First Order Logic: Syntax and Semantics

COMP30411
Sean Bechhofer
sean.bechhofer@manchester.ac.uk

Problems

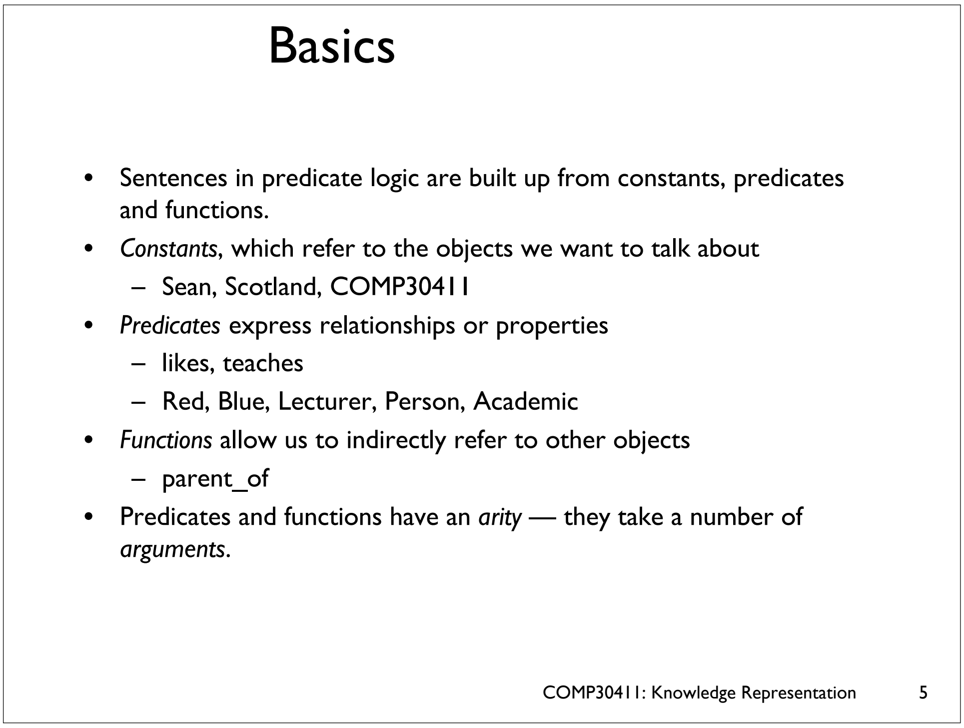
- Propositional logic isn't very expressive
- As an example, consider
  p = Scotland won on Saturday
- We have no access to any of the constituent parts of this proposition
  q = Sean drank beer on Saturday
- Can we tell from $p \wedge q$ that these both refer to the same day?

Problems

- Consider also the example
  Sean is a lecturer, lecturers are academics
  thus Sean is an academic
- What about:
  Bijan is a lecturer, lecturers are academics
  thus Bijan is an academic
- We can’t capture this kind of inference using propositional logic.
- The propositions don’t give us a general mechanism for talking about individuals in the world

