General first-order languages

- Sometimes we'll ask you to translate English sentences into a first-order language that you design yourself.
- This sort of task can be done in many ways.
  - Consider how we might translate 'John prefers logic to mathematics'. The most natural way to do it is to use a ternary predicate and three individual constants: 'Prefers(john, logic, mathematics)'.
  - But you *could* do it with a binary predicate and two constants: 'PrefersToMathematics(john, logic)', or 'JohnPrefers(logic, mathematics). Or even a unary predicate: 'JohnPrefersToMathematics(logic)'

- The three-place predicate is obviously more flexible. In general, when you’re doing this kind of exercise, you want to aim for naturalness and flexibility.

Function symbols

- Some FOL dialects include an additional sort of vocabulary, *function symbols*.
- Function symbols can be used to make *complex terms*, which function grammatically just like individual constants.
  - EG: favouriteactor(cian), favouriteactor(father(cian)), favouriteactor(favourtieactor(cian))
- A function symbol has an *arity* just like a predicate, but we'll write function symbols in lower case so there's never any confusion.

- (In prefix notation:) A complex term is the result of writing an *n*-ary function symbol followed by *n* terms (in parentheses, separated by commas), which may themselves be simple, i.e. individual constants, or complex.
- An *atomic sentence* is the result of writing an *n*-ary predicate letter followed by *n* terms (in parentheses, separated by commas).
    - Happy(father(joe)); OlderThan(father(joe), joe).
    - This is nonsense: Happy(Happy(joe)).

- Just as we require all individual constants to denote exactly one thing, so we require all complex terms to denote exactly one thing.
  - So we can't have a function symbol 'sonOf', unless everyone in the domain we're talking about is a person with exactly one son.
- Just as the identity predicate '=' is traditionally written in 'infix' notation, so certain function symbols are traditionally written in infix notation.

The language of arithmetic

- Binary predicates: $=$, $<$
- Individual constants: 0, 1
- Binary function symbols: $+$, $\times$

Proofs

- We'll introduce three methods for showing that a certain claim is a logical consequence of certain premises: *informal proof*, *formal proof* and *truth-tables*.
- In a *proof*, we start with the given premises, and step by step we establish intermediate conclusions that *obviously* follow from things we've already said, until eventually we reach the desired conclusion.
  - Actually that's only the most basic sort of proof: later on we'll introduce more complicated *methods of proof* that rely on *subproofs*.

- In an informal proof, you’re allowed to make any step provided it’s obvious to your audience how it follows from what you’ve already said.
  - Unless your audience is a logic teacher, in which case the standard for ‘obviousness’ is higher.
  - Informal proofs in logic should be completely rigorous. You’ll have to develop a special writing style: a useful skill.
- In a *formal* proof, the allowable steps are codified into a fixed set of mechanical rules.

Informal proofs using atomic sentences

- Given what we know about the meaning of the predicates in the blocks language, there are plenty of obviously valid arguments, e.g.:
  - 'LeftOf(a, b)' entails 'RightOf(b, a)'
  - 'LeftOf(a, b)' and 'LeftOf(b, c)' entail 'LeftOf(a, c)'
  - 'LeftOf(a, b)' and 'SameCol(b, c)' entail 'LeftOf(a, c)'
- Too many to list or codify in a formal system of proof.

- The *identity* predicate ‘=’ is of special interest: it’s one of the bits of vocabulary that logic has traditionally been especially concerned with.
  - The most important method of proof involving identity goes by the names *identity elimination*, *substitution*, *the indiscernability of identicals* and *Leibniz’s Law*.
    - Roughly: if we have established from our given premises that $a=b$, we can infer that whatever is true of a is true of b.
    - Given a premise that involves a certain name, say a, and a premise of the form $a = b$, we can infer the result of substituting b for a in the first premise. Familiar from algebra.

- We also have the rule of *identity introduction*, aka *the reflexivity of identity*: this lets us infer a sentence of the form ‘a=a’ from whatever premises we please, or from none at all.
  - *Logically true* sentences are sentences that *must* be true. They follow from every other sentence. We can also say that they follow from the null set of premises.
  - The assumption that all names have referents is playing a crucial role here. Is ‘Santa is identical to Santa’ a true sentence in English?

- Other useful principles:
  - the *symmetry of identity* (from ‘a = b’ we can conclude ‘b = a’)
  - the *transitivity* of identity (from ‘a = b’ and ‘b = c’, conclude ‘a = c’)
- These can in fact be derived from the first two principles. (B&E derive symmetry on p. 50)

Problems for next week:

- 1.9 (30%)
- 1.11 (25%)
- 2.6 (15%)
- 2.8 - 2.13 (30%)