Resolution (14A)

Young W. Lim
4/23/14

based on the following document:
http://www.learnprolognow.org/ Learn Prolog Now!

Please send corrections (or suggestions) to youngwlim@hotmail.com.

This document was produced by using LibreOffice/OpenOffice.

Modus Ponens

The Prolog resolution algorithm
based on the modus ponens form of inference

a general rule – the major premise and
a specific fact – the minor premise

All men are mortal
Socrates is a man
Socrates is mortal

Facts	a	a	Facts	man('Socrates').
Rules	$a \rightarrow b$	b :- a	Rules	mortal(X) :- man(X).
Conclusion	b	b	Conclusion	mortal('Socrates').

Lists

A reversed modus ponens is used in Prolog

Prolog tries to prove that
a query (b) is a consequence of
the database content (a, a ⇒ b).

Using the major premise, it goes from b to a,
and using the minor premise, from a to true.

Such a sequence of goals is called a **derivation**.

A derivation can be **finite** or **infinite**.

b :- a
a   true

Facts	a	a
Rules	a → b	b :- a
Conclusion	b	b

a
major premise   b :- a   minor premise
b

Lists

A categorical syllogism consists of three parts:

Major premise:	All humans are mortal.
Minor premise:	All Greeks are humans.
Conclusion:	All Greeks are mortal.

Each part - a categorical proposition - two categorical terms

In Aristotle, each of the premises is in the form

"All A are B"	universal proposition
"Some A are B"	particular proposition
"No A are B"	universal proposition
"Some A are not B"	particular proposition

Each of the premises has one term in common with the conclusion:
this common term is called
a major term in a major premise (the predicate of the conclusion)
a minor term in a minor premise (the subject of the conclusion)

Mortal is the major term,
Greeks the minor term.
Humans the middle term

Lists

- **Initialization**

  Initialize **Resolvent** to **Q**, the initial goal of the resolution algorithm.

  Initialize the final substitution σ to {}

  Initialize **failure** to false

- **Loop with Resolvent = G1, G2, ..., Gi, ..., Gm**

  while (**Resolvent** ≠ ∅) {

  1. Select the goal **G**i ∈ **Resolvent**;
  2. If **G**i == true, delete it and continue;
  3. Select the rule **H :- B**1, ..., **B**n in the database

     such that **G**i and **H** unify with the **MGU** θ.

     If there is no such a rule then set **failure** to true; break;
  4. Replace **G**i with **B**1, ..., **B**n in **Resolvent**

     % **Resolvent** = **G**1,...,**G**i−1, **B**1,...,**B**n, **G**i+1,..., **G**m
  5. Apply θ to **Resolvent** and to **Q**;
  6. Compose σ with θ to obtain the new current σ; %the final substitution

  }

Most General Unifier

G1, **G**2, ..., **G**i, ..., **G**m

↑

H :- B1, ..., **B**n

Resolution (14A)

References

[1] en.wikipedia.org
[2] en.wiktionary.org
[3] U. Endriss, "Lecture Notes : Introduction to Prolog Programming"
[4] http://www.learnprolognow.org/ Learn Prolog Now!
[5] http://www.csupomona.edu/~jrfisher/www/prolog_tutorial
[6] www.cse.unsw.edu.au/~billw/cs9414/notes/prolog/intro.html
[7] www.cse.unsw.edu.au/~billw/dictionaries/prolog/negation.html
[8] http://ilppp.cs.lth.se/, P. Nugues, An Intro to Lang Processing with Perl and Prolog