OWL Tutorial

An Example OWL Ontology

We will present a small OWL ontology

- to demonstrate the syntaxes of OWL
- to demonstrate how to use OWL
- to demonstrate the utility of OWL
- to demonstrate reasoning in OWL

Abstract syntax version of the ontology is attached.
OWL Tutorial

Reasoning Services

Reasoning services help knowledge engineers and users to build and use ontologies

(Many of the following slides have been taken from a longer tutorial on *Logical Foundations for the Semantic Web* by Ian Horrocks and Ulrike Sattler)
Ontology engineering tasks:
- design
- evolution
- inter-operation and Integration
- deployment

Further complications are due to
- sheer size of ontologies
- number of persons involved
- users not being knowledge experts
- natural laziness
- etc.
Reasoning Services: what we might want in the Design Phase

- be warned when making **meaningless** statements
  - test **satisfiability** of defined concepts

\[ \text{SAT} (C, \mathcal{T}) \text{ iff there is a model } \mathcal{I} \text{ of } \mathcal{T} \text{ with } C^\mathcal{I} \neq \emptyset \]

unsatisfiable, defined concepts are signs of faulty modelling

- see **consequences** of statements made
  - test defined concepts for **subsumption**

\[ \text{SUBS} (C, D, \mathcal{T}) \text{ iff } C^\mathcal{I} \subseteq D^\mathcal{I} \text{ for all model } \mathcal{I} \text{ of } \mathcal{T} \]

unwanted or missing subsumptions are signs of imprecise/faulty modelling

- see **redundancies**
  - test defined concepts for **equivalence**

\[ \text{EQUIV} (C, D, \mathcal{T}) \text{ iff } C^\mathcal{I} = D^\mathcal{I} \text{ for all model } \mathcal{I} \text{ of } \mathcal{T} \]

knowing about “redundant” classes helps avoid misunderstandings
Reasoning Services: what we might want when Modifying Ontologies

- the same system services as in the design phase, plus

- automatic generation of concept definitions from examples
  - given individuals \( o_1, \ldots, o_n \) with assertions (“ABox”) for them, create
    a (most specific) concept \( C \) such that each \( o_i \in C^I \) in each model \( I \) of \( T \)
    “non-standard inferences”

- automatic generation of concept definitions for too many siblings
  - given concepts \( C_1, \ldots, C_n \), create
    a (most specific) concept \( C \) such that \( \text{SUBS}(C_i, C, T) \)
    “non-standard inferences”

- etc.
Reasoning Services: what we might want when Integrating and Using Ontologies

For integration:

- the same system services as in the design phase, plus
- the possibility to abstract from concepts to patterns and compare patterns

\[ \text{e.g., compute those concepts } D \text{ defined in } \mathcal{T}_2 \text{ such that} \]

\[
\text{SUBS}(\text{Human } \sqcap (\forall \text{child.}(X \sqcap \forall \text{child}.Y)), D, \mathcal{T}_1 \cup \mathcal{T}_2)
\]

“non-standard inferences”

When using ontologies:

- the same system services as in the design phase and the integration phase, plus
- automatic classification of individuals

\[ \text{given individual } o \text{ with assertions, return all defined concepts } D \text{ such that} \]

\[ o \in D^\mathcal{I} \text{ for all models } \mathcal{I} \text{ of } \mathcal{T} \]
(many) reasoning problems are **inter-reducible**:

\[
\begin{align*}
\text{EQUIV}(C, D, T) & \text{ iff } \text{sub}(C, D, T) \text{ and } \text{sub}(D, C, T) \\
\text{SUBS}(C, D, T) & \text{ iff } \neg \text{SAT}(C \sqcap \neg D, T) \\
\text{SAT}(C, T) & \text{ iff } \neg \text{SUBS}(C, A \sqcap \neg A, T) \\
\text{SAT}(C, T) & \text{ iff } \text{cons}\{o: C\}, T
\end{align*}
\]

\[\rightarrow\] In the following, we concentrate on \text{SAT}(C, T)
We know $\text{SAT}$ is reducible to $\text{co-SUBS}$ and vice versa

Hence $\text{SAT}$ is undecidable iff $\text{SUBS}$ is

$\text{SAT}$ is semi-decidable iff $\text{co-SUBS}$ is

If $\text{SAT}$ is undecidable but semi-decidable, then

there exists a complete $\text{SAT}$ algorithm:

$\text{SAT}(C, T) \Leftrightarrow \text{“satisfiable”}$, but might not terminate if not $\text{SAT}(C, T)$

there is a complete $\text{co-SUBS}$ algorithm:

$\text{SUBS}(C, T) \Leftrightarrow \text{“subsumption”}$, but might not terminate if $\text{SUBS}(C, D, T)$

1. Do expressive ontology languages exist with decidable reasoning problems?

2. Is there a practical difference between ExpTime-hard and non-terminating?
Do Reasoning Services need to be Decidable?

We know $\text{SAT}$ is reducible to $\text{co-SUBS}$ and vice versa.

Hence

$\text{SAT}$ is undecidable iff $\text{SUBS}$ is

$\text{SAT}$ is semi-decidable iff $\text{co-SUBS}$ is

$\rightarrow$ if $\text{SAT}$ is undecidable but semi-decidable, then

there exists a complete $\text{SAT}$ algorithm:

$\text{SAT}(C, \mathcal{T}) \iff \text{"satisfiable"}, \text{but might not terminate if not } \text{SAT}(C, \mathcal{T})$

there is a complete $\text{co-SUBS}$ algorithm:

$\text{SUBS}(C, \mathcal{T}) \iff \text{"subsumption"}, \text{but might not terminate if } \text{SUBS}(C, D, \mathcal{T})$

1. Do expressive ontology languages exist with decidable reasoning problems?
   Yes: DAML+OIL and OWL DL

2. Is there a practical difference between ExpTime-hard and non-terminating?
   let’s see
\textbf{Relationship with other Logics}

- \textbf{SHI} is a fragment of \textbf{first order logic}

- \textbf{SHIQ} is a fragment of \textbf{first order logic with counting quantifiers equality}

- \textbf{SHI} without transitivity is a fragment of first order with \textbf{two variables}

- \textbf{ALC} is a notational variant of the \textbf{multi modal logic K}
  
  \textbf{inverse} roles are closely related to converse/past modalities
  
  \textbf{transitive} roles are closely related to transitive frames/axiom 4
  
  \textbf{number restrictions} are closely related to deterministic programs in PDL
Deciding Satisfiability of $SHIQ$

Remember: $SHIQ$ is OWL DL without datatypes and nominals

Next: tableau-based decision procedure for SAT $(C, T)$

The algorithm proceeds by trying to construct a representation of a model $I$ for $C$
This can be done because there always is such a representation, and the representation is at most of size exponential in the size of the ontology
### Complexity of DLs: Summary

Deciding satisfiability (or subsumption) of

<table>
<thead>
<tr>
<th>concepts in</th>
<th>Definition</th>
<th>without a TBox is</th>
<th>w.r.t. a TBox is</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>$\mathcal{ALC}$</strong></td>
<td>$\sqcap, \sqcup, \neg, \exists R.C, \forall R.C,$</td>
<td>PSpace-c</td>
<td>ExpTime-c</td>
</tr>
<tr>
<td><strong>$S$</strong></td>
<td>$\mathcal{ALC} + \text{transitive roles}$</td>
<td>PSPace-c</td>
<td>ExpTime-c</td>
</tr>
<tr>
<td><strong>$\mathcal{SI}$</strong></td>
<td>$\mathcal{SI} + \text{inverse roles}$</td>
<td>PSPace-c</td>
<td>ExpTime-c</td>
</tr>
<tr>
<td><strong>$\mathcal{SH}$</strong></td>
<td>$S + \text{role hierarchies}$</td>
<td>ExpTime-c</td>
<td>ExpTime-c</td>
</tr>
<tr>
<td><strong>$\mathcal{SHIQ}$</strong></td>
<td>$\mathcal{SHI} + \text{number restrictions}$</td>
<td>ExpTime-c</td>
<td>ExpTime-c</td>
</tr>
<tr>
<td><strong>$\mathcal{SHIQO}$</strong></td>
<td>$\mathcal{SHI} + \text{nominals}$</td>
<td>NExpTime-c?</td>
<td>NExpTime-c?</td>
</tr>
<tr>
<td><strong>$\mathcal{SHIQ}^+$</strong></td>
<td>$\mathcal{SHIQ} + \text{“naive number restrictions”}$</td>
<td>undecidable</td>
<td>undecidable</td>
</tr>
<tr>
<td><strong>$\mathcal{SH}^+$</strong></td>
<td>$\mathcal{SH} + \text{“naive role hierarchies”}$</td>
<td>undecidable</td>
<td>undecidable</td>
</tr>
</tbody>
</table>
Complexity of $SHIQ$ (Roughly OWL Lite)

