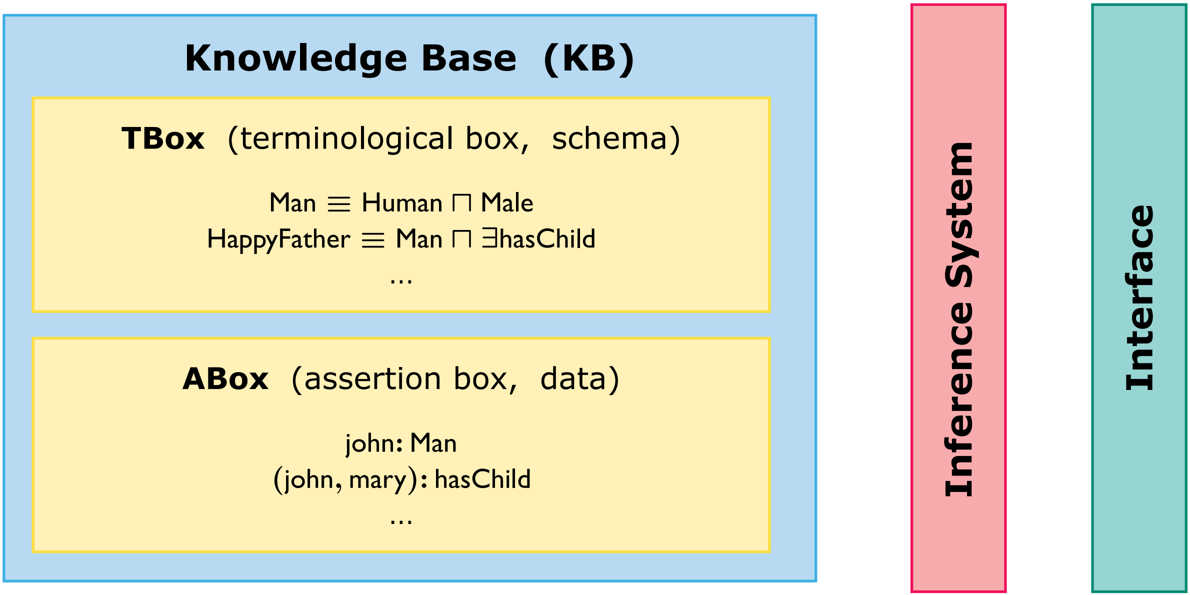
ABox: данные и дескрипционная логика



Базы знаний

База знаний $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ состоит из теории $\mathcal{T}$ и ABox $\mathcal{A}$.

Заметим, что данное определение покрывает ABox со сложными утверждениями вида $a : C$. Если сложное утверждение вида $a : C$ (т.е. C — сложный концепт) входит в $\mathcal{A}$, мы

- заменим его на $a : A$, где A — новое имя;
- добавим $A \equiv C$ к $\mathcal{T}$.

Назовем полученную базу знаний $(\mathcal{T}', \mathcal{A}')$. Тогда $(\mathcal{T}', \mathcal{A}')$ дает те же ответы на все запросы, не содержащие A что и $(\mathcal{T}, \mathcal{A})$.

Пример

Рассмотрим TBox Uni:

- **BritishUnivty** $\sqsubseteq$ **University**;
- **University** $\sqcap$ **Student** $\equiv \bot$;
- $\top \sqsubseteq \forall$**registered_at.University**;
- $\top \sqsubseteq \forall$**student_at.University**;
- $\exists$**student_at.**$\top \sqsubseteq$ **Student**;
- **Student** $\sqsubseteq \exists$**student_at.**$\top$;
- **NonBritishUni** $\equiv$ **University** $\sqcap \neg$**BritishUnivty**.

Пример

и $\mathrm{Uni}_{\mathcal{A}}$:

- $\mathbf{CMU} : \mathbf{NonBritishUni}$
- $\mathbf{Harvard} : \mathbf{Inst}, \quad \mathbf{FUBerlin} : \mathbf{Inst}$
- $\mathbf{LU} : \mathbf{BritishUnivty}, \quad \mathbf{MU} : \mathbf{BritishUnivty};$
- $\mathbf{Tim} : \mathbf{Student};$
- $(\mathbf{Tim}, \mathbf{LU}) : \mathbf{registered\_at}, \quad (\mathbf{Bob}, \mathbf{MU}) : \mathbf{registered\_at};$
- $(\mathbf{Tom}, \mathbf{Harvard}) : \mathbf{student\_at}.$

Обозначим by $\mathrm{Uni}_{\mathcal{R}}$ базу данных с той же информацией.

Ответы

Query	Rel. DB Uni_R	Abox Uni_A	KB $(\mathrm{Uni}, \mathrm{Uni}_A)$
University(CMU)	No	Don't know	Yes
University(Harvard)	No	Don't know	Yes
NonBritishUni(CMU)	Yes	Yes	Yes
Student(Tim)	Yes	Yes	Yes
Student(Tom)	No	Don't know	Yes
$\exists$**student_at.**$\top$(**Tom**)	Yes	Yes	Yes
$\exists$**student_at.**$\top$(**Tim**)	No	Don't know	Yes
(**Student** $\sqcap$ $\neg$**Univty**)(**Tim**)	Yes	Don't know	Yes
(**Inst** $\sqcap$ $\neg$**Univty**)(**FUBerlin**)	Yes	Don't know	Don't know

BritishUnivty $\sqsubseteq$ **University**
University $\sqcap$ **Student** $\equiv$ $\bot$
$\top$ $\sqsubseteq$ $\forall$**registered_at.University**
$\top$ $\sqsubseteq$ $\forall$**student_at.University**
$\exists$**student_at.**$\top$ $\sqsubseteq$ **Student**
Student $\sqsubseteq$ $\exists$**student_at.**$\top$
NonBritishUni $\equiv$ **University** $\sqcap$ $\neg$**BritishUnivty**

CMU : NonBritishUni
Harvard : Inst,
FUBerlin : Inst
LU : BritishUnivty
MU : BritishUnivty
Tim : Student
(**Tim, LU**) **: registered_at**
(**Bob, MU**) **: registered_at**
(**Tom, Harvard**) **: student_at.**

Формальное определение ответа на запрос к базе знаний

Говорят, что интерпретация $\mathcal{I}$ является моделью базы знаний $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ если

- $\mathcal{I}$ является моделью $\mathcal{T}$ и
- $\mathcal{I}$ является моделью $\mathcal{A}$: все имена a из $\mathcal{A}$ интерпретированы как $a^{\mathcal{I}}$ в $\Delta^{\mathcal{I}}$, $a^{\mathcal{I}} \in A^{\mathcal{I}}$ когда $a : A \in \mathcal{A}$, $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$ когда $(a, b) : R \in \mathcal{A}$.

Для данных $\mathcal{K}$ и $q = q(x_1, \ldots, x_n)$ точным ответом на q по отношению к $\mathcal{K}$,

$$\textbf{CertAnswer}(\mathcal{K}, q),$$

является множество всех $(a_1, \ldots, a_n)$ т.ч. $\mathcal{I} \models q(a_1, \ldots, a_n)$, для всех $\mathcal{I}$ — моделей $\mathcal{K}$.

Для запросов без свободных переменных q мы говорим, что

- ответ, задаваемый $\mathcal{K}$, ”Да”, если $\mathcal{I} \models q$ для всех $\mathcal{I}$ — моделей $\mathcal{K}$;
- ответ, задаваемый $\mathcal{K}$, ”Нет”, если $\mathcal{I} \not\models q$ для всех $\mathcal{I}$ — моделей $\mathcal{K}$;
- иначе, ответ “не знаю”.

Язык запросов

- Для запросов к БД используется **SQL** = логика предикатов
- Использование произвольных запросов в семантике ABox приводит к неразрешимости:

  q — произвольная замкнутая формула логики предикатов

  $\textbf{CertAnswer}(\emptyset, \mathbf{q}) =$ "Да" т. и.т.т., когда для любой $\mathcal{I}$: $\mathcal{I} \models q$.

  – Проверка принадлежности

  $$(\mathcal{T}, \mathcal{A}) \models C(a)$$

    * Обычно, решается теми же алгоритмами, что и классификация

  – (Объединение) конъюнктивных запросов

  $$q(x_1, \ldots, x_n) = \exists y_1 \ldots \exists y_m \phi,$$

  где ϕ — конъюнкция выражений $A(t)$ и $r(t, t')$.

$\mathcal{EL}$

Рассматриваем более богатый язык с номиналами:

- все имена концептов и $\top$ – $\mathcal{ELO}$ концепты
- Если C и D являются $\mathcal{ELO}$-концептами, а r — имя роли, то

$$(C \sqcap D), \quad \exists r.C$$

  являются $\mathcal{EL}$-концептами
- Если a имя индивида, то $\{a\}$ – $\mathcal{ELO}$-концепт.

$$C_{\mathcal{A}} := \bigsqcap_{C(a)\in\mathcal{A}} \exists u.(\{a\} \sqcap C) \sqcap \bigsqcap_{r(a,b)\in\mathcal{A}} \exists u.(\{a\} \sqcap \exists r\{b\}),$$

$(\mathcal{T}, \mathcal{A}) \models C(a)$ т. и т.т., когда $\mathcal{T} \models \{a\} \sqcap C_{\mathcal{A}} \sqsubseteq C$.

