

# Методы искусственного интеллекта

Лекция 2. Методы, основанные на знаниях. Онтологии

## Тема 4. Методы, основанные на знаниях

# Агенты, основанные на знаниях

- **Идея:** представлять имеющуюся информацию о задаче (наблюдаемом состоянии и закономерностях) в таком виде, чтобы можно было *манипулировать* этой информацией, в том числе, получая *новую информацию, явным образом не заданную*
  - Вывод (логический?)
  - Характер *манипуляций* зависит от *способа представления знаний*
    - Та или иная формальная модель
      - Логика высказываний (пропозициональная логика)
      - Логика предикатов (первого порядка)
      - Дескрипционные логики
      - ...
- **Достоинства:**
  - Способны принимать к исполнению новые задачи
  - Способны получать новые инструкции и усваивать новые знания
  - Способны приспосабливаться к изменениям в среде, обновляя знания

# 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** an *action*  
    **persistent:** *KB*, a knowledge base  
                  *t*, a counter, initially 0, indicating time  
  
    TELL(*KB*, MAKE-PERCEPT-SENTENCE(*percept*, *t*))  
    *action*  $\leftarrow$  ASK(*KB*, MAKE-ACTION-QUERY(*t*))  
    TELL(*KB*, MAKE-ACTION-SENTENCE(*action*, *t*))  
    *t*  $\leftarrow$  *t* + 1  
    **return** *action*

# Общие термины

**Синтаксис** языка представления – правила формирования высказываний.

**Семантика** определяет истинность каждого из высказываний применительно к каждому из *возможных миров*.

- Модель – сопоставление конкретных значений компонентам высказывания.
  - Высказывание  $\alpha$  является истинным в модели  $m$
  - $m$  является моделью высказывания  $\alpha$

$\alpha \models \beta$  – “высказывание  $\alpha$  влечет за собой высказывание  $\beta$ ”

Тогда и только тогда, когда в любой модели, в которой высказывание  $\alpha$  является истинным,  $\beta$  также истинно.

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

# 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 (адекватность модели)

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





# Логика первого порядка

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

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

**Примеры синтаксически корректных высказываний:**

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

$\text{LeftLeg}(\text{John}) = \text{Mary}$

# Семантика

Семантика связывает высказывания с объектами и отношениями реального мира, 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.

## Пример интерпретации:

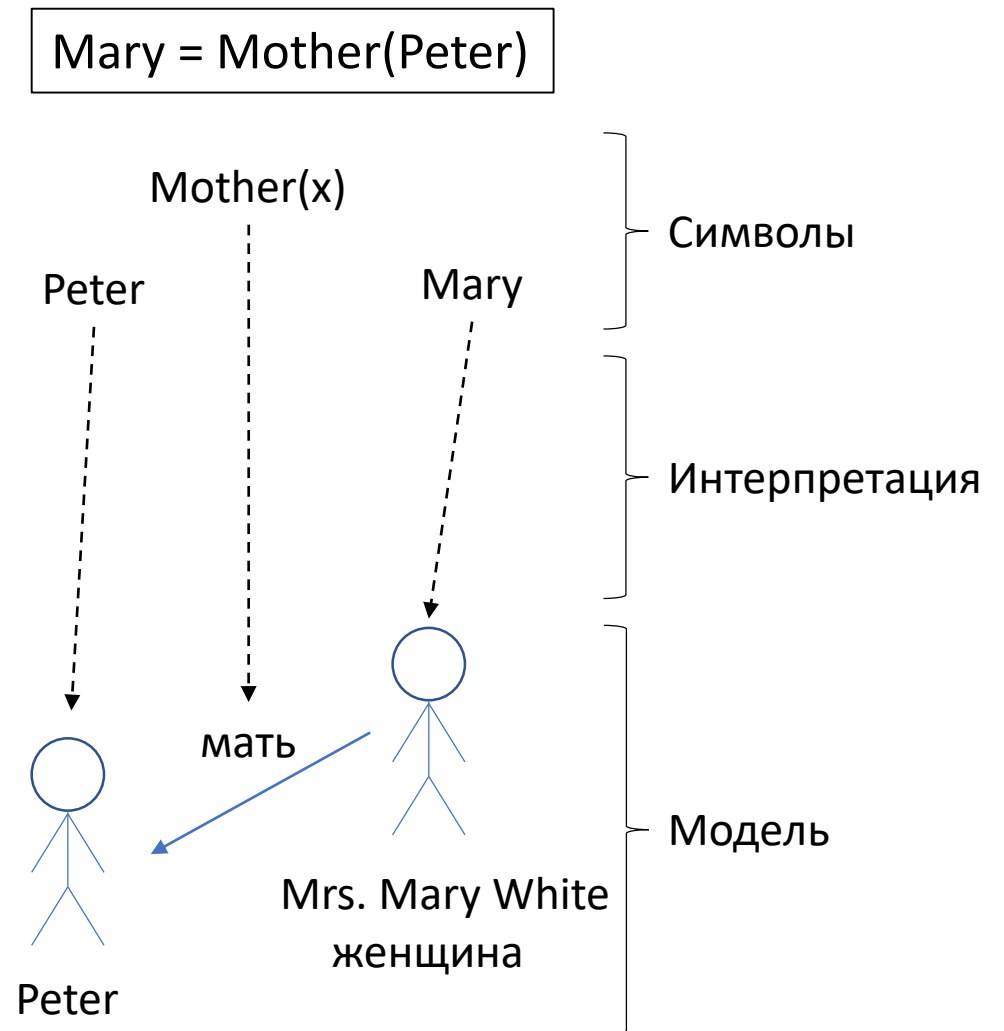
*Woman(x)* – «быть женщиной»

*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  $\forall x P$  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  $\exists x P$  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.

