OWL and tractability

Based on slides from
Ian Horrocks and Franz Baader
Where are we?

- XML
- RDF(S)/SPARQL
- PL/FOL
- OWL
- DL Extensions
- OWL Reasoning
- OWL in practice
- Scalability
- Practical Topics
Repetition: DL tableaux algorithm

- How many Rules?
- Main data structure?
- Open ABox?
- Complete ABox?
- How to deal with TBox-Axioms?
- Order of rule application: does it matter?
Today

• DLs
  – Adding GCI’s: blocking
  – Adding number restrictions

• OWL (1.1)

• Tractability
  – Lightweight sub-languages
  – (Polynomial time) decision procedures for these languages
A finite set of GCIs can be encoded into one GCI of the form $\top \subseteq C$:

$$\{C_1 \subseteq D_1, \ldots, C_n \subseteq D_n\} \quad \rightarrow \quad \{\top \subseteq (\neg C_1 \cup D_1) \cap \ldots \cap (\neg C_n \cup D_n)\}$$

Consider a GCI $\top \subseteq C$ where $C$ is in NNF.

The GCI-rule for $\top \subseteq C$

**Condition:** $\mathcal{A}$ contains the individual name $a$, but not $C(a)$

**Action:** $\mathcal{A}' := \mathcal{A} \cup \{C(a)\}$
Adding GCIs

- local correctness, completeness, and soundness are easy to show
- termination does not hold:

Test consistency of \( \{P(a)\} \) w.r.t. the GCI \( \top \subseteq \exists r. P \)

Solution: blocking

- \( y \) is blocked by \( x \) iff \( L(y) \subseteq L(x) \)
- to avoid cyclic blocking we fix an enumeration of the individual names, and add to the blocking condition that \( y \) comes after \( x \) in the enumeration
- generating rules are not applied to blocked individuals
• Adding GCIs

consistency of \{(\forall r. Q)(a), \ P(a)\}
w.r.t. the GCI \( \top \subseteq \exists r. P \)

does this yield a decision procedure?

blocked by a
• Adding number restrictions

Number restrictions: \((\geq n \cdot r.C), (\leq n \cdot r.C)\) with semantics

\[
\begin{align*}
(\geq n \cdot r.C)^I & := \{d \in \Delta^I \mid \text{card}(\{e \mid (d, e) \in r^I \land e \in C^I\}) \geq n\} \\
(\leq n \cdot r.C)^I & := \{d \in \Delta^I \mid \text{card}(\{e \mid (d, e) \in r^I \land e \in C^I\}) \leq n\}
\end{align*}
\]

Negation normal form:

\[
\neg(\geq n + 1 \cdot r.C) \iff (\leq n \cdot r.C) \\
\neg(\geq 0 \cdot r.C) \iff \bot \\
\neg(\leq n \cdot r.C) \iff (\geq n + 1 \cdot r.C)
\]

Extension of algorithm:

• new rules: \(\geq\)-rule and \(\leq\)-rule

• new assertions: inequality assertions of the form \(x \neq y\) with obvious semantics \(x^I \neq y^I\) viewed as symmetric

• new obvious contradictions
Adding number restrictions

The $\geq$-rule

**Condition:** $\mathcal{A}$ contains $(\geq n \ r. C)(a)$, but there are no $c_1, \ldots, c_n$ with
$$\{r(a, c_1), C(c_1), \ldots, r(a, c_n), C(c_n)\} \cup \{c_i \neq c_j \mid 1 \leq i < j \leq n\} \subseteq \mathcal{A}$$

**Action:** $\mathcal{A}' := \mathcal{A} \cup \{r(a, b_1), C(b_1), \ldots, r(a, b_n), C(b_n)\} \cup \{b_i \neq b_j \mid 1 \leq i < j \leq n\}$

where $b_1, \ldots, b_n$ are new individual names

The $\leq$-rule

**Condition:** $\mathcal{A}$ contains $(\leq n \ r. C)(a)$, and there are $b_1, \ldots, b_{n+1}$ with
$$\{r(a, b_1), C(b_1), \ldots, r(a, b_{n+1}), C(b_{n+1})\} \subseteq \mathcal{A},$$
but $\{b_i \neq b_j \mid 1 \leq i < j \leq n + 1\} \not\subseteq \mathcal{A}$