где u — новое имя роли.

Алгоритм для $\mathcal{ELO}$

(simpleR) If $A' \in S(A)$ and $A' \sqsubseteq B \in T$ and $B \notin S(A)$, then

$$S(A) := S(A) \cup \{B\}.$$

(conjR) If $A_1, A_2 \in S(A)$ and $A_1 \sqcap A_2 \sqsubseteq B \in T$ and $B \notin S(A)$, then

$$S(A) := S(A) \cup \{B\}.$$

(rightR) If $A' \in S(A)$ and $A' \sqsubseteq \exists r.B \in T$ and $(A, B) \notin R(r)$, then

$$R(r) := R(r) \cup \{(A, B)\}.$$

(leftR) If $(A, B) \in R(r)$ and $B' \in S(B)$ and $\exists r.B' \sqsubseteq A' \in T$ and $A' \notin S(A)$, then

$$S(A) := S(A) \cup \{A'\}.$$

(nom) If $\{a\} \in S(A) \cap S(B)$, $R^*(A, B)$ and $S(B) \not\subseteq S(A)$, then

$$S(A) := S(A) \cup S(B)$$

Логический анализ ABox в $\mathcal{ALC}$: пример

Given: Сэм живет в Германии. Сэм пьет пиво и сидр. Баварец это человек, живущий в Германии, пьет пиво и только пиво.

Q: Баварец ли Сэм?

ABox $\mathcal{A}$

$$\begin{array}{l} \text{sam: Person} \\ \text{sam: } \exists \text{livesIn.Germany} \\ \text{sam: } \exists \text{drinks.Beer} \\ (\text{sam, cider})\text{: drinks} \end{array}$$

TBox $\mathcal{T}$

$$\begin{array}{rl} \text{Bavarian} \equiv & \text{Person} \sqcap \exists \text{livesIn.Germany} \\ & \sqcap \exists \text{drinks.Beer} \sqcap \forall \text{drinks.Beer} \end{array}$$

Является ли $\boxed{\text{sam}}$ примером $\boxed{\text{Bavarian}}$?

1. Сведение к реализуемости ABox

   Сэм является баварцем т. и т.т., когда $\mathcal{A} \cup \{ \text{ sam: } \neg\text{Bavarian } \}$ не реализуется

2. Предваренная форма $\neg$Bavarian:

$$\boxed{\neg\text{Person} \sqcup \forall \text{livesIn.}\neg\text{Germany} \sqcup \forall \text{drinks.}\neg\text{Beer} \sqcup \exists \text{drinks.}\neg\text{Beer}}$$

Логический анализ ABox в $\mathcal{ALC}$: пример

$$
\begin{aligned}
S_0 \quad &= \{\ \text{sam: Person},\ \text{sam: } \exists \text{livesIn.Germany}, \\
&\qquad \text{sam: } \exists \text{drinks.Beer},\ (\text{sam, cider})\text{: drinks}, \\
&\quad \text{sam: } \neg\text{Person} \sqcup \forall \text{livesIn.} \neg \text{Germany} \\
&\qquad \sqcup\ \forall \text{drinks.} \neg \text{Beer} \sqcup \exists \text{drinks.} \neg \text{Beer}\ \} \\
S_0 \rightarrow_{\sqcup} S_{1.1} &= S_0 \cup \{\ \text{sam: } \neg \text{Person}\ \}
\end{aligned}
$$

clash

Логический анализ ABox в $\mathcal{ALC}$: пример

$$
\begin{aligned}
S_0 \quad &= \{\ \text{sam: Person},\ \ \text{sam: } \exists\text{livesIn.Germany},\\
&\qquad \text{sam: } \exists\text{drinks.Beer},\ \ (\text{sam, cider}): \text{drinks},\\
&\qquad \text{sam: } \neg\text{Person} \ \sqcup\ \forall\text{livesIn.}\neg\text{Germany}\\
&\qquad\qquad \sqcup\ \forall\text{drinks.}\neg\text{Beer} \ \sqcup\ \exists\text{drinks.}\neg\text{Beer}\ \}
\end{aligned}
$$

$S_0 \rightarrow_{\sqcup} S_{1.1} = S_0 \cup \{\ \text{sam: } \neg\text{Person}\ \}$ **clash**

$S_0 \rightarrow_{\sqcup} S_{1.2} = S_0 \cup \{\ \text{sam: } \forall\text{livesIn.}\neg\text{Germany}\ \}$

$S_{1.2} \rightarrow_{\exists} S_{2.2} = S_{1.2} \cup \{\ (\text{sam}, x): \text{livesIn},\ \ x: \text{Germany}\ \}$

$S_{2.2} \rightarrow_{\forall} S_{3.2} = S_{2.2} \cup \{\ x: \neg\text{Germany}\ \}$ **clash**

$S_0 \rightarrow_{\sqcup} S_{1.3} = S_0 \cup \{\ \text{sam: } \forall\text{drinks.}\neg\text{Beer}\ \}$

$S_{1.3} \rightarrow_{\exists} S_{2.3} = S_{1.3} \cup \{\ (\text{sam}, x): \text{drinks},\ \ x: \text{Beer}\ \}$

$S_{2.3} \rightarrow_{\forall} S_{3.3} = S_{2.3} \cup \{\ x: \neg\text{Beer}\ \}$ **clash**

$S_0 \rightarrow_{\sqcup} S_{1.4} = S_0 \cup \{\ \text{sam: } \exists\text{drinks.}\neg\text{Beer}\ \}$ (... см. след. слайд)

Логический анализ ABox в $\mathcal{ALC}$: пример

$$\begin{aligned}
S_0 \quad &= \{\ \text{sam: Person},\ \ \text{sam: } \exists\text{livesIn.Germany}, \\
&\qquad \text{sam: } \exists\text{drinks.Beer},\ \ (\text{sam, cider})\text{: drinks}, \\
&\qquad \text{sam: } \neg\text{Person} \ \sqcup\ \forall\text{livesIn.}\neg\text{Germany} \\
&\qquad\quad \sqcup\ \forall\text{drinks.}\neg\text{Beer} \ \sqcup\ \exists\text{drinks.}\neg\text{Beer}\ \} \\
S_0 \rightarrow_{\sqcup} S_{1.4} \quad &= S_0 \cup \{\ \text{sam: } \exists\text{drinks.}\neg\text{Beer}\ \} \\
S_{1.4} \rightarrow_{\exists} S_{2.4} &= S_{1.4} \cup \{\ (\text{sam}, x)\text{: drinks},\ \ x\text{: } \neg\text{Beer}\ \} \\
S_{2.4} \rightarrow_{\exists} S_{3.4} &= S_{2.4} \cup \{\ (\text{sam}, y)\text{: drinks},\ \ y\text{: Beer}\ \}
\end{aligned}$$

Логический анализ ABox в $\mathcal{ALC}$: пример

$$
\begin{aligned}
S_0 \quad &= \{\ \text{sam}: \text{Person},\ \ \text{sam}: \exists\text{livesIn}.\text{Germany}, \\
&\qquad \text{sam}: \exists\text{drinks}.\text{Beer},\ \ (\text{sam}, \text{cider}): \text{drinks}, \\
&\qquad \text{sam}: \neg\text{Person} \ \sqcup\ \forall\text{livesIn}.\neg\text{Germany} \\
&\qquad\qquad \sqcup\ \forall\text{drinks}.\neg\text{Beer}\ \sqcup\ \exists\text{drinks}.\neg\text{Beer}\ \} \\
S_0 \rightarrow_{\sqcup} S_{1.4} \quad &= S_0 \cup \{\ \text{sam}: \exists\text{drinks}.\neg\text{Beer}\ \} \\
S_{1.4} \rightarrow_{\exists} S_{2.4} &= S_{1.4} \cup \{\ (\text{sam}, x): \text{drinks},\ \ x: \neg\text{Beer}\ \} \\
S_{2.4} \rightarrow_{\exists} S_{3.4} &= S_{2.4} \cup \{\ (\text{sam}, y): \text{drinks},\ \ y: \text{Beer}\ \} \\
S_{3.4} \rightarrow_{\exists} S_{4.4} &= S_{3.4} \cup \{\ (\text{sam}, z): \text{livesIn},\ \ z: \text{Germany}\ \}
\end{aligned}
$$

$S_{4.4}$ полная непротиворечивая таблица. Т.о.,

$$\mathcal{A} \cup \{\text{sam}: \neg\text{Bavarian}\}$$

реализуем и Сэм не **не является** баварцем.

Сложность запросов к базам знаний $(\mathcal{T}, \mathcal{A})$

Для простоты, рассмотрим запросы без свободных переменных. Три способа измерить сложность.

- Сложность относительно данных: зафиксировать TBox $\mathcal{T}$ и запрос q. Сложность оценивается как функция от размера $\mathcal{A}$.
- Сложность относительно онтологии и данных: зафиксировать запрос q и оценивать сложность как функцию от размера $\mathcal{T}$ и $\mathcal{A}$.
- Комбинированная сложность: сложность оценивается как функция от размера запроса q, TBox $\mathcal{T}$, и ABox $\mathcal{A}$.

