First-order Predicate Logic (FOPL) as an Ontology Language

Ontology Languages Based on First-order predicate Logic (FOPL)

The following standardized languages are based on FOPL:

- Common Logic
- CycL
- KIF

They are typically much more expressive than description logics and reasoning is typically undecidable. There are many upper level ontologies (domain independent ontologies that fix concepts across domains) formulated in FOPL. An example is DOLCE (Descriptive Ontology for Linguistic and Cognitive Engineering).

DOLCE

DOLCE defines general concepts such as 'participation', 'dependence', 'constitution' in an extension of FOPL.

For example, DOLCE contains axioms that fix the meaning of concepts related to 'parthood': Assume $\mathbf{P}(x, y)$ says that x is a part of y. Then

- $\mathbf{PP}(x, y) = \mathbf{P}(x, y) \wedge \neg\mathbf{P}(y, x)$ defines proper parthood;
- $\mathbf{O}(x, y) = \exists z(\mathbf{P}(z, x) \wedge \mathbf{P}(z, y))$ defines overlap;
- $\mathbf{AT}(x) = \neg\exists y\mathbf{PP}(y, x)$ defines atoms.

Syntax of FOPL: signature

A signature for FOPL consists of

- a set $\mathcal{V}$ of **variables** denoted by $x, x_0, x_1, \ldots, y, y_0, y_1, \ldots, z, z_0, z_1 \ldots$;
- a set $\mathcal{P}$ of **predicate symbols** each of which comes with a positive number as its arity. Predicates of arity one are called unary predicates and are often denoted by $A, A_0, A_1 \ldots, B, B_0, B_1, \ldots$ In description logic, they correspond to **concept names**, so we sometimes call them concept names. Predicates of arity two are called binary predicates and are often denoted by $r, r_0, r_1, \ldots, s, s_0, s_1, \ldots$. In description logic, they correspond to **roles**, so we sometimes call them roles.
- a "special" binary relation symbols "$=$".

Syntax of FOPL: formulas

Formulas of FOPL are defined by induction:

- If P is a predicate of arity k and $x_1, \ldots, x_k$ are terms, then $P(x_1, \ldots, x_k)$ is a formula;
- in particular, if x_1, x_2 are terms, then $x_1 = x_2$ is a formula;
- If F is a formula, then $\neg F$ is a formula;
- If F and G are formulas, then $F \wedge G$ and $F \vee G$ are formulas;
- If F is a formula and x a variable, then $\forall x.F$ and $\exists x.F$ are formulas.

We abbreviate

- $F_1 \rightarrow F_2 = \neg F_1 \vee F_2$;
- $F_1 \leftrightarrow F_2 = (F_1 \rightarrow F_2) \wedge (F_2 \rightarrow F_1)$.

Translation of concept inclusions into FOPL: idea

- $\textbf{Father} \equiv \textbf{Person} \sqcap \textbf{Male} \sqcap \exists \textbf{hasChild}.\top$.
- $\textbf{Student} \equiv \textbf{Person} \sqcap \exists \textbf{is\_registered\_at}.\textbf{University}$.
- $\textbf{Father} \sqsubseteq \textbf{Person}$.
- $\textbf{Father} \sqsubseteq \exists \textbf{hasChild}.\top$.

are translated to

- $\forall x.(\textbf{Father}(x) \leftrightarrow (\textbf{Person}(x) \wedge \textbf{Male}(x) \wedge \exists y.\textbf{hasChild}(x, y)))$.
- $\forall x.(\textbf{Student}(x) \leftrightarrow (\textbf{Person}(x) \wedge \exists y.(\textbf{is\_registered\_at}(x, y) \wedge \textbf{University}(y))))$.
- $\forall x.(\textbf{Father}(x) \rightarrow \textbf{Person}(x))$.
- $\forall x.(\textbf{Father}(x) \rightarrow \exists y.\textbf{hasChild}(x, y))$.

Semantics of FOPL: interpretation

An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ for FOPL consists of

- a non-empty set $\Delta^{\mathcal{I}}$ (the domain);
- an interpretation function that maps
  - every k-ary predicate symbol P to a k-ary relation over $\Delta^{\mathcal{I}}$. In particular,
    - every unary predicate symbol A is mapped to a subset $A^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$ and
    - every binary predicate symbol r is a mapped to a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$.

