THE UNIVERSITY of EDINBURGH
informatics

Semantic Web Systems

Ontological Representation

Jacques Fleuriot

School of Informatics

In the previous lecture

- Ontologies
  - "a formal, explicit specification of a shared conceptualisation"
  - A way of encoding domain knowledge
- Frames
  - Class taxonomy, slot with values
  - ISA vs IO
- Folksonomy
  - "tagging that works"

In this lecture

- Ontology components
  - What to represent?
- Representation considerations
  - How to represent it?

Ontology Components

Ontology Components

Possible components of ontologies contain:

- individuals
- classes
- attributes
- relations
- functions
- axioms
- planning rules

Ontology Components: **Individuals**

Individuals are **instances** or **objects**

These are usually **concrete**
(e.g. rick_grimes, uk_prime_minister, uoe_student_1389203)

but they can be **abstract**
(e.g. numbers and words)

Two individuals may be equivalent
(e.g. uk_prime_minister, david_cameron)

It is not always clear whether something ought to be an individual or a class (e.g. uk_prime_minister)

Ontology Components: **Classes**

Classes are used to group things together.

- In most representations, members of classes must be *individuals*.
- In more expressive representations, *classes* may be also be allowed to be members of other classes. This can lead to complications.
- Classes can be *subsumed by*, or can *subsume* other classes
  ⇒ **subclasses** and **superclasses**.
- This leads to the class hierarchy, which is central to most ontologies.
- Some ontologies consist **only** of a class hierarchy – these are called **taxonomy** and opinion is divided as to whether they are ontologies at all.

Ontology Components: **Attributes**

Attributes are aspects, properties, features, characteristics, or parameters that objects and classes can have.

For example, the slots described in the previous lecture are a kind of attributes. *Frames* are a way of assigning attributes to classes.

Attributes can link objects and classes to

- Boolean values (true/false)
- Specific values (integers, individuals or other literals)
- Classes
- Complex data types (e.g. enumerated lists)

Ontology Components: **Relations**

Relations describe how classes and individuals relate to one another.

Typically, relations are defined between classes, and instantiations of relations are between individuals.

- course(Course_Name, Lecturer, Level, Credits, Year)
- course(sws, jacques_fleuriot, 11, 10, 2015/2016)

Expressive representations allow *n*-ary relations: relations with *n* arguments, where *n* is unlimited.

More restricted representations may limit this, e.g. only allow binary relations.

Ontology Components: **Functions**

Functions are relations such that, for a function with *n*+1 arguments, if the first n arguments are defined, the *n*+1th is defined.

e.g. *plus(Addend, Addend, Result)* is a function: if the two Addends are instantiated, there is only one possible value for Result.

The functional nature of relations is often indicated by using the representation: *plus(Addend, Addend) = Result*

Is *course(Course_Name, Lecturer)* functional?

- course(sws, Lecturer)
- course(ar, Lecturer)

Ontology Components: **Axioms**

Axioms describe how new facts can be derived from existing ones in the ontology.

$$sibling(X,Y) \wedge male(X) \rightarrow brother(X,Y)$$

It is not necessary to store all the facts about brothers: if information exists about gender of individuals and sibling relations, then information about brothers can be derived when required.

$$brother(X,Y) \vee sister(X,Y) \rightarrow sibling(X,Y)$$

Note that this notion of axiom is **different** to the notion used in formal logic, where the axioms are the facts known *a priori*.

Ontology Components: **Planning rules**

Rules describe how the world may be changed.

They consist of **antecedents** (things that must be true before the rule can be used and **consequents** (things that are made true by applying the rule). e.g. *Buy*:

in_stock(Item) ∧ has_money(Person, Amount, Time1) ∧ cost(Item,Price) ∧ Amount > Price
→ has(Item, Person) ∧ has_money(Person, New_amount, Time2) ∧ New_amount = Amount - Price

Because there is an implied before and after in a planning rule, it is necessary to have some way of identifying **time** (see has_money).

Many (most?) common ontology representations do **not** allow planning rules.

Note: it is common for axioms to be described as rules as well, so in general a rule in an ontology may be considered to be either something that describes how the world can be changed or something that describes how facts can be derived.

Ontology Components

Possible components of ontologies contain:

- individuals
- classes
- attributes
- relations
- functions
- axioms
- planning rules

Ontology Components

Possible components of ontologies contain:

- individuals
- classes
- attributes
- relations
- functions
- axioms
- planning rules

The more expressive a representation, the more of these components it will allow, and the fewer restrictions it will place on them.

Representation choices

What kind of representation?

There are many different kinds of representations which one can use for databases and ontologies.

The ones we will cover in the course are:

- Resource Description Framework (RDF)
- Resource Description Framework Scheme (RDFS)
- Description Logic (DL)
- Web Ontology Language (OWL):
  - OWL-full
  - OWL-DL
  - OWL-lite

There are also many others.

What kind of representation?

Deciding which representation to use essentially comes down to three issues:

Convenience of use and popularity of format
This is (largely) an implementational issue

The ability to say everything you want to say:
Expressivity

The ability to reason over your ontology:
Efficiency

The **tension between expressivity and efficiency** is at the heart of choosing an appropriate format.