$SHIQ$ is ExpTime-hard because $ALC$ with TBoxes is and $SHIQ$ can internalise TBoxes: polynomially reduce $\text{SAT}(C, \mathcal{T})$ to $\text{SAT}(C_{\mathcal{T}}, \emptyset)$

$$C_{\mathcal{T}} := C \cap \bigcap_{C_i \subseteq D_i \in \mathcal{T}} (C_i \Rightarrow D_i) \cap \forall U. \bigcap_{C_i \subseteq D_i \in \mathcal{T}} (C_i \Rightarrow D_i)$$

for $U$ new role with $\text{trans}(U)$, and

$$R \sqsubseteq U, R^- \sqsubseteq U$$

for all roles $R$ in $\mathcal{T}$ or $C$

Lemma: $C$ is satisfiable w.r.t. $\mathcal{T}$ iff $C_{\mathcal{T}}$ is satisfiable

Why is $SHIQ$ in ExpTime?

Tableau algorithms runs in worst-case non-deterministic double exponential space using double exponential time....
**$\text{SHIQ}$ is in ExpTime**

Translation of $\text{SHIQ}$ into Büchi Automata on infinite trees

$$C, \mathcal{T} \leadsto A_{C,\mathcal{T}}$$

such that

1. $\text{SAT}(C, \mathcal{T})$ iff $L(A_{C,\mathcal{T}}) \neq \emptyset$
2. $|A_{C,\mathcal{T}}|$ is exponential in $|C| + |\mathcal{T}|$
   (states of $C, \mathcal{T}$ are sets of subconcepts of $C$ and $\mathcal{T}$)

This yields ExpTime decision procedure for $\text{SAT}(C, \mathcal{T})$ since
emptyness of $L(A)$ can be decided in time polynomial in $|A|$

**Problem** $A_{C,T}$ needs (?) to be constructed before being tested: best-case ExpTime
**SHIQO** (roughly OWL DL) is NExpTime-hard

**Fact:** for **SHIQ** and **SHOQ**, \( \text{SAT}(C, T) \) are ExpTime-complete

* stands for “with inverse roles”, *O* for “with nominals”

**Lemma:** their combination is NExpTime-hard

even for **ALCQIO**, \( \text{SAT}(C, T) \) is NExpTime-hard
Naive implementation of $SHIQ$ tableau algorithm is doomed to failure:

Construct a tree of exponential depth in a non-deterministic way
\[ \sim \] requires backtracking in a deterministic implementation

Optimisations are crucial

A selection of some vital optimisations:
Classification: reduce number of satisfiability tests when classifying TBox
Absorption: replace globally disjunctive axioms by local versions
Optimised Blocking: discover loops in proof process early
Backjumping: dependency-directed backtracking
SAT optimisations: take good ideas from SAT provers
Missing in \textit{SHIQ} from OWL DL: Datatypes and Nominals

(Remember: \textit{I} stands for “with inverse roles”, \textit{O}” for “with nominals”)

So far, we discussed DLs that are fragments of OWL DL

\textit{SHIQ} + Nominals = \textit{SHIQO}

- we have seen: \textit{SHIQO} is \textit{NExpTime-hard}
- so far: no “goal-directed” reasoning algorithm known for \textit{SHIQO}
- unclear: whether \textit{SHIQO} is “practicable”
- but: t-algorithm designed for \textit{SHOQ}

\textit{SHIQ} + Datatypes = \textit{SHIQ(D_n)}

\textit{SHOQ} + Datatypes = \textit{SHOQ(D_n)}

- extend \textit{SHIQ} with concrete data and built-in predicates
- extend \textit{SHIQ} with, e.g.,
  \( \exists \text{age.} > 18 \) or
  \( \exists \text{age, shoeSize.} = \)

- relevant in many ontologies
- dangerous, but well understood extension
- currently being implemented and tested for \textit{SHOQ (D)}
In DLs, datatypes are known as **concrete domains**

