316 12-10-09

Note Title 12/10/2009

- PROLOG
  - Terms - Symbolic data objects of PROLOG
    - Variables = Identifiers beginning with an uppercase letter
    - Constants = Identifiers beginning with a lowercase letter - corresponds to atoms of LISP
    - structured terms formed from variables, constants, and functors
      - functors = Identifiers beginning with a lowercase letter

$\underset{\uparrow \text{ functor}}{\underline{f}}(X, Y)$, $\underset{\uparrow \text{ functor}}{\underline{\text{course}}}(\text{CourseNum}, \text{CourseCode})$, course($\underline{\text{cs316}}$, $\underline{\text{a012b}}$), $\underline{f}(\underline{g}(a, 3), \underline{f}(x, Y, \text{cs316}), \underline{h}(\text{abc}, 55))$

constants

functors

Used as data objects, not as function applications

- Lists are special kinds of terms

  $[e_1, \ldots, e_n]$, $n \geq 0$

  $[\,]$, called nil, denotes the empty list

  Head to tail notation

  $[e_1, \ldots, e_n] = [e_1 \mid [e_2, \ldots, e_n]]$

  $= [e_1 \mid [e_2 \mid [e_3, \ldots, e_n]]]$

  $\vdots$

  $= [e_1 \mid [e_2 \mid [e_3 \mid \ldots [e_n \mid [\,]] \ldots]]]$

$[1,2,3] = [1 \mid [2,3]]$

$= [1 \mid [2 \mid [3]]]$

$= [1 \mid [2 \mid [3 \mid [\,]]]]$

Programs

$A :- B_1, \ldots, B_n.$ $n \geq 1$ A, B_i are atomic formulas

A is true if $B_1 \wedge \ldots \wedge B_n$ is True

(This is a conditional clause)

A. $\leftarrow$ A is true (fact clause)

Atomic formulas are syntactically identical to structural terms with functors

$p(x, a, Y)$, $p(f(x, a), g(Y))$ etc

The outermost functors are called predicates, or relation symbols, and denote k-ary relations over k terms

Atomic formulas are true or false, not data objects.

Define "length" relation for lists

```
length([], 0).  /* The length of the empty list [] is 0 */
length([H|T]) :- length(T, Y), L is Y+1.
/* The length of [H|T] is L if the lengths of T is Y and L is Y+1 */
```

Head element ↑ ↑ Tail list

Built-in arithmetic predicates

x is $e_1 + e_2$

x is $e_1 - e_2$

x is $e_1 * e_2$

x is e_1 / e_2

Prolog computation is "generalized BNF reduction sequences" with pattern matching called unification

length([a,b,c], Z), /* compute the length Z of [a, b, c] */

unifier substitution → $\mid \Theta_1 = \{H_1 = a,\ T_1 = [b, c],\ L_1 = Z\}$

length([b,c], Y_1), L_1 is $Y_1 + 1$

|

$|\ \theta_2 = \{H_2 = b, T_2 = [c], L_2 = Y_1\}$

length([c], Y_2), L_2 is $Y_2 + 1$, L_1 is $Y_1 + 1$

$|\ \theta_3 = \{H_3 = c, T_3 = [\,], L_3 = Y_2\}$

length([ ], Y_3), L_3 is $Y_3 + 1$, L_2 is $Y_2 + 1$, L_1 is $Y_1 + 1$

$|\ \theta_4 = \{Y_3 = 0\}$

L_3 is $0+1$, L_2 is $Y_2 + 1$, L_1 is $Y_1 + 1$

$|\ \theta_5 = \{L_3 = 1\}$

L_2 is $1+1$, L_1 is $Y_1 + 1$

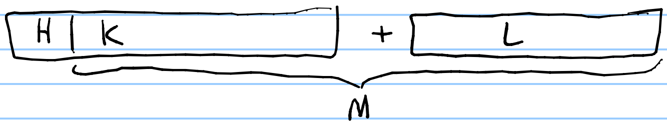
$|\ \theta_6 = \{L_2 = 2\}$

L_2 is $2+1$

$|\ \ \theta_7 = \{L_1 = 3\}$

success

```
/* append(L1,L2,L) holds if L is the result of appending L2 to L1 */
append([], L, L).   /* L is the result of appending [] and L */
append([H|K], L, [H|M]) :- append(K, L, M).
```



Prolog can be used for database applications

Parallel implementation of functional languages and Prolog