Во многих случаях, размер $\mathcal{T}$ и q много меньше размера $\mathcal{A}$. Тем не менее, иногда $\mathcal{T}$ может быть очень большим и его размером нельзя пренебрегать.

Сложность конъюнктивных запросов к $(\mathcal{T}, \mathcal{A})$

- Для $\mathcal{ALC}$ и $\mathcal{SHOIQ}$ сложность относительно данных: coNP-полная задача.

  сложность относительно онтологии и данных, а также комбинированная сложность: (N)ExpTime-полная задача.

- Для $\mathcal{EL}$: сложность относительно данных и относительно онтологии и данных: PTime.

  Комбинированная сложность (с учетом запросов): NP-полная задача.

Как улучшить?

Дескрипционные логики семейства DL-Lite: терминологическая часть

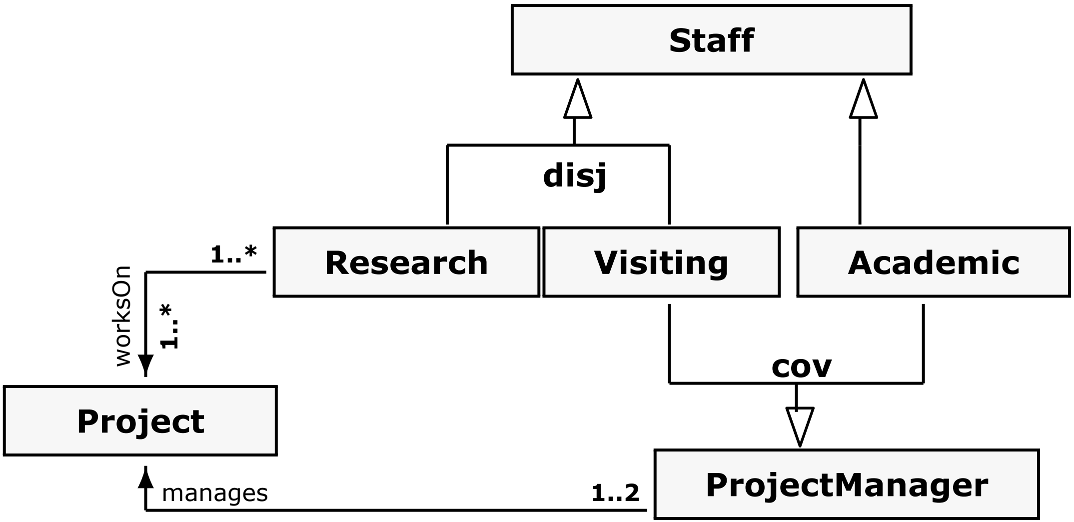
DL-Lite

Семейство дескрипционных логик DL-Lite было введено чтобы

- описать свойства стандартных концептуальных моделей таких как ER- и UML-диаграммы;
- придать им формальную семантику;
- обеспечить эффективный доступ к данным с использованием концептуальной модели описанной как DL-Lite-TBox.

Мы рассмотрим три языка: DL-Lite$_{horn}$, DL-Lite$_{bool}$ и DL-Lite$_{core}$.

UML-диаграмма



Синтаксис DL-Lite$_{horn}$

- **Язык DL-Lite$_{horn}$-концептов (классов)**
  - имена концептов $A_0, A_1, \ldots$
  - имена ролей $r_0, r_1, \ldots$
  - концепт $\top$ (“thing”)
  - концепт $\bot$ (пустой класс, “nothing”)
  - связка $\sqcap$ (пересечение, конъюнкция или просто “и”).
  - квантор $\exists$.
  - связка $\geq n$, где $n > 0$ — натуральное число.
  - связка $\cdot^-$ (обратная роль).

DL-Lite$_{horn}$

Роль это имя роли или обратное имя роли

Базовые DL-Lite концепты определяются следующим образом:

- имена концептов, $\top$ и $\bot$ являются базовыми DL-Lite концептами;
- $\exists \boldsymbol{r}.\top$ является базовым DL-Lite концептом для всех ролей $\boldsymbol{r}$;
- $(\geq \boldsymbol{n}\ \boldsymbol{r}.\top)$ является базовым DL-Lite концептом для всех ролей $\boldsymbol{r}$.

Импликация DL-Lite$_{horn}$ концептов имеет вид

$$\boldsymbol{B}_1 \sqcap \cdots \sqcap \boldsymbol{B}_n \sqsubseteq \boldsymbol{B},$$

где $\boldsymbol{B}_1, \ldots, \boldsymbol{B}_n, \boldsymbol{B}$ базовые DL-Lite концепты.

DL-Lite$_{horn}$ TBox это конечное множество импликаций DL-Lite$_{horn}$ концептов.

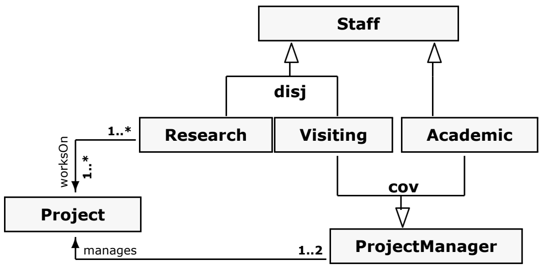
Пример

- **Person** $\sqcap$ ($\geq$ **5 hasChild.**$\top$) (человек, у которого по крайней мере 5 детей);
- **Person** $\sqcap$ ($\geq$ **7 hasChild**$^{-}$**.**$\top$) (человек, у которого по крайней мере 7 родителей (?));
- **Chair** $\sqcap$ **Table** $\sqsubseteq \bot$ (столы и стулья дизъюнктны);

- $\mathcal{EL}$-концепт **Person** $\sqcap \exists$**hasChild.Person** не выразим в DL-Lite$_{horn}$;
- $\mathcal{EL}$-концепт **Person** $\sqcap \exists$**hasChild.**$\exists$**hasChild.**$\top$ не выразим DL-Lite$_{horn}$;

- $\exists \boldsymbol{r}.\top$ эквивалентно $(\geq \mathbf{1}\ \boldsymbol{r}.\top)$.

Моделирование UML-диаграммы в DL-Lite$_{horn}$ (почти)



$$
\begin{array}{llllll}
\exists\,\text{manages}.\top & \sqsubseteq & \text{ProjectManager}, & \exists\,\text{worksOn}.\top & \sqsubseteq & \text{Research},\\
\exists\,\text{manages}^-.\top & \sqsubseteq & \text{Project}, & \exists\,\text{worksOn}^-.\top & \sqsubseteq & \text{Project},\\
\text{Project} & \sqsubseteq & \exists\,\text{manages}^-.\top, & \text{Research} & \sqsubseteq & \exists\,\text{worksOn}.\top,\\
(\geq 3\ \text{manages}^-.\top) & \sqsubseteq & \bot, & \text{Project} & \sqsubseteq & \exists\,\text{worksOn}^-.\top,\\
\text{Research} & \sqsubseteq & \text{Staff}, & \text{Visiting} & \sqsubseteq & \text{Staff},\\
\text{Research} \sqcap \text{Visiting} & \sqsubseteq & \bot, & \text{Academic} & \sqsubseteq & \text{Staff},\\
\text{Visiting} & \sqsubseteq & \text{ProjectManager}, & \text{Academic} & \sqsubseteq & \text{ProjectManager}.
\end{array}
$$

Не можем выразить

“every Projectmanager is VistingStaff or AcademicStaff”

Семантика DL-Lite$_{horn}$

- **Интерпретация** это структура $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ в которой
  - $\Delta^\mathcal{I}$ is the **носитель** (непустое множество)
  - $\cdot^\mathcal{I}$ является **интерпретирующей функцией** которая отображает:
    * каждое имя концепта A в подмножество $A^\mathcal{I}$ of $\Delta^\mathcal{I}$ $(A^\mathcal{I} \subseteq \Delta^\mathcal{I})$
    * каждое имя роли r в бинарное отношение $r^\mathcal{I}$ над $\Delta^\mathcal{I}$ $(r^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I})$
- Интерпретация обратной роли $(r^-)^\mathcal{I}$ обратна к $r^\mathcal{I}$:

$$(r^-)^\mathcal{I} = \{(d, e) \mid (e, d) \in r^\mathcal{I}\}$$

- Интерпретация $B^\mathcal{I}$ базового DL-Lite концепта B определяется следующим образом:
  - $(\top)^\mathcal{I} = \Delta^\mathcal{I}$ и $(\bot)^\mathcal{I} = \emptyset$
  - $(\geq n\ r.\top)^\mathcal{I}$ это множество $x \in \Delta^\mathcal{I}$ т.ч. число y в $\Delta^\mathcal{I}$ с $(x, y) \in r^\mathcal{I}$ не меньше n
  - $(\exists r.\top)^\mathcal{I} = \{x \in \Delta^\mathcal{I} \mid \exists y \in \Delta^\mathcal{I} \text{ such that } (x, y) \in r^\mathcal{I}\}$

Логический анализ DL-Lite$_{horn}$

- **Поглощение**. $\mathcal{T}$ — TBox а $B_1 \sqcap \cdots \sqcap B_n \sqsubseteq B$ импликация концептов. Говорят, что $B_1 \sqcap \cdots \sqcap B_n \sqsubseteq B$ **следует из** $\mathcal{T}$ т. и т.т., когда каждая модель $\mathcal{T}$ является моделью $B_1 \sqcap \cdots \sqcap B_n \sqsubseteq B$. Обозначение:
  - $\mathcal{T} \models B_1 \sqcap \cdots \sqcap B_n \sqsubseteq B$ or
  - $B_1 \sqcap \cdots \sqcap B_n \sqsubseteq_{\mathcal{T}} B$.
- **Выполнимость TBox** . TBox $\mathcal{T}$ выполним т. и т.т., когда существует модель $\mathcal{T}$.