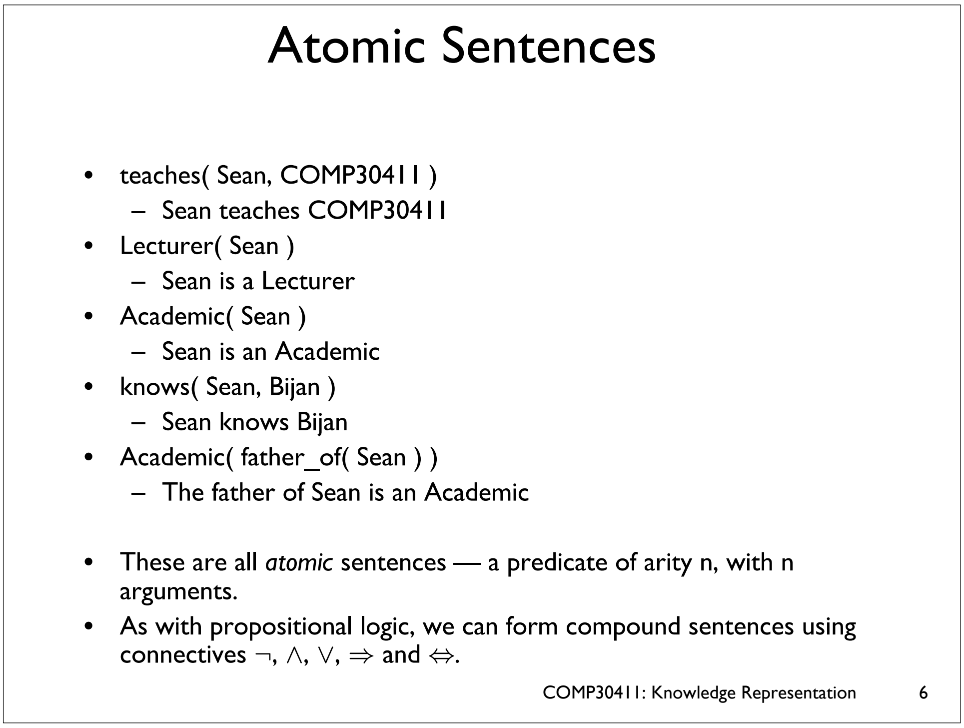
Predicate Calculus

- First order or predicate calculus provides a solution to this
- Predicates allow us to express relations or properties
- Variables allow abstract and generalisation



Basics

- Sentences in predicate logic are built up from constants, predicates and functions.
- *Constants*, which refer to the objects we want to talk about
  - Sean, Scotland, COMP30411
- *Predicates* express relationships or properties
  - likes, teaches
  - Red, Blue, Lecturer, Person, Academic
- *Functions* allow us to indirectly refer to other objects
  - parent_of
- Predicates and functions have an *arity* — they take a number of *arguments*.



Atomic Sentences

- teaches(Sean, COMP30411)
  - Sean teaches COMP30411
- Lecturer(Sean)
  - Sean is a Lecturer
- Academic(Sean)
  - Sean is an Academic
- knows(Sean, Bijan)
  - Sean knows Bijan
- Academic(father_of(Sean))
  - The father of Sean is an Academic

- These are all *atomic* sentences — a predicate of arity n, with n arguments.
- As with propositional logic, we can form compound sentences using connectives ¬, ∧, ∨, ⇒ and ⇔.

Compound Sentences

- teaches(Sean, COMP30411) ∧ Academic (Sean)
  - Sean teaches COMP30411 and Sean is an Academic
- knows(Sean, Bijan) ⇒ Lecturer(Sean)
  - If Sean knows Bijan then Sean is a Lecturer
- Academic(Sean) ∨ drinks(Sean, Beer)
  - Sean is an academic or Sean drinks Beer (or both)

- This is all very well, but all these predicates are about constants (or functions applied to constants).
- How can we talk about more general situations?

Variables & Quantifiers

- *Variables* allow us to assert more general or abstract sentences that don't talk about particular objects

- Academic(x) ⇒ drinks(x, Beer)
  - If x is an Academic, then x drinks Beer
- teaches(x, COMP30411) ∧ knows(x, y) ∧ ¬ Academic(y)
  - x teaches COMP30411 and x knows y and y is not an Academic

- *Quantifiers* tell us how to interpret the variables
- ∃ (exists) is used to indicate that there is something that can be subsituted for the variable such that the sentence holds (but doesn't necessarily tell us what it is).
- ∀ (for all) is used to indicate that the sentence should hold when anything is substituted for the variable

Quantification

- $\exists$ x. Red(x)
  - There is a red thing.
- $\forall$ x.$\exists$ y. teaches(x, y) $\Rightarrow$ Lecturer(x)
  - For any x, if there is some y that x teaches, then x is a Lecturer.

Formal Definition of Syntax

- *Symbols*
  - A set of constant symbols
  - A set of variables
  - A set of function symbols (with associated arities)
  - A set of predicate symbols (with associated arities)
- A *term* is defined as either
  - A constant
  - A variable
  - A function expression $f(t_1,\ldots,t_n)$, where f has arity n and $t_1,\ldots t_n$ are terms. Note that n may be zero

Well Formed Formulae

- An expression $P(t_1,\ldots,t_n)$ where P is a predicate symbol of arity n and $t_1,\ldots t_n$ are terms is a WFF
- If S is a WFF then so is $\neg S$
- If S_1 and S_2 are WFFs then so is $S_1 \land S_2$
- If S_1 and S_2 are WFFs then so is $S_1 \land S_2$
- If S_1 and S_2 are WFFs then so is $S_1 \land S_2$
- If x is a variable and S is a WFF then so is $\forall$ x. S
  - We say that any occurrence of x in S is *bound*.
- If x is a variable and S is a WFF then so is $\exists$ x. S
  - We say that any occurrence of x in S is *bound*.