**Action:** for all $i < j$ with $b_i \neq b_j \not\in \mathcal{A}$

$\mathcal{A}_{i,j} := \mathcal{A}[b_i \leftarrow b_j]$ where $b_i$ replaced by $b_j$
Combining the strengths of UMIST and The Victoria University of Manchester

- Adding number restrictions

- \( \mathcal{A} \) contains \( (\leq n \, r.C)(a) \), and there are \( b_1, \ldots, b_{n+1} \) with

\[
\{ r(a, b_1), C(b_1), \ldots, r(a, b_{n+1}), C(b_{n+1}) \} \subseteq \mathcal{A}
\]

and

\[
\{ b_i \neq b_j \mid 1 \leq i < j \leq n + 1 \} \subseteq \mathcal{A}
\]

- \( \mathcal{A} \) contains \( a \neq a \) for some individual name \( a \)
• For $\mathcal{ALC}$, the subsumption problem and the instance problem are PSpace-complete.

The **tableau algorithm** as described needs exponential space, but it can be modified such that it needs only polynomial space.

• Both TBoxes and number restrictions can be added without increasing the complexity.

• W.r.t. general TBoxes, the subsumption and the instance problem are ExpTime-complete.

**Tableau algorithms** do not “easily” yield this upper bound, but they are more practical than the worst-case optimal automata-based algorithms.

• The **tableau algorithms** implemented in systems like FaCT and Racer are highly optimized, and behave quite well on large knowledge bases.
DL Complexity Navigator ...
Combining the strengths of UMIST and The Victoria University of Manchester

OWL
Use a (Description) Logic

- OWL DL based on $\text{SHIQ}$ Description Logic
  - In fact it is equivalent to $\text{SHOIN(D)}$ DL
- OWL DL Benefits from many years of DL research
  - Well defined semantics
  - Formal properties well understood (complexity, decidability)
  - Known reasoning algorithms
  -Implemented systems (highly optimised)
- In fact there are three “species” of OWL (!)
  - OWL full is union of OWL syntax and RDF
  - OWL DL restricted to First Order fragment
  - OWL Lite is “simpler” subset of OWL DL
## Class/Concept Constructors

<table>
<thead>
<tr>
<th>Constructor</th>
<th>DL Syntax</th>
<th>Example</th>
<th>FOL Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>intersectionOf</td>
<td>$C_1 \sqcap \ldots \sqcap C_n$</td>
<td>Human $\sqcap$ Male</td>
<td>$C_1(x) \land \ldots \land C_n(x)$</td>
</tr>
<tr>
<td>unionOf</td>
<td>$C_1 \sqcup \ldots \sqcup C_n$</td>
<td>Doctor $\sqcup$ Lawyer</td>
<td>$C_1(x) \lor \ldots \lor C_n(x)$</td>
</tr>
<tr>
<td>complementOf</td>
<td>$\neg C$</td>
<td>$\neg$ Male</td>
<td>$\neg C(x)$</td>
</tr>
<tr>
<td>oneOf</td>
<td>${x_1} \sqcup \ldots \sqcup {x_n}$</td>
<td>{john} $\sqcup$ {mary}</td>
<td>$x = x_1 \lor \ldots \lor x = x_n$</td>
</tr>
<tr>
<td>allValuesFrom</td>
<td>$\forall P.C$</td>
<td>$\forall$ hasChild.Doctor</td>
<td>$\forall y.P(x,y) \rightarrow C(y)$</td>
</tr>
<tr>
<td>someValuesFrom</td>
<td>$\exists P.C$</td>
<td>$\exists$ hasChild.Lawyer</td>
<td>$\exists y.P(x,y) \land C(y)$</td>
</tr>
<tr>
<td>maxCardinality</td>
<td>$\leq n P$</td>
<td>$\leq$ 1 hasChild</td>
<td>$\exists y.P(x,y)$</td>
</tr>
<tr>
<td>minCardinality</td>
<td>$\geq n P$</td>
<td>$\geq$ 2 hasChild</td>
<td>$\exists \leq n y.P(x,y)$</td>
</tr>
</tbody>
</table>