Теорема. Существует полиномиальный алгоритм, решающий задачу поглощения концептов относительно DL-Lite$_{horn}$-TBox.

DL-Lite$_{bool}$

DL-Lite$_{bool}$ это расширение DL-Lite$_{horn}$, полученное добавлением

- связки $\sqcup$ (объединение, дизъюнкция или просто “или”)

к DL-Lite$_{horn}$.

Импликация DL-Lite$_{bool}$ концептов это выражение

$$B_1 \sqcap \cdots \sqcap B_n \sqsubseteq E_1 \sqcup \cdots \sqcup E_m,$$

где $B_1, \ldots, B_n, E_1, \ldots, E_m$ являются базовыми DL-Lite концептами.

DL-Lite$_{bool}$ TBox это конечное множество импликаций DL-Lite$_{bool}$ концептов.

Интерпретация $\mathcal{I}$:

$$(B_1 \sqcup \cdots \sqcup B_n)^{\mathcal{I}} = B_1^{\mathcal{I}} \cup \cdots \cup B_n^{\mathcal{I}}.$$

(остальное как в DL-Lite$_{horn}$).

Логический анализ в DL-Lite$_{bool}$

В DL-Lite$_{bool}$ можно выразить

every Projectmanager is VistingStaff or AcademicStaff

используя импликацию

$$\mathsf{ProjectManager} \sqsubseteq \mathsf{Academic} \sqcup \mathsf{Visiting}.$$

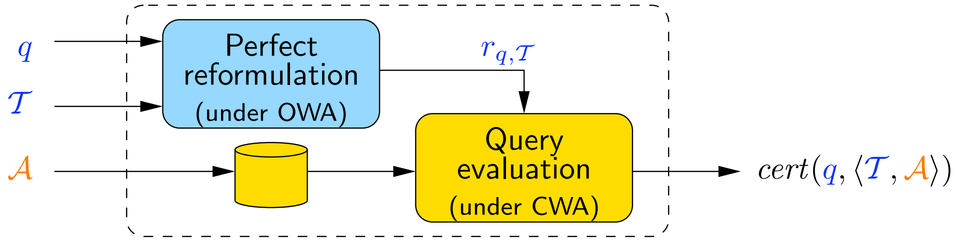
приходится платить высокую цену:

Теорема. Задача проверки выполнимости DL-Lite$_{bool}$-TBox является NP-полной.

Проверка поглощения является coNP-полной задачей.

ABox и DL-Lite$_{core}$

И снова БД



- Переписываем запрос в новый язык
- Используем базы данных для поиска всех точных ответов
- Предполагаем, что $\mathcal{T}$ выполним

DL-Lite$_{core}$

Выражения вида

Положительные включения

$$C \sqsubseteq D$$

Отрицательные включения

$$C \sqsubseteq \neg D$$

где C и D выражения вида B, $\exists s.\top$, $\exists s^{-}.\top$

Пример

из презентации Diego Calvanese

$$q(x) \leftarrow \textbf{Professor}(x)$$

$$\textbf{AssistantProfessor} \sqsubseteq \textbf{Professor}$$

Пример

из презентации Diego Calvanese

$$q(x) \leftarrow \textbf{Professor}(x)$$

$$\textbf{AssistantProfessor} \sqsubseteq \textbf{Professor}$$

$$\textbf{Professor}(x) \leftarrow \textbf{AssistantProfessor}(x)$$

Пример

из презентации Diego Calvanese

$$q(x) \leftarrow \textbf{Professor}(x)$$

$$\textbf{AssistantProfessor} \sqsubseteq \textbf{Professor}$$

$$\textbf{Professor}(x) \leftarrow \textbf{AssistantProfessor}(x)$$

Если атом в запросе унифицируется с головой правила то

заменить голову на тело

$$q'(x) \leftarrow \textbf{AssistantProfessor}(x)$$

Пример

$$q(x) \leftarrow \textbf{teaches}(x, y), \textbf{Course}(y)$$

$$\exists\textbf{teaches}.\top \sqsubseteq \textbf{Course}$$
$$\textbf{Course}(z_2) \leftarrow \textbf{teaches}(z_1, z_2)$$

Пример

$$q(x) \leftarrow \textbf{teaches}(x, y), \textbf{Course}(y)$$

$$\exists \textbf{teaches}.\top \sqsubseteq \textbf{Course}$$
$$\textbf{Course}(z_2) \leftarrow \textbf{teaches}(z_1, z_2)$$

$$q'(x) \leftarrow \textbf{teaches}(x, y), \textbf{teaches}(z_1, y)$$

Пример

$$q(x) \leftarrow \textbf{teaches}(x, y), \textbf{Course}(y)$$

$$\exists\textbf{teaches}.\top \sqsubseteq \textbf{Course}$$
$$\textbf{Course}(z_2) \leftarrow \textbf{teaches}(z_1, z_2)$$

$$q'(x) \leftarrow \textbf{teaches}(x, y), \textbf{teaches}(z_1, y)$$

Сколемизация $\exists$

$$q(x) \leftarrow \textbf{teaches}(x, y)$$

$$\textbf{Professor} \sqsubseteq \exists\textbf{teaches}.\top$$
$$\textbf{teaches}(z, f(z)) \leftarrow \textbf{Professor}(z)$$

$$q'(x) \leftarrow \textbf{Professor}(x)$$

Пример

Однако...

$$q(x) \leftarrow \textbf{teaches}(x, \textbf{databases})$$

$$\textbf{Professor} \sqsubseteq \exists \textbf{teaches}.\top$$
$$\textbf{teaches}(z, f(z)) \leftarrow \textbf{Professor}(z)$$

не унифицируется!

Пример

Однако...

$$q(x) \leftarrow \textbf{teaches}(x, \textbf{databases})$$

$$\textbf{Professor} \sqsubseteq \exists \textbf{teaches}.\top$$
$$\textbf{teaches}(z, f(z)) \leftarrow \textbf{Professor}(z)$$

не унифицируется!

С выделенными переменными обращаемся как с константами

$$q(x, \mathbf{y}) \leftarrow \textbf{teaches}(x, \mathbf{y})$$

Пример

Однако...

$$q(x) \leftarrow \textbf{teaches}(x, \textbf{databases})$$

$$\textbf{Professor} \sqsubseteq \exists \textbf{teaches}.\top$$
$$\textbf{teaches}(z, f(z)) \leftarrow \textbf{Professor}(z)$$

не унифицируется!

С выделенными переменными обращаемся как с константами

$$q(x, \mathrm{y}) \leftarrow \textbf{teaches}(x, \mathrm{y})$$

Как и с переменными, входящими в другие атомы

$$q(x) \leftarrow \textbf{teaches}(x, \mathrm{y}), (z, \mathrm{y}),$$

Пример

Редукция

$$q(x) \leftarrow \textbf{teaches}(x, \mathbf{y}), (z, \mathbf{y}),$$

(унифицируя атомы)

$$q'(x) \leftarrow \textbf{teaches}(x, y)$$

Пример

Редукция

$$q(x) \leftarrow \textbf{teaches}(x, \mathrm{y}), (z, \mathrm{y}),$$

(унифицируя атомы)

$$q'(x) \leftarrow \textbf{teaches}(x, y)$$

и тогда с помощью

$$\textbf{Professor} \sqsubseteq \exists \textbf{teaches}.\top$$
$$\textbf{teaches}(z, f(z)) \leftarrow \textbf{Professor}(z)$$

получаем

$$q'(x) \leftarrow \textbf{Professor}(x)$$

Алгоритм

```
PR := {q};
repeat
   PR′ := PR;
   for all q ∈ PR′ do
      for all g ∈ q do
         for all CI I in T do
            if I is applicable to g then
               PR := PR ∪ {q[g/(g, I)]}
            end if
         end for
         for all g1, g2 ∈ q do
            if g1 and g2 unify then
               PR := PR ∪ {(Reduce(q, g1, g2))};
            end if
         end for
      end for
   end for
until PR′ = PR
return PR
```

Пример

TBox: $\mathsf{Person} \sqsubseteq \exists \mathsf{hasFather}$
$\exists \mathsf{hasFather}^- \sqsubseteq \mathsf{Person}$

ABox: $\mathsf{Person}(\mathtt{mary})$

Query: $\mathsf{q}(x) \leftarrow \mathsf{Person}(x), \mathsf{hasFather}(x, y_1), \mathsf{hasFather}(y_1, y_2), \mathsf{hasFather}(y_2, y_3)$

