Week 4

COMP62342
Sean Bechhofer, Uli Sattler
sean.bechhofer@manchester.ac.uk,
uli.sattler@manchester.ac.uk
Today

- Some clarifications from last week’s coursework
- More on reasoning:
  - extension of the tableau algorithm & discussion of blocking
  - traversal or “how to compute the inferred class hierarchy”
  - OWL profiles
- The OWL API: a Java API and reference implementation for
  - creating,
  - manipulating and
  - serialising OWL Ontologies and
  - interacting with OWL reasoners
- Lab:
  - OWL API for coursework
  - Ontology Development
Some clarifications from last week’s coursework
Ontologies, inference, entailments, models

- OWL is based on Description Logics
  - we can use DL syntax
  - e.g., $C \sqsubseteq D$ for $C$ SubClassOf $D$

- An OWL ontology $O$ is a **document**:
  - therefore, it cannot **do** anything - as it isn’t a piece of software!
  - in particular, it cannot **infer** anything

- An OWL ontology $O$ is a **web document**:
  - with ‘import’ statements, annotations, …
  - corresponds to a **set of logical OWL axioms**
  - the OWL API (today) helps you to
    - parse an ontology
    - access its axioms
  - a **reasoner** is only interested in this set of axioms
    - **not** in annotation axioms
    - see https://www.w3.org/TR/owl2-primer/#Document_Information_and_Annotations
    - https://www.w3.org/TR/2012/REC-owl2-syntax-20121211/#Annotations
We have defined what it means for O to entail an axiom C SubClassOf D written $O \models C \text{ SubClassOf } D$ or $O \models C \sqsubseteq D$

based on the notion of a model I of O

- i.e., an interpretation I that satisfies all axioms in O
- don’t confuse ‘model’ with ‘ontology’
  - one ontology can have many models
  - the more axioms in O the fewer models O has

A DL reasoner can be used to check entailments of an OWL ontology O and compute the inferred class hierarchy of O

  - this is also known as classifying O
  - e.g., by using a tableau algorithm
More on Reasoning
Recall Week 2: OWL 2 Semantics: Entailments

Let O be an ontology, α an axiom, and A, B classes, b an individual name:

- **O** is **consistent** if there exists some model I of O
  - i.e., there is an interpretation that satisfies all axioms in O
  - i.e., O isn’t self contradictory

- **O** **entails** α (written \( O \models \alpha \)) if α is satisfied in all models of O
  - i.e., α is a consequence of the axioms in O

- **A** is **satisfiable** w.r.t. O if \( O \not\models A \text{ SubClassOf Nothing} \)
  - i.e., there is a model I of O with \( A^I \neq \{\} \)

- b is an **instance of** A w.r.t. O (written \( O \models b:A \)) if \( b^I \subseteq A^I \) in every model I of O

**Theorem:**

1. O is consistent iff \( O \not\models \text{ Thing SubClassOf Nothing} \)
2. A is satisfiable w.r.t. O iff \( O \cup \{n:A\} \) is consistent (where n doesn’t occur in O)
3. b is an instance of A in O iff \( O \cup \{b:\text{not}(A)\} \) is not consistent
4. O entails A SubClassOf B iff \( O \cup \{n:A \text{ and not}(B)\} \) is inconsistent
Recall Week 2: OWL 2 Semantics: Entailments etc.

Let $O$ be an ontology, $\alpha$ an axiom, and $A$, $B$ classes, $b$ an individual name:

- $O$ is **consistent** if there exists some model $I$ of $O$
  - i.e., there is an interpretation that satisfies all axioms in $O$
  - i.e., $O$ isn’t self contradictory
- $O$ **entails** $\alpha$ (written $O \models \alpha$) if $\alpha$ is satisfied in all models of $O$
  - i.e., $\alpha$ is a consequence of the axioms in $O$
- $A$ is **satisfiable** w.r.t. $O$ if $O \models A \text{ SubClassOf Nothing}$
  - i.e., there is a model $I$ of $O$ with $A^I \neq \emptyset$
- $b$ is an **instance of** $A$ w.r.t. $O$ if $b^I \subseteq A^I$ in every model $I$ of $O$

