MANCHESTER 1824

The University of Manchester

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

The University of Manchester

MANCHESTER 1824

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:**
  - therefor, 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

Ontologies, inference, entailments, models (2)

- We have defined what it means for O to **entail** an axiom C SubClassOf D
  - written O ⊨ C SubClassOf D or O ⊨ C ⊑ 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**

The University of Manchester

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 ⊧ α) 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 ⊭ A 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 ⊧ b:A) if $b^I \subseteq A^I$ in every model I of O

Theorem:

1. O is consistent iff O ⊭ Thing SubClassOf Nothing
2. A is satisfiable w.r.t. O iff O ∪ {n:A} is consistent (where n doesn't occur in O)
3. b is an instance of A in O iff O ∪ {b:not(A)} is not consistent
4. O entails A SubClassOf B iff O ∪ {n:A and not(B)} is inconsistent

Recall Week 2: OWL 2 Semantics: Entailments etc.

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

  …and returning the result in a suitable form: O's **inferred class hierarchy**

Week 3 (before Easter): how to test satisfiability …

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

x • $\{C_1 \sqcap C_2, \ldots\}$	$\rightarrow_{\sqcap}$	x • $\{C_1 \sqcap C_2, C_1, C_2, \ldots\}$
x • $\{C_1 \sqcup C_2, \ldots\}$	$\rightarrow_{\sqcup}$	x • $\{C_1 \sqcup C_2, C, \ldots\}$ For $C \in \{C_1, C_2\}$
x • $\{\exists R.C, \ldots\}$	$\rightarrow_{\exists}$	x • $\{\exists R.C, \ldots\}$ ↓ R y • $\{C\}$
x • $\{\forall R.C, \ldots\}$ ↓ R y • $\{\ldots\}$	$\rightarrow_{\forall}$	x • $\{\forall R.C, \ldots\}$ ↓ R y • $\{C, \ldots\}$

MANCHESTER 1824

The University of Manchester

Mini-exercise

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

$\{A \sqsubseteq B \sqcap \exists P.C,$
$B \sqsubseteq C \sqcap \forall P.(\neg C \sqcup D)\}$

{A SubClassOf B and (P some C),
A SubClassOf C and (P only (not C or D)}

This week: GCIs and tableau algorithm

- When writing an OWL ontology in Protégé,
  - axioms are of the form A SubClassOf B with A a class **name**
  - (or A 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 SubClass (P some A)

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

$$x \bullet \{\ldots\} \quad \rightarrow_{GCI} \quad x \bullet \{\neg C \sqcup D, \ldots\}$$

for each $C \sqsubseteq D \in O$

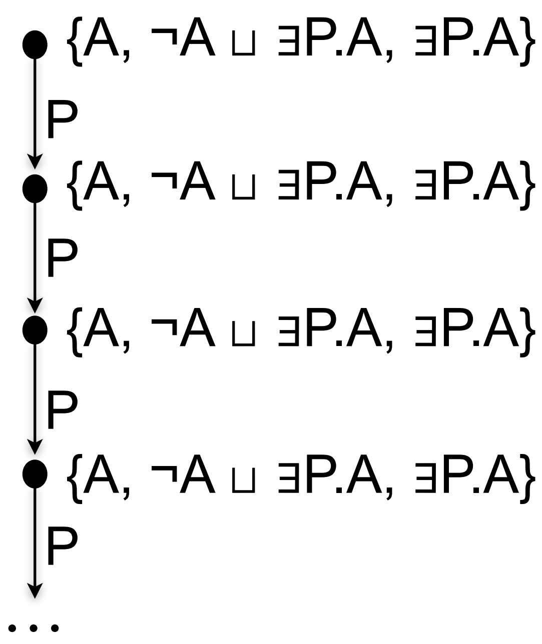
GCIs and tableau algorithm

$$x \bullet \{\ldots\} \quad \to_{GCI} \quad x \bullet \{\neg C \sqcup D, \ldots\}$$