- $C$ is a concept (class); $P$ is a role (property); $x$ is an individual name
RDFS Syntax

E.g., Person \( \sqcap \forall \text{hasChild.}(\text{Doctor} \sqcup \exists \text{hasChild.}\text{Doctor}) \):

```xml
<owl:Class>
  <owl:intersectionOf rdf:parseType=" collection">
    <owl:Class rdf:about="#Person"/>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasChild"/>
      <owl:toClass>
        <owl:unionOf rdf:parseType=" collection">
          <owl:Class rdf:about="#Doctor"/>
          <owl:Restriction>
            <owl:onProperty rdf:resource="#hasChild"/>
            <owl:hasClass rdf:resource="#Doctor"/>
          </owl:Restriction>
        </owl:unionOf>
      </owl:toClass>
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>
```
Task

• Define an OWL class expression for the all females who have a husband which works as a lawyer.
Ontologies / Knowledge Bases

• **OWL ontology** equivalent to a DL Knowledge Base

• **OWL ontology** consists of a set of **axioms and facts**
  – *Note: an ontology is usually thought of as containing only Tbox axioms (schema)---OWL is non-standard in this respect*

• Recall that a DL KB $\mathcal{K}$ is a pair $\langle \mathcal{T}, \mathcal{A} \rangle$ where
  – $\mathcal{T}$ is a set of “terminological” axioms (the Tbox)
  – $\mathcal{A}$ is a set of “assertional” axioms (the Abox)
## Ontology/Tbox Axioms

<table>
<thead>
<tr>
<th>OWL Syntax</th>
<th>DL Syntax</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>subClassOf</td>
<td>$C_1 \sqsubseteq C_2$</td>
<td>Human $\sqsubseteq$ Animal $\sqcap$ Biped</td>
</tr>
<tr>
<td>equivalentClass</td>
<td>$C_1 \equiv C_2$</td>
<td>Man $\equiv$ Human $\sqcap$ Male</td>
</tr>
<tr>
<td>subPropertyOf</td>
<td>$P_1 \sqsubseteq P_2$</td>
<td>hasDaughter $\sqsubseteq$ hasChild</td>
</tr>
<tr>
<td>equivalentProperty</td>
<td>$P_1 \equiv P_2$</td>
<td>cost $\equiv$ price</td>
</tr>
<tr>
<td>transitiveProperty</td>
<td>$P^+ \sqsubseteq P$</td>
<td>ancestor$^+$ $\sqsubseteq$ ancestor</td>
</tr>
</tbody>
</table>
### Ontology Facts / Abox Axioms

<table>
<thead>
<tr>
<th>OWL Syntax</th>
<th>DL Syntax</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>$a : C'$</td>
<td>John : Happy-Father</td>
</tr>
<tr>
<td>property</td>
<td>$\langle a, b \rangle : R$</td>
<td>$\langle John, Mary \rangle : \text{has-child}$</td>
</tr>
</tbody>
</table>

- Note: using **nominals** (e.g., in $\text{SHOIN}$), can reduce Abox axioms to concept inclusion axioms
  - $a : C'$ equivalent to $\{a\} \sqsubseteq C$
  - $\langle a, b \rangle : R$ equivalent to $\{a\} \sqsubseteq \exists R.\{b\}$
Features of OWL language layers

• OWL Lite
  – (sub)classes, individuals
  – (sub)properties, domain, range
  – conjunction
  – (in)equality
  – cardinality 0/1
  – datatypes
  – inverse, transitive, symmetric properties
  – someValuesFrom
  – allValuesFrom

• OWL DL
  – Negation
  – Disjunction
  – Full cardinality
  – Enumerated types
  – hasValue

