Math Foundations: Predicate Calculus

- A symbolic form of logic that deals with expressing and reasoning about **propositions**
- Statements/queries about state of the "universe"
- Simplest form: **atomic proposition**
- Form: functor (parameters)
- Examples: man (jake)
  like (bob, redheads)
- Can either assert truth ("jake is a man") or query existing knowledge base ("is jake a man?")
- Can contain variables, which can become bound
  man (x)

Compound Propositions

- Contain two or more atomic propositions connected by various logical operators:

Name	Symbol	Example	Meaning
negation	$\neg$	$\neg$ a	not a
conjunction	$\wedge$	a $\wedge$ b	a and b
disjunction	$\vee$	a $\vee$ b	a or b
equivalence	$\equiv$	a $\equiv$ b	a is equivalent to b
implication	$\Rightarrow$	a $\Rightarrow$ b	a implies b
	$\Leftarrow$	a $\Leftarrow$ b	b implies a

Predicate Calculus: Quantifiers

- Quantifiers bind variables in propositions
  - Universal quantifier: $\forall$
  - $\forall$x.P -- means "for all x, P is true"
  - Existential quantifier: $\exists$
    $\exists$x.P -- means "there exists a value of x such that P is true"
  - Examples:
    $\forall$x.(woman(x) $\Rightarrow$ human(x))
    $\exists$x.(mother(mary, x) $\wedge$ male (x))

Clausal Form

- "Clausal form" is a canonical form for propositions:

  $B1 \vee B2 \vee \ldots \vee Bn \Leftarrow A1 \wedge A2 \wedge \ldots \wedge Am$
- Means: if all of the A's are true, at least one of the B's must be true
- Right side is the **antecedent**; left side is the **consequent**
- Examples:

likes(bob, mary) $\Leftarrow$ likes(bob, redheads) $\wedge$ redhead(mary)

father(louis, al) $\vee$ father(louis, violet) $\Leftarrow$ father(al, bob) $\wedge$ mother(violet, bob) $\wedge$ grandfather(louis,bob)

Horn Clauses

- A proposition with zero or one term in the consequent (left) is called a **Horn clause**
- If there are no terms, it is called a **headless** Horn clause:

  man(jake)
- If there is one term, it is a **headed** Horn clause:

  person(jake) $\Leftarrow$ man(jake)

Resolution

- The process of computing inferred propositions from given propositions
- Example:
  if we know:
  - older(joanne, jake) $\Leftarrow$ mother(joanne, jake)
  - wiser(joanne, jake) $\Leftarrow$ older(joanne, jake)

  we can infer the proposition:
  - wiser(joanne, jake) $\Leftarrow$ mother(joanne, jake)
- There are several logic rules that can be applied in resolution. In practice, the process can be quite complex.

Prolog — Control

- The right hand sides of predicates are "evaluated" left to right
- On a right hand side, a false predicate causes the system to return to the last predicate to its left having a true value; a true result allows the evaluation of the right hand side to continue to the right.
- Collections of predicates are "examined" in their lexical (textual) order — top to bottom, first to last
- Recursion!

Prolog Control

- A reference to a predicate is like a "function call" to the collection of predicates of that name
- State of the program contains markers to last successful (i.e. True) instantiation in collections of facts or rules to support backtracking in recursion
- When all markers are beyond end of all applicable predicate collections, result is "no"

Prolog — Modularity and Abstraction

- Facts and predicates of the same name (and same # of parameters) are collected by a Prolog system to form modules — the pieces do not have to be textually contiguous
- Collections of facts and rules may be stored in separate named files
- Files are "consulted" to bring them into a workspace

Imperatives Continued

- Comparison Operators
  =, \=, <, >, >=, =<, =:=
- Most Prologs support integer arithmetic expressions
- SWI Prolog supports integer and floating point math expressions well
- "Assignment" (local) uses the infix "is" operator; assigns right hand side value to variable on left:

  X is (3+4)