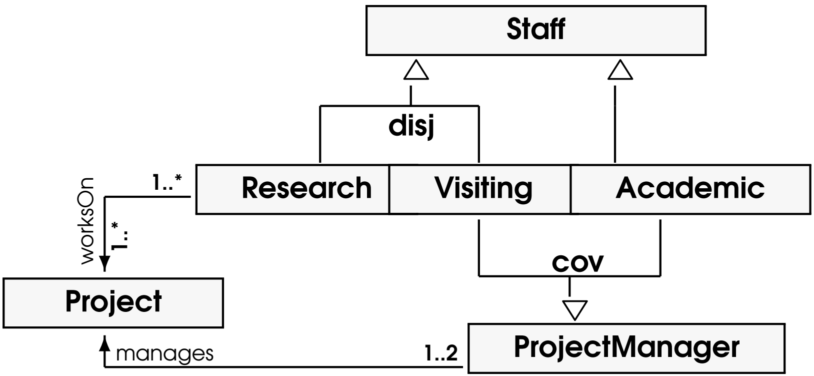
Description logics from the DL-Lite family: the terminological part

DL-Lite

DL-Lite is a family of description logics that has been designed to

- capture (at least) standard conceptual models such as ER and UML diagrams;
- provide a formal semantics for them;
- enable efficient access to data using a conceptual model represented as a DL-Lite TBox.

More details on what the last point means later. Here we introduce the terminological part of a member of the DL-Lite family which, for simplicity, we just call DL-Lite.



A UML diagram

How to represent a conceptual model such as this one in description logic?

DL-Lite (syntax)

- **Language for DL-Lite concepts (classes)**
  - concept names $A_0, A_1, \ldots$
  - role names $r_0, r_1, \ldots$
  - the concept $\top$ (often called "thing") 同上，域中全部元素
  - the concept $\bot$ (stands for the empty class)
  - the concept constructor $\sqcap$ (often called intersection, conjunction, or simply "and").
  - the concept constructor $\exists$ (often called existential restriction).
  - the concept constructor $\geq n$, where $n > 0$ is a natural number (called number restriction). eg: $\{Jack\} \sqsubseteq \geq 3\, haschild.\top$ (指Jack至少有3个孩子)
  - the role constructor $\cdot^-$ (called inverse role constructor).

Jack is father of Tom

$\Leftrightarrow$ Tom has father Jack

is father of$^-$ = has father

DL-Lite

DL中只能 $\cdots.\top$

A **role** is either a role name or the inverse r^- over a role name r.

DL-Lite concepts are defined as follows:

但EL中有 $\cdots.C$

(C是某集合)

- All concept names, $\top$ and $\bot$ are DL-Lite concepts;
- $\exists r.\top$ is a DL-Lite concept, for every role r;
- $(\geq n\ r.\top)$ is a DL-Lite concept, for every role r.
- If B_1 and B_2 are DL-Lite concepts, then $B_1 \sqcap B_2$ is a DL-Lite concept.

A **DL-Lite concept-inclusion** is of the form

$$B_1 \sqsubseteq B_2$$

where B_1 and B_2 are DL-Lite concepts.

A **DL-Lite TBox** is a finite set of DL-Lite concept inclusions.

$$A \underset{\text{是孩子}}{\overset{\text{是妈}}{\rightleftarrows}} B$$

EL中没有⊥，无法表示两集合不相交

Examples, discussion, and comparison with $\mathcal{EL}$

- **Person** $\sqcap$ ($\geq$ **5 hasChild.**$\top$) (a person with at least 5 children);
- **Person** $\sqcap$ ($\geq$ **7 hasChild**$^{-}$**.**$\top$) (a person with at least 7 parents (?));
- **Chair** $\sqcap$ **Table** $\sqsubseteq$ $\bot$ (the classes of chairs and tables are disjoint);
- The $\mathcal{EL}$-concept **Person** $\sqcap$ $\exists$**hasChild.Person** cannot be expressed in DL-Lite;
- The $\mathcal{EL}$-concept **Person** $\sqcap$ $\exists$**hasChild.**$\exists$**hasChild.**$\top$ cannot be expressed in DL-Lite;

DL中必须是⊤

- $\exists \boldsymbol{r}.\top$ is equivalent to ($\geq \mathbf{1}\ \boldsymbol{r}.\top$).