**Concrete domain** \( D + (\text{dom}(D), \text{pred}) \) consists of

- a set \( \text{dom}(D) \), e.g., integers, strings, lists of reals, etc.
- a set \( \text{pred} \) of **predicates**, each predicate \( P \in \text{pred} \) comes with
  - **arity** \( n \in \mathbb{N} \) and
  - a (fixed!) **extension** \( P^n \subseteq \text{dom}(D)^n \)
- e.g. predicates on \( \mathbb{Q} \): unary \( =_3 \), \( \leq_7 \), binary \( \leq, = \), ternary \( \{(x, y, z) \mid x + y = y\} \)
We know

- how to reason in $\mathcal{SHIQ}$ (proven to be ExpTime-complete) implementations and optimisations well understood
- how to reason in $\mathcal{SHOQ}(D)$ (decidable, exact complexity unknown) optimisation for nominals $\mathcal{O}$ need more investigations optimisation for $(D)$ are currently being investigated
- that their combination, OWL DL$^1$, is more complex: NExpTime-hard so far, no “goal-directed” reasoning algorithm known for OWL DL

- accept an incomplete algorithm for OWL DL
- use a first-order prover for reasoning (and accept possibility of non-termination)
- live with OWL Lite while waiting for complete OWL DL algorithm

__________________________
1. $\mathcal{SHIQO}(D)$ with number restrictions restricted to $\geq nR.\top$, $\leq nR.\top$
ABoxes and Instances

Remember: when using ontologies, we would like to automatically classify individuals described in an ABox

an ABox $A$ is a finite set of assertions of the form

$$C(a) \text{ or } R(a, b)$$

$I$ is a model of $A$ if

$$a^I \in C^I \text{ for each } C(a) \in A$$
$$\langle a^I, b^I \rangle \in R^I \text{ for each } R(a, b) \in A$$

Cons($A$, $\mathcal{T}$) if there is a model $I$ of $A$ and $\mathcal{T}$

Inst($a$, $C$, $A$, $\mathcal{T}$) if $a^I \in C^I$ for each model $I$ of $A$ and $\mathcal{T}$

Easy: Inst($a$, $C$, $A$, $\mathcal{T}$) iff not Cons($A \cup \{\neg C(a)\}$, $\mathcal{T}$)

Example: $A = \{A(a), R(a, b), A(b), S(b, c), B(c)\}$
$\mathcal{T} = \{A \sqsubseteq \leq 1R. \top\}$
Inst($a$, $\forall R.A$, $A$, $\mathcal{T}$) but not Inst($b$, $\forall S.B$, $A$, $\mathcal{T}$)
How to decide whether $\text{Cons}(A, \mathcal{T})$?

~ extend tableau algorithm to start with ABox 

$C(a) \in A \implies C \in \text{L}(a)$

$R(a, b) \in A \implies (a, R, y)$

this yields a graph—in general, not a tree

work on forest—rather than on a single tree

i.e., trees whose root nodes intertwine in a graph

theoretically not too complicated

many problems in implementation

Current Research: how to provide ABox reasoning for huge ABoxes

approach: restrict relational structure of ABox
For Ontology Engineering, useful reasoning services can be based on SAT and SUBS.

Are all useful reasoning services based on SAT and SUBS?

Remember: to support modifying ontologies, we wanted

- automatic generation of concept definitions from examples
  - given ABox A and individuals $a_i$ create
    a (most specific) concept $C$ such that each $a_i \in C^I$ in each model $I$ of $T$
    $$\text{msc}(a_1, \ldots, a_n), A, T$$

- automatic generation of concept definitions for too many siblings
  - given concepts $C_1, \ldots, C_n$, create
    a (most specific) concept $C$ such that $\text{SUBS}(C_i, C, T)$
    $$\text{lcs}(C_1, \ldots, C_n), A, T$$
Unlike SAT, SUBS, etc., msc and lcs are computation problems.

