порядка.

Эпистемологический вклад – возможные состояния знаний, которые она позволяет выразить в отношении каждого факта. И в пропозициональной логике, и в логике первого порядка любое высказывание представляет собой факт, и агент либо доверяет утверждению, что это высказывание истинно, либо доверяет утверждению, что оно ложно, либо не имеет мнения на этот счет.

ЕЩЁ О ПРЕДСТАВЛЕНИИ

(продолжение)

Язык	Онтологический вклад (что существует в мире)	Эпистемологический вклад (степень доверия фактам)
Пропозициональная логика	Факты	Истинно/ложно/неизвестно
Логика первого порядка	Факты, объекты, отношения	Истинно/ложно/неизвестно
Временная логика	Факты, объекты, отношения, интервалы времени	Истинно/ложно/неизвестно
Теория вероятностей	Факты	Степень доверия [0, 1]
Нечёткая логика	Факты со степенью истинности [0, 1]	Известное интервальное значение

ЭЛЕМЕНТЫ СИТУАЦИОННОГО ИСЧИСЛЕНИЯ

(Элемент инженерии знаний, который призван показать, как можно применять логику первого порядка не совсем тривиальным образом при организации интеллектуального поведения в динамической системе.)

Ситуационное исчисление – описание ситуаций, обозначающих состояния, возникающие в результате выполнения действий.

• Действия представляют собой логические термы, такие как Forward и Turn (Right).
• Ситуации представляют собой логические термы, состоящие из начальной ситуации (обычно называемой S0) и всех ситуаций, которые создаются в результате применения некоторого действия в некоторой ситуации. Функция Result(a, s) (иногда называемая Do) обозначает ситуацию, возникающую в результате выполнения действия a в ситуации s.
• **Флюентными** называются функции и предикаты, которые изменяются от одной ситуации к другой. В соответствии с общепринятым соглашением ситуация — это всегда последний параметр флюентного высказывания.
• Допускается также использовать вневременные, или неизменные предикаты и функции.

ЭЛЕМЕНТЫ СИТУАЦИОННОГО ИСЧИСЛЕНИЯ: НАВИГАЦИЯ В ЛАБИРИНТЕ

Флюентный предикат:

At(o, x, s) – агент *o* находится в клетке *x* в ситуации (момент времени) *s.*

Начальные условия:

$At(o, x, s) \Leftrightarrow (o = Agent \wedge x = Cell11 \wedge S_0)$

Цель:

Получить план действий как ответ на запрос

$\exists seq\ At(Agent, Cell73, Result(seq, S_0))$

Описание действий:

Аксиома возможности:

Предусловия $\Rightarrow Poss(a, s)$

Аксиома состояния-преемника:

Действие возможно ⇒ (Флюентное высказывание является истинным в результирующем состоянии ⇔ Оно стало истинным в результате действия ∨ оно было истинным прежде, а действие оставило его неизменным)

$$Poss(a, s) \Rightarrow$$
$$At\big(Agent, y, Result(a, s)\big) \Leftrightarrow a = Go(x, y) \vee At(Agent, y, s) \wedge a \neq Go(y, z)$$

ОПИСАТЕЛЬНЫЕ ЛОГИКИ

Синтаксис логики первого порядка предназначен для упрощения процедуры формирования высказываний об объектах, а **описательные логики** представляют собой системы обозначений, которые предназначены для упрощения процедуры описания определений и свойств категорий.

Основные задачи логического вывода для описательных логик сводятся к **обобщению** (проверке того, является ли одна категория подмножеством другой путем сравнения их определений) и **классификации** (определению принадлежности некоторого объекта к какой-то категории). В некоторых системах предусматривается также проверка непротиворечивости категории.

Следствием этой ориентации на описание определений и свойств категорий является то, что описательные логики ложатся в основу онтологий. В частности, в настоящее время ДЛ являются важным в концепции Семантической паутины, где их предполагается использовать при построении онтологий.

Фрагменты OWL-DL и OWL-Lite языка веб-онтологий OWL также основаны на ОЛ.

ОПИСАТЕЛЬНЫЕ ЛОГИКИ

Описательные логики оперируют понятиями **концéпт** и **роль**, соответствующими в других разделах математической логики понятиям «одноместный предикат» (или множество, класс) и «двуместный предикат» (или бинарное отношение).

Интуитивно, концепты используются для описания *классов* некоторых объектов, например, «Люди», «Женщины», «Машины». Роли используются для описания двуместных отношений между объектами, например, на множестве людей имеется двуместное отношение «X есть_родитель_для Y», а между людьми и машинами имеется двуместное отношение «X имеет_в_собственности Y».

С помощью языка описательных логик можно формулировать утверждения общего вида — о классах вообще (всякий Студент есть Человек) и частного вида — о конкретных объектах (Иван есть Студент).

СИНТАКСИС ЛОГИКИ ALC

- Всякий атомарный концепт является концептом;
- выражения ⊥ и T являются концептами;
- если *C* есть концепт, то его дополнение $\neg C$ является концептом;
- если *C* и *D* есть концепты, то их пересечение $C \sqcap D$ и объединение $C \sqcup D$ являются концептами;
- если *C* есть концепт, а *R* есть роль, то выражения $\forall R.C$ и $\exists R.C$ являются концептами.

СЕМАНТИКА ЛОГИКИ ALC

Интерпретация $\mathcal{I}$ состоит из непустого множества $\Delta^{\mathcal{I}}$ (домена) и интерпретирующей функции, которая сопоставляет каждому атомарному концепту A некоторое подмножество $A^{\mathcal{I}}$, а каждой атомарной роли R — некоторое подмножество $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. Если пара индивидов принадлежит интерпретации некоторой роли R, то есть $(e, d) \in R^{\mathcal{I}}$, то говорят, что индивид d является R-последователем индивида e.

Составные концепты:

- T интерпретируется как весь домен;
- $\bot$ интерпретируется как пустое множество;
- дополнение концепта интерпретируется как дополнение множества;
- пересечение концептов интерпретируется как пересечение множеств;
- объединение концептов интерпретируется как объединение множеств;
- выражение $\forall R.C$ интерпретируется как множество тех индивидов, у которых все R-последователи принадлежат интерпретации концепта C;
- выражение и $\exists R.C$ интерпретируется как множество тех индивидов, у которых имеется R-последователь, принадлежащий интерпретации концепта C.

БАЗЫ ЗНАНИЙ ОПИСАТЕЛЬНОЙ ЛОГИКИ

Знания подразделяются на:

- набор терминологических аксиом или TBox;
- набор утверждений об индивидах или ABox.

Терминологические аксиомы:

$$Woman \equiv Person \sqcap Female$$

$$Mother \equiv Woman \sqcap \exists hasChild.\top$$

$$Doctor \sqsubseteq Person$$

Аксиома эквивалентности концептов

Аксиома вложенности концептов

Утверждения об индивидах:

$$Mary : Woman \sqcap \neg Doctor$$

$$Peter : Doctor \sqcap \forall hasChild.\bot$$