CS2136: Paradigms of Computation

Class 05: Lists Operators

Complicated Rules

Copyright 2001, 2002, 2003 Michael J. Ciaraldi and David Finkel
Equality and Inequality
Equality and Inequality

* z Equals: “=“
* z Not Equals: “\=“
Rules for Equality

z $X = Y$ returns “Yes” if...

y $X$ is uninstantiated and $Y$ is instantiated:
  x $X$ becomes instantiated to same object as $Y$.

y $X$ and $Y$ are the same atom or integer
  x Or are instantiated to it.

y $X$ and $Y$ are structures with the same functor and the same number of components, and each component in $X$ is equal to the corresponding component in $Y$.

y $X$ and $Y$ are uninstantiated:
  x They “share” a.k.a. “co-refer”.
Sharing

Once two Prolog variables share, when one gets instantiated to a value, the other gets instantiated to the same value.

Y=W, owns(X,Y), owns(Z,W).
Lists
Lists

- Prolog supports lists
  - Should be familiar from CS 2135
  - Notation is completely different from Scheme

- Prolog notation:
  - Square brackets, with list elements separated by commas
  - [john, mary, wine, football]
Heads and Tails I

- Key operation on lists is to separate the head of the list from the rest of the list (the tail of the list).
- Head of the list is the first element of the list
- Tail of the list is the list of everything after the first element
Heads and Tails II

z For the list [john, mary, wine, football], the head is john and the tail is [mary, wine, football].

z Special notation for heads and tails:
  - \([X|Y]\) means the list with head X and tail Y.
Example with Lists

If we have a fact

students([john, mary, sam, sally]).

And we ask the question

?- students([X|Y]).

Prolog will make the instantiations

X = john, Y = [mary, sam, sally]
Member

\[ \text{member}(X, [Y|_]) :- X=Y. \]

\[ \text{member}(X, [_|Y]) :- \text{member}(X,Y). \]

How does this work?

It’s recursive, but where is the base case?

Suppose we ask

?- member(george, [john, mary, sam, sally]).

See example on page 55 of C&M.
Operators
Math Operators

z Infix
  y + - * /

z Prefix
  y -

z Postfix
  y !  [Factorial in math, not in Prolog; in Prolog it is “cut”.]
Math Operators II

z Cannot just use parentheses to control precedence.

z Infix operators can be written as prefix, with parentheses.

y \ a - b / c \ = \ - (a, / (b, c)).

z These create structures, not numbers.

z Use the “is” predicate to actually evaluate the arithmetic expression.
Math Example

z Database of countries, populations, and areas.
pop(usa,203).
pop(india,548).
pop(china,800).
pop(brazil,108).
area(usa,3).
area(india,1).
area(china,4).
area(brazil,3).

z Calculate density

density(X,Y) :- pop(X,P), area(X,A), Y is P/A.

y The “is” predicate evaluates the right side as an arithmetic expression, then compares to left.
Counting the number of elements on a list

\texttt{z} \texttt{listlen([],0). Meaning – the length of an empty list is 0}

\texttt{z} \texttt{listlen([H|T], N) :-}
\hspace{1cm} \texttt{listlen(T, N1), N is N1+1.}

\texttt{z} See alternate approach using an “accumulator” in C&M Section 3.7.
More Complicated Rules
Creating
More Complicated Rules

z See files:
  y family.pl
  y altFamily1.pl
  y altFamily2.pl
  y altFamily3.pl

z What’s wrong with each of these?
Creating More Complicated Rules II

z General approaches:
  y Build up complicated rules from simpler rules.
  y Have Prolog rules match English rules.
    x e.g. “same_mother” should be independent of gender of child.
  y Don’t rely on tricks; code the rules to make the test.
    x Or at least document your tricks & assumptions!
Next Time

z More Prolog:
  y Satisfying multiple subgoals.
  y Cut!