$\mathsf{q}(x) \leftarrow \mathsf{Person}(x), \mathsf{hasFather}(x, y_1), \mathsf{hasFather}(y_1, y_2), \mathsf{hasFather}(y_2, \_)$

$\Downarrow$ **Apply** $\mathsf{Person} \sqsubseteq \exists \mathsf{hasFather}$ to the atom $\mathsf{hasFather}(y_2, \_)$

$\mathsf{q}(x) \leftarrow \mathsf{Person}(x), \mathsf{hasFather}(x, y_1), \mathsf{hasFather}(y_1, y_2), \mathsf{Person}(y_2)$

$\Downarrow$ **Apply** $\exists \mathsf{hasFather}^- \sqsubseteq \mathsf{Person}$ to the atom $\mathsf{Person}(y_2)$

$\mathsf{q}(x) \leftarrow \mathsf{Person}(x), \mathsf{hasFather}(x, y_1), \mathsf{hasFather}(y_1, y_2), \mathsf{hasFather}(\_, y_2)$

$\Downarrow$ **Unify** atoms $\mathsf{hasFather}(y_1, y_2)$ and $\mathsf{hasFather}(\_, y_2)$

$\mathsf{q}(x) \leftarrow \mathsf{Person}(x), \mathsf{hasFather}(x, y_1), \mathsf{hasFather}(y_1, y_2)$

$\Downarrow$

$\ldots$

$\mathsf{q}(x) \leftarrow \mathsf{Person}(x), \mathsf{hasFather}(x, \_)$

$\Downarrow$ **Apply** $\mathsf{Person} \sqsubseteq \exists \mathsf{hasFather}$ to the atom $\mathsf{hasFather}(x, \_)$

$\mathsf{q}(x) \leftarrow \mathsf{Person}(x)$

Сложность конъюнктивных запросов к $(\mathcal{T}, \mathcal{A})$

- Для DL-Lite$_{core}$, DL-Lite$_{horn}$ и некоторых других логик семейства DL-Lite, сложность относительно данных LogSpace-полна

При условии уникальности имен

для любых $a, b \in \mathcal{A}$ мы имеем $a^{\mathcal{I}} \neq b^{\mathcal{I}}$

- **Без условия уникальности имен**: co-NP

Язык описания онтологий OWL

OWL (Язык описания Web-онтологий)

OWL это язык описания онтологий, принятый в качестве стандарта для Web (точнее, Semantic Web). Используется и для других приложений.

- OWL 1 рекомендован W3C с февраля 2004 года.
- OWL 2 новый стандарт, рекомендован с октября 2009.

Описание

Из описания OWL1: The OWL Web Ontology Language is designed for use by applications that need to process the content of information instead of just presenting information to humans. OWL facilitates greater machine interpretability of Web content than that supported by XML, RDF, and RDF Schema (RDF-S) by providing additional vocabulary along with a formal semantics.

OWL has three increasingly-expressive sublanguages: **OWL Lite**, **OWL DL**, and **OWL Full**.

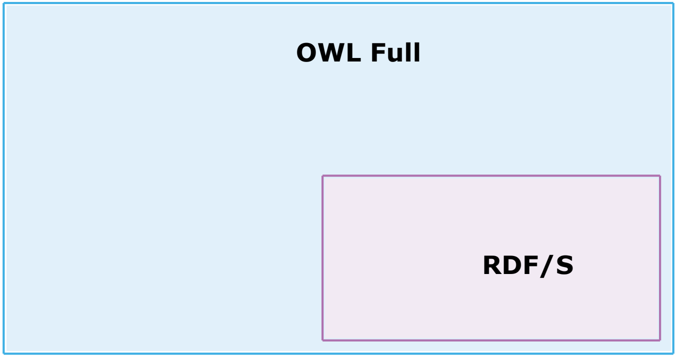
Из описания OWL2: The OWL 2 Web Ontology Language, informally OWL 2, is an ontology language for the Semantic Web with formally defined meaning. OWL 2 ontologies provide classes, properties, individuals, and data values and are stored as Semantic Web documents. OWL 2 ontologies can be used along with information written in RDF, and OWL 2 ontologies themselves are primarily exchanged as RDF documents.

Три вида OWL

- **OWL Full**
  - Очень выразителен
  - Нестандартная семантика (наследие RDF)
  - Полностью совместим с RDF (синтаксически и семантически):
    - любой RDF-документ является OWL Full-документом
    - любое истинное RDF/S-утверждение является истинным OWL Full-утверждением
  - **неразрешим** (нет полных (и эффективных) процедур автоматического анализа)
- **OWL DL**
  - Подмножество OWL Full, соответствует $\mathcal{SHOIN}$ с XML типами данных;
    * OWL 2 DL соответствует $\mathcal{SROIQ}$
  - эффективные (но не полиномиальные) процедуры анализа;
- **OWL Lite**
  - Подмножество OWL 1 DL, соответствующее $\mathcal{SHIN}$ (без номиналов) и без XML типов данных;
  - Высокая вычислительная сложность
  - Не так важен (2009) как в 2004.

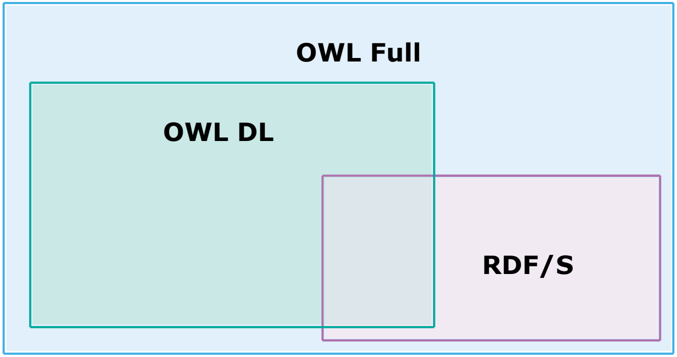
Три вида OWL

Совместимость между видами OWL и RDF/S



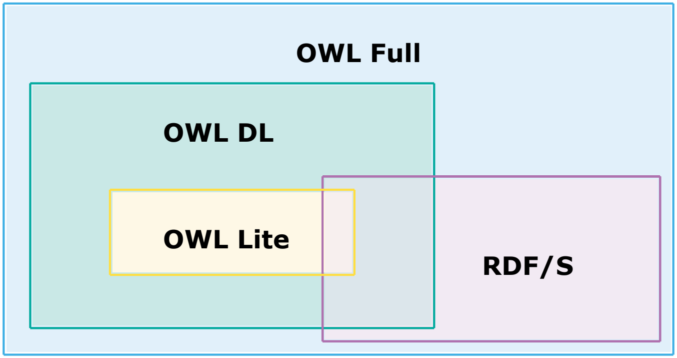
Три вида OWL

Совместимость между видами OWL и RDF/S



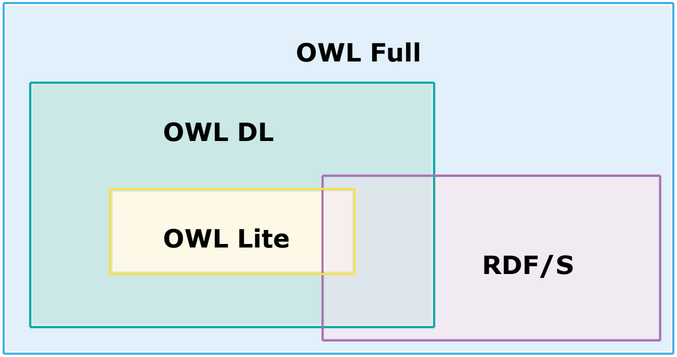
Три вида OWL

Совместимость между видами OWL и RDF/S



Три вида OWL

Совместимость между видами OWL и RDF/S



- В каком-то смысле, OWL это надстройка над RDF/S:
  - Все виды OWL поддерживают синтаксис на основе RDF
  - OWL-конструкции, такие как **owl:Class**, **owl:DatatypeProperty** and **owl:ObjectProperty** являются специализацией RDF/S-аналогов

Профили OWL 2

Профили OWL 2 это подмножества OWL 2, имеющие определенные преимущества для конкретных приложений.

- OWL 2 EL основан на $\mathcal{EL}$. Полиномиальный алгоритм логического анализа.
- OWL 2 QL Эффективные запросы к БД, основан на семействе DL-Lite
- OWL 2 RL Правила (продукции) для RDF

Синтаксис языка OWL

Несколько синтаксических форм

- **Основанный на RDF (XML)** синтаксис (основной)
- **Основанный на XML** синтаксис, не следует соглашениям RDF

  (легче понимать) http://www.w3.org/TR/owl-xmlsyntax/
- **Абстрактный** синтаксис

  (гораздо легче понимать см. http://www.w3.org/TR/owl-semantics/
- Графическое представление, основанное на **UML**

  (Unified Modelling Language)
- ... (?)

