Reasoning in Description Logics:
Expressive Power vs. Computational Complexity

Carsten Lutz

University of Bremen, Germany
Description Logic is a subfield of KR concerned with terminological knowledge:

Describe the central notions of the application domain (its terminology) and their interrelations.

E.g. in medical applications:

Tissue, Inflammation, Pericadium, Pericarditis, etc.

DLs play an important role as logical foundation of ontology languages:

- OWL is the W3C-standard for a Web Ontology Language
  - OWL 1 in 2004
  - OWL 2 in 10/2009

- OWL is essentially a description logic with an XML syntax
Main reason for popularity:

attractive compromise between expressive power and computational complexity

<table>
<thead>
<tr>
<th>Propositional Logic</th>
<th>First-Order Logic</th>
<th>Modal Logic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Efficient reasoning via SAT solvers, but often too inexpressive</td>
<td>Very expressive reference formalism, but reasoning too costly</td>
<td>\approx</td>
</tr>
</tbody>
</table>

Not one DL, but a large toolbox of formalisms:

- DLs cover broad range of responses to “complexity vs. expressive power”
- OWL 2 contains different profiles (3 inexpressive, 1 expressive, 1 not a logic)
Before break:

- brief introduction to description logics
- complexity and expressive power of expressive DLs
- complexity and expressive power of lightweight DLs, part I

After break:

- complexity and expressive power of lightweight DLs, part II
- instance data and query answering
Introduction to Description Logics
Some DL Basics

Knowledge is (mainly) stored in the TBox, e.g.:

\[
\begin{align*}
\text{Pericardium} & \sqsubseteq \text{Tissue} \sqcap \exists \text{partOf.Heart} \\
\text{Pericarditis} & \doteq \text{Inflammation} \sqcap \exists \text{location.Pericardium} \\
\text{Inflammation} & \sqsubseteq \text{Disease} \sqcap \exists \text{actsOn.Tissue} \\
\text{Tissue} \sqcap \text{Disease} & \sqsubseteq \bot
\end{align*}
\]

\(\text{TBox} = \text{“Terminology Box”;} \ \text{modern \ view: \ TBox = ontology}\)

Formally, a TBox is a finite set of

concept inclusions \(C \sqsubseteq D\) \quad \text{and} \quad \text{concept definitions} \ C \doteq D

where \(C, D\) are concepts (\(\approx\) formulas) in the DL used.
Some DL Basics

Different concept constructors give rise to different DLs / OWL dialects:

\[ \text{PTIME} \subseteq \text{EL} \subseteq \text{ACC} \subseteq \text{SHIQ} \subseteq \text{SHOIQ} \approx \text{OWL 1} \subseteq \text{SROIQ} \approx \text{OWL 2} \]

\[ \text{PTIME} \subseteq \text{ExpTime} \subseteq \text{ExpTime} \subseteq \text{NExpTime} \subseteq 2\text{NExpTime} \]
The Description Logic \( \mathcal{ALC} \)

Fix a countably infinite supply of

- concept names (\( \sim \) unary predicates)
- role names (\( \sim \) binary predicates)

Concept language of \( \mathcal{ALC} \):

\[ C ::= A \mid \top \mid \bot \mid \neg C \mid C \sqcap D \mid C \sqcup D \mid \exists r.C \mid \forall r.C \]

\( \exists r.C \): existential restriction

\( \forall r.C \): universal restriction / value restriction

For example:

\[ \text{Disease} \sqcap \exists \text{actsOn.} \text{Organ} \sqcap \forall \text{cause.} \neg \text{Genetic} \]
The Description Logic $\mathcal{ALC}$

**DL interpretation $\mathcal{I}$:**

FO structure with only unary+binary predicates = Kripke structure

**DL-style notation:** interpretation $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ with

- $\Delta^\mathcal{I}$ a non-empty set, the **domain**
- $\cdot^\mathcal{I}$ the **interpretation function** which assigns
  - a set $A^\mathcal{I} \subseteq \Delta^\mathcal{I}$ to each concept name $A$
  - a binary relation $r^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$ to each role name $r$

We now extend $\cdot^\mathcal{I}$ to composite concepts
DL concepts $\approx$ FO formulas with exactly 1 free variable $\approx$ modal formulas

\[
\begin{align*}
A & \quad A(x) & \quad \mathcal{P}_A \\
\neg C & \quad \neg C(x) & \quad \neg C \\
C \sqcup D & \quad C(x) \lor D(x) & \quad C \lor D \\
C \sqcap D & \quad C(x) \land D(x) & \quad C \land D \\
\exists r. C & \quad \exists y. ( r(x, y) \land C(y) ) & \quad \langle r \rangle . C \\
\forall r. C & \quad \forall y. ( r(x, y) \rightarrow C(y) ) & \quad [r] . C
\end{align*}
\]

Note: 2 variables / guarded formulas suffices

We use $C^\mathcal{I}$ to denote the set \{ $d \in \Delta^\mathcal{I} \mid \mathcal{I} \models C(d)$ \}
TBoxes correspond to FO sentences:

\[ C \subseteq D \quad \quad \forall x.( C(x) \rightarrow D(x) ) \]

\[ C \models D \quad \quad \forall x.( C(x) \leftrightarrow D(x) ) \]

\[ \exists \varphi \quad \quad \bigwedge_{\varphi \in \mathcal{T}} \varphi \]

Example:

\[ \text{Pericardium} \subseteq \text{Tissue} \cap \exists \text{partOf.Heart} \]

translates to

\[ \forall x. ( \text{Pericardium}(x) \rightarrow ( \text{Tissue}(x) \land \exists y. ( \text{partOf}(x, y) \land \text{Heart}(y) ) ) ) \]
Reasoning

