# A rapid run through the basics of Propositional and Predicate Logic

Mark Neal
mjn@aber.ac.uk

## Logic

- Blame the Greeks
  - Aristotle and others started to think about how people used language and made arguments
  - Syllogisms
  - All men are mortal. All Greeks are men. All Greeks are mortal.
- Modern Logic
  - Formal structure
  - Well defined notation and functionality
  - Mathematical
  - Huge number of logics for different purposes
  - This module: Propositional and First Order Predicate Logic

## Components

- Constants
  - Values for things
  - Numeric:     11
  - Labels:     Clegg
- Variables
  - Just as in programming languages
- Functions
  - Return a unique value about something
  - NumberofLies('Clegg')

# Logical Connectives

- Negation – inverts truth of a term
  - NOT ¬
- Conjunction – requires both sides to be true to be true
  - AND ^
- Disjunction – either side can be true to be true
  - OR v
- Implication – if then (a bit odd though) material implication
  - IF =>
- Equality – if and only if – implication in both directions
  - IFF <=>

## Quantifiers

- For All
  - ∀ is like a lot of AND connectives
  - ∀x likes(X, icecream) – everyone likes icecream
- There Exists an
  - ∃x is like a lot of OR connectives
  - ∃x ¬likes(X, Clegg) – someone does not like Clegg
- Notice that you can do things with negation:
  - ∃x ¬likes(X, Clegg) is equivalent to
  - ¬∀x likes(X, Clegg)

# Propositional Logic

- A special case that uses these symbols and ideas
- Things are either **true** or **false**
- Nothing else
  - 'Clegg_was_elected' is a proposition...
  - …that is **true**
  - 'Clegg_is_in_power' is a proposition...
  - …that is only partially **true**
- So it would be very difficult to encapsulate a discussion about Clegg's political situation in Propositional Logic
  - ('Clegg_was_elected') ^ ('Mark_dislikes_Clegg') => ('Mark_will_probably_vote_for_Plaid_Cymru_next_time')
- But we can still make interesting expressions

## Predicate Logic

- Much richer
  - Represents objects and their states or actions
  - Which allow relations to be constructed:
  - elected('clegg', 'mark') => disappointed('mark')
  - Literals that are always true can also be used:
  - optimistic('mark') politician('clegg') catholic('pope')
- Could you do this in propositional logic?
- What would it look like?
- Need a lot of unrelated propositions because there's no possibility of encapsulating information:
  - 'mark' or 'clegg'
  - No generalization like elected('Clegg', X)

# Horn Clauses and Logic Programming

- Clauses are just a bunch of literals joined up by connectives and an implication:
  - A v B ^ C ^ ¬D => E
- Horn clauses are special:
  - A ^ B ^ C ^ D => E
  - Used as the basis for logic programming systems
  - Equivalent to Turing machines: Turing complete
- Prolog turns it around:
  - E :- A,B,C,D
  - Uses implication on predicates to search for matches to queries using Horn clauses to get there

# Things to try in Prolog

- **assert** some facts
  - assert(politician(clegg)).
  - assert(politician(cameron)).
  - assert(politician(opik)).
  - assert(male(clegg)).
- Run some **queries**
  - politician(X).
  - Try pressing enter after this
  - Then try pressing semicolon
- Run a more interesting **query**
  - politician(X),male(X).
  - Try pressing semicolon
  - Make other politicians male with assertions
- Don't forget the full-stops at the end of the lines
- Use *swipl* to start and *halt.* to close prolog

# Prolog: unification and backtracking

- Prolog uses a depth first search to find something that matches the “goal”
- It will attempt to match:
  - Name of predicate
  - Number of arguments
  - The arguments themselves
- `politician(X)`
  - Goes through database looking for facts that match
  - Then instantiates `X` with first value it finds
  - Asking for the next one `(;)` continues the search

## How is this useful?

- You can populate the database with whatever facts you have about some topic
- Then you can set goals for Prolog to attempt to match
- Prolog will find any set of facts that matches your goal no matter how many predicates it needs to go through to do it
- So you can concentrate on the “facts” that you know about your problem...
- ...rather than about how to fit them together to answer the question that you are tackling