Fix a DL $\mathcal{L}$. Define

$$ C = \text{msc}(a_1, \ldots, a_n, A, \mathcal{T}) \iff a_i^\mathcal{T} \in C^\mathcal{T} \ \forall 1 \leq i \leq n \text{ and } \forall \mathcal{I} \text{ model of } A \text{ and } \mathcal{T} $$

$C$ is the smallest such concept, i.e.,

if $a_i^\mathcal{T} \in C'^\mathcal{T} \ \forall 1 \leq i \leq n$ and $\forall \mathcal{I}$ model of $A$ and $\mathcal{T}$

then $\text{SUBS}(C, C', \mathcal{T})$

$$ C = \text{lcs}(C_1, \ldots, C_n, \mathcal{T}) \iff \text{SUBS}(C_i, C, \mathcal{T}) \ \forall 1 \leq i \leq n $$

$C$ is the smallest such concept, i.e.,

if $C_i \in C' \ \forall 1 \leq i \leq n$

then $\text{SUBS}(C, C', \mathcal{T})$

Clear: $\text{msc}(a_1, \ldots, a_n, A, \mathcal{T}) = \text{lcs}(\text{msc}(a_1, A, \mathcal{T}), \ldots, \text{msc}(a_n, A, \mathcal{T}))$

$$ \text{lcs}(C_1, C_2, C_3, \mathcal{T}) = \text{lcs}(\text{lcs}(C_1, C_2, \mathcal{T}), C_3, \mathcal{T})) $$
Non-Standard Reasoning Services: msc and lcs

Known Results:

- **lcs in DLs with □** is **useless**: \( \text{lcs}(C_1, C_2, \mathcal{T}) = C_1 \cup C_2 \)

- **msc}(a, A, \mathcal{T})** might **not exist**: \( \mathcal{L} = \mathcal{ALC} \)
  \[ \mathcal{T} = \emptyset \]
  \[ A = \{ A(a), R(a, a) \} \]
  \[ \text{msc}(a, A, \mathcal{T}) = A \cap \exists R.A? A \cap \exists R.(A \cap \exists R.A)? \]

- **∃ DLs: (SUBS, SAT) msc, lcs are decidable/computable in polynomial time**
  \( \mathcal{EL} \) with cyclic TBoxes (only \( \sqcap \) and \( \exists R.C \))

- **∃ DLs: lcs can be computed, but might be of exponential size**
  \( \mathcal{ALE} \) (only \( \sqcap \), primitive \( \neg \), \( \forall R.C \), \( \exists R.C \))
Non-Standard Reasoning Services: other

**concept pattern:** concept with variables in the place of concepts

The following non-standard reasoning services also come w.r.t. TBoxes

**unification:** \( C \equiv? D \) for \( C, D \) concept patterns

- **solution to** \( C \equiv? D \): a substitution \( \sigma \) (replacing variables with concepts) such that \( \sigma(C) \equiv \sigma(D) \)

  - Goal: decide unification problem and find a (most specific) such substitution

**matching:** \( C \equiv? D \) for \( C \) concept patterns and \( D \) a concept

- **solution to** \( C \equiv? D \): a substitution \( \sigma \) with \( \sigma(C) \equiv D \)

**approximation:** given DLs \( \mathcal{L}_1, \mathcal{L}_2 \) and \( \mathcal{L}_1 \)-concept \( C \), find \( \mathcal{L}_2 \)-concept \( \hat{C} \) with \( \text{SUBS}(C, \hat{C}) \) and \( \text{SUBS}(C, D) \) implies \( \text{SUBS}(\hat{C}, D) \) for all \( \mathcal{L}_2 \)-concepts \( D \)

**rewriting** given \( C, \mathcal{T} \), find “shortest” \( \hat{C} \) such that \( \text{EQUIV}(C, \hat{C}, \mathcal{T}) \)
Resources

**ESSLI Tutorial by Ian Horrocks and Ulrike Sattler**
http://www.cs.man.ac.uk/~horrocks/ESSLI203/

**W3C Webont Working Group Documents**  http://www.w3.org/2001/sw/WebOnts/
**Particularly OWL Web Ontology Language Guide**  http://www.w3.org/TR/owl-guide/

**W3C RDF Core Working Group Documents**  http://www.w3.org/2001/sw/RDFCore/
**Particularly RDF Primer**  http://www.w3.org/TR/rdf-primer/

**Description Logics Handbook**  http://books.cambridge.org/0521781760.htm

**RDF and OWL Tutorials by Roger Costello and David Jacobs**
http://www.xfront.com/rdf/
http://www.xfront.com/rdf-schema/
http://www.xfront.com/owl-quick-intro/
http://www.xfront.com/owl/