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)

Complexity of Ontology engineering

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
    \[ \text{SAT}(C, T) \text{ iff there is a model } I \text{ of } T \text{ with } C^I \neq \emptyset \]
    unsatisfiable, defined concepts are signs of faulty modelling
- see consequences of statements made
  - test defined concepts for subsumption
    \[ \text{SUBS}(C, D, T) \text{ iff } C^I \subseteq D^I \text{ for all model } I \text{ of } T \]
    unwanted or missing subsumptions are signs of imprecise/faulty modelling
- see redundancies
  - test defined concepts for equivalence
    \[ \text{EQUIV}(C, D, T) \text{ iff } C^I = D^I \text{ for all model } I \text{ of } T \]
    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 \(C_1, \ldots, C_m\), 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
  ➔ e.g., compute those concepts $D$ defined in $T_2$ such that
  \[
  \text{SUBS}(\text{Human} \sqcap (\forall \text{child}.(X \sqcap \forall \text{child}.Y)), D, T_1 \cup T_2)
  \]
  “non-standard inferences”

When using ontologies:
- the same system services as in the design phase and the integration phase, plus
- automatic classification of individuals
  ➔ given individual $o$ with assertions, return all defined concepts $D$ such that
  \[
  o \in D^2 \text{ for all models } \mathcal{I} \text{ of } T
  \]

Do Reasoning Services need to be Decidable?

We know SAT is reducible to co-SUBS and vice versa

Hence  SAT is undecidable iff SUBS is
  SAT is semi-decidable iff co-SUBS is
  ➔ if SAT is undecidable but semi-decidable, then
  there exists a complete SAT algorithm:
  \[
  \text{SAT}(C, T) \iff \text{“satisfiable”}, \text{ but might not terminate if not SAT}(C, T)
  \]
  there is a complete co-SUBS algorithm:
  \[
  \text{SUBS}(C, T) \iff \text{“subsumption”}, \text{ but might not terminate if SUBS}(C, D, T))
  \]

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

Reasoning Services: what we can do

(many) reasoning problems are inter-reducible:

\[
\begin{align*}
\text{EQUIV}(C, D, T) & \text{ iff } \text{sub}(C, D, T) \text{ and } \text{sub}(D, C, T) \\
\text{SUBS}(C, D, T) & \text{ iff } \text{not SAT}(C \sqcap \neg D, T) \\
\text{SAT}(C, T) & \text{ iff } \text{not SUBS}(C, A \sqcap \neg A, T) \\
\text{SAT}(C, T) & \text{ iff } \text{cons}({o: C}, T)
\end{align*}
\]

➔ In the following, we concentrate on SAT($C, T$)

Relationship with other Logics

- $SHI$ is a fragment of first order logic
- $SHIQ$ is a fragment of first order logic with counting quantifiers equality
- $SHI$ without transitivity is a fragment of first order with two variables
- $ALC$ is a notational variant of the multi modal logic K
  inverse roles are closely related to converse/past modalities
  transitive roles are closely related to transitive frames/axiom 4
  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>$ALC$</td>
<td>$\forall R.C, \forall R.C.$</td>
<td>PSpace-c</td>
<td>ExpTime-c</td>
</tr>
<tr>
<td>$S$</td>
<td>$ALC +$ transitive roles</td>
<td>PSPace-c</td>
<td>ExpTime-c</td>
</tr>
<tr>
<td>$SI$</td>
<td>$SI +$ inverse roles</td>
<td>PSpace-c</td>
<td>ExpTime-c</td>
</tr>
<tr>
<td>$SH$</td>
<td>$S +$ role hierarchies</td>
<td>ExpTime-c</td>
<td>ExpTime-c</td>
</tr>
<tr>
<td>$SHIQ$</td>
<td>$SH +$ number restrictions</td>
<td>ExpTime-c</td>
<td>ExpTime-c</td>
</tr>
<tr>
<td>$SHIQO$</td>
<td>$SH +$ nominals</td>
<td>NExpTime-c</td>
<td>NExpTime-c</td>
</tr>
<tr>
<td>$SHIQ^+$</td>
<td>$SHIQ +$ &quot;naive number restrictions&quot;</td>
<td>undecidable</td>
<td>undecidable</td>
</tr>
<tr>
<td>$SH^+$</td>
<td>$SH +$ &quot;naive role hierarchies&quot;</td>
<td>undecidable</td>
<td>undecidable</td>
</tr>
</tbody>
</table>

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

$$C_T := C \cap \bigcap_{C_i \in D_i \in T} (C_i \Rightarrow D_i) \cap \forall U \bigcap_{C_i \in D_i \in T} (C_i \Rightarrow D_i)$$

for $U$ new role with $\text{trans}(U)$, and $R \subseteq U, R^\rightarrow \subseteq U$ for all roles $R$ in $T$ or $C$

Lemma: $C$ is satisfiable w.r.t. $T$ iff $C_T$ is satisfiable.