- $O$ is **coherent** if every class name that occurs in $O$ is satisfiable w.r.t $O$
- **Classifying** $O$ is a reasoning service consisting of
  1. testing whether $O$ is consistent; if yes, then
  2. checking, for each pair $A,B$ of class names in $O$ plus Thing, Nothing $O \models A \text{ SubClassOf B}$
  3. checking, for each individual name $b$ and class name $A$ in $O$, whether $O \not\models b:A$

...and returning the result in a suitable form: $O$’s **inferred class hierarchy**
Last week, you saw a **tableau algorithm** that

- takes a class expression C and decides **satisfiability of C**
- i.e., it
  - answers ‘yes’ if C is satisfiable
  - ‘no’ if C is not satisfiable
  - sound, complete, and terminating

- we saw this for the ALC fragment of OWL
  - (ALC is a Description Logic that forms logical basis of OWL)
  - only and, or, not, some, only
- works by trying to generate an interpretation with an instance of C
  - by breaking down class expressions (in NNF!)
  - generating new P-successors for some-values from restrictions
    (\(\exists P.C\) restrictions in DL)

- we can handle an ontology that is a set of **acyclic SubClassOf axioms**
  - via **unfolding** (check Week 3 slides!)
Week 3 (before Easter): tableau rules

\[
\begin{align*}
\times & \bullet \{C_1 \land C_2, \ldots \} & \rightarrow \land & \times & \bullet \{C_1 \land C_2, C_1, C_2, \ldots \} \\
\times & \bullet \{C_1 \lor C_2, \ldots \} & \rightarrow \lor & \times & \bullet \{C_1 \lor C_2, C, \ldots \} \\
\times & \bullet \{\exists R.C, \ldots \} & \rightarrow \exists & \times & \bullet \{\exists R.C, \ldots \} \\
\times & \bullet \{\forall R.C, \ldots \} & \rightarrow \forall & \times & \bullet \{\forall R.C, \ldots \} \\
\end{align*}
\]

For \( C \in \{C_1, C_2\} \)

Algorithm Examples

- Test the satisfiability of \( 9R.A \lor 8R.B \)
- \( 9R.A \lor 8R.\neg A \)
- \( 9R.(A \lor 9R.B) \lor 8R.\neg A \lor 8R.\neg B) \)
Mini-exercise

- Apply the tableau algorithm to test whether A is satisfiable w.r.t.

\[
\{ A \sqsubseteq B \cap \exists P.C, \\
B \sqsubseteq C \cap \forall P.(\neg C \cup D) \} \]

\[
\{ A \text{ SubClassOf } B \text{ and } (P \text{ some } C), \\
A \text{ SubClassOf } C \text{ and } (P \text{ only } \neg(C \text{ or } D)) \} 
\]
This week: GCIs and tableau algorithm

- When writing an OWL ontology in Protégé,
  - axioms are of the form $A \text{ SubClassOf } B$ with $A$ a class name
  - (or $A \text{ EquivalentTo } B$ with $A$ a class name)

- last week’s tableau handles these via unfolding:
  - works only for acyclic ontologies
  - e.g., not for $A \text{ SubClassOf } (P \text{ some } A)$

- but OWL allows for general class inclusions (GCIs),
  - axioms of the form $A \text{ SubClassOf } B$ with $A$ a class expression
  - e.g., $(\text{eats some Thing}) \text{ SubClassOf } \text{Animal}$
  - e.g., $(\text{like some Dance}) \text{ SubClassOf } (\text{like some Music})$
  - this requires basically another rule:

\[
\begin{align*}
  x \bullet \{\ldots\} \quad \rightarrow_{\text{GCI}} \quad x \bullet \{\neg C \sqcup D,\ldots\}
\end{align*}
\]

for each $C \sqsubseteq D \in O$
GCIs and tableau algorithm

\[ \begin{align*}
\times \quad & \bullet \{ \ldots \} \quad \rightarrow_{\text{GCI}} \quad \times \quad & \bullet \{ \neg C \sqcup D, \ldots \} \\
\text{for each } & C \sqsubseteq D \in O
\end{align*} \]

- E.g., test whether A is satisfiable w.r.t. \( \{\text{A SubClassOf (P some A)}\} \) or \( \{\text{A }\sqsubseteq \exists P.A\} \)
- This rule easily causes non-termination
  - if we forget to block
### Blocking