# 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:

$$\text{Tell}(\text{KB}, \text{King}(\text{John}))$$
$$\text{Tell}(\text{KB}, \forall x \text{King}(x) \Rightarrow \text{Person}(x))$$

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

$$\text{Ask}(\text{KB}, \text{King}(\text{John}))$$

Вопросы, заданные с помощью операции Ask, называются **запросами**, или **целями**. На любой запрос, который логически следует из базы знаний, *должен* быть получен утвердительный ответ:

$$\text{Ask}(\text{KB}, \text{Person}(\text{John})) \quad \text{--- True}$$
$$\text{Ask}(\text{KB}, \exists x \text{Person}(x)) \quad \text{--- True (И???)...}$$

# Подстановка (список связывания)

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

То есть:

$\text{Ask}(\text{KB}, \exists x \text{Person}(x)) \quad \text{---} \quad \{x/\text{John}\}$

# 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

Мать — это родитель женского пола:

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

Муж — это супруг мужского пола:

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

Мужчины и женщины — непересекающиеся категории людей:

$$\forall x \text{ Male}(x) \Leftrightarrow \neg \text{Female}(x)$$

Отношения между родителями и детьми являются взаимно противоположными:

$$\forall p, c \text{ Parent}(p, c) \Leftrightarrow \text{Child}(c, p)$$

Дедушка или бабушка — это родитель родителя:

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

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.

# Вывод в логике первого порядка

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

# Хорновские выражения

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

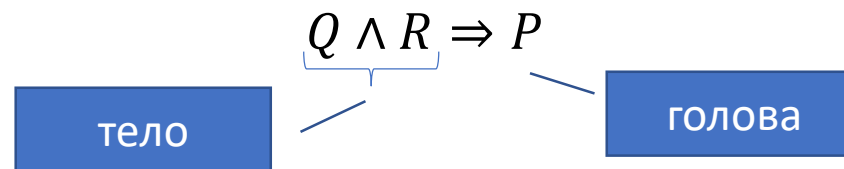
$P \vee \neg Q \vee \neg R$  – хорновское

$\neg Q \vee \neg R$  – хорновское

$P \vee Q$  – **не** хорновское

**Зачем они нужны:**

1) Может быть записано как импликация вида:



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

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

**В логике первого порядка:**

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

# Обобщенное правило отделения (modus ponens)

Для атомарных высказываний  $p_i$ ,  $p'_i$  и  $q$ , если существует подстановка  $\theta$ , такая, что  $\text{Subst}(\theta, p'_i) = \text{Subst}(\theta, p_i)$ , то для всех  $i$  имеет место следующее:

$$\frac{p'_1, p'_2, \dots, p'_n, (p_1 \wedge p_2 \wedge \dots \wedge p_n \Rightarrow q)}{\text{Subst}(\theta, q)}$$

**Например:**

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

$\text{Greedy}(m)$

$\text{King}(\text{John})$

**Тогда:**

$p'_1 - \text{King}(\text{John}); p'_2 - \text{Greedy}(m)$

$p_1 - \text{King}(x); p_2 - \text{Greedy}(x); q - \text{Evil}(x)$

$\theta - \{x/\text{John}, m/\text{John}\}$

$\text{Subst}(\theta, q) - \text{Evil}(\text{John})$

# 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.

Unify принимает на входе два высказывания и возвращает для них **унификатор**, если таковой существует:

$$Unify(p, q) = \theta, \text{ где } Subst(\theta, p) = Subst(\theta, q)$$

**Например:**

$$Unify(Knows(John, x), Knows(John, Jane)) = \{x/Jane\}$$

$$Unify(Knows(John, x), Knows(y, Bill)) = \{x/Bill, y/John\}$$

$$Unify(Knows(John, x), Knows(y, Mother(y))) = \{y/John, x/Mother(John)\}$$

$$Unify(Knows(John, x), Knows(y, z)) = \{y/John, x/z\}$$

# 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

# Простой алгоритм прямого логического вывода

- **Идея:** В каждой итерации добавлять к базе знаний KB 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( $KB, \alpha$ ) returns a substitution or false
  inputs:  $KB$ , the knowledge base, a set of first-order definite clauses
            $\alpha$ , the query, an atomic sentence

