Logic Programming

Louden
12 Logic (1)

- formal specifications
- axiomatic semantics.

- theorem provers. → Prolog.

12.1 First-Order Predicate Calculus

consts – numbers or names.
predicates – fns that are true or false.
functions – others.
variables – unspecified objs.
connectives – and, or, not, impl, equiv
quantifiers – ∃, ∀
punctuation.

12.2 Horn clause

$$\underset{\text{head}}{b} \leftarrow \underbrace{a_1 \wedge a_2 \wedge a_3 \cdots \wedge a_n}_{\text{body}}$$

ex: sort (x, y) ← permute(x, y) and sorted(y)

no "or", no "not", ∴ logically incomplete
(or) rep by mult clauses.
failure as false

ex: natural (0).
natural (X) :– natural (predecessor (X)).

head vars – universally quantified (∀)
body vars – existentially quantified (∃)

Louden
12 Logic (2)

12.2 Horn Clauses

$$b_1 \leftarrow a_{11}, a_{12}, \ldots, a_{1k}$$
$$b_2 \leftarrow a_{21}, a_{22}, \ldots, a_{2j}$$

comma means "and"
next rule means "or"

sort (x,y) ← permute (x,y), sorted (y).

Bozo sort $O(n!)$

12.3 Resolution

head of 1st clause in body of 2nd, subst.

ex
$$a \leftarrow a_1, a_2, \ldots, a_n$$
$$b \leftarrow b_1, b_2, \ldots, b_m$$

suppose b_i matches a
then we infer
$$b \leftarrow b_1, \ldots, b_{i-1}, a_1, \ldots, a_n, b_{i+1}, \ldots, b_m$$

goal = Horn clause w/o head.

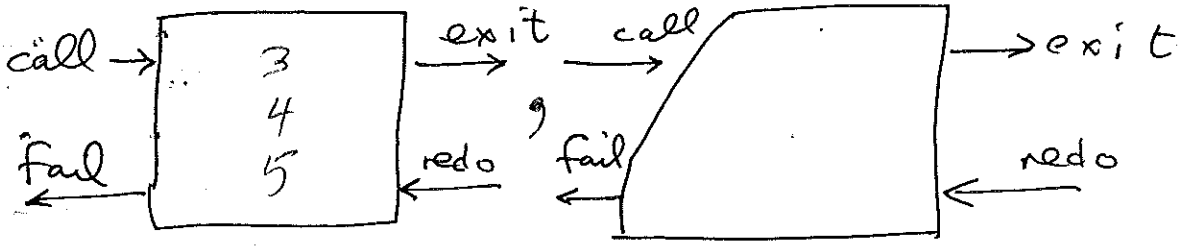
if ← a is a goal
then ← $a_1, \ldots, a_n$ is a subgoal.

Pattern matching → unification

```
Script started on Tue Mar  6 21:09:45 2012
bash-3.2$ cat -n he\007lloworld.pl
     1  % $Id: helloworld.pl,v 1.3 2011-05-19 19:53:59-07 - - $ */
     2
     3  hello :-
     4     write( 'Hello, World!' ), nl.
     5
bash-3.2$ gprolog
GNU Prolog 1.3.1
By Daniel Diaz
Copyright (C) 1999-2009 Daniel Diaz
| ?- [helloworld].
compiling /afs/cats.ucsc.edu/courses/cmps112-wm/Languages/prolog/Examples/hellow
orld.pl for byte code...
/afs/cats.ucsc.edu/courses/cmps112-wm/Languages/prolog/Examples/helloworld.pl co
mpiled, 5 lines read - 472 bytes written, 10 ms

yes
| ?- hello.
Hello, World!

yes
| ?-

bash-3.2$ exit

Script done on Tue Mar  6 21:09:59 2012
```