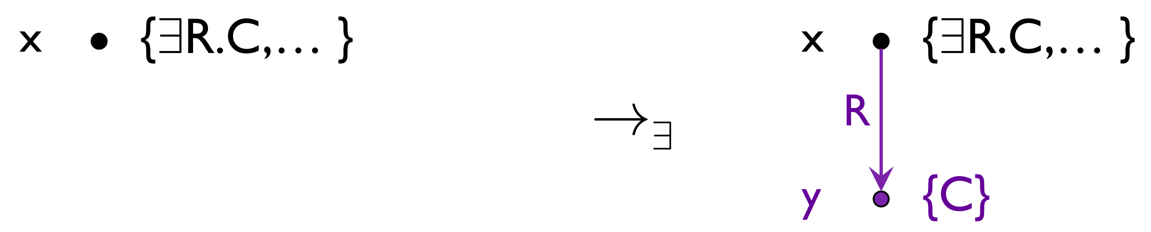
for each $C \sqsubseteq D \in O$

- E.g., test whether A is satisfiable
  w.r.t. {A SubClassOf (P some A)}
  or $\{A \sqsubseteq \exists P.A\}$

- This rule easily causes **non-termination**
  - if we forget to **block**

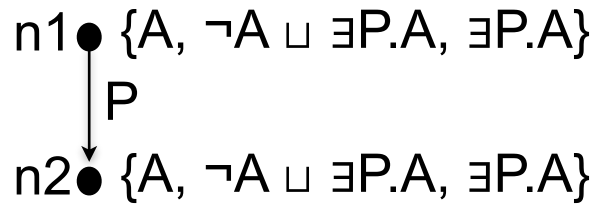


Blocking



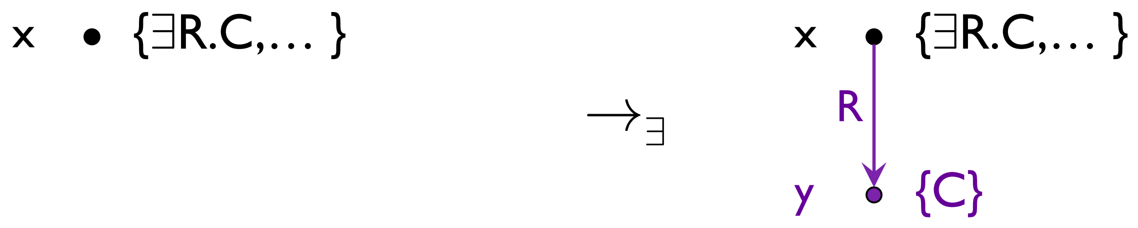
only if x's node label isn't contained in the node label of a predecessor of x

- 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*"



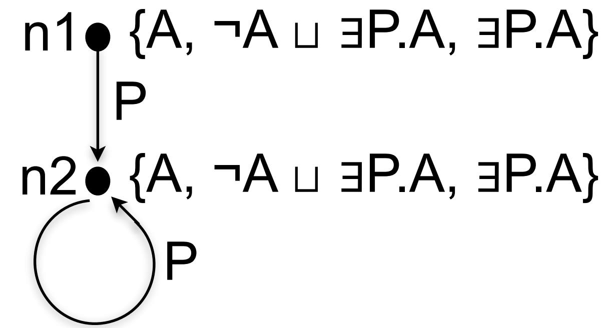
- E.g., test whether A is satisfiable
  w.r.t. {A SubClassOf (P some A)}
  - here, n2 is blocked by n1

Blocking



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

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



MANCHESTER 1824

The University of Manchester

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!

Tableau algorithm & complexity

not too bad:
bounded by number
of 'some'
expressions in O

Our ALC tableau algorithm has a few sources of **complexity**

- the breadth/out-degree of the tree constructed
- the depth of/path length in tree constructed
- non-determinism due to $\rightarrow_{\sqcup}$ rule from
  - disjunctions in O, e.g., A SubClassOf B or C
  - SubClassOf axioms in O
  - EquivalentTo axioms 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

hopefully not
too bad

1 disjunction
per axiom in O
for each node in tree

2 disjunctions
per axiom in O
for each node in tree

3 nodes with
25 SubClassOf
axioms →
how many
choices?

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

   Test:
   is Thing satisfiable w.r.t. O?