  while true do
     $new \leftarrow \{ \}$       // The set of new sentences inferred on each iteration
    for each rule in  $KB$  do
       $(p_1 \wedge \dots \wedge p_n \Rightarrow q) \leftarrow \text{STANDARDIZE-VARIABLES}(\textit{rule})$ 
      for each  $\theta$  such that  $\text{SUBST}(\theta, p_1 \wedge \dots \wedge p_n) = \text{SUBST}(\theta, p'_1 \wedge \dots \wedge p'_n)$ 
        for some  $p'_1, \dots, p'_n$  in  $KB$ 
           $q' \leftarrow \text{SUBST}(\theta, q)$ 
          if  $q'$  does not unify with some sentence already in  $KB$  or  $new$  then
            add  $q'$  to  $new$ 
             $\phi \leftarrow \text{UNIFY}(q', \alpha)$ 
            if  $\phi$  is not failure then return  $\phi$ 
    if  $new = \{ \}$  then return false
  add  $new$  to  $KB$ 
```

# Простой алгоритм прямого логического вывода

**function** FOL-FC-ASK( $KB, \alpha$ ) **returns** a substitution or *false*

**inputs:**  $KB$ , the knowledge base, a set of first-order definite clauses  
 $\alpha$ , the query, an atomic sentence

**while true do**

$new \leftarrow \{ \}$       *// The set of new sentences inferred on each iteration*

**for each rule in**  $KB$  **do**

$(p_1 \wedge \dots \wedge p_n \Rightarrow q) \leftarrow \text{STANDARDIZE-VARIABLES}(\text{rule})$

**for each**  $\theta$  such that  $\text{SUBST}(\theta, p_1 \wedge \dots \wedge p_n) = \text{SUBST}(\theta, p'_1 \wedge \dots \wedge p'_n)$   
            for some  $p'_1, \dots, p'_n$  in  $KB$

$q' \leftarrow \text{SUBST}(\theta, q)$

**if**  $q'$  does not unify with some sentence already in  $KB$  or  $new$  **then**

                add  $q'$  to  $new$

$\phi \leftarrow \text{UNIFY}(q', \alpha)$

**if**  $\phi$  is not *failure* **then return**  $\phi$

**if**  $new = \{ \}$  **then return false**

    add  $new$  to  $KB$

**База знаний:**

1. little(cat)
  2. middle(tiger)
  3. big(elephant)
  4. strong(tiger)
  5. middle(X)  $\wedge$  strong(X)  $\Rightarrow$  powerful(X)
  6. big(X)  $\Rightarrow$  powerful(X)
  7. powerful(X)  $\Rightarrow$  dangerous(X)
- } Факты ( $p_i$ )

**Запрос:**

ASK(dangerous(tiger))

# Свойства рассмотренного алгоритма прямого логического вывода

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.

Улучшения: например, Rete-алгоритм

# 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( $KB, query$ ) **returns** a generator of substitutions  
    **return** FOL-BC-OR( $KB, query, \{\}$ )

**function** FOL-BC-OR( $KB, goal, \theta$ ) **returns** a substitution  
    **for each**  $rule$  **in** FETCH-RULES-FOR-GOAL( $KB, goal$ ) **do**  
         $(lhs \Rightarrow rhs) \leftarrow$  STANDARDIZE-VARIABLES( $rule$ )  
        **for each**  $\theta'$  **in** FOL-BC-AND( $KB, lhs, UNIFY(rhs, goal, \theta)$ ) **do**  
            **yield**  $\theta'$

**function** FOL-BC-AND( $KB, goals, \theta$ ) **returns** a substitution  
    **if**  $\theta = failure$  **then return**  
    **else if** LENGTH( $goals$ ) = 0 **then yield**  $\theta$   
    **else**  
         $first, rest \leftarrow$  FIRST( $goals$ ), REST( $goals$ )  
        **for each**  $\theta'$  **in** FOL-BC-OR( $KB, SUBST(\theta, first), \theta$ ) **do**  
            **for each**  $\theta''$  **in** FOL-BC-AND( $KB, rest, \theta'$ ) **do**  
                **yield**  $\theta''$

# Обратный логический вывод и 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):

- $\forall x A(x) \wedge W(x) \wedge S(x, y, z) \Rightarrow C(x) \dashv\vdash \neg A(x) \vee \neg W(x) \vee \neg S(x, y, z) \vee C(x)$

2) Использование правила резолюции:

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

где  $Unify(l_i, \neg m_j) = \theta$

- **Применение:** системы автоматического доказательства теорем.

# Тема 5. Онтологии



# Понятие онтологии

**Онтолóгия** (новолат. ontologia от др.-греч. ὄν, род. п. ὄντος — сущее, то, что существует + λόγος — учение, наука) — учение о сущем; учение о бытии как таковом; раздел философии, изучающий фундаментальные принципы бытия, его наиболее общие сущности и категории, структуру и закономерности.

# Понятие онтологии

**Онтоlóгия** (новолат. *ontologia* от др.-греч. ὄν, род. п. ὄντος — сущее, то, что существует + λόγος — учение, наука) — учение о сущем; учение о бытии как таковом; раздел философии, изучающий фундаментальные принципы бытия, его наиболее общие сущности и категории, структуру и закономерности.

**Онтоlóгия в информатике** — это попытка всеобъемлющей и подробной формализации некоторой области знаний с помощью *концептуальной схемы*. Обычно такая схема состоит из структуры данных, содержащей все релевантные классы объектов, их связи и правила (теоремы, ограничения), принятые в этой области.

- Составляющие:
  - Терминология, словарь предметной области, описывающий взаимосвязь между терминами (T-Box)
    - *Протон является элементарной частицей*
  - Утверждения, касающиеся конкретных объектов (A-Box)
    - *У наблюдаемого объекта нулевой заряд, полуцелый спин и большое время жизни*

# Пример онтологии и рассуждений с ее помощью

- Терминологические аксиомы (Abox)

# Пример онтологии и рассуждений с ее помощью

- Терминология (T-Box)
  1. «*быть родителем*» означает либо «*быть отцом*», либо «*быть матерью*»
  2. если *a* является *родителем b*, то *b* является *потомком a*
  3. *сын* – это *потомок* мужского пола
  4. *дочь* – это *потомок* женского пола
- Описания объектов (A-Box)
  - Евграф – мужского пола
  - Фёкла – мать Евграфа
- Возможный вывод
  - 1: Фёкла – *родитель* Евграфа
  - 2: Евграф – *потомок* Фёклы
  - 3: Евграф – *сын* Фёклы

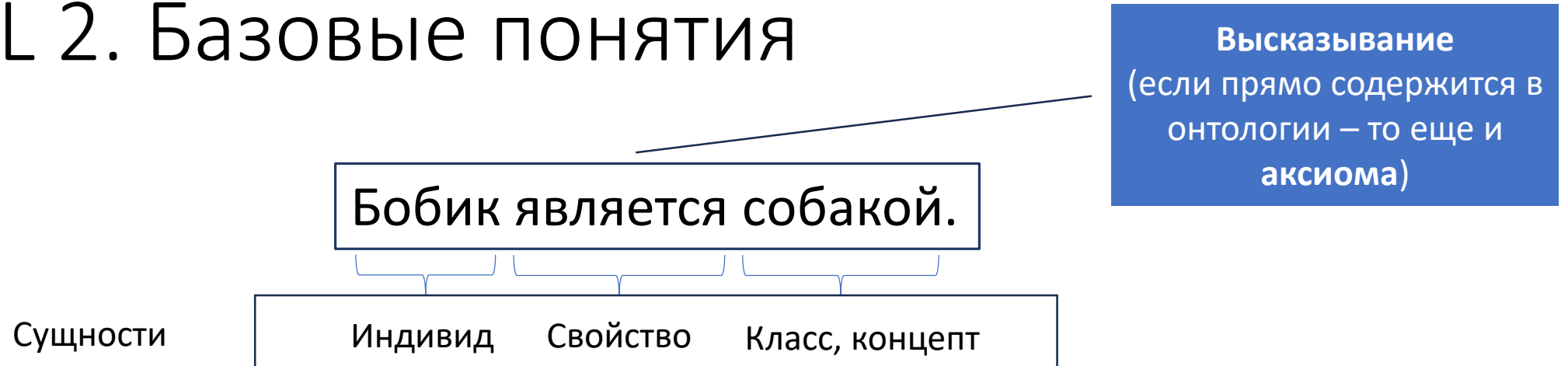
# Язык OWL 2

- **OWL 2 Web Ontology Language** (язык веб-онтологий или онтологий для веб) – это формальный язык онтологий, предназначенный (в первую очередь) для использования в стеке технологий *Семантического Веб* (Semantic Web).
- Онтологии OWL 2 определяют классы, свойства, индивиды и значения, которые могут сохраняться как документы Семантического Веб.
- Синтаксис:
  - *Функциональный*
  - Манчестерский
  - Диктуемые местом в стеке Семантического Веб:
    - RDF/XML
    - OWL XML

# OWL 2. Базовые понятия

- **Аксиомы** (Axioms) – базовые высказывания, которые выражает онтология
- **Сущности** (Entities) – элементы, используемые для обозначения объектов реального мира
  - Индивиды, свойства, классы
- **Выражения** (Expressions) – комбинации сущностей для формирования сложных определений из простых
  - Пересечение, объединение
- Все знания (с которыми имеет дело онтология) состоят из *высказываний*:
  - «идет дождь», «ласточка – птица»
- Набор высказываний может *влечь за собой* (entail) другое высказывание

# OWL 2. Базовые понятия



- **Сущности (Entities)** – элементы, используемые для обозначения объектов реального мира
  - Индивиды – объекты реального мира
  - Классы, концепты – категории объектов
  - Свойства – отношения между объектами
    - Объектные свойства (object properties)
    - Свойства-данные (datatype properties)
    - Свойства-аннотации (annotation properties)

# OWL 2. Классы

`ClassAssertion( :Person :Mary )`

Аксиома, утверждающая, что именованная сущность Mary относится к классу Person

`ClassAssertion( :Woman :Mary )`

Аксиома, утверждающая, что именованная сущность Mary относится к классу Woman

`SubClassOf( :Woman :Person )`

Аксиома, утверждающая, что класс Woman является подклассом класса Person (т.е. любой индивид, который относится к классу Woman также относится и к Person)

`EquivalentClasses( :Human :Person )`

Аксиома эквивалентности классов (синонимы)

`DisjointClasses( :Woman :Man )`

(Принципиально) отсутствуют индивиды, которые могут относиться к перечисленным классам



# OWL 2. Объектные свойства

<code>ObjectPropertyAssertion( :hasWife :John :Mary )</code>	Значением объектного свойства hasWife для индивида John является индивид Mary
<code>NegativeObjectPropertyAssertion( :hasWife :John :Mary )</code>	Значением объектного свойства hasWife для индивида John <u>НЕ</u> является индивид Mary
<code>SubObjectPropertyOf( :hasWife :hasSpouse )</code>	«Подсвойство», конкретизирующее другое свойство
<code>ObjectPropertyDomain( :hasWife :Man )</code>	Область определения свойства (экземпляры каких классов могут иметь свойство)
<code>ObjectPropertyRange( :hasWife :Woman )</code>	Область значений свойства (экземплярами каких классов может быть значение свойства)

# OWL 2. Свойства-данные

DataPropertyAssertion(  
  :hasAge :John "51"^xsd:integer )

Значением свойства hasAge для индивида John является целое число 51

NegativeDataPropertyAssertion(  
  :hasWife :Jack "53"^xsd:integer )

Значением свойства hasAge для индивида Jack НЕ является 53

DataPropertyDomain( :hasAge :Person )

Область определения свойства (экземпляры каких классов могут иметь свойство)

DataPropertyRange(  
  :hasAge xsd:nonNegativeInteger)

Область значений свойства (тип данных)

# OWL 2. Ещё о классах