This is almost the same definition as for description logics. The only difference is that we have predicate symbols of arity larger than 2.

Example

Consider

- the concept names **University**, **BritishUniversity**, **Student**,
- the role name **student_at**.

An interpretation $\mathcal{I}_U$ is given by

- $\Delta^{\mathcal{I}_U} = \{\textsf{LU}, \textsf{MU}, \textsf{CMU}, \textsf{Tim}, \textsf{Tom}, \textsf{Rob}, a, b, c\}$.
- $\textsf{University}^{\mathcal{I}_U} = \{\textsf{LU}, \textsf{MU}, \textsf{CMU}, a, b\}$;
- $\textsf{BritishUniversity}^{\mathcal{I}_U} = \{\textsf{LU}, \textsf{MU}\}$;
- $\textsf{Student}^{\mathcal{I}_U} = \{\textsf{Tim}, \textsf{Tom}, c\}$;
- $\textsf{student\_at}^{\mathcal{I}_U} = \{(\textsf{Tim}, \textsf{LU}), (\textsf{Rob}, \textsf{CMU})\}$.

Semantics of FOPL: Satisfaction

A **variable assignment** α for an interpretation $\mathcal{I}$ is a function that maps every variable $x \in \mathcal{V}$ to an element $\alpha(x) \in \Delta^{\mathcal{I}}$.

We define the **satisfaction relation** "$\mathcal{I}, \alpha \models F$" between a formula F and an interpretation $\mathcal{I}$ w.r.t. an assignment α inductively as follows:

- $\mathcal{I}, \alpha \models P(x_1, \ldots, x_k)$ if, and only if, $(\alpha(x_1), \ldots, \alpha(x_k)) \in P^{\mathcal{I}}$;
- $\mathcal{I}, \alpha \models x_1 = x_2$ if, and only if, $\alpha(x_1) = \alpha(x_2)$;
- $\mathcal{I}, \alpha \models \neg F$ if, and only if, not $\mathcal{I}, \alpha \models F$;
- $\mathcal{I}, \alpha \models F \wedge G$ if, and only if, $\mathcal{I}, \alpha \models F$ and $\mathcal{I}, \alpha \models G$;
- $\mathcal{I}, \alpha \models F \vee G$ if, and only if, $\mathcal{I}, \alpha \models F$ or $\mathcal{I}, \alpha \models G$;

Satisfaction continued

- $\mathcal{I}, \alpha \models \exists x.F$ if, and only if, there exists a $d \in \Delta^{\mathcal{I}}$ such that $\mathcal{I}, \alpha[x \mapsto d] \models F$
- $\mathcal{I}, \alpha \models \forall x.F$ if, and only if, for all $d \in \Delta^{\mathcal{I}}$ we have $\mathcal{I}, \alpha[x \mapsto d] \models F$.

where $\alpha[x \mapsto d]$ stands for the assignment that coincides with α except that x is mapped to d.

A variable x is called **free** in a formula F if x has an occurrence in F such that no occurrence of $\exists x$ or $\forall x$ is in front of it. To indicate that the free variables in a formula F are among $x_1, \ldots, x_k$, we sometimes write $F(x_1, \ldots, x_k)$ instead of F.

To test whether $\mathcal{I}, \alpha \models F(x_1, \ldots, x_k)$ it is enough to know the $\alpha(x_1), \ldots, \alpha(x_k)$. Thus, we write

$$\mathcal{I} \models F(a_1, \ldots, a_k)$$

if $\mathcal{I}, \alpha \models F(x_1, \ldots, x_k)$ holds for some assignment (equivalently, all assignments) α with $\alpha(x_1) = a_1, \ldots, \alpha(x_k) = a_k$.

