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)

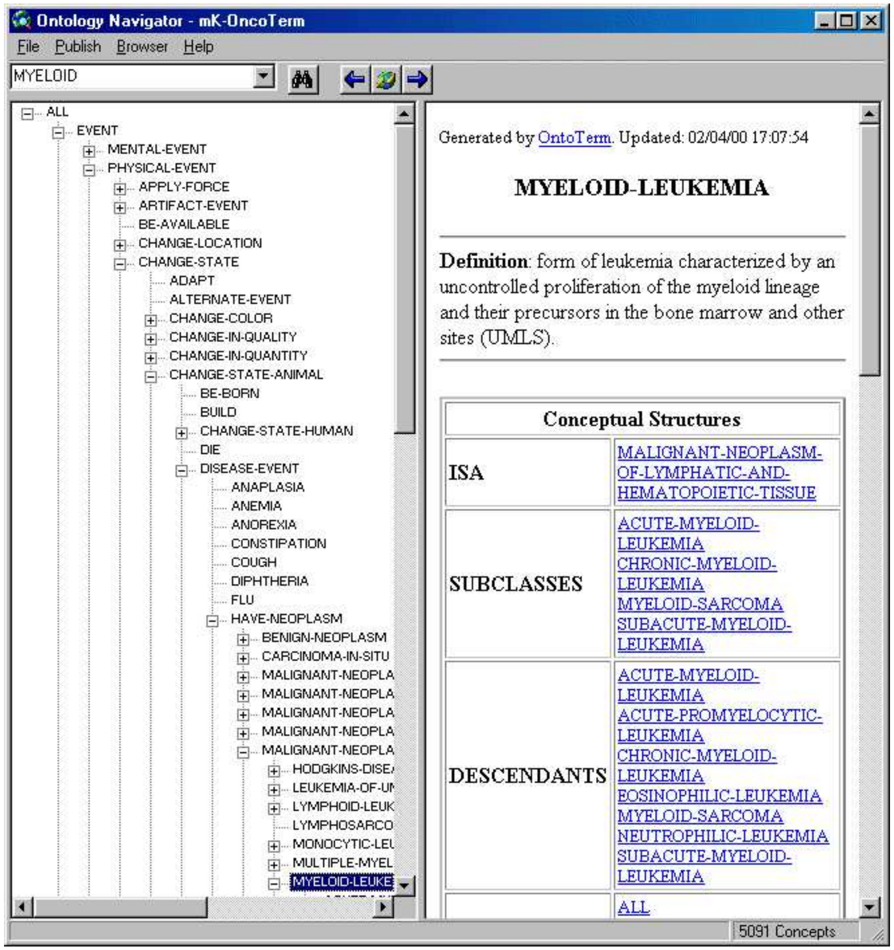
Complexity of Ontology engineering

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

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

  unsatisfiable, defined concepts are signs of faulty modelling

- see **consequences** of statements made

  ➠ test defined concepts for **subsumption**

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

  unwanted or missing subsumptions are signs of imprecise/faulty modelling

- see **redundancies**

  ➠ test defined concepts for **equivalence**

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

  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 $o_i \in C^{\mathcal{I}}$ in each model $\mathcal{I}$ of $\mathcal{T}$

  "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 **SUBS**$(C_i, C, \mathcal{T})$

  "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 $\mathcal{T}_2$ such that

$$\mathsf{SUBS}(\mathtt{Human} \sqcap (\forall \mathtt{child}.(X \sqcap \forall \mathtt{child}.Y)), D, \mathcal{T}_1 \cup \mathcal{T}_2)$$

“non-standard inferences”

When using ontologies:

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

  ➠ given individual o with assertions, return all defined concepts D such that

$$o \in D^{\mathcal{I}} \text{ for all models } \mathcal{I} \text{ of } \mathcal{T}$$

Reasoning Services: what we can do

(many) reasoning problems are **inter-reducible**:

$$\textbf{EQUIV}(C, D, \mathcal{T}) \text{ iff } \textbf{sub}(C, D, \mathcal{T}) \text{ and } \textbf{sub}(D, C, \mathcal{T})$$

$$\textbf{SUBS}(C, D, \mathcal{T}) \text{ iff } \textbf{not SAT}(C \sqcap \neg D, \mathcal{T})$$

$$\textbf{SAT}(C, \mathcal{T}) \text{ iff } \textbf{not SUBS}(C, A \sqcap \neg A, \mathcal{T})$$

$$\textbf{SAT}(C, \mathcal{T}) \text{ iff } \textbf{cons}(\{o \colon C\}, \mathcal{T})$$