```
EquivalentClasses(  
  :Mother  
  ObjectIntersectionOf( :Woman :Parent )  
)
```

«Сложные» классы. А также  
ObjectUnionOf, ObjectComplementOf

```
EquivalentClasses(  
  :Parent  
  ObjectSomeValuesFrom( :hasChild :Person )  
)
```

Existential quantification. Также может  
быть universal quantification –  
ObjectAllValuesFrom

```
EquivalentClasses(  
  :JohnsChildren  
  ObjectHasValue( :hasParent :John )  
)
```

```
ClassAssertion(  
  ObjectMaxCardinality( 4 :hasChild :Parent )  
  :John  
)
```

А также минимальная кардинальность  
свойства (ObjectMinCardinality), точная  
кардинальность (ObjectExactCardinality)

# OWL 2. Ещё о свойствах

InverseObjectProperties( :hasParent :hasChild )

SymmetricObjectProperty( :hasSpouse)

А также асимметричные, транзитивные,  
рефлексивные, антирефлексивные,  
функциональные

SubObjectPropertyOf(

ObjectPropertyChain( :hasParent :hasParent )  
:hasGrandparent

)

Т.н. «цепочки свойств» - property chains.

# OWL 2. Управление онтологиями

```
Prefix(:=<http://example.com/owl/families/>)
Prefix(otherOnt:=<http://example.org/otherOntologies/families/>)
Prefix(xsd:=<http://www.w3.org/2001/XMLSchema#>)
Prefix(owl:=<http://www.w3.org/2002/07/owl#>)

Ontology(<http://example.com/owl/families>
  ...