Traditional reasoning problems:

- **satisfiability**: given $C$ and $\mathcal{T}$, is there a model $\mathcal{I}$ of $\mathcal{T}$ with $C^\mathcal{I} \neq \emptyset$?
  
  used for detecting **modelling mistakes**

- **subsumption**: given $C$, $D$ and $\mathcal{T}$, does $\mathcal{T} \models C \subseteq D$?
  
  i.e., do all models $\mathcal{I}$ of $\mathcal{T}$ satisfy $C^\mathcal{I} \subseteq D^\mathcal{I}$?

  used to arrange all concepts in a TBox in a **subsumption hierarchy**

  makes structure explicit, facilitates **browsing** and **navigation**
Reasoning

Traditional reasoning problems:

- **satisfiability**: given $C$ and $\mathcal{T}$, is there a model $\mathcal{I}$ of $\mathcal{T}$ with $C^\mathcal{I} \neq \emptyset$?

  used for detecting **modelling mistakes**

- **subsumption**: given $C$, $D$ and $\mathcal{T}$, does $\mathcal{T} \models C \sqsubseteq D$?

  i.e., do all models $\mathcal{I}$ of $\mathcal{T}$ satisfy $C^\mathcal{I} \subseteq D^\mathcal{I}$?

  used to arrange all concepts in a TBox in a **subsumption hierarchy**

  makes structure explicit, facilitates **browsing** and **navigation**

Note:

- $C$ satisfiable w.r.t. $\mathcal{T}$ iff $\mathcal{T} \not \models C \sqsubseteq \bot$

- $\mathcal{T} \models C \sqsubseteq D$ iff $C \cap \neg D$ unsatisfiable w.r.t. $\mathcal{T}$
On the Role of Complexity

Is DL all about computational complexity?

What complexity theory can do for us:

- help to understand the expressive power of the formalism
  to prove hardness results, one must show that something can be expressed

- provide performance guarantees or show that they do not exist

What it cannot do for us (so far):

- tell us whether something will work in practice or not
Expressive Description Logics
(i.e.: $\mathcal{ALC}$ and above)
**A Bit of History**

Stone age of description logics (until mid-1990ies):

“We have to offer efficient reasoning and thus cannot include all Booleans”

“Every application needs at least conjunction and universal restriction”

(and thus reasoning is co-NP-complete)

The *SHIQ* era (since mid-1990ies):

“ExpTime DLs can be implemented efficiently” (FaCT system by Horrocks)

“We do need the Booleans and much, much more (but want to stay decidable)!”
Expressive Power of $\mathcal{ALC}$

Central notion for understanding expressive power of $\mathcal{ALC}$:

Relation $\rho \subseteq \Delta^\mathcal{I}_1 \times \Delta^\mathcal{I}_2$ is **bisimulation** between interpretations $\mathcal{I}_1$ and $\mathcal{I}_2$ if $d \rho d'$ implies that

- $d$ and $d'$ satisfy same concept names
- each successor of $d$ has $\rho$-related counterpart at $d'$
- each successor of $d'$ has $\rho$-related counterpart at $d$
$(\mathcal{I}_1, d_1) \sim (\mathcal{I}_2, d_2)$: there is a bisimulation $\rho$ between $\mathcal{I}_1$ and $\mathcal{I}_2$ with $d_1 \rho d_2$
Lemma. \( \mathcal{ALC} \) is invariant under bisimulations, i.e.,

If \((I_1, d_1) \sim (I_2, d_2)\), then \(d_1 \in C^{I_1}\) iff \(d_2 \in C^{I_2}\)

for all \(\mathcal{ALC}\)-concepts \(C\).

Together with example from previous slide:

\(\mathcal{ALC}\) lacks expressive power for counting successors!
The converse is false in general:

\[ \mathcal{I}_1 \quad \mathcal{I}_2 \]

**Theorem.** An FO-formula \( \varphi \) with one free variable is equivalent to an ALC-concept iff it is invariant under bisimulation. [vanBenthem76]
Theorem. If an $\mathcal{ALC}$-concept $C$ is satisfiable w.r.t. an $\mathcal{ALC}$-TBox $\mathcal{T}$, then there is a tree-shaped model of $C$ and $\mathcal{T}$.

Proof via unraveling:
Decidability of $\mathcal{ALC}$

Benefits of tree model property:

- tree models *computationally much simpler* than graph models
  recall, e.g., Rabin’s theorem
- there are *powerful tools* for logics on trees (e.g. automata, games)

**Theorem.** In $\mathcal{ALC}$, satisfiability (and subsumption) is ExpTime-complete.

Many kinds of algorithms, e.g. based on:

- tree automata *(ExpTime upper bound, best case exponential)*
- tableau calculus *(no ExpTime upper bound, used by most reasoners)*
- Pratt-style *type elimination* *(ExpTime upper bound, conceptually simple)*
ExpTime-hardness: reduce word problem of alternating Turing machines whose tape is bounded polynomially [FischerLadner79]

Central ideas:

- ATMs generalize non-deterministic TMs:
  - linear TM computations generalized to ATM computation trees
- alternating PSpace = ExpTime
- polysize tape can be represented using a single domain element
  - (concept names such as $A_{a,i}$, $A_{h,i}$, $A_q$)
- $\mathcal{ALC}$ tree models can represent ATM computation trees
From $\mathcal{ALC}$ to OWL

From an application perspective, the expressive power of $\mathcal{ALC}$ is limited

OWL enriches $\mathcal{ALC}$ in many ways, including:

- concepts ($\leq 1 \ r$) expressing local functionality of roles
  
  e.g. Disease $\sqcap \exists\text{hasCause}.\text{Infection} \sqcap (\leq 1 \ \text{hasCause})$

  formal semantics: $\forall y, y'. (r(x, y) \land r(x, y') \rightarrow y = y')$

- concepts ($\leq 1 \ r^{-}$) for the converse of roles

- nominals, a new sort that identifies a unique domain element
  
  e.g. Pope, SoccerWorldChampion, but possibly also Red, Blue

Call the resulting description logic OWL1 Core (DL Name: $\mathcal{ALCFIO}$)
In OWL1 Core, the tree model property is lost rather dramatically:

already in $\mathcal{ALC}$, we can easily generate a tree

\[
\begin{align*}
L_0 &\sqsubseteq \exists x. L_1 \cap \exists y. L_1 \\
L_1 &\sqsubseteq \exists x. L_2 \cap \exists y. L_2 \\
L_2 &\sqsubseteq \exists x. L_3 \cap \exists y. L_3
\end{align*}
\]
In OWL1 Core, the tree model property is **lost rather dramatically**: 

- already in $\mathcal{ALC}$, we can easily generate a tree
- now make $L_4$ a nominal

\[
\begin{align*}
L_0 & \subseteq \exists x. L_1 \land \exists y. L_1 \\
L_1 & \subseteq \exists x. L_2 \land \exists y. L_2 \\
L_2 & \subseteq \exists x. L_3 \land \exists y. L_3
\end{align*}
\]
In OWL1 Core, the tree model property is lost rather dramatically:

- already in $\mathcal{ALC}$, we can easily generate a tree
- now make $L_4$ a nominal
- make the converses of $x$ and $y$ functional

\[
L_0 \subseteq \exists x. L_1 \cap \exists y. L_1 \\
L_1 \subseteq \exists x. L_2 \cap \exists y. L_2 \\
L_2 \subseteq \exists x. L_3 \cap \exists y. L_3
\]
In OWL1 Core, the tree model property is lost rather dramatically:

- already in $\mathcal{ALC}$, we can easily generate a tree
- now make $L_4$ a nominal
- make the converses of $x$ and $y$ functional

$$L_0 \subseteq \exists x. L_1 \land \exists y. L_1$$
$$L_1 \subseteq \exists x. L_2 \land \exists y. L_2$$
$$L_2 \subseteq \exists x. L_3 \land \exists y. L_3$$
In OWL1 Core, the tree model property is **lost rather dramatically**: 

- already in $\mathcal{ALC}$, we can easily generate a tree 
- now make $L_4$ a nominal 
- make the converses of $x$ and $y$ functional

\[
\begin{align*}
L_0 &\subseteq \exists x. L_1 \cap \exists y. L_1 \\
L_1 &\subseteq \exists x. L_2 \cap \exists y. L_2 \\
L_2 &\subseteq \exists x. L_3 \cap \exists y. L_3 \\
L_4 &
\end{align*}
\]
In OWL1 Core, the tree model property is lost rather dramatically:

- already in $\mathcal{ALC}$, we can easily generate a tree
- now make $L_4$ a nominal
- make the converses of $x$ and $y$ functional

$$L_0 \subseteq \exists x. L_1 \cap \exists y. L_1$$
$$L_1 \subseteq \exists x. L_2 \cap \exists y. L_2$$
$$L_2 \subseteq \exists x. L_3 \cap \exists y. L_3$$
From $\mathcal{ALC}$ to OWL

Consequences:

- the tree model property is lost in a rather dramatic way

- grids can represent computations of non-deterministic Turing machines

- with a small trick, we can generate a grid of exponential size

  (count levels in binary, not in unary)

- it follows that OWL1Core is $\text{NExpTime}$-hard, in fact $\text{NExpTime}$-complete

  \cite{tobies}

In OWL2, we can even enforce grids of 2-exponential size

\[ \Rightarrow 2\text{NExpTime-completeness} \quad \text{\cite{kazakov}} \]
• OWL1 and OWL2 are rather expressive
  close to, and sometimes beyond the **2-variable fragment of FO**

• OWL1 and OWL2 are **computationally very costly** (worst case!)

• with the transition

  \[ ALC \rightarrow SHIQ \rightarrow OWL1 \rightarrow OWL2 \]

  the promise of efficiency on natural inputs got **increasingly untrue**

• there are applications and reasoning tasks where this is unacceptable
Lightweight Description Logics
A Bit of History

Stone age of description logics (until mid-1990ies):

“We have to offer efficient reasoning and thus cannot include all Booleans”

“Every application needs at least conjunction and universal restriction”

(and thus reasoning is co-NP-complete)

The $\text{SHIQ}$ era (since mid-1990ies until ??):

“ExpTime DLs can be implemented efficiently” (FaCT system by Horrocks)

“We do need the Booleans and much, much more (but want to stay decidable)!”
The $\mathcal{EL}$ and DL-Lite era (since $\approx 2005$):

“Applications need existential restrictions rather than universal ones”

“Lightweight DLs are sufficient for many applications and can be scalable”
The Description Logic EL

Dominating constructors in many large-scale ontologies:

conjunction and existential restrictions

```
Pericardium ⊑ Tissue ⊓ ∃partOf.Heart
Pericarditis ⊑ Inflammation ⊓ ∃location.Pericardium
Inflammation ⊑ Disease ⊓ ∃actsOn.Tissue
Tissue ⊓ Disease ⊑ ⊥
```

Large-scale ontologies usually require a highly abstract conceptual modeling
The Description Logic $\mathcal{EL}$

Concept language of $\mathcal{EL}$ is “half of $\mathcal{ALC}$”:

$$C ::= A \mid \top \mid \bot \mid C \cap D \mid \exists r.C$$

Most prominent $\mathcal{EL}$-ontology: SNOMED CT

- large scale, professionally developed medical ontology ($\sim 400,000$ concepts)
- used to systematize health care terminology, standard e.g. in US, Canada, etc.