Question: Why don't we just put DL-Lite and $\mathcal{EL}$ together? Isn't this much better than introducing two languages...?

Domain Restriction in DL-Lite

The UML class diagram above says that

"everybody who manages something is a manager"

or, equivalently, "only managers manage something".

In other words, it states that every object in the domain of the relation "manages" is a Manager. This can be expressed in DL-Lite as

$$\exists \mathbf{manages}.\top \sqsubseteq \mathbf{Manager}$$

Inclusions of the form

$$\exists r.\top \sqsubseteq C$$

are called **domain restrictions**.

A manages B

domain range

Range Restrictions in DL-Lite

The UML class diagram above says that

"everything that is managed by something is a project"

or, equivalently, "only projects are managed".

In other words, it states that every object in the range of the relation "manages" is a Project. This can be expressed in DL-Lite as

$$\exists \mathbf{manages}^{-}.\top \sqsubseteq \mathbf{Project}$$

$\approx$ be managed

Inclusions of the form

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

are called **range restrictions**.

Disjointness Statements

The UML class diagram above says that

"nobody is in both classes Research and Visiting"

or, equivalently, the classes Research and Visiting are disjoint.

This can be expressed in DL-Lite as

$$\textbf{Research} \sqcap \textbf{Visiting} \sqsubseteq \bot$$

Inclusions of the form

$$\boldsymbol{A} \sqcap \boldsymbol{B} \sqsubseteq \bot$$

are called **disjointness statements**.

The UML class diagram above in DL-Lite (almost)

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

The only missing information is the covering constraint that

every Projectmanager is VistingStaff or AcademicStaff.

This cannot be expressed in DL-Lite.

DL-Lite (semantics)

域 函数

Interpretations are defined as before:

- Recall that an **interpretation** is a structure $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ in which
  - $\Delta^{\mathcal{I}}$ is the **domain** (a non-empty set)
  - $\cdot^{\mathcal{I}}$ is an **interpretation function** that maps:
    - every concept name A to a subseteq $A^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$ $\quad (A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}})$
    - every role name r to a binary relation $r^{\mathcal{I}}$ over $\Delta^{\mathcal{I}}$ $\quad (r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}})$
- The interpretation of an inverse role $(r^-)^{\mathcal{I}}$ is the inverse of $r^{\mathcal{I}}$:

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

通常concept name大、可以解释为

eg. $\Delta^{\mathcal{I}} = \boxed{a, b, c, d, e, 1, 2, 3, 4, 5, 6}$

$Person^{\mathcal{I}} = \{a, b\}$

$Animal^{\mathcal{I}} = \{c, d\}$

$r^{\mathcal{I}} = \{ (a,2), (a,4), (b,3), (d,1), (d,4), (d,7) \}$

DL-Lite (semantics)

The interpretation $B^{\mathcal{I}}$ of a DL-Lite concept B in $\mathcal{I}$ is defined inductively as follows:

$(\geq 1\ r.\top)^{\mathcal{I}} = \{ a, b, d \}$

$(\geq 2\ r.\top)^{\mathcal{I}} = \{ a, d \}$

$(\geq 3\ r.\top)^{\mathcal{I}} = \{ d \}$

- $(\top)^{\mathcal{I}} = \Delta^{\mathcal{I}}$;
- $(\bot)^{\mathcal{I}} = \emptyset$;
- $(\geq n\ r.\top)^{\mathcal{I}}$ is the set of all $x \in \Delta^{\mathcal{I}}$ such that the number of y in $\Delta^{\mathcal{I}}$ with $(x, y) \in r^{\mathcal{I}}$ is at least n; 至少有n个满足r关系.
- $(\exists r.\top)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid$ there exists $y \in \Delta^{\mathcal{I}}$ such that $(x, y) \in r^{\mathcal{I}}\}$;
- $(B_1 \sqcap B_2)^{\mathcal{I}} = B_1^{\mathcal{I}} \cap B_2^{\mathcal{I}}$.