2. checking, for each pair A,B of class names in O plus Thing, Nothing whether O ⊧ A SubClassOf B

   Test:
   is A ⊓¬B **un**satisfiable w.r.t. O?

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

   Test:
   is O ∪ {b:¬A} is **in**consistent?

…and returning the result in a suitable form: O's **inferred class hierarchy**

MANCHESTER 1824

The University of Manchester

Naive Classification

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

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

Test: is Thing satisfiable w.r.t. O?	1 test

2. checking, for each pair A,B of class names in O plus Thing, Nothing whether O ⊧ A SubClassOf B

Test: is A ⊓¬B **un**satisfiable 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 ⊧ b:A

Test: is O ∪ {b:¬A} is **in**consistent?	*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

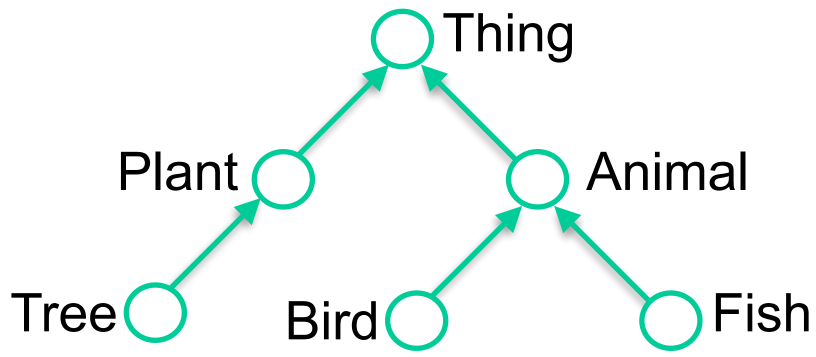
- **Naive Classification of O** requires $1 + n^2 + nm$ expensive satisfiability/consistency tests
- …can we do better?

➡ 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 ⊨ Oak ⊑ Plant?
    yes - continue with Plant’s child
  - O ⊨ Oak ⊑ Animal?
    no - ignore Animal’s children!
  - O ⊨ Oak ⊑ Tree?
    yes - done!

  ➡ 2 entailment tests saved!

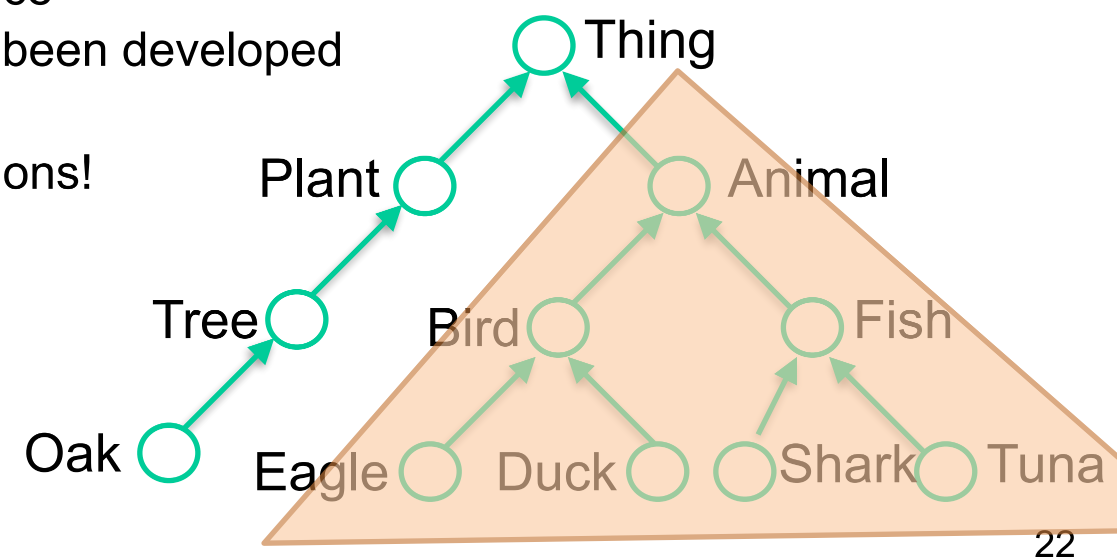


Enhanced Traversal

- **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!