  SameIndividual( :John otherOnt:JohnBrown )
  SameIndividual( :Mary otherOnt:MaryBrown )
  EquivalentClasses( :Adult otherOnt:Grownup )
  EquivalentObjectProperties( :hasChild otherOnt:child )
  EquivalentDataProperties( :hasAge otherOnt:age )

)
```

# Профили OWL 2

- **Профиль OWL 2 (фрагмент)** – «урезанная» разновидность языка, жертвующая частью выразительных возможностей в пользу эффективности вывода.
- Профили:
  - OWL 2 EL – полезен в приложениях, где онтологии содержат очень большое число классов и/или свойств. Основные задачи вывода имеют полиномиальное время от размера онтологии.
    - EL-логики, только existential qualification
  - OWL 2 QL – очень много индивидов, ответ на запросы (поиск индивидов по ограничениям) – самая важная задача вывода.
    - Переписывание на query language
  - OWL 2 RL – наиболее выразительный профиль, сохраняющий, однако полиномиальность базовых алгоритмов вывода (алгоритмов проверки согласованности, допустимости класса, проверки принадлежности классу и ряда других)
    - Вывод может быть реализован на языке правил – rule language

# Основные задачи онтологического вывода

- Согласованность онтологий (ontology consistency)
- Допустимость определения класса(class expression satisfiability)
  - Возможен ли класс с заданным определением, могут ли у него быть индивиды?
- Проверка отношения между классами (class expression subsumption)
  - Является ли один класс подклассом другому
- Проверка индивидов (instance checking)
  - Является ли заданный индивид экземпляром заданного класса
- Обработка запросов ((Boolean) conjunctive query answering)
  - Поиск всех индивидов, удовлетворяющих заданным условиям

# Онтологии и логика

- **Логика первого порядка**
- 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. **Разрешимые фрагменты** логики первого порядка.



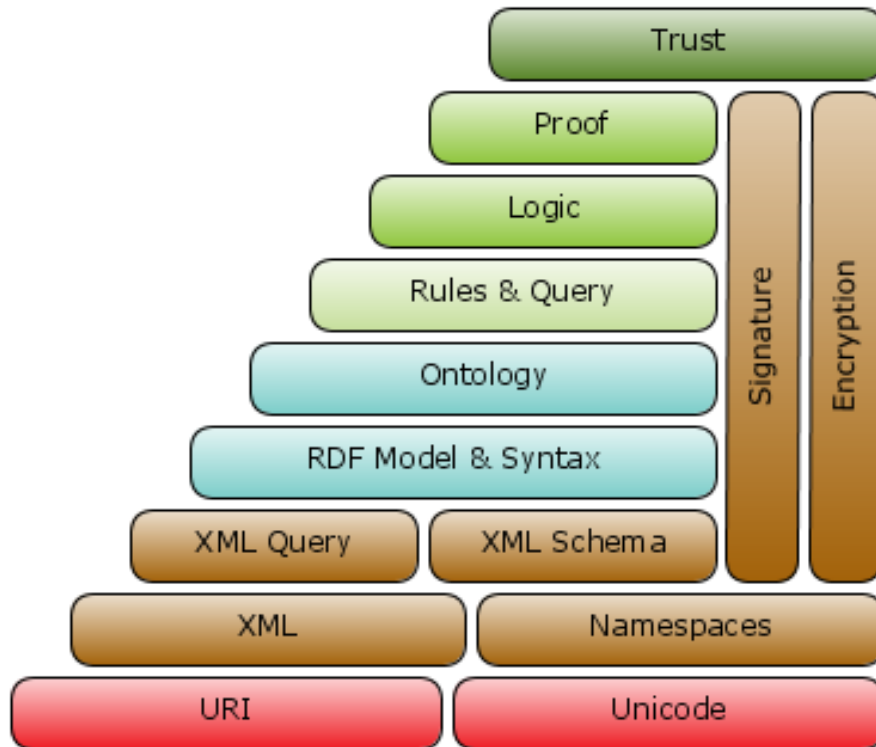
# Онтологии и логика

Описательная логика (ALC)	Теоретико-множественная семантика	Логика первого порядка
Концепт $A$	Множество $A \subseteq \Delta^{\mathcal{I}}$	Одноместный предикат $A$
Роль $R$	Множество кортежей $R \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$	Бинарный предикат $R$
Индивид $d$	Элемент множества $\Delta^{\mathcal{I}}$	Объект $d$
$Doctor \sqcup Lawyer$	$Doctor \cup Lawyer$	$Doctor(x) \vee Lawyer(x)$
$Rich \sqcap Happy$	$Rich \cap Happy$	$Rich(x) \wedge Happy(x)$
$Cat \sqcap \exists sitsOn. Mat$	$Cat \cap \{x \in \Delta^{\mathcal{I}} \mid \exists d \in \Delta^{\mathcal{I}}: (x, d) \in sitsOn \wedge d \in Mat\}$	$\exists y. (Cat(x) \wedge sitsOn(x, y))$
$Peter : Doctor$ $Doctor(Peter)$	$Peter \in Doctor$	$Doctor(Peter)$
$sitsOn(Tom, Mat1)$	$(Tom, Mat1) \in sitsOn$	$sitsOn(Tom, Mat1)$

# Онтологии в контексте Семантического веб

- **Семантический веб (Semantic Web)** – предложенная в 1998 г. Тимом Бернерсом-Ли концепция, в соответствии с которой «человекочитаемая» Всемирная паутина (World Wide Web) будет дополнена «машиночитаемой», основанной на наборе стандартов семантической разметки:
  - Гигантский граф
  - Запросы к графу, ответ на которые получается с помощью вывода и рассуждений

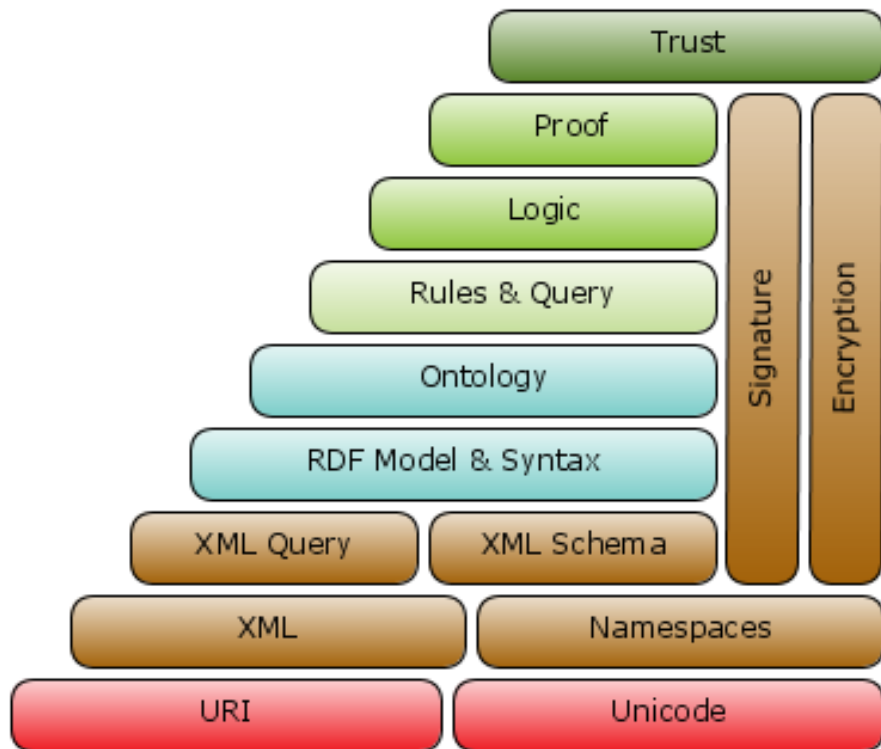
# Онтологии в контексте Семантического веб. URI



URI для уникальных идентификаторов всего – объектов, понятий, свойств в целях разрешения неоднозначности.

- Лук у специалистов по метательному оружию:
  - <http://archery.su/ontology/Лук>
    - Длина плечей, фунтаж и пр.
- Лук у специалистов по сельскому хозяйству:
  - <http://mcx.gov.ru/ontology/Лук>
    - Семейство, типовой вид и пр.

# Онтологии в контексте Семантического веб. XML

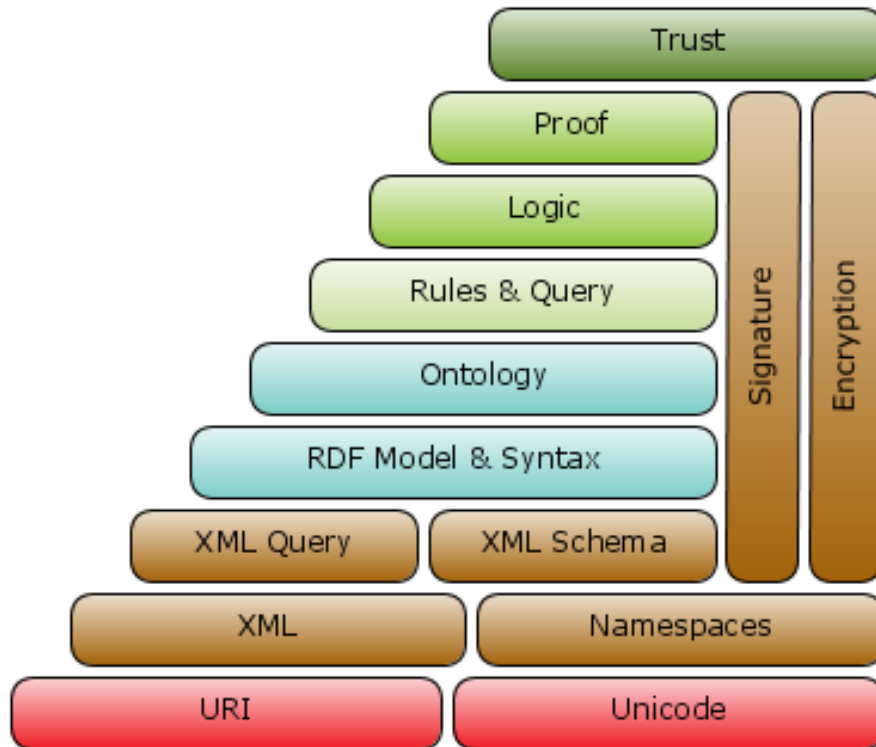


XML (eXtensible Markup Language) – исключительно *синтаксис*. Предназначен для создания *расширений* – конкретных грамматик, представленных словарем тегов и атрибутов и набором правил.