Satisfiability and subsumption still interreducible:

- $C$ satisfiable w.r.t. $\mathcal{T}$ iff $\mathcal{T} \not\models C \subseteq \bot$
- $\mathcal{T} \models C \subseteq D$ iff $C \cap A$ unsatisfiable w.r.t. $\mathcal{T} \cup \{C \cap A \cap D \subseteq \bot\}$
Central notion for understanding expressive power of $\mathcal{EL}$:

Relation $\rho \subseteq \Delta^{\mathcal{I}_1} \times \Delta^{\mathcal{I}_2}$ is simulation from interpretation $\mathcal{I}_1$ to $\mathcal{I}_2$ if $d \rho d'$ implies that

- $d'$ satisfies all concept names that $d$ satisfies
- each successor of $d$ has $\rho$-related counterpart at $d'$
- nothing else

\[\mathcal{I} \quad \frac{d}{\rho} \quad \mathcal{I}'\]

$A, B$ 

$r$ 

$A, B, B'$ 

$r$
(\mathcal{I}_1, d_1) \preceq (\mathcal{I}_2, d_2): \text{ there is a simulation } \rho \text{ from } \mathcal{I}_1 \text{ to } \mathcal{I}_2

\text{ with } d_1 \rho d_2
Lemma. $\mathcal{EL}$ is preserved under simulations, i.e.,

if $(I_1, d_1) \sim (I_2, d_2)$, then $d_1 \in C_{I_1}$ implies $d_2 \in C_{I_2}$

for all $\mathcal{EL}$-concepts $C$.

Thus $\mathcal{EL}$ cannot distinguish $(I_1, d_1)$ from $(I_2, d_2)$ if they mutually simulate

This is not the same as bisimulation:
Since $\mathcal{EL}$ is a fragment of $\mathcal{ALC}$: $\mathcal{EL}$ has tree model property

But $\mathcal{EL}$ satisfies a much stronger property: it has canonical tree models

**Theorem.** If an $\mathcal{EL}$-concept $C$ is satisfiable w.r.t. an $\mathcal{EL}$-TBox $\mathcal{T}$, then there is a tree-shaped model $(\mathcal{M}, d)$ of $C$ and $\mathcal{T}$ such that for all models $\mathcal{I}$ of $\mathcal{T}$ and all $e \in C^\mathcal{I}$: $(\mathcal{M}, d) \preceq (\mathcal{I}, e)$

Intuition: the canonical model can be found in any other model (in terms of a simulation)
As an example, take

\[ C = A \cap \exists r \cdot B \quad \quad \mathcal{T} = \{ A \subseteq \exists s \cdot B \} \]

Models of \( \mathcal{T} \) e.g.:

Canonical model:
Canonical models can be constructed in a straightforward way:

\[ A \subseteq B_1 \quad B_1 \subseteq \exists r.B_1 \quad \exists r.B_1 \subseteq B_2 \]
\[ B_1 \cap B_2 \subseteq \exists s.B_2 \]

- This is a (tree) model of \( A \) and \( \mathcal{T} \)
- Everything we have generated must be present in every model of \( A \) and \( \mathcal{T} \)!
Due to $\bot$, the canonical model construction can fail

and that happens exactly when $C$ is unsatisfiable w.r.t. $\mathcal{T}$:

- If we derive $\bot$, then $\bot$ is a logical consequence of $C$ and $\mathcal{T}$
  thus $C$ is unsatisfiable w.r.t. $\mathcal{T}$

- If we do not derive $\bot$, then $\mathcal{M}$ is a model of $C$ and $\mathcal{T}$
  thus $C$ is satisfiable w.r.t. $\mathcal{T}$

This is the basis for a satisfiability algorithm in $\mathcal{EL}$. 
**Theorem.** In $\mathcal{EL}$, satisfiability (and subsumption) are in PTime.

[BaaderBrandtL__05]

Proof approach:

- We cannot construct the infinite tree-shaped model $\mathcal{M}$

- Instead use a *compact version* of the canonical model $\mathcal{M}_c$
Canonical models can be constructed in a straightforward way:

\[ A \subseteq B_1 \quad B_1 \subseteq \exists r. B_1 \quad \exists r. B_1 \subseteq B_2 \]
\[ B_1 \cap B_2 \subseteq \exists s. B_2 \]

- The unraveling of \( \mathcal{M}_c \) is exactly \( \mathcal{M} \)
  \[ \implies \] construction of \( \mathcal{M}_c \) fails iff construction of \( \mathcal{M} \) fails

- \( \mathcal{M}_c \) is of polynomial size, can be constructed in polynomial time
Some additional virtues of $\mathcal{M}_c$

- $\mathcal{M}_c$ is a model of $\mathcal{C}$ and $\mathcal{T}$, too.

- just like $\mathcal{M}$, $\mathcal{M}_c$ simulates every model of $\mathcal{C}$ and $\mathcal{T}$:

$$\sim \quad (\mathcal{I}, e)$$

$$\sim \quad (\mathcal{M}, d)$$

$$\sim \quad (\mathcal{M}_c, d)$$

**Theorem.** An FO-formula $\varphi$ with one free variable is equivalent to an $\mathcal{EL}$-concept iff it is preserved under simulation and has a canonical model.  

[PirolaL--Wolter10]
Extensions of $\mathcal{EL}$

PTime upper bound can be generalized to $\mathcal{EL}^{++}$, i.e., $\mathcal{EL}$ extended with

- **range restrictions** on roles, i.e., $T \sqsubseteq \forall r.C$

(  
- **domain restrictions** on roles, i.e., $T \sqsubseteq \forall r^{-}.C$
  )

- **role implications**, i.e., TBox statements $r_1 \circ \cdots \circ r_n \sqsubseteq r$

  \[ \cdots \]

Other extensions cause a jump back to ExpTime, e.g.