• OWL Full
  – Meta-classes
  – Modify language
Intractability – one main problem of Semantic Web technologies
Big problem: intractability

• All OWL profiles turned out intractable in the end

• Challenge:
  – find DLs for which reasoning is tractable,
  – i.e, which have polynomial-time decision procedures

• Candidates?
Task

• Order the constructors of description logic SHOIQ by what you feel their contribution to complexity is; most hard constructor first!
Big problem: intractability

- Tractable DLs cannot allow for all Boolean operators: satisfiability in propositional logic is already NP-complete

- Conjunction ($\sqcap$) is indispensable: otherwise one cannot require several properties simultaneously

- Negation plus conjunction is propositionally complete: full negation must be disallowed

- No DL without roles: either value or existential restrictions should be present

Two minimal DLs satisfying these requirements:

- $\mathcal{FL}_0$: conjunction ($\sqcap$) and value restrictions ($\forall r.C$)

- $\mathcal{EL}$: conjunction ($\sqcap$) and existential restrictions ($\exists r.C$)
FL0 (conjunction and forall)

Satisfiability: is trivial

\[
\text{every } \mathcal{FL}_0\text{-concept description is satisfiable}
\]

just interpret all concept names as the whole domain

Subsumption: is the interesting inference problem for \( \mathcal{FL}_0\)-concept descriptions

\[
\forall r. (\forall s. B \cap \forall s. \forall r. A) \cap \forall r. (A \cap B) \subseteq \forall r. (A \cap \forall s. (B \cap \forall r. A))
\]

Structural subsumption algorithm:

1. Normalize the concept descriptions
2. Compare the structure of the normalized descriptions
Normalization of $\mathcal{FL}_0$-concept descriptions proceeds in several steps. 

equivalence preserving
Combining the strengths of UMIST and The Victoria University of Manchester

**Step 2:** use words over $N_R$

$n = 0: \ w = \varepsilon$
• Normal form of $\mathcal{FL}_0$-concept descriptions

\[ NF(C) = \forall L_1.A_1 \cap \ldots \cap \forall L_m.A_m \]

\[ C \subseteq D \]

*can be checked in polynomial time*
Combining the strengths of UMIST and The Victoria University of Manchester

• Extension to acyclic TBoxes

Subsumption in $\mathcal{FL}_0$ w.r.t. acyclic TBoxes corresponds to the inclusion problem for acyclic finite automata.

Inclusion problem:

Given two acyclic finite automata $\mathcal{A}, \mathcal{B}$

Question does $L(\mathcal{A}) \subseteq L(\mathcal{B})$ hold

$\text{coNP-complete}$

i.e., non-inclusion is $\text{NP-complete}$

Note:

Acyclic automata define finite sets of words, however, they can do this exponentially more succinct than the enumeration of all elements

\[
\begin{array}{c}
\text{a} \\
\text{b}
\end{array} 
\rightarrow \begin{array}{c}
\text{a} \\
\text{b}
\end{array} 
\rightarrow \ldots \rightarrow 
\begin{array}{c}
\text{a} \\
\text{b}
\end{array} 
\rightarrow 
\begin{array}{c}
\text{a} \\
\text{b}
\end{array} 
\rightarrow 
\begin{array}{c}
\text{a} \\
\text{b}
\end{array}
\]
• From acyclic TBoxes to acyclic automata

\[
A \equiv \forall r.B \sqcap \forall r.C \\
B \equiv \forall r.C \\
C \equiv \forall s.P
\]
Combining the strengths of UMIST and The Victoria University of Manchester

• From acyclic TBoxes to acyclic automata

\[ A \equiv \forall r. B \sqcap \forall r. C \]
\[ B \equiv \forall r. C \]
\[ C \equiv \forall s. P \]

Complications:
\[ A \equiv \forall r. \forall s. B \sqcap \ldots \quad \text{introduce auxiliary states} \]
\[ A \equiv B \sqcap \ldots \quad \text{introduce and then eliminate } \varepsilon\text{-transitions} \]