➠ In the following, we concentrate on **SAT**$(C, \mathcal{T})$

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:
SAT$(C, \mathcal{T}) \Leftrightarrow$ "satisfiable", but might not terminate if not **SAT**$(C, \mathcal{T})$

there is a complete co-**SUBS** algorithm:
SUBS$(C, \mathcal{T}) \Leftrightarrow$ "subsumption", but might not terminate if **SUBS**$(C, D, \mathcal{T})$)

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 **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:
SAT$(C, \mathcal{T}) \Leftrightarrow$ "satisfiable", but might not terminate if not **SAT**$(C, \mathcal{T})$

there is a complete co-**SUBS** algorithm:
SUBS$(C, \mathcal{T}) \Leftrightarrow$ "subsumption", but might not terminate if **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

Relationship with other Logics

- $\mathcal{SHI}$ is a fragment of **first order logic**
- $\mathcal{SHIQ}$ is a fragment of **first order logic with counting quantifiers**

  equality
- $\mathcal{SHI}$ without transitivity is a fragment of first order with **two variables**
- $\mathcal{ALC}$ is a notational variant of the **multi modal logic** $\mathbf{K}$

  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 $\mathcal{SHIQ}$

Remember: $\mathcal{SHIQ}$ is OWL DL without datatypes and nominals

Next: **tableau-based decision procedure for SAT (C,$\mathcal{T}$)**

The algorithm proceeds by trying to construct a representation of a **model** $\mathcal{I}$ 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

concepts in	Definition	without a TBox is	w.r.t. a TBox is
$\mathcal{ALC}$	$\sqcap$, $\sqcup$, $\neg$, $\exists R.C$, $\forall R.C$,	PSpace-c	ExpTime-c
$\mathcal{S}$	$\mathcal{ALC}$ + transitive roles	PSPace-c	ExpTime-c
$\mathcal{SI}$	$\mathcal{SI}$ + inverse roles	PSPace-c	ExpTime-c
$\mathcal{SH}$	$\mathcal{S}$ + role hierarchies	ExpTime-c	ExpTime-c
$\mathcal{SHIQ}$	$\mathcal{SHI}$ + number restrictions	ExpTime-c	ExpTime-c
$\mathcal{SHIQO}$	$\mathcal{SHI}$ + nominals	NExpTime-c?	NExpTime-c?
$\mathcal{SHIQ}^+$	$\mathcal{SHIQ}$ + "naive number restrictions"	undecidable	undecidable
$\mathcal{SH}^+$	$\mathcal{SH}$ + "naive role hierarchies"	undecidable	undecidable

Complexity of $\mathcal{SHIQ}$ (Roughly OWL Lite)

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

$$C_\mathcal{T} := C \sqcap \bigsqcap_{C_i \dot\sqsubseteq D_i \in \mathcal{T}} (C_i \Rightarrow D_i) \sqcap \forall U. \bigsqcap_{C_i \dot\sqsubseteq D_i \in \mathcal{T}} (C_i \Rightarrow D_i)$$

for U new role with **trans**(U), and

$R \dot\sqsubseteq U, R^- \dot\sqsubseteq U$ 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 $\mathcal{SHIQ}$ in ExpTime?

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

$\mathcal{SHIQ}$ is in ExpTime

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

$$C, \mathcal{T} \;\rightsquigarrow\; A_{C,\mathcal{T}}$$

such that

1. **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 **SAT**$(C, \mathcal{T})$ since

emptyness of $L(A)$ can be decided in time polynomial in $|A|$

Problem $A_{C,\mathcal{T}}$ needs (?) to be constructed before being tested: best-case ExpTime

$\mathcal{SHIQO}$ (roughly OWL DL) is NExpTime-hard

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

$\mathcal{I}$ stands for "with inverse roles", $\mathcal{O}$" for "with nominals"

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

Implementing OWL Lite or OWL DL

Naive implementation of $\mathcal{SHIQ}$ 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 $\mathcal{SHIQ}$ from OWL DL: Datatypes and Nominals

(Remember: $\mathcal{I}$ stands for "with inverse roles", $\mathcal{O}$" for "with nominals")

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

$\mathcal{SHIQ}$ + **Nominals** = $\mathcal{SHIQO}$

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

➠ live without nominals **or** inverses

$\mathcal{SHIQ}$ + Datatypes = $\mathcal{SHIQ}(D_n)$
$\mathcal{SHOQ}$ + Datatypes = $\mathcal{SHOQ}(D_n)$