Заголовок XML/RDF

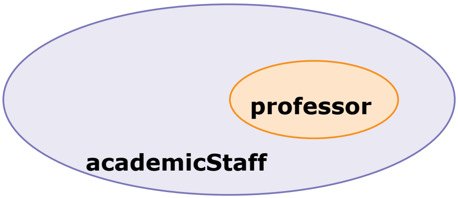
```
<rdf:RDF xmlns:owl="http://www.w3.org/2002/07/owl#"
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
        xml:base="http://www.dcs.bbk.ac.uk/">

  <owl:Ontology rdf:about="">
    <rdfs:comment>An example OWL ontology</rdfs:comment>
    <owl:priorVersion rdf:resource="http://www.dcs.bbk.ac.uk/uni-old-ns"/>
    <owl:imports rdf:resource="http://www.dcs.bbk.ac.uk/person"/>
    <rdfs:label>SCSIS Ontology</rdfs:label>
  </owl:Ontology>

  ...

</rdf:RDF>
```

Классы в XML/RDF

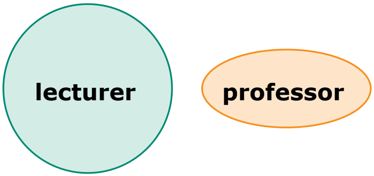
Классы (концепты) определяются при помощи **owl:Class**

(**owl:Class** подкласс **rdfs:Class**)

```
<owl:Class rdf:ID="professor">
  <rdfs:subClassOf rdf:resource="#academicStaff"/>
</owl:Class>
```




```
<owl:Class rdf:about="#professor">
  <owl:disjointWith rdf:resource="#lecturer"/>
</owl:Class>
```



OWL: абстрактный синтаксис

Подробности: http://www.w3.org/TR/owl-semantics/syntax.html#2.3.2.1

Классы: примитивные или определяемые

описания

Class(*name* partial ...)

‘**all** *name* ...’

примитивные концепты

$\sqsubseteq$

определения

Class(*name* complete ...)

‘a *name* is **anything** that ...’

определяемые концепты

$\equiv$

Пример:

Class(MargheritaPizza **partial**
  Pizza
  restriction(*hasTopping*
    someValuesFrom(Mozzarella))
  restriction(*hasTopping*
    someValuesFrom(Tomato)))

‘All Margherita pizzas have, amongst other things, some mozzarella topping and also some tomato topping’

Class(CheesyPizza **complete**
  Pizza
  restriction(*hasTopping*
    someValuesFrom(Cheese)))

‘A cheesy pizza is any pizza that has, amongst other things, some cheese topping’

Классы: дизъюнктность

В OWL классы могут пересекаться если только **не объявлены дизъюнктными**:

DisjointClasses(*class*$_1$ … *class*$_n$)

Пример:

```
DisjointClasses(
        Vegetable Meat Seafood Cheese)
```

PizzaTopping

- Vegetable
  - Tomato
  - Pepper
  - Mushroom
- Meat
  - SpicyBeef
  - Pepperoni
- Seafood
  - Tuna
  - Prawn
  - Anchovy
- Cheese
  - Mozzarella
  - Parmesan

Кванторы

экзистенциальный

restriction(*prop*
someValuesFrom(*class*))

'some', 'at least one'

$\exists$

универсальный

restriction(*prop*
allValuesFrom(*class*))

'only', 'no value except'

$\forall$

Пример:

Class(DogOwner complete
Person
restriction(hasPet
someValuesFrom(Dog)))

'A dog owner is any person who
has as a pet some dog'

Class(FirstClassLounge complete
Lounge
restriction(hasOccupants
allValuesFrom(FirstCPassenger)))

'A first class lounge is any lounge where
the occupants are
only first class passengers'

'A first class lounge is any lounge where
there are no occupants except
first class passengers'

Кванторы

экзистенциальный	универсальный
$\exists$	$\forall$

Пример:

```
Class(DogOwner partial
   Person
   restriction(hasPet
               someValuesFrom(Dog)))
```

‘Dog owners are people
and have as a pet some dog’

```
Class(FirstClassLounge partial
   Lounge
   restriction(hasOccupants
               allValuesFrom(FirstCPassenger)))
```

‘All first class lounges have
only occupants who are
first class passengers’

‘All first class lounges
have no occupants except
first class passengers’

‘All first class lounges
have no occupants who are
not first class passengers’

Логические связки

объединение (дизьюнкция)

unionOf(*class*$_1$ …*class*$_n$)

'*class*$_1$ **and/or** *class*$_2$'

$\sqcup$

пересечение (конъюнкция)

intersectionOf(*class*$_1$ …*class*$_n$)

'**both** *class*$_1$ **and also** *class*$_2$'

$\sqcap$

Пример:

```
Class(VegetarianPizza complete
  Pizza
  restriction(hasTopping
     allValuesFrom(
            unionOf(Vegetable Cheese))))
```

'A vegetarian pizza is any pizza which, amongst other things, has only vegetable and/or cheese toppings'

```
Class(ProteinLoversPizza complete
  Pizza
  restriction(hasTopping
     allValuesFrom(
            intersectionOf(Meat Seafood))))
```

'A protein lover's pizza is any pizza that, amongst other things, has only toppings that are both meat and also seafood'

NO topping is both meat and also seafood **!**

(therefore, the intersection is empty)

Отрицание

complementOf(*class*) $\neg$

- **complementOf(intersectionOf(*class*$_1$ *class*$_2$))**
  — 'not all of' / 'not both *class*$_1$ and also *class*$_2$'
- **complementOf(unionOf(*class*$_1$ *class*$_2$))** — 'neither *class*$_1$ nor *class*$_2$'
- **restriction(*prop* someValuesFrom(complementOf(*class*)))**
  — 'has some *prop* that are not *class*'
- **complementOf(restriction(*prop* someValuesFrom(*class*)))**
  — 'does not have any *prop* that are *class*'
- **restriction(*prop* allValuesFrom(complementOf(*class*)))**
  — 'has *prop* no *class*' / 'has only *prop* that are not *class*'
- **complementOf(restriction(*prop* allValuesFrom(*class*)))**
  — 'does not have only *prop* that are *class*'

Мощностные ограничения

restriction(*prop*
minCardinality(*n*))

'**at least** *n* (distinct) *prop*'

$\geq$

restriction(*prop*
maxCardinality(*n*))

'**at most** *n* (distinct) *prop*'

$\leq$

Пример:

Class(InterestingPizza complete
  Pizza
  restriction(hasTopping
    minCardinality(3)))

'An interesting pizza is any pizza that,
amongst other things, has
at least 3 (distinct) toppings'

Class(Pizza partial
  restriction(hasBase
    maxCardinality(1)))

'Any pizza, amongst other things,
has at most 1 pizza base'

Домен / область значений

ObjectProperty(*name* … domain(*classD*) range(*classR*))

range

Class(**owl:Thing** partial
restriction(*name*
allValuesFrom(*classR*)))

'All things have no *name* except *classR*'

domain

SubClassOf(restriction(*name*
someValuesFrom(owl:Thing))
classD)

'Having a *name* implies being *classD*'

Пример

```
ObjectProperty(hasTopping
            domain(Pizza))
```

‘Having a topping implies being pizza’

Рожки для мороженного:

```
Class(IceCreamCone partial
     restriction(hasTopping
            someValuesFrom(IceCream)))
```

‘All ice-cream cones,
amongst other things,
have some ice-cream topping’

OWL как ДЛ: Классы

A	A
owl:Thing	$\top$
owl:Nothing	$\bot$

OWL как ДЛ: Классы

A	A
owl:Thing	$\top$
owl:Nothing	$\bot$
intersectionOf$(C_1\ C_2\ ...C_n)$	$C_1 \sqcap C_2 \sqcap \ldots \sqcap C_n$
unionOf$(C_1\ C_2\ ...C_n)$	$C_1 \sqcup C_2 \sqcup \ldots \sqcup C_n$
complementOf(C)	$\neg C$

OWL как ДЛ: Классы

A	A
owl:Thing	$\top$
owl:Nothing	$\bot$
intersectionOf$(C_1\ C_2\ ...C_n)$	$C_1 \sqcap C_2 \sqcap \ldots \sqcap C_n$
unionOf$(C_1\ C_2\ ...C_n)$	$C_1 \sqcup C_2 \sqcup \ldots \sqcup C_n$
complementOf(C)	$\neg C$
oneOf$(a_1\ a_2\ ...a_n)$	$\{a_1\} \sqcup \{a_2\} \sqcup \ldots \sqcup \{a_n\}$

OWL как ДЛ: Классы

A	A
owl:Thing	$\top$
owl:Nothing	$\bot$
intersectionOf$(C_1\ C_2\ ...C_n)$	$C_1 \sqcap C_2 \sqcap \ldots \sqcap C_n$
unionOf$(C_1\ C_2\ ...C_n)$	$C_1 \sqcup C_2 \sqcup \ldots \sqcup C_n$
complementOf(C)	$\neg C$
oneOf$(a_1\ a_2\ ...a_n)$	$\{a_1\} \sqcup \{a_2\} \sqcup \ldots \sqcup \{a_n\}$
restriction$(R\ ...)$	
allValuesFrom(C)	$\forall R.C$
someValuesFrom(C)	$\exists R.C$