```
<?xml version="1.0">  
<Human name="John">  
    <height>182</height>  
</Human>
```

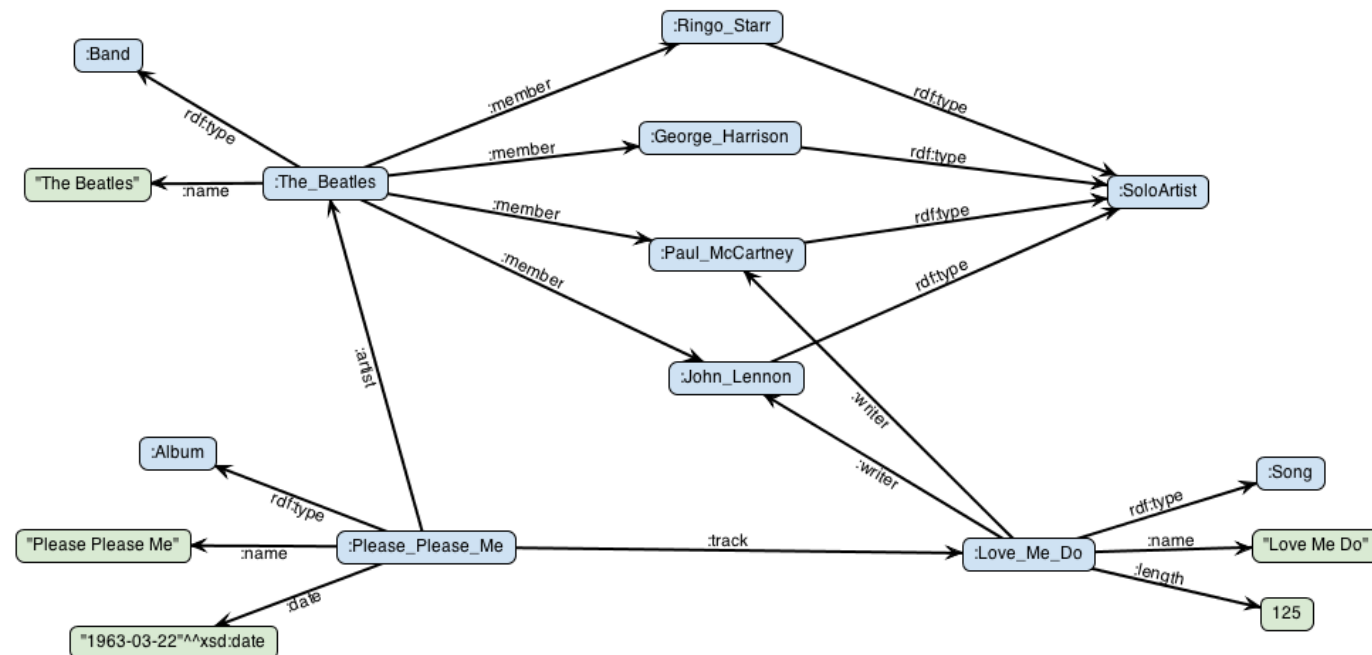
# Онтологии в контексте Семантического веб.

## RDF



RDF (Resource Description Framework) – модель представления данных. В основе модели – тройки (триплеты), состоящие из:

- субъекта
- предиката
- объекта



# Онтологии в контексте Семантического веб. OWL