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

Missing in $\mathcal{SHIQ}$ from OWL DL: Datatypes

In DLs, datatypes are known as **concrete domains**

Concrete domain $D + (\mathbf{dom}(D), \mathbf{pred})$ consists of

- a set $\mathbf{dom}(D)$, e.g., integers, strings, lists of reals, etc.
- a set **pred** of **predicates**, each predicate $P \in \mathbf{pred}$ comes with
  - **arity** $n \in \mathbb{N}$ and
  - a (fixed!) **extension** $P^n \subseteq \mathbf{dom}(D)^n$
- e.g. predicates on $\mathbb{Q}$: unary $=_3$, $\leq_7$, binary $\leq, =$, 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 $\mathcal{SHOQ}(\mathbf{D})$ (decidable, exact complexity unknown)
  optimisation for nominals $\mathcal{O}$ need more investigations
  optimisation for $(\mathbf{D})$ 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. $\mathcal{SHIQO}(\mathbf{D})$ with number restrictions restricted to $\geqslant n R.\top$, $\leqslant n R.\top$

ABoxes and Instances

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

an **ABox** Ais a finite set of **assertions** of the form

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

$\mathcal{I}$ is **a model of A** if

$$\begin{aligned} a^{\mathcal{I}} \in C^{\mathcal{I}} &\quad \text{for each } C(a) \in A \\ (a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}} &\quad \text{for each } R(a, b) \in A \end{aligned}$$

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

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

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

Example:

$$\begin{aligned} A &= \{A(a), R(a, b), A(b), S(b, c), B(c)\} \\ \mathcal{T} &= \{A \mathrel{\dot\sqsubseteq} \leqslant 1 R.\top\} \end{aligned}$$

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

ABoxes and Instances

How to decide whether **Cons**$(A, \mathcal{T})$?

⇝ extend tableau algorithm to start with ABox $C(a) \in A \Rightarrow C \in \mathrm{L}(a)$

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

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**
  - ➠ given ABox Aand individuals a_i create
    a (most specific) concept C such that each $a_i \in C^{\mathcal{I}}$ in each model $\mathcal{I}$ of $\mathcal{T}$

$$\textsf{msc}(a_1, \ldots, a_n), A, \mathcal{T})$$

- automatic generation of concept definitions for **too many siblings**
  - ➠ given concepts $C_1, \ldots, C_n$, create
    a (most specific) concept C such that **SUBS**$(C_i, C, \mathcal{T})$

$$\textsf{lcs}(C_1, \ldots, C_n), A, \mathcal{T})$$

Non-Standard Reasoning Services: msc and lcs

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

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

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

C **is the smallest such concept**, i.e.,
if $a_i^{\mathcal{I}} \in C'^{\mathcal{I}}\ \forall 1 \leq i \leq n$ and $\forall\ \mathcal{I}$ model of Aand $\mathcal{T}$
then **SUBS**$(C, C', \mathcal{T})$

$C = \textbf{lcs}(C_1, \ldots, C_n, \mathcal{T})$ iff **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 **SUBS**$(C, C', \mathcal{T})$

Clear:

$$\textbf{msc}(a_1, \ldots, a_n, A, \mathcal{T}) = \textbf{lcs}(\textbf{msc}(a_1, A, \mathcal{T}), \ldots, \textbf{msc}(a_n, A, \mathcal{T}))$$
$$\textbf{lcs}(C_1, C_2, C_3, \mathcal{T}) = \textbf{lcs}(\textbf{lcs}(C_1, C_2, \mathcal{T}), C_3, \mathcal{T}))$$

Non-Standard Reasoning Services: **msc** and **lcs**

Known Results:

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

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

$$\begin{aligned} \mathcal{L} &= \mathcal{ALC} \\ \mathcal{T} &= \emptyset \\ \mathcal{A} &= \{A(a), R(a,a)\} \\ \textbf{msc}(a, \mathcal{A}, \mathcal{T}) &= A \sqcap \exists R.A?\ A \sqcap \exists R.(A \sqcap \exists R.A)? \end{aligned}$$

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

Non-Standard Reasoning Services: other

concept pattern: concept with **variabels** 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 σ (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 σ 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 **SUBS**$(C, \hat{C})$ and
SUBS(C, D) implies **SUBS**$(\hat{C}, D)$ for all $\mathcal{L}_2$-concepts D

rewriting given C, $\mathcal{T}$, find "shortest" $\hat{C}$ such that **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/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/`