Sentences

- WFF may still contain unbound variables, for example:
  - $\forall$ x. teaches(x, y) $\land$ Lecturer(x)
- This can cause us problems when we try and interpret them as it's not clear what we should do with the variable.
- To cope with this we define a further class of formulae known as sentences.
- A sentence is a WFF with no unbound (or free) variables.
  - Any variable occurring in the formula is bound by a quantifier

Interpretations

- How do we interpret the truth (or falsehood) of a statement?
- For example, given

  teaches(Sean, COMP30411)
- how do I know whether this is true or not?
- As with propositional logic, the truth of this statement has to be determined within some *context*.
  - It depends on my interpretation of the constants Sean and COMP30411

Interpretations

- In the propositional case, our interpretations were based on a truth valuation that assigned T or F to each atomic proposition. We then used truth tables to determine whether a sentence was true or not.
- In the predicate case, our interpretations are based on functions that map the symbols in the language to elements and sets of objects in a domain.
- The intuition here is that the domain consists of the objects that we're talking about, while the interpretation tells us how the symbols in the language map to those objects.

Interpretations

- To keep things simple here, we'll only consider a language with:
  - A set of constants C
  - A set of variables V
  - A set of unary predicates P
  - A set of binary predicates R

Interpretations

- An interpretation *I* consists of a set Δ known as the domain, along with functions:
  - I_C: C → Δ
    - Constants map to elements of the domain
  - I_P: P → *P*(Δ)
    - Unary predicates map to sets of elements of the domain
  - I_R: R → *P*(Δ × Δ)
    - Binary predicates map to sets of pairs of elements of the domain
- The functions tell us how to interpret atomic predicates.

Interpretations: Atoms

- Given an atomic sentence (e.g. one with no quantifiers or connectives), we define the notion of satisfaction:
  $I \models A(c)$ iff $I_C(c) \in I_P(A)$ for A a unary predicate
- A(c) is satisfied iff the interpretation of c is in the interpretation of A
  $I \models S(c,d)$ iff $(I_C(c),I_C(d)) \in I_R(S)$ for S a binary predicate
- S(c,d) is satisfied iff the pair of objects given by the interpretations of c and d are in the interpretation of S

Interpretations: Connectives

- As with the propositional case, we can extend the interpretation function to sentences that contain connectives
- For sentences A, B we define

$I \models \neg A$	if, and only if	$I \not\models A$
$I \models A \wedge B$	if, and only if	$I \models A$ and $I \models B$
$I \models A \vee B$	if, and only if	$I \models A$ or $I \models B$
$I \models A \Rightarrow B$	if, and only if	If $I \models A$ then $I \models B$
$I \models A \Leftrightarrow B$	if, and only if	$I \models A$ if and only if $I \models B$

Interpretations: Quantifiers

- We can also extend our interpretations to cover sentences that contain quantifiers

  I ⊨ ∃x.A(x) iff

  there is *some* object e in the domain such that A(e) is true
- Similarly for universal quantification

  I ⊨ ∀x.A(x) iff

  A(e) is true for *all* objects e in the domain.

Tautologies and Satisfiability

- A sentence is *valid* or is a *tautology* iff it is true under *every* interpretation. If A is a tautology, this is written

  ⊨ A
  - Just as in the propositional case
- A formula is said to be *satisfiable* (or *consistent*) iff it is true under *at least one* interpretation.
- A formula is said to be *unsatisfiable* (or *inconsistent* or *contradictory*) iff there is *no interpretation* under which it is true
- If a formula A is a *tautology* then ¬A is *unsatisfiable*.

Logical Consequences

- B is a consequence of $A_1, \ldots, A_n$ if it's the case that *whenever* all the A_i are true, then B must be true also. In other words, for any interpretation where $A_1, \ldots, A_n$ are satisfied then B is satisfied. We write:

$$A_1, \ldots, A_n \vDash B$$

  - Again, like the propositional case
- For example, if we know that both A(e) and $\forall x.(A(x) \Rightarrow B(x))$ hold, then we might want to be able to conclude that B(e) also holds.
- In the propositional case, we could do this by exhaustively writing down all the possible interpretations (truth tables) and checking them all.
- We can't do the same here though
  - There might be an infinite number of different interpretations.