```
Prefix(:=<urn:webprotege:ontology:4512c73f-900e-414e-9c54-7194715ef272#>)
Prefix(owl:=<http://www.w3.org/2002/07/owl#>)
Prefix(rdf:=<http://www.w3.org/1999/02/22-rdf-syntax-ns#>)
Prefix(xml:=<http://www.w3.org/XML/1998/namespace>)
Prefix(xsd:=<http://www.w3.org/2001/XMLSchema#>)
Prefix(rdfs:=<http://www.w3.org/2000/01/rdf-schema#>)
```

```
Ontology(<urn:webprotege:ontology:4512c73f-900e-414e-9c54-7194715ef272>
```

```
Declaration(Class(<http://avponomarev.bitbucket.io/aim/ontology/Human>))
Declaration(DataProperty(<http://avponomarev.bitbucket.io/aim/ontology/height>))
Declaration(NamedIndividual(<http://avponomarev.bitbucket.io/aim/ontology/John>))
```

```
AnnotationAssertion(rdfs:label
  <http://avponomarev.bitbucket.io/aim/ontology/height> "height")
SubDataPropertyOf(<http://avponomarev.bitbucket.io/aim/ontology/height>
  owl:topDataProperty)
DataPropertyDomain(<http://avponomarev.bitbucket.io/aim/ontology/height>
  <http://avponomarev.bitbucket.io/aim/ontology/Human>)
DataPropertyRange(<http://avponomarev.bitbucket.io/aim/ontology/height>
  xsd:positiveInteger)
```

```
AnnotationAssertion(rdfs:label
  <http://avponomarev.bitbucket.io/aim/ontology/Human> "Human")
```

```
AnnotationAssertion(rdfs:label
  <http://avponomarev.bitbucket.io/aim/ontology/John> "John")
DataPropertyAssertion(<http://avponomarev.bitbucket.io/aim/ontology/height>
  <http://avponomarev.bitbucket.io/aim/ontology/John> "182"^^xsd:integer))
```

```
<?xml version="1.0"?>
<rdf:RDF xmlns="urn:webprotege:ontology:4512c73f-900e-414e-9c54-7194715ef272#"
  xml:base="urn:webprotege:ontology:4512c73f-900e-414e-9c54-7194715ef272"
  xmlns:aim="http://avponomarev.bitbucket.io/aim/ontologies/"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:xml="http://www.w3.org/XML/1998/namespace"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:ontology="http://avponomarev.bitbucket.io/aim/ontology/">
```

```
<owl:Ontology rdf:about="urn:webprotege:ontology:4512c73f-900e-414e-9c54-7194715ef272"/>
```

```
<owl:DatatypeProperty rdf:about="http://avponomarev.bitbucket.io/aim/ontology/height">
  <rdfs:subPropertyOf rdf:resource="http://www.w3.org/2002/07/owl#topDataProperty"/>
  <rdfs:domain rdf:resource="http://avponomarev.bitbucket.io/aim/ontology/Human"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#positiveInteger"/>
  <rdfs:label>height</rdfs:label>
</owl:DatatypeProperty>
```

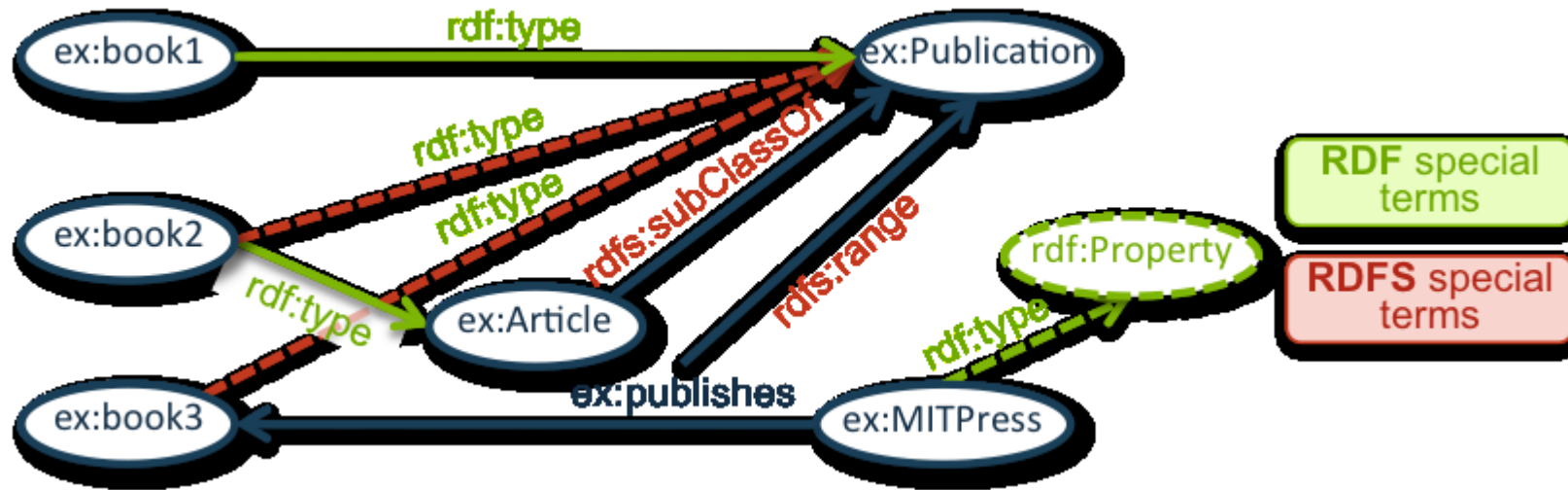
```
<owl:Class rdf:about="http://avponomarev.bitbucket.io/aim/ontology/Human">
  <rdfs:label>Human</rdfs:label>
</owl:Class>
```