- **disjunctions** $C \sqcup D$

- **universal restrictions** $\forall r.C$

- **number restrictions** ($\geq 2^r$)

Interesting: no extension between PTime and ExpTime known (dichotomy?)
Extensions of $\mathcal{EL}$

**Theorem.** In $\mathcal{EL} + \sqcap$, satisfiability (and subsumption) are ExpTime-complete. [BaaderBrandtL_05]

Proof: **reduction from satisfiability** of concept name $A_0$ w.r.t. $\mathcal{ALC}$-TBox $T$

**Step 1:** Replace universal restrictions in $T$ with existential ones:

$$\forall r.C \quad \text{becomes} \quad \neg \exists r.\neg C$$

**Step 2:** Modify $T$ so that negation is applied only to concept names

$$A \sqsubseteq \exists s.(B' \sqcup \neg \exists r.B) \quad \text{becomes} \quad A \sqsubseteq \exists s.(B' \sqcup \neg X)$$

$$X \models \exists r.B$$

($X$ a fresh concept name)
Extensions of \( \mathcal{EL} \)

**Theorem.** In \( \mathcal{EL} + \sqcup \), satisfiability (and subsumption) are ExpTime-complete.
[BaaderBrandtL_05]

**Proof:** reduction from satisfiability of concept name \( A_0 \) w.r.t. \( \mathcal{ALC} \)-TBox \( T \)

**Step 3:** Remove negation entirely from \( T \)

- Replace each \( \neg X \) with \( \overline{X} \), \( \overline{X} \) a fresh concept name
- Ensure correct behaviour of \( \overline{X} \):

\[
T \subseteq X \cup \overline{X} \\
X \cap \overline{X} \subseteq \bot
\]

Resulting TBox \( T' \) is in \( \mathcal{EL} + \sqcup \) and \( A_0 \) sat w.r.t. \( T \) iff \( A_0 \) sat w.r.t. \( T' \)
Theorem. In $\mathcal{EL} + \forall r.C$ and $\mathcal{EL} + (\geq 2 r)$, satisfiability is ExpTime-complete.

[BaaderBrandtL\_05]

Proof: reduction from satisfiability of concept name $A_0$ w.r.t. $\mathcal{EL} + \sqcap$-TBox $\mathcal{T}$

We can assume that disjunction occurs only in the form

$$A_1 \sqcup A_2 \subseteq A$$

and

$$A \subseteq B_1 \sqcup B_2$$

\[= A_1 \sqsubseteq A, A_2 \sqsubseteq A\]  \hspace{1cm} \text{replace by}

$$A \sqcap \exists r.\top \sqsubseteq B_1$$

$$A \sqcap \forall r.X \sqsubseteq B_2$$

\[r, X \text{ fresh}\]
Theorem. In $\mathcal{EL} + \forall r. C$ and $\mathcal{EL} + (\geq 2 r)$, satisfiability is ExpTime-complete.

[BaaderBrandtL__05]

Proof: reduction from satisfiability of concept name $A_0$ w.r.t. $\mathcal{EL} + \sqcup$-$TBox T$

We can assume that disjunction occurs only in the form

\[
A_1 \sqcup A_2 \sqsubset A \quad \text{and} \quad A \sqsubset B_1 \sqcup B_2
\]

\[
= A_1 \sqsubset A, \ A_2 \sqsubset A \quad \text{replace by}
\]

\[
A \sqsubset \exists r. X \sqcap \exists r. Y
\]

\[
A \sqcap \exists r. (X \sqcap Y) \sqsubseteq B_1 \quad \text{r, X, Y fresh}
\]

\[
A \sqcap (\geq 2 r) \sqsubseteq B_2
\]
Call an extension of $\mathcal{EL}$ convex if:

$$\mathcal{T} \models C \subseteq D_1 \cup D_2 \quad \text{implies} \quad \mathcal{T} \models C \subseteq D_i \quad \text{for some} \ i \in \{1, 2\}$$

$\mathcal{EL} + \forall r.C$ is not convex:

$$\emptyset \not\models T \subseteq \exists r. T \cup \forall r. X,$$ but $\emptyset \not\models T \subseteq \exists r. T$ and $\emptyset \not\models T \subseteq \forall r. X$

The reductions show: if an extension of $\mathcal{EL}$ is not convex, it is ExpTime-hard.

Interestingly, the converse does not hold!

Easy to prove:

Existence of canonical models $\mathcal{M}$ implies convexity:
Extensions of $\mathcal{EL}$

Consider $\mathcal{EL}$ extended with inverse existential restrictions:

$$\exists r^- C \text{ has semantics } \{ d \in \Delta^I \mid \exists e \in C^I : (e, d) \in r^I \}$$

**Theorem.** $\mathcal{EL} + \exists r^- C$ is convex, but satisfiability is ExpTime-complete.

[BaaderBrandtL'05]

Here only: canonical models can become exponentially large

$$L_0 \sqsubseteq \exists r. (L_1 \cap A_1) \cap \exists r. (L_1 \cap \overline{A_1})$$

$$L_1 \sqsubseteq \exists r. (L_2 \cap A_2) \cap \exists r. (L_2 \cap \overline{A_2})$$

$$L_2 \cap \exists r^- A_1 \sqsubseteq A_1$$

$$L_2 \cap \exists r^- \overline{A_1} \sqsubseteq \overline{A_1}$$

Merging leaves destroys canonicity!
Discussion

- $\mathcal{EL}$ is a natural ontology language for a high level of abstraction

- satisfiability and subsumption can be computed in polytime

- this has led to standardization as OWL EL profile of OWL2

- efficient reasoners are available, e.g. CEL (Dresden), SnoRocket (Brisbane)
  based on canonical models, very robust, classify SNOMED CT in $< 10$ min

- algorithms have been generalized to Horn-$SHIQ$

reasoner CB (Oxford)
The historic choice of **universal restrictions** instead of **existential restrictions** leads to much worse computational behaviour.