Characterization of subsumption:
\[ A \sqsubseteq_T B \iff L(B, P) \subseteq L(A, P) \quad \text{for all primitive concepts } P \]

The subsumption problem in $\mathcal{FL}_0$ w.r.t. acyclic TBoxes is in coNP.
\[ T_C : \]

\[
\emptyset \rightarrow r \rightarrow \{P\} \rightarrow r \rightarrow \{P\}
\]

\[
\{P, Q\} \rightarrow r \rightarrow \{Q\} \rightarrow s \rightarrow \{P\}
\]

conjunction, existential restrictions, and top concept \( \top \)
Subsumption corresponds to existence of homomorphism
Subsumption corresponds to existence of homomorphism

\[ C \subseteq D \iff \text{there is a homomorphism from } T_D \text{ to } T_C. \]

\[
\exists r. (\exists r. Q \cap \exists s. Q) \cap \exists r. P \equiv P \cap \exists r. (\exists r. (P \cap Q) \cap \exists s. Q) \cap \exists r. (P \cap \exists s. P)
\]
• Existence of homomorphism

NP-complete for graphs

subtree of $T_1$ with root $u$
• Goal

• find DLs for which reasoning is tractable,
• i.e., which have polynomial-time decision procedures
Subsumption in the presence of GCIs remains polynomial if we add

- the bottom concept \(\bot\), which stands for the empty set;

- nominals, i.e., singleton concepts; \(\{\text{Denmark}\}\)

- restricted role-value-maps (RVMs), which can express transitivity and right-identities;

\(\text{Clinical\_finding} \sqcap \text{Body\_part} \sqsubseteq \bot\)
Restricted RVMs can express important properties of roles

\[ \epsilon \trianglelefteq \text{part_of} \]

\[ \text{part_of} \circ \text{part_of} \trianglelefteq \text{part_of} \]

\[ \text{proper_part_of} \trianglelefteq \text{part_of} \]

\[ \text{has_exact_location} \trianglelefteq \text{has_location} \]

\[ \text{has_location} \circ \text{part_of} \trianglelefteq \text{has_location} \]

Reflexivity

Transitivity

Role hierarchy

Role hierarchy

Right identity

\[
\begin{align*}
\text{Hand} & \xrightarrow{\text{part_of}} \text{Arm} \\
\text{has_location} & \quad \text{has_location} \\
\text{Hand_{injury}} & \quad \text{Hand_{amputation}} \\
\end{align*}
\]
Subsumption in the presence of GCIs remains polynomial if we add

- the bottom concept \( \bot \), which stands for the empty set;
- nominals, i.e., singleton concepts;
- restricted role-value-maps (RVMs), which can express transitivity and right-identities;
- domain and range restrictions for roles;
- restricted concrete domains, which exclude such as numbers, strings, \ldots in the definition of concepts.

\[
\text{domain}(\text{has\_location}) \sqsubseteq \text{Clinical\_finding} \\
\text{range}(\text{has\_location}) \sqsubseteq \text{Body\_part} \\
>_{180}(\text{has\_diastolic\_bp\_mmHg}) \sqsubseteq \text{Hypertension}
\]

Adding any of the other constructors available in OWL makes the subsumption problem intractable in the presence of GCIs.
Subsumption in the presence of GCIs remains polynomial if we add

- the bottom concept \( \perp \), which stands for the empty set;
- nominals, i.e., singleton concepts;
- restricted role-value-maps (RVMs), which can express transitivity and right-identities;
- domain and range restrictions for roles;
- restricted concrete domains, which enable using datatypes such as numbers, strings, ... in the definition of concepts.

To the more expressive DL \( \mathcal{EL}^{++} \) [Baader, Brandt, Lutz; 05, 08]
Back to the Cake ...

- Unique identification of resources
- A format for specifying structured data in a machine-readable form
- A language for querying information specified in RDF.
- A model for describing resources with properties and property values.
- Highly expressive ontology language for modelling complex knowledge domains.
- A language for describing a lightweight ontology.
What’s next?

Either

• Racer

or

• OWL 2