16. Logical Programming

Logic Programming

- Neither imperative nor functional
- Deals with relations, not functions
  - Arguments and results are treated uniformly
  - daughter(sue,john)
  - lessthan(3,10)
- Separate logic from control
  - Programmer declares what facts and relations are true
  - System determines how to use facts to solve problems

Relations (Predicates)

- A table with $n \geq 0$ columns and a possibly infinite set of rows
- A tuple $(a_1,a_2,\ldots,a_n)$ is in a relation if a_i appears in column i, $1 \leq i \leq n$
- Example: Relation *append* is a set of tuples of the form (X,Y,Z), where Z consists of the elements of X followed by the elements of Y
- Relations can be specified using Horn clauses (rules)

Horn Clauses (Rules)

- A Horn clause:

  P if Q_1 and Q_2 and … and Q_k, $k \geq 0$

  can be interpreted as:

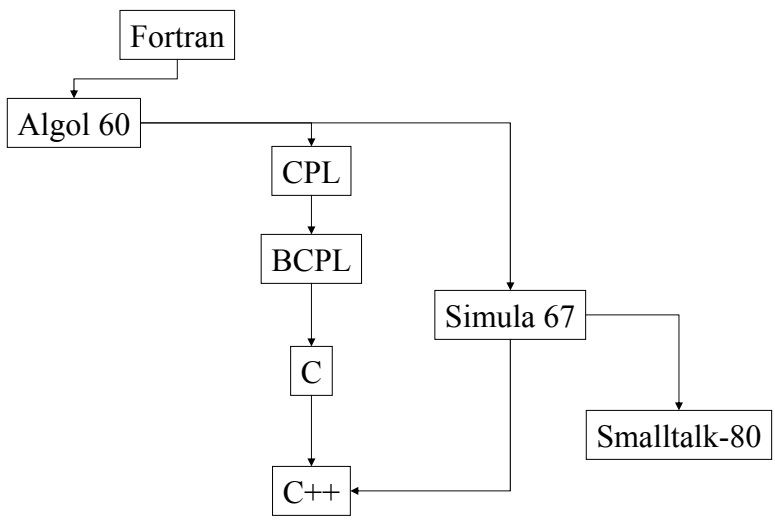
  The consequent, P, is true if the antecedents Q_1, $Q_2,\ldots, Q_k$, $k \geq 0$ are all true

  - Antecedents: conjunction of zero or more conditions that are atomic formulae in predicate logic
  - Consequent: an atomic formula in predicate logic

Horn Clause Terminology

- Horn Clause ⇔ Clause
- Consequent ⇔ Goal ⇔ Head
- Antecedents ⇔ Subgoals ⇔ Tail
- Horn clause with no tail ⇔ Fact
- Horn clause with tail ⇔ Rule
- P if Q_1 and Q_2 and … and Q_k ⇔ p :- q_1 , …, q_n

Example: Links Between Languages



Example Prolog Facts and Rules

- A prolog program starts with a collection of facts

```
link(fortran, algol60)
link(algol60, cpl)
link(cpl, bcpl)
link(bcpl, c)
link(c, cplusplus)
link(algol60, simula67)
link(simula67, cplusplus)
link(simula67, smalltalk80)
path(L, L)
path(L,M) :- link(L,X), path(X,M)
```

In Prolog

- A simple term is a number, variable or an atom that stands for itself
  - All variables start with capital letters
  - All constants are in lower case
- A compound term consists of an atom followed by a parenthesized sequence of subterms.
  - The atom is called a functor
  - Subterms are called arguments
  - All predicates are in lower case

Horn Clauses with Variables

- Variables may appear in the antecedents and the consequent of a Horn clause:
  - $p(X_1,X_2,\ldots,X_n)$:- $h(X_1,X_2,\ldots,X_m)$
  - $p(X_1,X_2,\ldots,X_n)$:- $h(X_1,X_2,\ldots,X_m,Y_1,Y_2,\ldots,Y_k)$

Queries

- Simplest form: Does a particular tuple belong to a relation
  - path(algol60,smalltalk80).
- Can't ask: Does a particular tuple not belong to a relation
  - Yes/Fail rather then Yes/No answers, where Fail indicates a failure to deduce an answer.
- More interesting: Is there an X such that a specific clause containing X evaluates to yes
  - path(X,cplusplus)

Examples

```
1 ?- consult(myrules).
myrules compiled, 0.00 sec, 1,640 bytes.
Yes
2 ?- path(X,cplusplus).
X = cplusplus ;
X = fortran ;
X = fortran ;
X = algol60 ;
X = cpl ;
X = bcpl ;
X = c ;
X = algol60 ;
X = simula67 ;
No
3 ?- path(c,Y).
Y = c ;
Y = cplusplus ;
No
4 ?- path(algol60,smalltalk80).
Yes
```

More Examples

```
?- link(N,M), link(L,M).
N = fortran
M = algol60
L = fortran
Yes
?- link(N,M), link(L,M), not(L=N).
N = c
M = cplusplus
L = simula67
Yes
```