Complexity of **subsumption** in $\mathcal{FL}_0$, constructors $\top$, $(\bot)$, $\sqcap$, $\forall r.C$:

- **empty TBox**: tractable [BrachmanLevesque84]
- **acyclic TBox**: co-NP-complete [Nebel90]
- **cyclic TBox**: PSpace-complete [KazakovDeNivelle03]
- **general TBox**: ExpTime-complete [BaaderBrandtL_05,Hofmann05]
A Glimpse at $\mathcal{FL}_0$

Clearly,

$$\forall r. (A \sqcap B) \equiv \forall r. A \sqcap \forall r. B$$

Thus, every $\mathcal{FL}_0$-concept is equivalent to one of the form

$$\forall r_{1,1}. \forall r_{1,2}. \cdots \forall r_{1,n_1}. A_1$$

$$\sqcap \forall r_{2,1}. \forall r_{2,2}. \cdots \forall r_{2,n_2}. A_2$$

$$\cdots$$

$$\sqcap \forall r_{k,1}. \forall r_{k,2}. \cdots \forall r_{k,n_k}. A_k$$

(finite) words over the alphabet of role names

Grouping according to concept name achieves the following normal form

$$\forall L_1. A_1 \sqcap \forall L_2. A_2 \sqcap \cdots \sqcap \forall L_m. A_m$$

finite formal languages over the alphabet of role names
We consider subsumption instead of satisfiability

Subsumption in $\mathcal{FL}_0$ (without TBoxes):

\[ C = \forall L_1.A_1 \sqcap \forall L_2.A_2 \sqcap \cdots \sqcap \forall L_m.A_m \]

\[ D = \forall M_1.A_1 \sqcap \forall M_2.A_2 \sqcap \cdots \sqcap \forall M_m.A_m \]

Then $C \sqsubseteq D$ iff $L_i \supseteq M_i$ for $1 \leq i \leq m$ \hspace{1cm} (\!*\!)

Theorem. Subsumption in $\mathcal{FL}_0$ without TBoxes is in PTime.

Intuitively (\!*\!) still holds with acyclic TBoxes,

but sets $L_i$ can be described compactly, get exponentially large
A Glimpse at $\mathcal{FL}_0$

Reduction from 3SAT to $\mathcal{FL}_0$-subsumption w.r.t. TBoxes:

Take a 3-formula

$$\varphi = (l_{1,1} \lor l_{1,2} \lor l_{1,3}) \land \cdots \land (l_{n,1} \lor l_{n,2} \lor l_{n,3})$$

over the variables $x_1, \ldots, x_k$

Ideas:

- use two role names $t$ and $f$ representing “true” and “false”
- represent truth assignments as words over $\{t, f\}$ of length $k$
- as the target subsumption $C \sqsubseteq D$, use

$$C = \forall L_C.A, \quad L_C \text{ the set of truth assignments that make } \varphi \text{ false}$$

$$D = \forall L_D.A, \quad L_D \text{ the set of all truth assignments}$$
To be done: describe $C$ and $D$ with polynomial-size TBox:

$D$ is easy:

$$L_i \equiv \forall t. L_{i+1} \cap \forall f. L_{i+1} \quad \text{for } 1 \leq i \leq n$$

$$L_{n+1} \equiv A$$

$$D \equiv L_0$$

$C$ too (basically)

**Theorem.** Subsumption in $\mathcal{FL}_0$ w.r.t. TBoxes is co-NP-hard.
Instance Data and Query Answering
In recent years, exciting new reasoning problems have popped up; e.g.:

- **conjunctive query answering** over instance data w.r.t. a background TBox

- problems related to the **modularity of TBoxes**:
  - does a given subset \( T' \subseteq T \) say **everything** about a given signature \( \Sigma \) that \( T \) does?
  - given a signature \( \Sigma \), **extract an as-small-as-possible subset** \( T' \subseteq T \) that says the same about \( \Sigma \) as \( T \)

- problems related to **privacy issues**
  - e.g. controlled interfaces to TBox / instance data
Ontologies are increasingly used with instance data, e.g.:

Clinical document architecture (CDA) becomes standard medical data format
CDA medical codes based on SNOMED CT terminology

Ontology can be exploited for interpreting data / deriving additional answers

**ABox**: finite set of ground facts, e.g.:

| Patient(p) | finding(p, d) | Pericarditis(d) |

Information in ABoxes is incomplete (open world semantics)
E.g., a patient record would not include Inflammation(d), though it is true.
TBox allows more complete query answers

\[
\begin{align*}
\text{ABox} & : \quad \text{Patient}(p) \quad \text{Inpatient}(p) \\
 & \quad \text{inWard}(p, w) \quad \neg \text{Intensive}(w)
\end{align*}
\]

\[
\begin{align*}
\text{TBox} & : \quad \text{Inpatient} \sqsubseteq \exists \text{finding}.\text{Disease} \\
 & \quad \exists \text{inWard}.\neg \text{Intensive} \sqsubseteq \forall \text{finding}.\neg \text{LiveThreatening}
\end{align*}
\]

Then \( p \) is an answer to query

\[\exists y. \text{Patient}(x) \land \text{finding}(x, y) \land \neg \text{LiveThreatening}(y)\]
More formally:

- **Model of ABox** $\mathcal{A}$: interpretation satisfying all facts in $\mathcal{A}$

- **Answers** to query $q$ for ABox $\mathcal{A}$ w.r.t. TBox $\mathcal{T}$:
  
  *Certain answers*, i.e., answers common to all models $\mathcal{I}$ of $\mathcal{A}$ and $\mathcal{T}$

Closely related to query answering in incomplete databases

  (but with a different kind of schema constraints)
Query Languages

- **Instance queries**
  take the form $C(v)$, $v$ a variable.
  technically close to subsumption, almost always of same complexity

- **Conjunctive queries**
  take the form $\exists \bar{v} . \varphi(\bar{v}, \bar{v}')$, with $\varphi$ a conjunction of atoms $A(v)$ or $r(v, v')$
  $\bar{v}'$ the answer variables, $\bar{v}$ the quantified variables

  generalize instance queries, but are more interesting
  Select-Project-Join fragment of SQL

- **FO/SQL queries**
  generalize conjunctive queries, but: FO sentence $\varphi$ valid iff $\emptyset, \emptyset \models \varphi$

  TBox     ABox
In patient databases and other large-scale applications:

- **Efficiency and scalability** of query answering is crucial
- Query answering in **expressive DLs** is computationally costly

<table>
<thead>
<tr>
<th></th>
<th>satisfiability</th>
<th>query answering</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \mathcal{ALC} )</td>
<td>( \text{ExpTime} )</td>
<td>( \text{ExpTime} )</td>
</tr>
<tr>
<td>( \mathcal{ALC} + \exists r^{-}.C )</td>
<td>( \text{ExpTime} )</td>
<td>( 2\text{ExpTime} )</td>
</tr>
<tr>
<td>( \mathcal{SHIQ} )</td>
<td>( \text{ExpTime} )</td>
<td>( 2\text{ExpTime} )</td>
</tr>
<tr>
<td>OWL1 Core</td>
<td>( \text{NExpTime} )</td>
<td>decidable</td>
</tr>
<tr>
<td>OWL1</td>
<td>( \text{NExpTime} )</td>
<td>decidability open</td>
</tr>
</tbody>
</table>
Most popular approach to achieve scalability:

Implement DL query answering based on relational database systems

Obvious problem: conventional RDBM unaware of TBoxes

- Solution I: query rewriting — “put TBox into query”
- Solution II: data completion — “put TBox into data”
Solution 1: query rewriting — “put TBox into query”
The query rewriting approach: [Calvanese, deGiacomo, Lenzerini et al.05]

- ABox stored in DB system as relational instance
- CQ is rewritten to FO/SQL query to incorporate TBox
- Rewritten query executed by relational DB system

Enables use of off-the-shelf DB systems!

Mission statement: given CQ $q$ and $\mathcal{T}$, rewrite $q$ into FO query $q'$ such that

$$\mathcal{A}, \mathcal{T} \models q[a_1, \ldots, a_n] \text{ iff } db_{\mathcal{A}} \models q'[a_1, \ldots, a_n] \text{ for all } \mathcal{A}, a_1, \ldots, a_n.$$
Query Rewriting—Example 1

Query: $\exists y. (A(x) \land r(x, y) \land B(y))$

TBox: $\exists s. \top \subseteq A$  $B' \subseteq B$

Rewritten query is disjunction of:

$A \overset{r}{\longrightarrow} B$

$s \overset{r}{\longleftarrow} B$

$A \overset{r}{\longrightarrow} B'$

$s \overset{r}{\longleftarrow} B'$
Query Rewriting—Example 2

Query

TBox

Rewritten query is disjunction of:

For which DLs does this work?
Query Rewriting

Data complexity:
- In DBs: measure complexity only in size of data, not of query
- In DLs: measure complexity only in size of data, neither of query nor TBox

Theorem. The query rewriting approach only works for DLs for which
CQ entailment is in AC₀ regarding data complexity. [Calvanese et al. 05]

Proof:
- FO query answering is in AC₀ regarding data complexity
- measured input (data) is left unchanged
- measured / non-measured inputs are not mixed in the rewriting
Query Rewriting

<table>
<thead>
<tr>
<th>Data complexity of DLs we have met:</th>
<th>$\mathcal{EL}$</th>
<th>$\mathcal{ALC}$ and above</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PTime-complete</td>
<td>NP-complete</td>
</tr>
</tbody>
</table>

Why query rewriting cannot be used for $\mathcal{EL}$:

Query

\[ A \quad A(x) \]

TBox

\[ \exists r. A \sqsubseteq A \]

Rewritten query is disjunction of:

\[ A \quad r \quad A \quad r \quad r \quad A \quad \ldots \]

We need $\exists y. r^*(x, y) \land A(y)$, but transitive closure not FO-expressible
**DL-Lite**: a lightweight DL with $AC_0$ data complexity [Calvanese et al.05]

Basic version: TBox statements of the form

$$C \sqsubseteq D \quad C \sqsubseteq \neg D$$

where $C, D$ are of the form $A, \exists r. T$, and $\exists r^- T$

For example: $\text{Professor} \sqsubseteq \exists \text{teachesTo}. T \quad \exists \text{teachesTo}^- T \sqsubseteq \text{Student}$

    $\text{Professor} \sqsubseteq \neg \text{Student}$

**DL-Lite:**

- Inexpressive, but can encode ER diagrams and UML class diagrams
- Admits the query rewriting approach
- Underlies OWL QL profile of OWL2
Solution II: data completion — “put TBox into data”
Limitations of the query rewriting approach:

- Works only for $AC_0$-DLs, i.e., only for DL-Lite

- Query rewriting blows up exponentially $O(|\mathcal{I}|^{|q|})$ performance problems with large queries / large TBoxes