expect

Prolog - Box Model



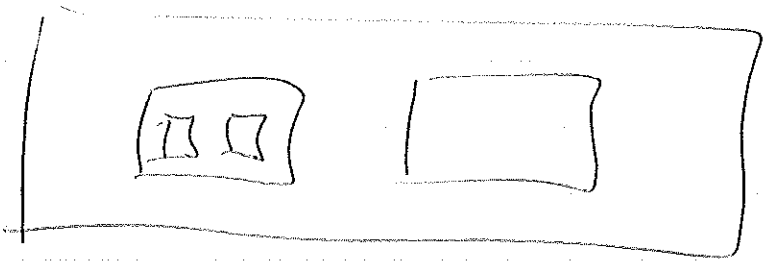
Call – 1st time soln is sought.
- seek clause that unifies goal

Fail – unification fails all attns.
- no unify head | orig invoc no soln
| bktrk no soln.

Exit – succeed.

Redo – proc call successful
- subsequent goal failed

Uses backtracking
Redo → Exit | Fail



oscillating recursion

```
laid(Chicken, Egg) :- laidby(Egg,Chicken).
laidby(Egg, Chicken) :- laid(Chicken, Egg).
```

∞!

```
1: laid(x, y).
?: laid(x, What).
   what = y ;
   what = y ;
   ⋮
   what = y ;
   ⋮
```

Nodes in general:

[H|T] is syntactic sugar for .(H,T)

```
?- .(H,T) = [a,b,c].
   H = a
   T = [b,c]
```

[a, b, c] ⟶ .(a, .(b, .(c, [])))

can simulate trees with terms.

Guess & Verify

Query: ?∃ S s.t. guess(s), verify(s) ?

subgoals.

ex: overlap (X, Y) :- member (M, X), member(M, Y).

guess M verify M.

?- overlap([a, b, c, d], [1, 2, c, d]).
yes.

?- member (M, [a, b, c, d]).

M = a;
M = b;
M = c;
M = d;
no.

efficiency: choose guess w/ fewer solns.

?- X = [1, 2, 3] , member (a, X)
no.

?- member (a, X), X = [1, 2, 3].
∞ computation
X = [a|_] ;
X = [_, a, |_] ;
X = [_, _, a, |_] ;
∞.

More list examples

```
append([], List, List).
append([Element|L1], L2, [Element|L3]) :-
        append(L1, L2, L3).
reverse([], []).
reverse([H|T], L) :- reverse(T, Result),
        append(Result, [H], L).
palindrome(L1, L2) :- append(L1, L3, L2),
        reverse(L1, L3).
```

Terms as Data

```
leaf
node(x, leaf, leaf)
node(x, node(y, leaf, leaf), node(x, leaf, leaf))

member(K, node(K, _, _)).
member(K, node(N, S, _)) :-
        K < N, member(K, S).
member(K, node(N, _, S)) :-
        K > N, member(K, S).
```

same as:

```
member(K, U) :-
        U = node(N, S, T),
        K = N.
```

```
f(X, Y) :- X = Y.
f(X, 2)
  X = 2
  f(2, Y)
    Y = 2
```

$B :- C_1 \ldots C_{j-1}, !, C_{j+1} \ldots C_K.$

- back track past $C_{j-1} \ldots C_1, B$ w/o considering remaining rules.

arithmetic

X is 3 + 4.

operators: + − * / // mod.

float int

compare numbers: =:= =\= > < >= =<

~~if (x<y)~~ ~~f(x,y)~~ fun min(x,y) = if x<y then x else y.

```
min(X, Y, M) :- X < Y, !, M is X.
min(X, Y, M) :- X >= Y, M is Y.
```

Back to Cuts ...

```
conclu(S) :- guess(S), !, verify(S)
```

↑ elim all but 1st guess.

prune search tree.

```
member(K, node(K, _, _)).
member(K, node(N, S, _)) :- K < N, !, member(K, S).
member(K, node(N, _, T)) :- K > N, !, member(K, T).
```

guess & verify