Expressive representations

- Expressive representations allow most or all of these components with few, if any, restrictions on them.
- The most expressive representations in common use are first-order, e.g. the Knowledge Interchange Format (**KIF**).
- Essentially, this means you can have quantified variables in predicates and functions.
- It is also possible to create **higher-order** ontologies (where you can have quantified predicates and/or functions) but these are hard to use in practice.

Advantages of Expressivity

You can describe a complex and fluid environment:

- Describe relationships between many objects
  - course(Course_Name, Lecturer, Level, Credits, Year)
- Describe how the world is changing and how to effect change in the world
- Describe things that are true at different times
  - course(masws, fiona_mcneill, 10/11, 10, 2012/2013)
  - course(masws, ewan_klein, 10/11, 10, 2011/2012)
- Anything you want!

Disadvantages of Expressivity

but...reasoning is hard.

Is *brother(peter, john)* true?

No inference rules ⇒ just look up whether or not this is true.

If there is a rule: $sibling(X,Y) \wedge male(X) \rightarrow brother(X,Y)$
we need to find if this sibling relationship exists and then check the value of *male(X).*
(Backward Chaining process)

If there is a rule: $parent(Z,X) \wedge parent(Z,Y) \wedge male(X) \rightarrow brother(X,Y)$
we cannot return no until we have checked every possible value of Z against this rule.

This gets very complicated very quickly!

Combinatorial Explosion

A small increase in the number of rules, functions and relations can increase the complexity of reasoning enormously.

Computing power is increasing all the time, meaning computers can reason faster.

But computing power increases linearly, while the number of potential combinations increases exponentially.

Desirable properties: **decidability**

A representation is decidable if any question asked of it will be answered with a *yes* or a *no* in finite time.

That is, an inference process can be developed such that the question

is statement X true within ontology Y?

will return a Boolean truth value for any statement X and ontology Y in the given representation, and will not loop indefinitely.

Many representations are not decidable.

Desirable properties: **soundness** and **completeness**

A representation is sound if any logical formula *provable* or *derivable* within that representation is *true*

you cannot prove things which are not true

A representation is complete if any logical formula which is *true* can be *proved* or *derived* from the representation

if it is true, you can prove it

It is easy to create representations that are **sound**, and prove that they are so.

Creating representations that are **complete** is more difficult, and depends on restricting expressivity. Proving results about completeness can also be hard.

Reasoning with common representations

- Full first-order representations are neither decidable nor complete
- Description Logics were created to be decidable fragments of first-order logic:

  taking as much of the expressivity of first-order logic as possible
- OWL-lite and OWL-DL are decidable. OWL-full is **not**.
- RDF is a restrictive representation which consists of triples (*subject-predicate-object*).

Creating triples from more expressive representations

It is always possible to translate *n*-ary relations into triples:

- course(Course_Name, Lecturer, Level, Credits,Year)
- course(ID, Course_Name), course(ID, Lecturer), course(ID, Level), course(ID, Credits), course(ID, Year)

but this is unwieldy and can lead to confusion.

Many organisations are currently in the process of translating legacy databases into RDF.

How important is decidability?

- Decidability is a really nice theoretical result, but ...
- There are no guarantees about time.
  - A response that is returned too late to be useful is effectively the same as no reply.
- In order to be **practicably** decidable, you need to either
  - design your ontology so that reasoning is fast,
  - introduce time-outs and proceed without answers.
- But this is the same for non-decidable ontologies!

Role of Semantic Web

What kind of representation is best for the Semantic Web?

- If we view the Semantic Web as a massively connected **data store**, simple representations are the best:
  ⇒ RDF is currently by far the most popular representation.

- If we view the SW as a kind of multi-agent system, where complex reasoning, planning and acting are going on, much more expressive representations are needed:
  ⇒ at least OWL, probably even more expressive

- If we require the SW to be both of these things, a mixture of representations is needed.

Reflection on Representation

- Finding the best representation is largely a matter of balancing **expressivity** and **efficiency of reasoning**.
- There is no ‘correct’ answer to this problem: the sweet spot depends on what tasks you will be using the representation for.
- **Translating** between different representations, with different levels of expressivity, is possible, but comes at a price.

Summary

- Ontology components:
  - Individuals
  - Classes
  - Attributes
  - Relations
  - Functions
  - Axioms
  - Planning rules
- Representation considerations:
  - Trade-off between expressivity and efficiency
  - Decidability, soundness, completeness

Task

- Consider the small ontology you built as the last task – this was most likely just a taxonomy.
- Think about the kinds of things you might want to talk about involving that ontology, e.g. if it was an ontology about places, you might want to talk about travelling to those places.
- Without restricting expressivity, write down a few relations that might be relevant, e.g. *currency(Country, Currency)*, or *hotel(Name, Location, Cost, Rating)*
- Think about whether any of these relations are functional. If you had to use a more restrictive representation, would you have to change much?

Reading

- Sections 1.1-1.4 in *Ontological Engineering* by Asunción Gómez-Pérez, Mariano Fernandez-Lopez and Oscar Corcho