Semantics: exactly the same as for $\mathcal{EL}$

Let $\mathcal{I}$ be an interpretation, $B_1 \sqsubseteq B_2$ a DL-Lite inclusion, and $\mathcal{T}$ a DL-Lite TBox.

a set of DL inclusion

- $\mathcal{I} \models B_1 \sqsubseteq B_2$ if, and only if, $B_1^{\mathcal{I}} \subseteq B_2^{\mathcal{I}}$. In words:
  - $\mathcal{I}$ satisfies $B_1 \sqsubseteq B_2$ or
  - $B_1 \sqsubseteq B_2$ is true in $\mathcal{I}$ or
  - $\mathcal{I}$ is a model of $B_1 \sqsubseteq B_2$.
- We set $\mathcal{I} \models \mathcal{T}$ if, and only if, $\mathcal{I} \models B_1 \sqsubseteq B_2$ for all $B_1 \sqsubseteq B_2$ in $\mathcal{T}$. In words:
  - $\mathcal{I}$ satisfies $\mathcal{T}$ or
  - $\mathcal{I}$ is a model of $\mathcal{T}$.

Reasoning Services for DL-Lite

- **Subsumption**. Let $\mathcal{T}$ be a TBox and $B_1 \sqsubseteq B_2$ a DL-Lite concept inclusion. We say that $B_1 \sqsubseteq B_2$ **follows from** $\mathcal{T}$ if, and only if, every model of $\mathcal{T}$ is a model of $B_1 \sqsubseteq B_2$. Again we often write
  - $\mathcal{T} \models B_1 \sqsubseteq B_2$ or
  - $B_1 \sqsubseteq_{\mathcal{T}} B_2$.

T的 model 必然是 $B_1 \sqsubseteq B_1$
的 model

- **TBox Satisfiablility**. A TBox $\mathcal{T}$ is satisfiable if, and only if, there exists a model of $\mathcal{T}$.

Theorem. For DL-Lite, there exist polytime algorithms deciding subsumption and TBox satisfiability.

Question: Why didn't we consider the TBox satisfiability problem for $\mathcal{EL}$-TBoxes?

Examples

Let $\mathcal{T} = \{A \sqsubseteq (\geq 2\ r.\top)\}$. Then

$$\mathcal{T} \not\models A \sqsubseteq (\geq 3\ r.\top).$$

To see this, construct an interpretation $\mathcal{I}$ such that

- $\mathcal{I} \models \mathcal{T}$;
- $\mathcal{I} \not\models A \sqsubseteq (\geq 3\ r.\top)$.

Namely, let $\mathcal{I}$ be defined by

- $\Delta^{\mathcal{I}} = \{a, b, c\}$;
- $A^{\mathcal{I}} = \{a\}$;
- $r^{\mathcal{I}} = \{(a, b), (a, c)\}$;

只有a有2个满足r关系的组合

Then $A^{\mathcal{I}} = \{a\} \subseteq \{a\} = (\geq 2\ r.\top)^{\mathcal{I}}$ and so $\mathcal{I} \models \mathcal{T}$. But $A^{\mathcal{I}} = \{a\} \not\subseteq \emptyset = (\geq 3\ r.\top)^{\mathcal{I}}$ and so $\mathcal{I} \not\models A \sqsubseteq (\geq 3\ r.\top)$.

Example: Satisfiability

Let

$$\mathcal{T} = \{A \sqsubseteq \exists r.\top, \exists r^-.\top \sqsubseteq B, \top \sqsubseteq A, B \sqsubseteq \bot\}$$

Then $\mathcal{T}$ is not satisfiable.

To see this, assume for a proof by contradiction that $\mathcal{I} \models \mathcal{T}$.

Let $a \in \Delta^{\mathcal{I}}$.

- Then $a \in \top^{\mathcal{I}}$ and so $a \in A^{\mathcal{I}}$.
- Hence $a \in (\exists r.\top)^{\mathcal{I}}$.
- Hence there exists $b \in \Delta^{\mathcal{I}}$ with $(a, b) \in r^{\mathcal{I}}$.
- Hence $b \in (\exists r^-.\top)^{\mathcal{I}}$.
- Hence $b \in B^{\mathcal{I}}$.
- But this contradicts $B \sqsubseteq \bot \in \mathcal{T}$.

$B^{\mathcal{I}} = \emptyset$

DL-Lite with "or"

DL-Lite with "or" is an extension of DL-Lite in which one can express covering constraints.