OWL как ДЛ: Классы

A	A
owl:Thing	$\top$
owl:Nothing	$\bot$
intersectionOf$(C_1\ C_2\ ...C_n)$	$C_1 \sqcap C_2 \sqcap \ldots \sqcap C_n$
unionOf$(C_1\ C_2\ ...C_n)$	$C_1 \sqcup C_2 \sqcup \ldots \sqcup C_n$
complementOf(C)	$\neg C$
oneOf$(a_1\ a_2\ ...a_n)$	$\{a_1\} \sqcup \{a_2\} \sqcup \ldots \sqcup \{a_n\}$
restriction$(R\ ...)$	
allValuesFrom(C)	$\forall R.C$
someValuesFrom(C)	$\exists R.C$
minCardinality(n)	$\geq n\ R$
maxCardinality(n)	$\leq n\ R$

OWL как ДЛ: Классы

A	A
owl:Thing	$\top$
owl:Nothing	$\bot$
intersectionOf$(C_1\ C_2\ ...C_n)$	$C_1 \sqcap C_2 \sqcap \ldots \sqcap C_n$
unionOf$(C_1\ C_2\ ...C_n)$	$C_1 \sqcup C_2 \sqcup \ldots \sqcup C_n$
complementOf(C)	$\neg C$
oneOf$(a_1\ a_2\ ...a_n)$	$\{a_1\} \sqcup \{a_2\} \sqcup \ldots \sqcup \{a_n\}$
restriction$(R\ ...)$	
allValuesFrom(C)	$\forall R.C$
someValuesFrom(C)	$\exists R.C$
minCardinality(n)	$\geq n\ R$
maxCardinality(n)	$\leq n\ R$
value(a)	$\exists R.\{a\}$

OWL как ДЛ: Классы

Class(A **partial** C_1 C_2 ...C_n) $$A \sqsubseteq C_1 \sqcap C_2 \sqcap \ldots \sqcap C_n$$

Class(A **complete** C_1 C_2 ...C_n) $$A \equiv C_1 \sqcap C_2 \sqcap \ldots \sqcap C_n$$

OWL как ДЛ: Классы

Class$(A$ **partial** $C_1\ C_2\ ...C_n)$	$A \sqsubseteq C_1 \sqcap C_2 \sqcap \ldots \sqcap C_n$
Class$(A$ **complete** $C_1\ C_2\ ...C_n)$	$A \equiv C_1 \sqcap C_2 \sqcap \ldots \sqcap C_n$
SubClassOf$(C\ \ D)$	$C \sqsubseteq D$
EquivalentClasses$(C_1\ C_2\ ...C_n)$	$C_1 \equiv C_2,\quad C_2 \equiv C_3,\ \ \ldots,\ C_{n-1} \equiv C_n$

OWL как ДЛ: Классы

Class(A **partial** $C_1\ C_2\ ...C_n$) $\quad A \sqsubseteq C_1 \sqcap C_2 \sqcap \ldots \sqcap C_n$

Class(A **complete** $C_1\ C_2\ ...C_n$) $\quad A \equiv C_1 \sqcap C_2 \sqcap \ldots \sqcap C_n$

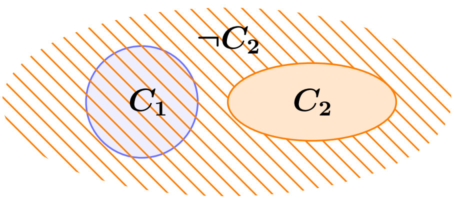
SubClassOf($C\ \ D$) $\quad C \sqsubseteq D$

EquivalentClasses($C_1\ C_2\ ...C_n$) $\quad C_1 \equiv C_2,\ \ C_2 \equiv C_3,\ \ \ldots,\ \ C_{n-1} \equiv C_n$

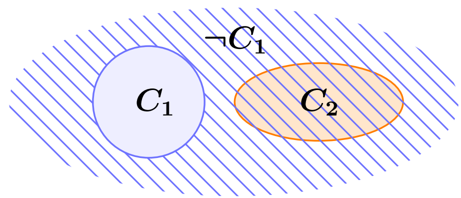
DisjointClasses($C_1\ C_2\ ...C_n$)

$$C_1 \sqsubseteq \neg C_2,\ C_1 \sqsubseteq \neg C_3,\ ...,\ C_1 \sqsubseteq \neg C_n$$
$$C_2 \sqsubseteq \neg C_3,\ ...,\ C_2 \sqsubseteq \neg C_n$$
$$...$$
$$C_{n-1} \sqsubseteq \neg C_n$$

Пример: C_1 и C_2 **дизъюнктны**:



$C_1 \sqsubseteq \neg C_2$ то же что



$C_2 \sqsubseteq \neg C_1$

OWL как ДЛ: свойства

SubPropertyOf$(R\ \ S)$	$R \sqsubseteq S$
EquivalentProperty$(R\ \ S)$	$R \equiv S$

OWL как ДЛ: свойства

SubPropertyOf($R\ \ S$) $\qquad R \sqsubseteq S$

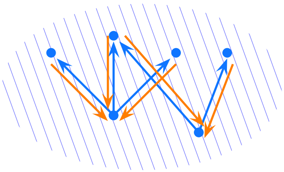
EquivalentProperty($R\ \ S$) $\qquad R \equiv S$

ObjectProperty($R\ \ \ldots$)

  super(S) $\qquad R \sqsubseteq S$

OWL как ДЛ: свойства

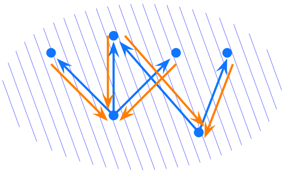
свойство R обратно к R^{-}



SubPropertyOf$(R\ \ S)$	$R \sqsubseteq S$
EquivalentProperty$(R\ \ S)$	$R \equiv S$
ObjectProperty$(R\ \ \ldots)$	
super(S)	$R \sqsubseteq S$
inverseOf(S)	$R \equiv S^{-}$

OWL как ДЛ: свойства

свойство R обратно к R^-



SubPropertyOf(R S) $R \sqsubseteq S$

EquivalentProperty(R S) $R \equiv S$

ObjectProperty(R ...)

super(S) $R \sqsubseteq S$

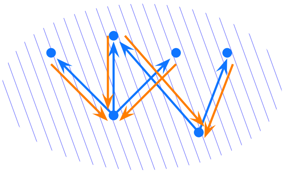
inverseOf(S) $R \equiv S^-$

Transitive *transitive*(R)

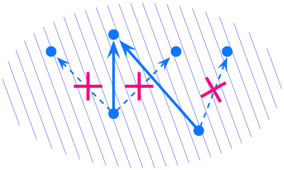
OWL как ДЛ: свойства

SubPropertyOf$(R\ \ S)$	$R \sqsubseteq S$
EquivalentProperty$(R\ \ S)$	$R \equiv S$
ObjectProperty$(R\ \ \ldots)$	
super(S)	$R \sqsubseteq S$
inverseOf(S)	$R \equiv S^{-}$
Transitive	*transitive*(R)
Functional	$\top \ \sqsubseteq\ \leq 1\ R$
InverseFunctional	$\top \ \sqsubseteq\ \leq 1\ R^{-}$

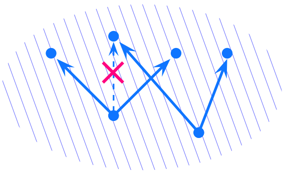
свойство R обратно к R^{-}



R функционально



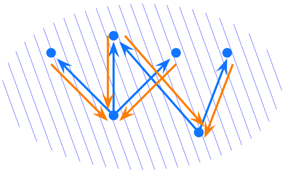
R обратная функция



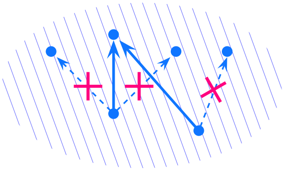
OWL как ДЛ: свойства

SubPropertyOf$(R\ \ S)$	$R \sqsubseteq S$
EquivalentProperty$(R\ \ S)$	$R \equiv S$
ObjectProperty$(R\ \ \ldots)$	
super(S)	$R \sqsubseteq S$
inverseOf(S)	$R \equiv S^{-}$
Transitive	*transitive*(R)
Functional	$\top \ \sqsubseteq\ \leq 1\ R$
InverseFunctional	$\top \ \sqsubseteq\ \leq 1\ R^{-}$
Symmetric	$R^{-} \sqsubseteq R$

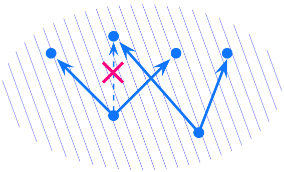
свойство R обратно к R^{-}



R функционально



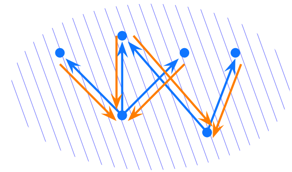
R обратная функция



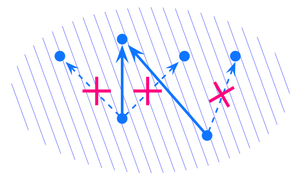
OWL как ДЛ: свойства

SubPropertyOf$(R \;\; S)$	$R \sqsubseteq S$
EquivalentProperty$(R \;\; S)$	$R \equiv S$
ObjectProperty$(R \;\; \ldots)$	
super(S)	$R \sqsubseteq S$
inverseOf(S)	$R \equiv S^{-}$
Transitive	*transitive*(R)
Functional	$\top \;\sqsubseteq\; \leq 1\, R$
InverseFunctional	$\top \;\sqsubseteq\; \leq 1\, R^{-}$
Symmetric	$R^{-} \sqsubseteq R$
range(C)	$\top \;\sqsubseteq\; \forall R.C$
domain(D)	$\exists R.\top \;\sqsubseteq\; D$