Example $\mathcal{I}_U$

- Let $\alpha(x) = \textbf{LU}$. Then $\mathcal{I}_U, \alpha \models \textbf{University}(x)$. We sometimes denote this by $\mathcal{I}_U \models \textbf{University}(\textbf{LU})$.
- Let $\alpha(x) = \textbf{Tim}$. Then $\mathcal{I}_U, \alpha \not\models \textbf{University}(x)$. We sometimes denote this by $\mathcal{I}_U \not\models \textbf{University}(\textbf{Tim})$.
- Let $\alpha(x) = \textbf{Tim}$ and $\alpha(y) = \textbf{LU}$. Then $\mathcal{I}_U, \alpha \models \textbf{student\_at}(x, y)$. We sometimes denote this by $\mathcal{I}_U \models \textbf{student\_at}(\textbf{Tim}, \textbf{LU})$.
- Let α be arbitrary. Then $\mathcal{I}_U, \alpha \models \exists x.\textbf{University}(x)$; since this does not depend on α, we denote this by $\mathcal{I}_U \models \exists x.\textbf{University}(x)$.
- We have $\mathcal{I}_U \models \exists x.\neg\textbf{University}(x)$.
- We have $\mathcal{I}_U \models \neg\forall x.(\textbf{Student}(x) \rightarrow \exists y.\textbf{student\_at}(x, y))$.

FOPL Ontologies and Reasoning

A closed FOPL formula (also called a sentence) is a FOPL formula without free variables.

An **FOPL ontology** $\boldsymbol{T}$ is a finite set of closed FOPL formulas.

For example, the following translation $\mathbf{MED}^\sharp$ of the $\mathcal{EL}$ TBoxes MED is an FOPL ontology:

$$\begin{aligned}
\forall x.(\mathbf{Pericardium}(x) &\rightarrow (\mathbf{Tissue}(x) \wedge \exists y.(\mathbf{cont_in}(x,y) \wedge \mathbf{Heart}(y))))\\
\forall x.(\mathbf{Pericarditis}(x) &\rightarrow (\mathbf{Inflammation}(x) \wedge \exists y.(\mathbf{has_loc}(x,y) \wedge \mathbf{Pericardium}(y))))\\
\forall x.(\mathbf{Inflammation}(x) &\rightarrow (\mathbf{Disease}(x) \wedge \exists y.(\mathbf{acts_on}(x,y) \wedge \mathbf{Tissue}(y))))\\
\forall x.(F(x) &\rightarrow (\mathbf{Heartdisease}(x) \wedge \mathbf{NeedTreatment}(x)))
\end{aligned}$$

where

$$F(x) = \mathbf{Disease}(x) \wedge \exists y.(\mathbf{has\_loc}(x,y) \wedge \exists z.(\mathbf{cont\_in}(y,z) \wedge \mathbf{Heart}(z)))$$

FOPL Ontologies and Reasoning

For an interpretation $\mathcal{I}$, we say that $\mathcal{I}$ is a model of $\boldsymbol{T}$ (or, equivalently, that $\mathcal{I}$ satisfies $\boldsymbol{T}$) and write

$$\mathcal{I} \models \boldsymbol{T},$$

if, and only if, $\mathcal{I} \models \boldsymbol{F}$ for all $\boldsymbol{F} \in \boldsymbol{T}$.

An FOPL sentence $\boldsymbol{F}$ follows from a FOPL ontology $\boldsymbol{T}$, in symbols

$$\boldsymbol{T} \models \boldsymbol{F},$$

if, and only if, for all interpretations $\mathcal{I}$ the following holds: If $\mathcal{I} \models \boldsymbol{T}$, then $\mathcal{I} \models \boldsymbol{F}$.

For example, "Pericarditis needs treatment" follows from **MED** if, and only if,

$$\mathbf{MED}^\sharp \models \forall x(\mathbf{Percarditis}(x) \rightarrow \mathbf{NeedsTreatment}(x)).$$

As mentioned above already, for FOPL ontologies $\boldsymbol{T}$ and FOPL sentences $\boldsymbol{F}$ the problem "$\boldsymbol{T} \models \boldsymbol{F}$" is undecidable!

First-order Predicate Logic (FOPL) as a Query Language

Data

A **ground sentence** F is of the form $P(c_1, \ldots, c_n)$, where P is an n-ary predicate and $c_1, \ldots, c_n$ are individual symbols.

A **database instance** is a finite set $\mathcal{D}$ of ground sentences. The set $\mathbf{Ind}(\mathcal{D})$ denote the set of individual symbols in $\mathcal{D}$.

For example, the following set $\mathcal{D}_{\text{uni}}$ is a database instance:

- $\mathbf{University}(\mathbf{LU})$, $\mathbf{University}(\mathbf{MU})$, $\mathbf{University}(\mathbf{CMU})$;
- $\mathbf{BritishUniversity}(\mathbf{LU})$
- $\mathbf{Student}(\mathbf{Tim})$, $\mathbf{Student}(\mathbf{Tom})$;
- $\mathbf{student\_at}(\mathbf{Tim}, \mathbf{LU})$, $\mathbf{student\_at}(\mathbf{Rob}, \mathbf{CMU})$.