It is obtained from DL-Lite by adding the

- concept constructor $\sqcup$ (often called union, disjunction, or simply "or")

to DL-Lite.

In the resulting description logic we can express the covering constraint that

every Projectmanager is VistingStaff or AcademicStaff

using the inclusion

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

The prize for this additional expressive power is high, however:

Theorem. Checking satisfiability in DL-Lite with "or" is NP-complete. Checking subsumption is coNP-complete.

Schema.org as a Description Logic: the terminological part

Schema.org

- Initiative by Google, Microsoft, Yahoo!, and Yandex.
- Provides vocabulary for structured **data markup** on webpages.
- Applications include: enhance presentation of search results (rich snippets) and import into Google (or other) Knowledge Graphs.
- Used by more than 15 million webpages and all major ones.

```
<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>
```

Schema.org: the Vocabulary/Ontology

- Constantly evolving (2015: 622 concept names (classes), 891 role names (properties))
- Information at http://schema.org

For example: the class **Movie** comes with properties:

actor, **director**, **duration**, **musicBy**,

productionCompany, **subtitleLanguage**, **trailer**

Movie is a subclass of **CreativeWork** that comes with approximately 50 properties, including

producer, **review**, **recordedAt**, etc

Schema.org: the underpinning Language

No official formal definition of syntax or semantics, but certain language patterns are described in natural language.

Formalisation as a Description Logic

A Schema.org ontology consists of concept inclusions of the form:

- $A \sqsubseteq B$, where A and B are concept names (called types in Schema.org). For example,

$$\textbf{Movie} \sqsubseteq \textbf{CreativeWork}$$

Schema.org: the underpinning Language

Inclusions

$$\exists r.\top \sqsubseteq A_1 \sqcup \cdots \sqcup A_n,$$

where r is a role name and $A_1, \ldots, A_n$ are concept names. For example,

$$\exists \textbf{musicBy}.\top \sqsubseteq \textbf{Episode} \sqcup \textbf{Movie} \sqcup \textbf{RadioSeries} \sqcup \textbf{TVSeries}$$

Recall that these expressions are called **domain restrictions** as they restrict the domain of the relation r to objects that are in A_1 or A_2, or A_3, and so on. Thus, the inclusion above says that everything in the domain of the relation **musicBy** is an Episode, a Movie, a RadioSeries, or a TVSeries.

Schema.org: the underpinning Language

Inclusions

$$\exists r^{-}.\top \sqsubseteq A_1 \sqcup \cdots \sqcup A_n,$$

where r is a role name and $A_1, \ldots, A_n$ are concept names. For example,

$$\exists \textbf{musicBy}^{-}.\top \sqsubseteq \textbf{Person} \sqcup \textbf{MusicGroup}$$

Recall that these expressions are called **range restrictions** as they restrict the range of the relation r to objects that are in A_1 or A_2, or A_3, and so on. Thus, the inclusion above says that everything in the range of the relation **musicBy** is a Person or a MusicGroup.

Schema.org: the underpinning Language

In addition, Schema.org has inclusions between role names

$$r \sqsubseteq s,$$

where r and s are role names. For example **sibling** $\sqsubseteq$ **relatedTo**.

Finally, Schema.org has **enumeration definitions**

$$A \equiv \{a_1, \ldots, a_n\},$$

where A is a concept name and $a_1, \ldots, a_n$ are individual names. Examples are

$$\textbf{Booktype} \equiv \{\textbf{ebook}, \textbf{hardcover}, \textbf{paperback}\},$$
$$\textbf{Colour} \equiv \{\textbf{blue}, \textbf{red}, \textbf{green}, \textbf{yellow}\}$$

Schema.org (semantics)

The semantics is exactly the same as for $\mathcal{EL}$ and DL-Lite, where in addition we require that in any interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ and for every individual name a we have

- $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$.

Then we say that an enumeration definition

$$A \equiv \{a_1, \ldots, a_n\}$$

is true in $\mathcal{I}$, in symbols $\mathcal{I} \models A \equiv \{a_1, \ldots, a_n\}$, if $A^{\mathcal{I}} = \{a_1^{\mathcal{I}}, \ldots, a_n^{\mathcal{I}}\}$.