OWL Profiles

- 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

MANCHESTER 1824

The University of Manchester

OWL Profiles

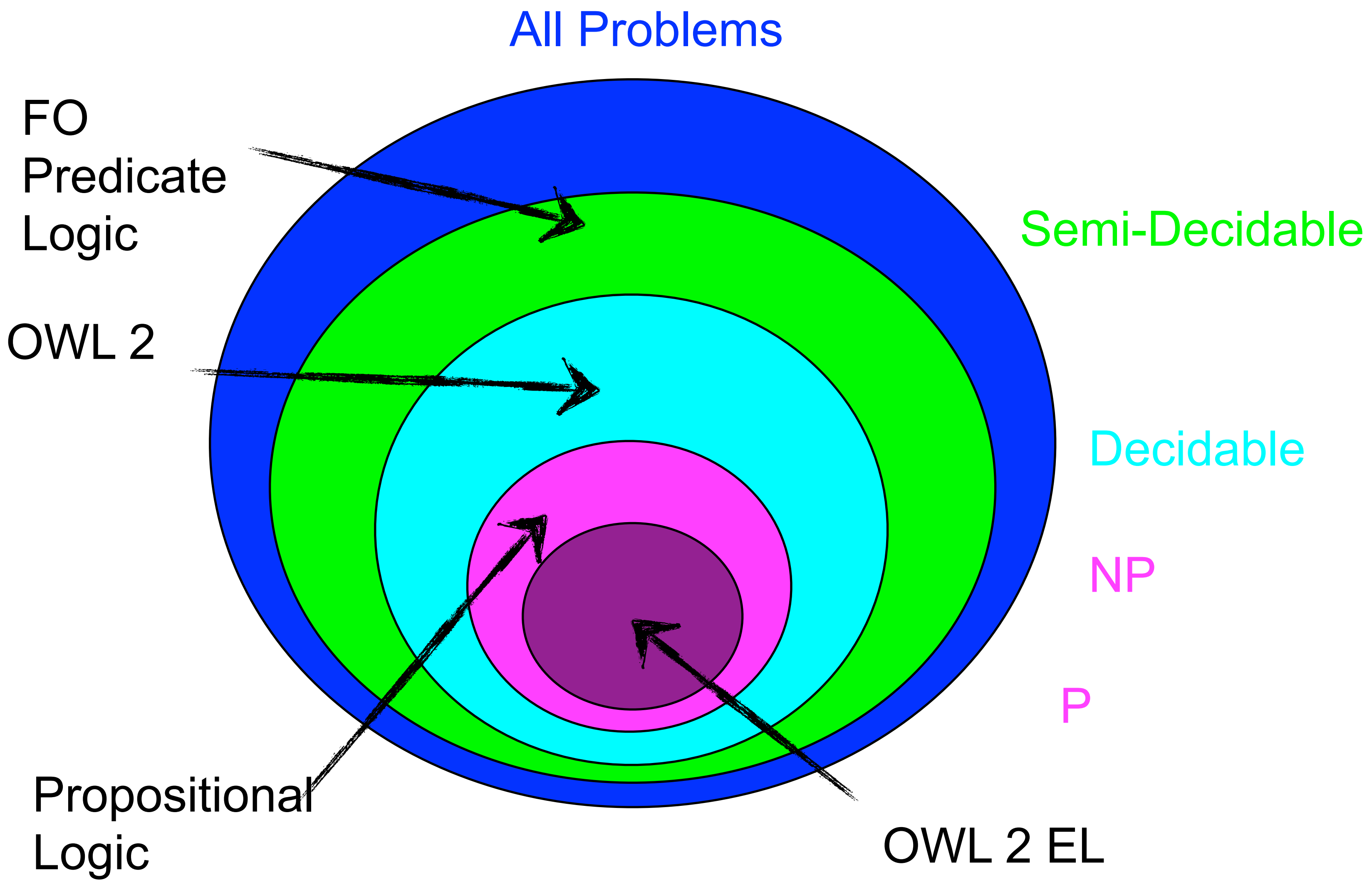
In a nutshell, these are the profiles of OWL 2:

- 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

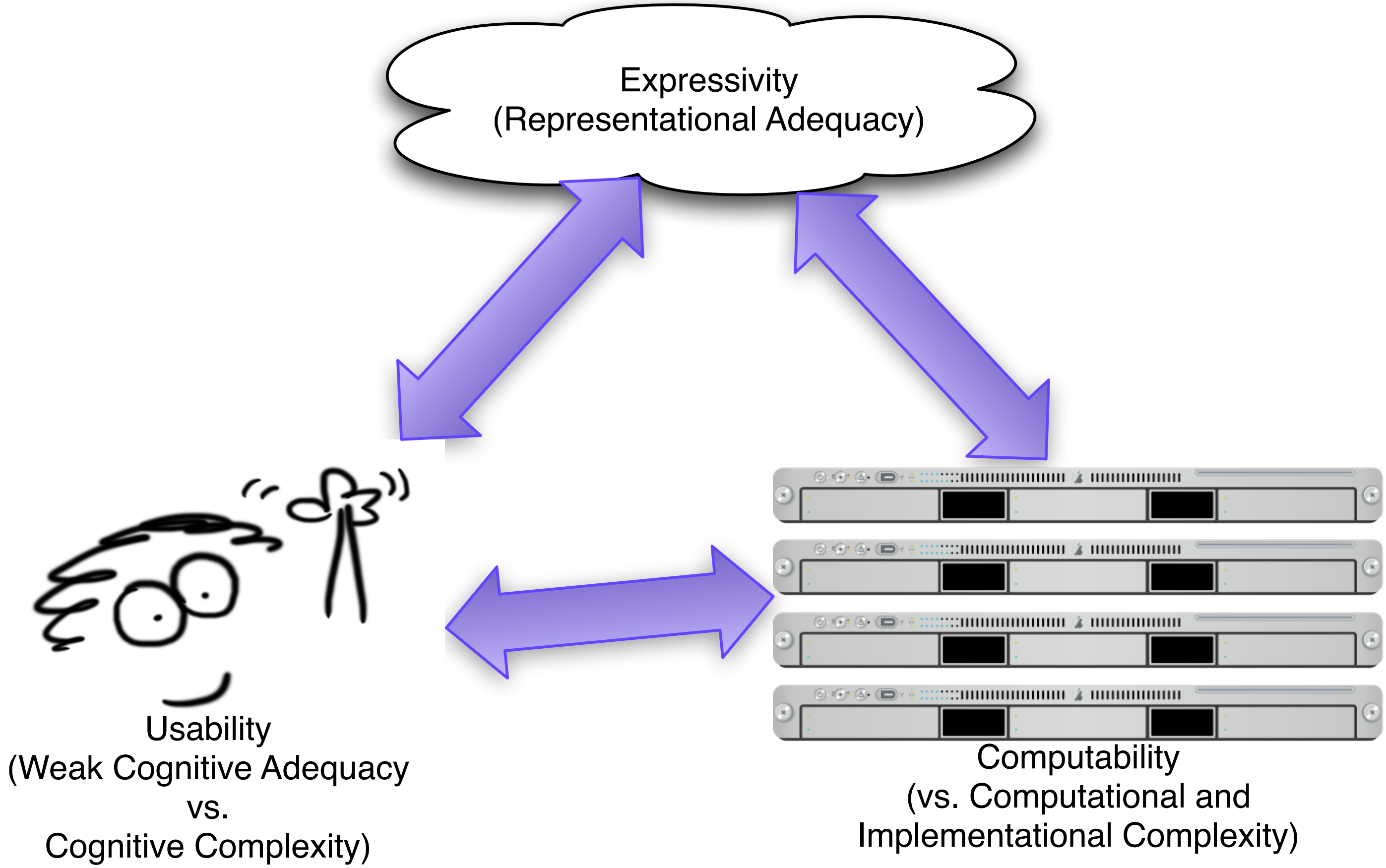
- For details, see OWL 2 specification!

- Note: OWL Lite was a profile of OWL (1).

Some Key Complexity Classes



The design triangle



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
  - …

Remember:

MANCHESTER
1824

The University of Manchester

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.

9