Here we have

$$\mathbf{Ind}(\mathcal{D}_{\text{uni}}) = \{\mathbf{LU}, \mathbf{MU}, \mathbf{CMU}, \mathbf{Tim}, \mathbf{Tom}, \mathbf{Rob}\}$$

Querying Database Instances

An n-ary query Q is a mapping that takes as input a database instance $\mathcal{D}$ and outputs a set $Q(\mathcal{D})$ of n-tuples from $\mathbf{Ind}(\mathcal{D})$.

If $n = 0$, then Q is called a **Boolean query** and outputs either 'Yes' or 'No' or 'Don't know':

$$Q(\mathcal{D}) \in \{ \text{Yes, No, Don't know} \}.$$

Consider the database instance $\mathcal{D}_{\mathbf{uni}}$ with the following queries:

- Boolean Query: Is every British university a university?
- Boolean Query: Does every university have a student?
- Unary query: Output all students that are not students at any university.
- Unary query: Output all universities that are not British universities.

Assumptions

The answers to the queries above depend on our assumptions about the way in which $\mathcal{D}_{\mathbf{uni}}$ represents the world. The main options are:

- Closed World Assumption: if something is not stated explicitly in $\mathcal{D}_{\mathbf{uni}}$ it is false. We assume complete knowledge about the domain: if some ground sentence is true, it is included in $\mathcal{D}_{\mathbf{uni}}$.
- Open World Assumption: if something is not stated in $\mathcal{D}_{\mathbf{uni}}$, then we do not know whether it is true or false. We assume incomplete knowledge about the domain: if some ground sentence is not in $\mathcal{D}_{\mathbf{uni}}$, it can still be true.

In relational databases we make the closed world assumption!

In Schema.org Markup we make the Open World Assumption

Consider the following Schema.org Markup:

```
<div vocab="http://schema.org/" typeof="Movie">
  <h1 property="name">Avatar</h1>
  <div property="director" typeof="Person">
  Director: <span property="name">James Cameron</span>
 (born <time property="birthDate" datetime="1954-08-16">August 16, 1954</time>)
  </div>
  <span property="genre">Science fiction</span>
</div>
```

Corresponds to the database instance:

$$\textbf{Movie}(m), \quad \textbf{name}(o, \text{Avatar}), \quad \textbf{director}(m, p)$$

$$\textbf{name}(p, \text{Cameron}), \quad \textbf{birthDay}(p, 1954), \quad \textbf{genre}(m, \text{ScienceFiction})$$

Clearly, we cannot be sure that by collecting markup of webpages we collect all movies!

FOPL Relational Database Querying

We formulate queries as FOPL formulas. The query defined by $\boldsymbol{F}$ is $\boldsymbol{n}$-ary if $\boldsymbol{F}$ has $\boldsymbol{n}$ free variables. A query is Boolean if it is defined by a closed formula (has no free variable).

We realize the closed world assumption by regarding a database instance as an interpretation.

The interpretation $\mathcal{I}_{\mathcal{D}}$ defined by a database instance $\mathcal{D}$ is given by setting:

- $\Delta^{\mathcal{I}_{\mathcal{D}}} = \mathbf{Ind}(\mathcal{D})$;
- For any n-ary predicate P and any $(a_1, \ldots, a_n)$ in $\mathbf{Ind}(\mathcal{D})$:

$$(a_1, \ldots, a_n) \in P^{\mathcal{I}_{\mathcal{D}}} \quad \text{if and only if} \quad P(a_1, \ldots, a_n) \in \mathcal{D}$$

Example

The interpretation $\mathcal{J} = \mathcal{I}_{\mathcal{D}_{\mathsf{uni}}}$ defined by $\mathcal{D}_{\mathsf{uni}}$ is given as follows:

- $\Delta^{\mathcal{J}} = \{\mathbf{LU}, \mathbf{MU}, \mathbf{CMU}, \mathbf{Tim}, \mathbf{Tom}, \mathbf{Rob}\}$;
- $\mathbf{University}^{\mathcal{J}} = \{\mathbf{LU}, \mathbf{MU}, \mathbf{CMU}\}$;
- $\mathbf{BritishUniversity}^{\mathcal{J}} = \{\mathbf{LU}\}$;
- $\mathbf{Student}^{\mathcal{J}} = \{\mathbf{Tim}, \mathbf{Tom}\}$;
- $\mathbf{student\_at}^{\mathcal{J}} = \{(\mathbf{Tim}, \mathbf{LU}), (\mathbf{Rob}, \mathbf{CMU})\}$.