Proof Rules

- We can provide a proof theory for FOL that allows us to determine logical consequence using some mechanical process.
- The rules are similar to those for the propositional case, but extended in order to deal with variables.
- For example, we have the ∀-elimination rule:

$$\frac{\forall x.A(x)}{A(e)} \quad \text{for any e in the domain}$$

- If we know that A holds for all x, then it must hold for some particular x.
- There are similar rules for ∃.

Proof

- By combining proof rules together, we can deduce consequences from premises.
- Ex: from $A(e)$ and $\forall x.(A(x) \Rightarrow B(x))$, can we prove $B(e)$? E.g. show that:

$$A(e), \forall x.(A(x) \Rightarrow B(x)) \vdash B(e)$$

1. $\forall x.(A(x) \Rightarrow B(x))$ [given]
2. $A(e)$ [given]
3. $A(e) \Rightarrow B(e)$ [1 ∀-elimination]
4. $B(e)$ [2,3 modus ponens] ✓

Proof Theory

- A proof theory allows us to determine satisfiability and entailments without having to consider the interpretations.
- It's similar to the propositional case, but a little more complex due to the extra machinery required for the quantifiers.

Relating Proofs and Semantics

- Again, we can relate the notions of *validity* (based on considering interpretations) and *provability* (based on proofs).
  - *Soundness* and *Completeness* are again the properties that relate the two
- A *Soundness* theorem states that:
  If $A_1,\ldots,A_n \vdash B$
  then $A_1 \wedge \ldots \wedge A_n \vDash B$ (i.e. $A_1 \wedge \ldots \wedge A_n \Rightarrow B$ is valid)
- A *Completeness* theorem states that:
  If $A_1 \wedge \ldots \wedge A_n \vDash B$ (i.e. $A_1 \wedge \ldots \wedge A_n \Rightarrow B$ is valid)
  then $A_1,\ldots,A_n \vdash B$

Soundness and Completeness

- Informally, soundness gives us an assurance that our proof theory is producing *correct* answers
- Completeness gives us an assurance that if a formula is valid, we can construct a proof of it using our proof theory.
- Again it is important to note that completeness doesn't say anything at all about how we might go about *constructing* such a proof — just that the proof *exists*.
- In the first order case, this is particularly important: provability in FOL is *undecidable*
  - There is *no* procedure that will guarantee to find us a proof of the validity of a sentence.
  - However, by restricting the expressivity of the language we can produce fragments of FOL that are decidable (e.g. Description Logics)

Knowledge Bases

- A Knowledge Base *K* = {$A_1,\ldots,A_n$} is a collection of sentences that describe facts about a situation of affairs that we are trying to model.

  K = {∀x.Lecturer(x) → Academic(x),
  ∀x.Academic(x) → (Hairy(x) ∨ Beardy(x)),
  Lecturer(Sean), Lecturer(Bijan), Lecturer(John)}
- We can then derive consequences from this knowledge base where B is a consequence if

$$A_1,\ldots,A_n \models B$$

Possible Worlds

- A key point here is that the sentences (or axioms) in the Knowledge Base do not describe a particular state of affairs but rather constrain the possible worlds that the knowledge describes.
- In the previous example, there are many different interpretations in which the axioms in *K* are satisfied
  - However, in each of those interpretations it must be the case that Academic(Sean) must be true.
  - It must also be the case that either Beardy(John) or Hairy(John) must be true, but this could differ in different interpretations.
- The consequences of the axioms are all those things that must hold in *any* possible interpretations
- This is in contrast to a database, where the facts in the database describe a *particular*, *single* state of affairs.

Possible Worlds

- This semantics based on interpretations allows us to represent *incomplete* or *partial* knowledge
  - As we will see later, this is a useful and powerful aspect.

Recap

- Propositional logic gives us a language for reasoning about propositions
  - The syntax defines the language
  - The semantics tells us how to interpret compound propositions
  - A proof theory allows us to reason about the consequences of propositions in a sound and complete way
- Predicate logic extends the propositional case allow us to talk about specific individuals, and generalise using variables.
  - Again a syntax, semantics and proof theory.
  - A Knowledge Base (KB) contains axioms that describe constraints on possible valid interpretations