```
<owl:NamedIndividual rdf:about="http://avponomarev.bitbucket.io/aim/ontology/John">
  <ontology:height rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">182</ontology:height>
  <rdfs:label>John</rdfs:label>
</owl:NamedIndividual>
</rdf:RDF>
```

# Язык запросов SPARQL

```
PREFIX mo: <http://purl.org/ontology/mo/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?img ?hp ?loc
WHERE {
    ?a a mo:MusicArtist ;
        foaf:name ?name .
    OPTIONAL { ?a foaf:img ?img }
    OPTIONAL { ?a foaf:homepage ?hp }
    OPTIONAL { ?a foaf:based_near ?loc }
}
```

# Язык запросов SPARQL и логические следствия (entailment regimes)



```
SELECT ?pub
WHERE {
  ?pub rdf:type ex:Publication
}
```

- «Простой» режим
  - ex:book1
- RDF
  - ex:book1
- RDFS
  - ex:book1, ex:book2



# Язык запросов SPARQL. Применения

← → ↻ dbpedia.org/sparql

SPARQL Query Editor About Tables ▾ Conductor Facet Browser Permalink

Extensions: cxml save to dav sponge User: SPARQL

Default Data Set Name (Graph IRI)

http://dbpedia.org

Query Text

```
select distinct ?Concept where {[] a ?Concept} LIMIT 100
```

Results Format

HTML ▾

Execute Query Reset



# Semantic Web Rule Language (SWRL): Правила

- **Классификация:**

`hasParent(?x1, ?x2) ^ hasBrother(?x2, ?x3) -> hasUncle(?x1, ?x3)`  
`hasParent(?x1, ?x2) -> Parent(?x2)`

- **Задание значения атрибуту:**

`hasParent(?x1, ?x2) ^ was_born(?x1, ?x3) -> became_parent(?x2, ?x3)`

- **Вычисления:**

`start_study(?x1, ?x2) ^`  
`study_duration(?x1, ?x3) ^`  
`swrlb:add(?x4, ?x2, ?x3) -> end_study(?x1, ?x4)`

# Применение онтологий

- Семантический веб в изначальном понимании (глобальная машиночитаемая сеть знаний), скорее мертв (и никогда не был жив)
- Онтологии  $\neq$  Семантический веб
- Локальные применения:
  - Производственные компании:
    - Festo (свойства компонентов и конфигурирование изделий)
    - Космическая отрасль (German Aerospace Center (DLR), European Space Agency, NASA, ЦНИИмаш)
  - Исследования
    - Gene Ontology
    - OBO Foundry

# Ограничения OWL

- Невозможность выполнять вычисления на основе нечеткой логики (и вообще, неопределенности)
  - Действительно, мы можем напрямую записать в модели факт «В Петербурге часто идет дождь», но это не позволит автоматически в одних случаях получать вывод о том, что дождь идет, а в других – нет.
- Невозможность использования модальных логик, позволяющих описывать не только фактическое состояние систем, но и выражать суждения о том, что может, должно, могло бы быть
- Невозможность выражать сведения о субъективной точке зрения: «Иван считает, что пингины живут в Арктике». Иван, конечно, ошибается, но инструментальных средств для выражения этого в OWL, казалось бы, нет
  - Однако никто не мешает в модели создать класс объектов «Мнение», который будет связывать субъекта с теми фактами, которые он считает истинными.
- Трудноразрешимой проблемой для OWL является представление утверждений о классах.
  - Если «Собака» - класс, то выразить напрямую сведения о том, что все собаки умеют лаять, будет сложно. Для этого потребуется создать класс «Умение», один из объектов которого будет соединять класс Собаки и умение лаять. ПО, использующее модель, сможет интерпретировать эту информацию, но задействовать её в получении логических выводов средствами reasoner'a не получится. Другой вариант решения проблемы – создание правила, которое гласит, что если X – собака, то X умеет лаять. Это сработает, но большими массивами правил трудно управлять, особенно если онтология подвергается рефакторингу.

# Онтологическое моделирование.

## Методология для новичков (Ной и МакГиннесс)

### 1. Определение области и масштаба онтологии

- Какую область будет охватывать онтология?
- Для чего собираемся ее использовать?
- На какие типы вопросов должна давать ответы онтология?
- Кто будет использовать и поддерживать онтологию?

### 2. Рассмотрение вариантов повторного использования существующих онтологий

### 3. Перечисление важных терминов в онтологии

- Просто список, без какого-то первоначального отсеивания

### 4. Определение классов и иерархии классов

- Нисходящая разработка (от самых общих понятий)
- Восходящая (от самых конкретных классов, обобщая их)
- Комбинированная

### 5. Определение свойств классов

### 6. Определение ограничений на значения свойств

### 7. Создание экземпляров

# Литература

- Методы на основе знаний (в целом):
  - Рассел, П. Норвиг Искусственный интеллект: современный подход, 4-е изд.
- Онтологии:
  - Т.А. Гаврилова, Д.В. Кудрявцев, Д.И. Муромцев Инженерия знаний. Модели и методы: учебник. – СПб., 2016.
  - OWL 2 Primer. URL: <https://www.w3.org/TR/owl2-primer/>
  - С. Горшков Введение в онтологическое моделирование. URL: <https://trinidata.ru/files/SemanticIntro.pdf>
  - Миф семантического веба. URL: <https://habr.com/ru/articles/502628/>