- Blocking ensures termination
  - even on cyclic ontologies
  - even with GCIs
- If x's node label is contained in the label of a predecessor y, we say "x is blocked by y"
- E.g., test whether A is satisfiable w.r.t. \{A SubClassOf (P some A)\}
  - here, n2 is blocked by n1

\[
x \bullet \{\exists R.C, \ldots \} \quad x \bullet \{\exists R.C, \ldots \}
\]

\[
\rightarrow \exists
\]

\[
R
\]

\[
y \bullet \{C\}
\]

only if x’s node label isn’t contained in the node label of a predecessor of x

\[
n1 \bullet \{A, \neg A \sqcup \exists P.A, \exists P.A\}
\]

\[
P
\]

\[
n2 \bullet \{A, \neg A \sqcup \exists P.A, \exists P.A\}
\]
Blocking

- When blocking occurs, we can build a **cyclic** model from a complete & clash-free completion tree.
  - hence soundness is preserved!

\[
\begin{align*}
\text{x} & \quad \bullet \{\exists R.C, \ldots \} \\
\rightarrow_\exists & \quad \text{x} \quad \bullet \{\exists R.C, \ldots \} \\
& \quad \quad \quad \quad \quad \quad \quad \text{only if x’s node label isn’t contained in} \\
& \quad \quad \quad \quad \quad \quad \quad \text{the node label of a} \\
& \quad \quad \quad \quad \quad \quad \quad \text{predecessor of x} \\
\end{align*}
\]
Tableau algorithm with blocking

Our ALC tableau algorithm with blocking is

- **sound**: if the algorithm stops and says “input ontology is consistent” then it is.
- **complete**: if the input ontology is consistent, then the algorithm stop and says so.
- **terminating**: regardless of the size/kind of input ontology, the algorithm stops and says
  - either “input ontology is consistent”
  - or “input ontology is **not** consistent”

- ...i.e., a decision procedure for ALC ontologies
  - even in the presence of cyclic axioms!
Our ALC tableau algorithm has a few sources of complexity:

- the breadth/out-degree of the tree constructed
- the depth/path length in tree constructed
- non-determinism due to $\sqcup$ rule from

  - disjunctions in O, e.g., A SubClassOf B or C
  - SubClassOf axioms in O
  - EquivalentTo axioms in O

Not too bad: bounded by number of ‘some’ expressions in O

Ok/linear for acyclic O
Bad/exponential for general O:
we can construct O of size $n$
where each model has a path of length $2^n$
Tableau algorithm & complexity

- Without further details: deciding ALC satisfiability
  - only of class expressions is PSpace-Complete
  - of class expressions w.r.t. ontology is ExpTime-complete
  - …much higher than intractable/SAT

- Implementation of ALC or OWL tableau algorithm requires optimisation
  - there has been a lot of work in the last ~25 years on this
  - you see the fruits in Fact++, Pellet, Hermit, Elk, …in Protégé
  - some of them from SAT optimisations, see COMP60332

- Next, I will discuss 1 optimisation: enhanced traversal
Naive Classification

- **Remember:** Classifying $O$ is a reasoning service consisting of
  
  1. testing whether $O$ is consistent; if yes, then
     
     \[
     \text{Test:} \quad \text{is Thing satisfiable w.r.t. } O? \]

  2. checking, for each pair $A, B$ of class names in $O$ plus Thing, Nothing
     whether $O \vDash A \text{ SubClassOf } B$
     
     \[
     \text{Test:} \quad \text{is } A \cap \neg B \text{ unsatisfiable w.r.t. } O? \]

  3. checking, for each individual name $b$ and class name $A$ in $O$, whether $O \vDash b: A$
     
     \[
     \text{Test:} \quad \text{is } O \cup \{b:\neg A\} \text{ is inconsistent?} \]

  ...and returning the result in a suitable form: $O$’s **inferred class hierarchy**
Naive Classification

- **Remember:** Classifying $O$ is a reasoning service consisting of

  1. testing whether $O$ is consistent; if yes, then

    Test: is $\text{Thing}$ satisfiable w.r.t. $O$?  
    1 test

  2. checking, for each pair $A, B$ of class names in $O$ plus $\text{Thing}, \text{Nothing}$ whether $O \nsubseteq A \text{ SubClassOf } B$ 

    Test: is $A \cap \neg B$ unsatisfiable w.r.t. $O$?  
    $n^2$ tests for $O$ with $n$ class names

  3. checking, for each individual name $b$ and class name $A$ in $O$, whether $O \nsubseteq b:A$

    Test: is $O \cup \{b:\neg A\}$ is inconsistent?  
    $nm$ tests for $O$ with $n$ class names, $m$ individuals

...and returning the result in a suitable form: $O$’s **inferred class hierarchy**
Enhanced Traversal

- **Naive Classification of O** requires $1 + n^2 + nm$ expensive satisfiability/consistency tests
- …can we do better?