FOPL Relational Database Queries (Closed World Semantics)

A tuple $(a_1, \ldots, a_n)$ in $\mathbf{Ind}(\mathcal{D})$ is an **answer** to an FOPL query $F(x_1, \ldots, x_n)$ in database instance $\mathcal{D}$ if

$$\mathcal{I}_{\mathcal{D}} \models F(a_1, \ldots, a_n)$$

We set

$$\mathbf{answer}(F(x_1, \ldots, x_n), \mathcal{D}) = \{(a_1, \ldots, a_n) \mid \mathcal{I}_{\mathcal{D}} \models F(a_1, \ldots, a_n)\}$$

The answer to a Boolean FOPL query F in $\mathcal{D}$ is 'Yes' if

$$\mathcal{I}_{\mathcal{D}} \models F$$

The answer to a Boolean FOPL query F in $\mathcal{D}$ is 'No' if

$$\mathcal{I}_{\mathcal{D}} \not\models F$$

Note that the answer 'Don't know' is never returned!

Examples

We query $\mathcal{D}_{\text{uni}}$ under closed world semantics:

- Output all universities. The corresponding query is

$$F(x) = \textbf{University}(x).$$

  The set of answers is $\textbf{answer}(F(x), \mathcal{D}_{\text{uni}}) = \{\textbf{LU}, \textbf{MU}, \textbf{CMU}\}$

- Is every British university a university? The corresponding query is

$$F = \forall x(\textbf{BritishUniversity}(x) \rightarrow \textbf{University}(x))$$

  The answer is 'Yes'.

Examples

- Does every university have a student? The corresponding query is

$$F = \forall x(\textbf{University}(x) \rightarrow \exists y \textbf{student\_at}(y, x))$$

  The answer is 'No'.

- Output all students that are not students at any university. The corresponding query is

$$F(x) = \textbf{Student}(x) \wedge \neg \exists y \textbf{student\_at}(x, y)$$

  The set of answers is $\textbf{answer}(F(x), \mathcal{D}_{\text{uni}}) = \{\textbf{Tom}\}$

- Output all universities that are not British universities. The corresponding query is

$$F(x) = \textbf{University}(x) \wedge \neg \textbf{BritishUniversity}(x)$$

  The set of answers is $\textbf{answer}(F(x), \mathcal{D}_{\text{uni}}) = \{\textbf{MU}, \textbf{CMU}\}$.

Complexity of querying relational databases

Consider, for simplicity, Boolean queries. There are two different ways of measuring the complexity of querying relational databases (RDBs):

- **Data complexity**: This measures the time/space needed (in the worst case) to evaluate a fixed query $\boldsymbol{F}$ in an instance $\mathcal{I}$ of a RDB (i.e., to decide whether $\mathcal{I} \models \boldsymbol{F}$). Thus, when measuring data complexity, the input variable is the size of the instance and the query is assumed to be fixed.
- **Combined Complexity**: This measures the time/space needed (in the worst case) to evaluate a query $\boldsymbol{F}$ in an instance $\mathcal{I}$ of a RDB. Thus, when measuring combined complexity, both the size of the instance and the query are input variables.

Data complexity is regarded as being more relevant as queries are typically rather small, but database instances are very large compared to the query.

FOPL: Data complexity is in LogSpace. So it is in polytime and, therefore, tractable. Combined complexity is PSpace-complete, and, therefore, not tractable.

FOPL Query Answering (Open World Semantics)

Let $\mathcal{D}$ be a database instance. An interpretation $\mathcal{I}$ is a model of $\mathcal{D}$ if

- $\mathbf{Ind}(\mathcal{D}) \subseteq \Delta^{\mathcal{I}}$;
- If $P(a_1, \ldots, a_n) \in \mathcal{D}$, then $(a_1, \ldots, a_n) \in P^{\mathcal{I}}$.