The data completion approach avoids both problems

in particular, it works for $\mathcal{EL}$-TBoxes
The data completion approach: [L__TomanWolter08]

- Incorporate TBox into the ABox, not into the query

- To deal with existential restrictions and avoid infinite databases:
  - eagerly reuse constants, producing spurious cycles (and more)
  - (similar to compact canonical model vs. canonical tree model)

- To nevertheless obtain correct answers: use query rewriting

Also enables use of off-the-shelf DB systems!
Data Completion—Example 1

TBox
\[ \exists r. A \sqsubseteq A \]

ABox
\[ a \xrightarrow{r} b \xrightarrow{r} c \xrightarrow{r} d \]

Completed ABox:
\[ A \]
\[ a \xrightarrow{r} b \xrightarrow{r} c \xrightarrow{r} d \]
\[ e, A \]

Query
\[ A(x) \]

Answer
\[ a, b, c, e \]
**Data Completion—Example 2**

**TBox**

\[
A \sqsubseteq \exists s.B \quad \exists s.B \sqsubseteq A' \quad \exists r.(A \cap A') \sqsubseteq B
\]

**ABox**

\[
a \xrightarrow{r} b
\]

**Completed ABox:**

\[
B \xrightarrow{r} A, A'
\]

\[
a \xrightarrow{r} b \xrightarrow{s} c \quad B, \text{Ex}
\]

**Query**

\[B(v)\]

**Answer**

\[a, c\]

**Rewritten query**

\[B(v) \land \neg \text{Ex}(v)\]

**Answer**

\[a\]
Data Completion—Example 2

TBox

\[ A \sqsubseteq \exists s. B \quad \exists s. B \sqsubseteq A' \quad \exists r. (A \cap A') \sqsubseteq B \]

ABox

\[ a \xrightarrow{r} b \]

Completed ABox:

\[ B \xrightarrow{r} A, A' \]

\[ a \xrightarrow{r} b \]

\[ s \]

\[ c \quad B, \text{ Ex} \]

ABox completion means building the canonical model

(for an ABox instead of for a concept)
Data Completion—Example 2

General shape of canonical model built for an ABox:

Problem: canonical model can get infinite, database can’t
Data Completion—Example 3

TBox

\[ A \sqsubseteq \exists r.A \]

ABox

\[ A \]
\[ a \]

Completed ABox:

\[ A \xrightarrow{r} A' \xrightarrow{r} A'' \]

Database cannot be infinite.

⇒ build compact canonical model!
Data Completion—Example 3

TBox

\[ A \subseteq \exists r.A \]

ABox

\[ A a \]

\[ r \]

\[ A b \]

Completed ABox:

\[ A a \]

\[ r \]

\[ A b \]

\[ \text{Wrong answer to some queries, e.g.} \]

\[ \exists y. r(x, y) \land r(y, y) \]

answer \( \{a, b\} \), should be \( \emptyset \)

\[ \exists y. r(x, y) \land r(x', y) \land r(x, x') \]

answer \( \{(a, b)\} \), should be \( \emptyset \)
Problem:

infinite, tree-shaped canonical model $\mathcal{M}$ gives correct answers to all queries, compact version $\mathcal{M}_c$ does not

Solution:

Rewrite CQ $q$ into FO query $q'$ so that

answers to $q'$ in $\mathcal{M}_c =$ answers to $q$ in $\mathcal{M}$

Implementation: add query conjuncts expressing that

- Variable on a query cycle cannot be mapped to an Ex element
- If $r(x, y), s(x', y)$ in query and $r \neq s$, then $y$ not mapped to Ex
- If $r(x, y), r(x', y)$ in query and $y$ mapped to Ex, then $x = x'$
Data Completion—Example 3

TBox

\[ A \sqsubseteq \exists r.A \]

ABox

\[
\begin{align*}
A & a \\
 r & \\
A & b \\
\end{align*}
\]

Completed ABox:

\[
\begin{align*}
A & a \\
r & \\
A & b \\
A & c \\
r & \\
Ex & \\
\end{align*}
\]

\[ q = \exists y. r(x, y) \land r(y, y) \]  
answer \{a, b\}

\[ q' = \exists y. r(x, y) \land r(y, y) \land \neg\text{Ex}(x) \land \neg\text{Ex}(y) \]  
answer \emptyset
Data Completion—Example 3

TBox

\[ A \sqsubseteq \exists r . A \]

ABox

\[ A \ a \]

\[ r \]

\[ A \ b \]

Completed ABox:

\[ A \ a \]

\[ r \]

\[ A \ c \]

\[ r \]

\[ A \ b \]

\[ r \] \[ A \ b \]

\[ Ex \]

\[ r \]

\[ A \]

\[ r \]

\[ A \ c \]

\[ r \]

\[ Ex \]

\[ r \]

\[ q = \exists y . r(x, y) \land r(x', y) \land r(x, x') \] answer \( \{(a, b)\} \)

\[ q' = \exists y . r(x, y) \land r(x', y) \land r(x, x') \] \land \neg\text{Ex}(x) \land \neg\text{Ex}(x') \land (\text{Ex}(y) \rightarrow x = x') \] answer \( \emptyset \)
Data Completion

Wrapup:

- Data completion approach works for EL and DL-Lite [KR10], results only in polynomial blowup of the query

- Requires authority over the data, blows up the data (polynomially)

- Extends to role hierarchies, domain and range restrictions
  (but transitive roles and general role inclusions are challenging)

- Limitation: for DLs whose data complexity is not in PTime
  there must be a (worst case) exponential blowup of the data
Questions?

PS: Slides are on my homepage

PPS: Somebody interested in a PhD/Postdoc position?