➡ Enhanced Traversal
  - idea: build inferred class hierarchy top-down and bottom-up, “trickling in” each class name in turn

- Assume you have, so far, constructed the right hierarchy for O
- Now you “trickle” Oak: check whether
  - $O \not\models Oak \sqsubseteq Plant$?
    - yes - continue with Plant’s child
  - $O \not\models Oak \sqsubseteq Animal$?
    - no - ignore Animal’s children!
  - $O \not\models Oak \sqsubseteq Tree$?
    - yes - done!

➡ 2 entailment tests saved!
Enhanced Traversal

- **Naive Classification of O** requires $1 + n^2 + nm$ expensive satisfiability/consistency tests
- …can we do better?
  ➡ Enhanced Traversal
  - idea: build inferred class hierarchy top-down and bottom-up, “trickling in” each class name in turn

- Potentially avoids many of the $n^2$ satisfiability/consistency tests
  - very important in practice
  - different variants have been developed

- Just one of many optimisations!
Despite all optimisations, classification of an ontology may still take too long if it is
- big and/or
  - 300,000 axioms or more
- rich
  - ALC plus inverse properties, atleast, atmost, sub-property chains,…

For OWL 2 [*], **profiles** have been designed
- syntactic fragments of OWL obtained by restricting constructors available

Each profile is
- **maximal**, i.e., we know that if we allow more constructors,
  then computational complexity of reasoning would increase
- **motivated** by a use case

[*] the one we talk about here/you use in Protégé
OWL Profiles

In a nutshell, these are the profiles of OWL 2:

- **OWL 2 EL:**
  - only ‘and’, ‘some’, SubProperty, transitive, SubPropertyChain
  - designed for big class hierarchies
- **OWL 2 QL:**
  - only restricted ‘some’, restricted ‘and’, inverseOf, SubProperty
  - designed for querying data in a database through a class-level ontology
- **OWL 2 RL:**
  - no ‘some’ on RHS of SubClassOf, …
  - designed to be implemented via a classic rule engine

- For details, see OWL 2 specification!

- Note: OWL Lite was a profile of OWL (1).
Some Key Complexity Classes

All Problems

FO
Predicate Logic

Semi-Decidable

OWL 2

Decidable

Propositional Logic

NP

P

OWL 2 EL
The design triangle

Expressivity
(Representational Adequacy)

Usability
(Weak Cognitive Adequacy vs. Cognitive Complexity)

Computability
(vs. Computational and Implementational Complexity)
Summary

OWL reasoning

- is unusual:
  - standard reasoning involves solving many reasoning problems/satisfiability tests
- is decidable:
  - for standard reasoning problems, we have decision procedures
  - i.e., a calculus that is sound, complete, and terminating
- can be complex
  - but we know the complexity for many different DLs/OWL variants/profiles
  - and implementations require many good optimisations!

- goes beyond what we have discussed here
  - entailment explanation
  - query answering
  - module extraction
  - ...
Coursework: The Sushi Ontology

- In addition to the weekly tasks, there is a modelling task that spans the entire course. We will go through steps of ontology building process including the identification of Competency Questions, performing Knowledge Acquisition, developing the Formalisation and Evaluating the results.
- The domain for this ontology will be Sushi.
- Ontology building is usually a collaborative process, and this task will be done in small groups (of 3 students).
  - You have already been allocated to groups (see BB), although formal group work does not begin until Week 2.
- Following the development of the ontology, you will be asked to provide an evaluation of two ontologies from other groups.
  - You will be assessed on how well you have performed this evaluation process, but the results of your evaluations will not contribute towards assessment of the other ontologies.
- You will also be required to provide individual reports discussing your ontology and the process that you went through in developing it.