The set of models of $\mathcal{D}$ is denoted by $\mathbf{Mod}(\mathcal{D})$.

Let $F(x_1, \ldots, x_n)$ be an FOPL query. Then $(a_1, \ldots, a_n)$ in $\mathbf{Ind}(\mathcal{D})$ is a **certain answer** to $F(x_1, \ldots, x_n)$ in $\mathcal{D}$, in symbols

$$\mathcal{D} \models F(a_1, \ldots, a_n),$$

if $\mathcal{I} \models F(a_1, \ldots, a_n)$ for all $\mathcal{I} \in \mathbf{Mod}(\mathcal{D})$.

The set of certain answers to $F(x_1, \ldots, x_n)$ in $\mathcal{D}$ is

$$\mathbf{certanswer}(F(x_1, \ldots, x_n), \mathcal{D}) = \{(a_1, \ldots, a_n) \mid \mathcal{D} \models F(a_1, \ldots, a_n)\}$$

FOPL Query Answering (Open World Semantics)

- 'Yes' is the certain answer to a Boolean query $\boldsymbol{F}$ if $\boldsymbol{\mathcal{I}} \models \boldsymbol{F}$ for all $\boldsymbol{\mathcal{I}} \in \mathbf{Mod}(\boldsymbol{\mathcal{D}})$.
- 'No' is the certain answer to a Boolean query $\boldsymbol{F}$ if $\boldsymbol{\mathcal{I}} \not\models \boldsymbol{F}$ for all $\boldsymbol{\mathcal{I}} \in \mathbf{Mod}(\boldsymbol{\mathcal{D}})$
- If neither 'Yes' nor 'No' is a certain answer, then we say that the certain answer is 'Don't know'.

Note that under closed world semantics the answer to a Boolean query is always either 'Yes' or 'No'.

Example

We consider again the database instance $\mathcal{D}_{\text{uni}}$:

- $\mathbf{University}(\mathbf{LU}), \quad \mathbf{University}(\mathbf{MU}), \quad \mathbf{University}(\mathbf{CMU})$;
- $\mathbf{BritishUniversity}(\mathbf{LU})$
- $\mathbf{Student}(\mathbf{Tim}), \quad \mathbf{Student}(\mathbf{Tom})$;
- $\mathbf{student\_at}(\mathbf{Tim}, \mathbf{LU}), \quad \mathbf{student\_at}(\mathbf{Rob}, \mathbf{CMU})$.

Example

Output all universities. The corresponding query is

$$\boldsymbol{F}(x) = \textbf{University}(x).$$

The set of certain answers is

$$\textbf{certanswer}(\boldsymbol{F}(x), \mathcal{D}_{\textsf{uni}}) = \{\textbf{LU}, \textbf{MU}, \textbf{CMU}\}$$

This coincides with $\textbf{answer}(\boldsymbol{F}(x), \mathcal{D}_{\textsf{uni}})$. To see this, we have to show:

- for all $\mathcal{I} \in \textbf{Mod}(\mathcal{D}_{\textsf{uni}})$: $\mathcal{I} \models \boldsymbol{F}(\textbf{LU})$, $\mathcal{I} \models \boldsymbol{F}(\textbf{MU})$, and $\mathcal{I} \models \boldsymbol{F}(\textbf{CMU})$.
- there exists $\mathcal{I} \in \textbf{Mod}(\mathcal{D}_{\textsf{uni}})$ with $\mathcal{I} \not\models \boldsymbol{F}(\textbf{Tim})$;
- there exists $\mathcal{I} \in \textbf{Mod}(\mathcal{D}_{\textsf{uni}})$ with $\mathcal{I} \not\models \boldsymbol{F}(\textbf{Tom})$.
- there exists $\mathcal{I} \in \textbf{Mod}(\mathcal{D}_{\textsf{uni}})$ with $\mathcal{I} \not\models \boldsymbol{F}(\textbf{Rob})$.

For Point 2 to 4 we just take $\mathcal{I}_{\mathcal{D}_{\textsf{uni}}}$!