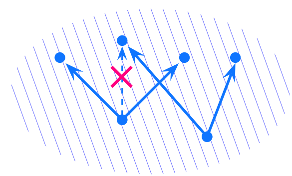
свойство R обратно к R^{-}



R функционально



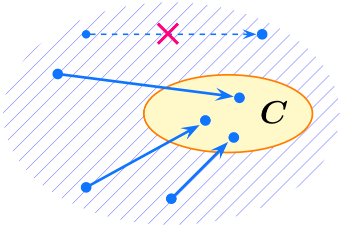
R обратная функция



OWL как ДЛ: ограничения домена и области значений

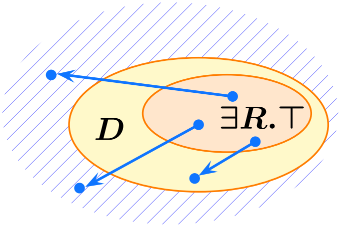
ObjectProperty$(R$ **range**$(C))$

$$\top \sqsubseteq \forall \boldsymbol{R}.\boldsymbol{C}$$



ObjectProperty$(R$ **domain**$(D))$

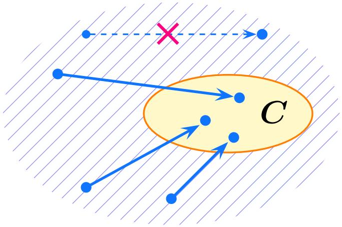
$$\exists \boldsymbol{R}.\top \sqsubseteq \boldsymbol{D}$$



OWL как ДЛ: ограничения домена и области значений

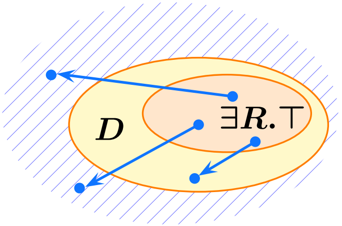
ObjectProperty$(R$ **range**$(C))$

$$\top \sqsubseteq \forall R.C$$



ObjectProperty$(R$ **domain**$(D))$

$$\exists R.\top \sqsubseteq D$$



NB: способ записать

ограничения области значений:

$$\exists R^{-}.\top \sqsubseteq C$$

ограничения домена:

$$\top \sqsubseteq \forall R^{-}.D$$

OWL как ДЛ: индивиды

DifferentIndividuals($a_1\ a_2\ ...a_n$)

$$a_1 : \neg\{a_2, a_3, \ldots, a_n\}$$
$$a_2 : \neg\{a_3, \ldots, a_n\}$$
...
$$a_{n-1} : \neg\{a_n\}$$

SameIndividuals($a_1\ a_2\ ...a_n$)

$$a_1 : \{a_2\}$$
$$a_2 : \{a_3\}$$
$$\ldots$$
$$a_{n-1} : \{a_n\}$$

OWL как ДЛ: индивиды

DifferentIndividuals(a_1 a_2 …a_n)

$$a_1 \colon \neg\{a_2, a_3, \ldots, a_n\}$$
$$a_2 \colon \neg\{a_3, \ldots, a_n\}$$
…
$$a_{n-1} \colon \neg\{a_n\}$$

SameIndividuals(a_1 a_2 …a_n)

$$a_1 \colon \{a_2\}$$
$$a_2 \colon \{a_3\}$$
$$\ldots$$
$$a_{n-1} \colon \{a_n\}$$

Предположение об уникальности имен (UNA) утверждает что

любые два индивида с разными именами интерпретируются по-разному

OWL как ДЛ: индивиды

DifferentIndividuals($a_1\ a_2 \ldots a_n$)

$$a_1 \colon \neg\{a_2, a_3, \ldots, a_n\}$$
$$a_2 \colon \neg\{a_3, \ldots, a_n\}$$
…
$$a_{n-1} \colon \neg\{a_n\}$$

SameIndividuals($a_1\ a_2 \ldots a_n$)

$$a_1 \colon \{a_2\}$$
$$a_2 \colon \{a_3\}$$
$$\ldots$$
$$a_{n-1} \colon \{a_n\}$$

Предположение об уникальности имен (UNA) утверждает что

любые два индивида с разными именами интерпретируются по-разному

OWL и DL не делают UNA

Пример: $\mathcal{T} = \{\top \sqsubseteq\ \leq 1R\}$ и $\mathcal{A} = \{(a,b)\colon R,\ (a,c)\colon R\}$

Совместен ли $\mathcal{A}$ с $\mathcal{T}$?

OWL как ДЛ: индивиды

DifferentIndividuals(a_1 a_2 …a_n)

$$a_1 \colon \neg\{a_2, a_3, \ldots, a_n\}$$
$$a_2 \colon \neg\{a_3, \ldots, a_n\}$$
…
$$a_{n-1} \colon \neg\{a_n\}$$

SameIndividuals(a_1 a_2 …a_n)

$$a_1 \colon \{a_2\}$$
$$a_2 \colon \{a_3\}$$
$$\ldots$$
$$a_{n-1} \colon \{a_n\}$$

Предположение об уникальности имен (UNA) утверждает что

любые два индивида с разными именами интерпретируются по-разному

OWL и DL не делают UNA

Пример: $\mathcal{T} = \{\top \sqsubseteq \leq 1 R\}$ и $\mathcal{A} = \{(a, b)\colon R,\ (a, c)\colon R\}$

Совместен ли $\mathcal{A}$ с $\mathcal{T}$?

ДА! рассмотрим $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, где $\Delta^{\mathcal{I}} = \{x, y\}$, $R^{\mathcal{I}} = \{(x, y)\}$,

$a^{\mathcal{I}} = x$, $b^{\mathcal{I}} = y$ и $c^{\mathcal{I}} = y$

От ДЛ к OWL

ДЛ аксиомы могут быть представлены разными способами

$$\boldsymbol{A \sqsubseteq B} \qquad (\boldsymbol{A} \text{ и } \boldsymbol{B} \text{ имена концептов})$$

может быть записано

1. **Class**(A **partial** B)

От ДЛ к OWL

ДЛ аксиомы могут быть представлены разными способами

$$A \sqsubseteq B$$ (A и B имена концептов)

может быть записано

1. **Class**(A **partial** B)
2. **SubClassOf**(A B)

От ДЛ к OWL

ДЛ аксиомы могут быть представлены разными способами

$A \sqsubseteq B$ (A и B имена концептов)

может быть записано

1. **Class**(A **partial** B)
2. **SubClassOf**(A B)
3. **DisjointClasses**(A **complementOf**(B))

От ДЛ к OWL

ДЛ аксиомы могут быть представлены разными способами

$$\boldsymbol{A \sqsubseteq B}$$ ($\boldsymbol{A}$ и $\boldsymbol{B}$ имена концептов)

может быть записано

1. **Class**(A **partial** B)
2. **SubClassOf**(A B)
3. **DisjointClasses**(A **complementOf**(B))

$$\boldsymbol{\top \sqsubseteq\ \leq 1\, R}$$ ($\boldsymbol{R}$ — имя роли)

может быть записано

1. **ObjectProperty**(R **functional**)

От ДЛ к OWL

ДЛ аксиомы могут быть представлены разными способами

$\boldsymbol{A \sqsubseteq B}$ (${\boldsymbol A}$ и ${\boldsymbol B}$ имена концептов)

может быть записано

1. **Class**(A **partial** B)
2. **SubClassOf**(A B)
3. **DisjointClasses**(A **complementOf**(B))

$\top \sqsubseteq \leq 1\, \boldsymbol{R}$ ($\boldsymbol{R}$ — имя роли)

может быть записано

1. **ObjectProperty**(R **functional**)
2. **SubClassOf**(**owl:Thing** **restriction**(R **maxCardinality**(1)))

От ДЛ к OWL

ДЛ аксиомы могут быть представлены разными способами

$\boldsymbol{A \sqsubseteq B}$ $\quad$ ($\boldsymbol{A}$ и $\boldsymbol{B}$ имена концептов)

может быть записано

1. **Class**(A **partial** B)
2. **SubClassOf**(A B)
3. **DisjointClasses**(A **complementOf**(B))

$\boldsymbol{\top \sqsubseteq \leq 1\, R}$ $\quad$ ($\boldsymbol{R}$ — имя роли)

может быть записано

1. **ObjectProperty**(R **functional**)
2. **SubClassOf**(**owl:Thing** **restriction**(R **maxCardinality**(1)))
3. **DisjointClasses**(**owl:Thing**
   complementOf(**restriction**(R **maxCardinality**(1))))