Why is $SHIQ$ in ExpTime?

Tableau algorithms runs in worst-case non-deterministic double exponential space using double exponential time....

$SHIQ$ is in ExpTime

Translation of $SHIQ$ into Büchi Automata on infinite trees

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

such that

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

This yields ExpTime decision procedure for $SAT(C,T)$ 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**, SAT\((C,T)\) are ExpTime-complete

I stands for "with inverse roles", O" for "with nominals"

**Lemma:** their combination is NExpTime-hard
even for **ALCQIO**, SAT\((C,T)\) is NExpTime-hard

**Implementing OWL Lite or OWL DL**

Naïve implementation of **SHIQ** tableau algorithm is doomed to failure:

Construct a tree of exponential depth in a non-deterministic way

\(\rightsquigarrow\) 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 SHIQ from OWL DL: Datatypes and Nominals**

(Remember: I stands for "with inverse roles", O" for "with nominals")

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

**SHIQ + Nominals = SHIQO**

- we have seen: **SHIQO** is NExpTime-hard
- so far: no "goal-directed" reasoning algorithm known for **SHIQO**
- unclear: whether **SHIQO** is "practicable"
- but: t-algorithm designed for **SHOQ**
  \(\Rightarrow\) live without nominals or inverses

**SHIQ + Datatypes = SHIQ(D_n)**

**SHOQ + Datatypes = SHOQ(D_n)**

- extend **SHIQ** with concrete data and built-in predicates
- extend **SHIQ** with, e.g.,
  \(\exists\)age. > 18 or \(\exists\)age, shoeSize. =
- relevant in many ontologies
- dangerous, but well understood extension
currently being implemented and tested for **SHOQ** (D)

**Missing in SHIQ from OWL DL: Datatypes**

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 \(=\), \(\le\), binary \(\le\), \(\le\), ternary \(\{(x, y, z) \mid x + y = y\}\)
Summing up: SAT and SUBS in OWL DL

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

$C(a)$ or $R(a, b)$

$I$ is a model of $A$ if $a^I \in C^I$ for each $C(a) \in A$

$(a^I, b^I) \in R^I$ for each $R(a, b) \in A$

Cons($A$, $T$) if there is a model $I$ of $A$ and $T$

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

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

Example: $A = \{A(a), R(a, b), A(b), S(b, c), B(c)\}$

$T = \{A \sqsubseteq \mathcal{L}R. \top\}$

Inst($a, \forall R.A, A, T$) but not Inst($b, \forall S.B, A, T$)

ABoxes and Instances

How to decide whether Cons($A$, $T$)?

$\triangleright$ extend tableau algorithm to start with ABox

$C(a) \in A \Rightarrow C \in \mathcal{L}(a)$

$R(a, b) \in A \Rightarrow (a, R, b)$

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

Non-Standard Reasoning Services

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

  $\triangleright$ 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

  $\triangleright$ given concepts $C_1, \ldots, C_n$, create

  a (most specific) concept $C$ such that $\text{SUBS}(C, C_1, T)$

  $\text{lcs}(C_1, \ldots, C_n), A, T$
Non-Standard Reasoning Services: msc and lcs

Unlike SAT, SUBS, etc., msc and lcs are computation problems

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

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

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

if $a_i^T \in C^T \forall 1 \leq i \leq n$ and $\forall \mathcal{I}$ model of $A$ and $T$

then $\text{SUBS}(C, C^0, T)$

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

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

if $C_i \in C^T \forall 1 \leq i \leq n$

then $\text{SUBS}(C, C^0, T)$

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

$\text{lcs}(C_1, C_2, C_3, T) = \text{lcs}(\text{lcs}(C_1, C_2, T), C_3, T)$

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 $C, T$, find “shortest” $\hat{C}$ such that $\text{EQUIV}(C, \hat{C}, T)$

Known Results:

- lcs in DLs with $\sqcup$ is useless: $\text{lcs}(C_1, C_2, T) = C_1 \sqcup C_2$

- $\text{msc}(a, A, T)$ might not exist: e.g., $\mathcal{L} = \mathcal{ALC}$

  $T = \emptyset$

  $A = \{A(a), R(a, a)\}$

  $\text{msc}(a, A, T) = A \sqcap \exists R.A? A \sqcap \exists R.(A \sqcap \exists R.A)$?

- $\exists$ DLs: (SUBS, SAT) msc, lcs are decidable/computable in polynomial time

  $\mathcal{EL}$ with cyclic TBoxes (only $\sqcap$ and $\exists R.C$)

- $\exists$ DLs: lcs can be computed, but might be of exponential size

  $\mathcal{ALC}$ (only $\sqcap$, primitive $\neg$, $\forall R.C$, $\exists R.C$)

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/Webont/
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/