Example

Is every British university a university? The corresponding query is

$$\boldsymbol{F} = \forall x(\textbf{BritishUniversity}(x) \rightarrow \textbf{University}(x))$$

The certain answer to $\boldsymbol{F}$ in $\mathcal{D}_{\textbf{uni}}$ is 'Don't know'. To see this we have to construct two interpretations $\mathcal{I} \in \textbf{Mod}(\mathcal{D}_{\textbf{uni}})$ and $\mathcal{J} \in \textbf{Mod}(\mathcal{D}_{\textbf{uni}})$ such that

- $\mathcal{I} \models \boldsymbol{F}$;
- $\mathcal{J} \not\models \boldsymbol{F}$.

For Point 1 we can take $\mathcal{I}_{\mathcal{D}_{\textbf{uni}}}$. For Point 2, we expand $\mathcal{I}_{\mathcal{D}_{\textbf{uni}}}$ by adding a new individual a to its domain $\Delta^{\mathcal{J}}$ and let $a \in \textbf{BritishUniversity}^{\mathcal{J}}$ and $a \notin \textbf{University}^{\mathcal{J}}$.

Example

Does every university have a student? The corresponding query is

$$\boldsymbol{F} = \forall x(\mathbf{University}(x) \rightarrow \exists y \mathbf{student\_at}(y, x))$$

The certain answer to $\boldsymbol{F}$ in $\mathcal{D}_{\text{uni}}$ is 'Don't know'. To see this we have to construct two interpretations $\mathcal{I} \in \mathbf{Mod}(\mathcal{D}_{\text{uni}})$ and $\mathcal{J} \in \mathbf{Mod}(\mathcal{D}_{\text{uni}})$ such that

- $\mathcal{I} \models \boldsymbol{F}$;
- $\mathcal{J} \not\models \boldsymbol{F}$.

For Point 2 we can take $\mathcal{J} = \mathcal{I}_{\mathcal{D}_{\text{uni}}}$ (since **MU** does not have a student in $\mathcal{I}_{\mathcal{D}_{\text{uni}}}$). For Point 1, we expand $\mathcal{I}_{\mathcal{D}_{\text{uni}}}$ to an interpretation $\mathcal{I}$ by adding a new individual a to its domain $\Delta^{\mathcal{I}}$ and let $(a, \mathbf{MU}) \in \mathbf{student\_at}^{\mathcal{I}}$.

Example

Output all students that are not students at any university. The corresponding query is

$$F(x) = \textbf{Student}(x) \wedge \neg \exists y \textbf{student\_at}(x, y)$$

The set of certain answers is $\textbf{certanswer}(F(x), \mathcal{D}_{\textbf{uni}}) = \emptyset$.

To see this, observe that $\mathcal{I}_{\mathcal{D}_{\textbf{uni}}}$ shows already that only **Tom** could be a certain answer. But **Tom** is not a certain answer since the interpretation $\mathcal{I}$ that expands $\mathcal{I}_{\mathcal{D}_{\textbf{uni}}}$ by adding $(\textbf{Tom}, \textbf{MU})$ to $\textbf{student\_at}^{\mathcal{I}}$ shows that there is a model of $\mathcal{D}_{\textbf{uni}}$ in which **Tom** is a student at some university.

Example

Output all universities that are not British universities. The corresponding query is

$$F(x) = \textbf{University}(x) \wedge \neg \textbf{BritishUniversity}(x)$$

The set of answers is $\textbf{certanswer}(F(x), \mathcal{D}_{\textbf{uni}}) = \emptyset$.

To see this, expand $\mathcal{I}_{\mathcal{D}_{\textbf{uni}}}$ by adding **MU** and **CMU** to $\textbf{BritishUniversity}^{\mathcal{I}}$. Then $\mathcal{I}$ is a model of $\mathcal{D}_{\textbf{uni}}$ in which there is no university that is not a British university.

The Relationship between Relational Databases and Ontologies

- FOPL database querying (closed world assumption) means checking

$$\mathcal{I} \models \boldsymbol{F}$$

- FOPL and Description Logic ontology querying means checking

$$\boldsymbol{T} \models \boldsymbol{F}$$

- Typically, an FOPL or Description Logic ontology $\boldsymbol{T}$ has many different interpretations (many different $\mathcal{I}$ such that $\mathcal{I} \models \boldsymbol{T}$). Data instances $\mathcal{D}$ have only one interpretation (under closed world assumption).
- FOPL and Description Logic ontologies make the open world assumption. For example, often individuals are not mentioned at all.