XSB Prolog (cont.)

CSE 392, Computers Playing Jeopardy!, Fall 2011

Stony Brook University

http://www.cs.stonybrook.edu/~cse392

Example

```
mother_child(trude, sally).
father_child(tom, sally).
father_child(tom, erica).
father_child(mike, tom).
parent_child(X, Y) :- father_child(X, Y).
parent_child(X, Y) :- mother_child(X, Y).
sibling(X, Y):- parent_child(Z, X), parent_child(Z, Y),
  X \= Y.

?- trace.
?- sibling(X, Y).
```

Evaluation

- **`?- father_child(Father, Child).`**
  enumerates all valid answers on backtracking.

Append example

```
append([],L,L).
append([X|L], M, [X|N]) :- append(L,M,N).
```

```
append([1,2],[3,4],X)?
```

Append example

```
append([],L,L).
append([X|L],M,[X|N]) :- append(L,M,N).
```

append([1,2],[3,4],X)?	X=1,L=[2],M=[3,4],A=[X|N]

Append example

```
append([],L,L).
append([X|L],M,[X|N]) :- append(L,M,N).
```

append([2],[3,4],N)?	
append([1,2],[3,4],X)?	X=1,L=[2],M=[3,4],A=[X|N]

Append example

```
append([],L,L).
append([X|L],M,[X|N’]) :- append(L,M,N’).
```

append([2],[3,4],N)?	X=2,L=[],M=[3,4],N=[2|N’]
append([1,2],[3,4],X)?	X=1,L=[2],M=[3,4],A=[1|N]

Append example

```
append([],L,L).
append([X|L],M,[X|N']) :- append(L,M,N').
```

append([],[3,4],N')?	
append([2],[3,4],N)?	X=2,L=[],M=[3,4],N=[2|N']
append([1,2],[3,4],X)?	X=1,L=[2],M=[3,4],A=[1|N]

Append example

```
append([],L,L).
append([X|L],M,[X|N']) :- append(L,M,N').
```

append([],[3,4],N')?	L = [3,4], N' = L
append([2],[3,4],N)?	X=2,L=[],M=[3,4],N=[2|N']
append([1,2],[3,4],X)?	X=1,L=[2],M=[3,4],A=[1|N]

Append example

```
append([],L,L).
append([X|L],M,[X|N’]) :- append(L,M,N’).
```

```
A = [1|N]
 N = [2|N’]
 N’= L
 L = [3,4]
```

Answer: **A** = **[1,2,3,4]**

append([],[3,4],N’)?	L = [3,4], N’ = L
append([2],[3,4],N)?	X=2,L=[],M=[3,4],N=[2|N’]
append([1,2],[3,4],X)?	X=1,L=[2],M=[3,4],A=[1|N]

Quicksort Example

```
partition([], _, [], []).
partition([X|Xs], Pivot, Smalls, Bigs) :-
    (   X @< Pivot ->
        Smalls = [X|Rest],
        partition(Xs, Pivot, Rest, Bigs)
    ;   Bigs = [X|Rest],
        partition(Xs, Pivot, Smalls, Rest)
    ).
quicksort([])     --> [].
quicksort([X|Xs]) -->
    { partition(Xs, X, Smaller, Bigger) },
    quicksort(Smaller), [X], quicksort(Bigger).
```

Interfaces to Java

- XSB Prolog: InterProlog (native || sockets)
- SWI-Prolog: JPL (native)
- Sicstus: PrologBeans (sockets)

More Examples

```
member(X,[X|R]).
member(X,[Y|R]) :- member(X,R)
```

- *X is a member of a list whose first element is X.*
- *X is a member of a list whose tail is R if X is a member of R.*

```
?- member(2,[1,2,3]).
Yes
?- member(X,[1,2,3]).
X = 1 ;
X = 2 ;
X = 3 ;
No
```

More Examples

select(X,[X|R],R).

select(X,[F|R],[F|S]) :- select(X,R,S).

- *When X is selected from [X|R], R results.*
- *When X is selected from the tail of [X|R], [X|S] results, where S is the result of taking X out of R.*

?- select(X,[1,2,3],L).

X=1 L=[2,3] ;

X=2 L=[1,3] ;

X=3 L=[1,2] ;

No

More Examples

```
append([],X,X).
append([X|Y],Z,[X|W]) :- append(Y,Z,W).

?- append([1,2,3],[4,5],X).
X=[1,2,3,4,5]
Yes
```

More Examples

```
reverse([X|Y],Z,W) :- reverse(Y,[X|Z],W).
reverse([],X,X).

?- reverse([1,2,3],[],X).
X = [3,2,1]
Yes
```

More Examples

```
perm([],[]).
perm([X|Y],Z) :- perm(Y,W), select(X,Z,W).

?- perm([1,2,3],P).
P = [1,2,3] ;
P = [2,1,3] ;
P = [2,3,1] ;
P = [1,3,2] ;
P = [3,1,2] ;
P = [3,2,1]
```

More Examples

- Sets

```
union([X|Y],Z,W) :- member(X,Z),  union(Y,Z,W).
union([X|Y],Z,[X|W]) :- \+ member(X,Z), union(Y,Z,W).
union([],Z,Z).

intersection([X|Y],M,[X|Z]) :- member(X,M), intersection(Y,M,Z).
intersection([X|Y],M,Z) :- \+ member(X,M), intersection(Y,M,Z).
intersection([],M,[]).
```

Definite clause grammar (DCG)

- A **DCG** is a way of expressing grammar in a logic programming language such as Prolog
- The definite clauses of a DCG can be considered a set of axioms where the fact that it has a parse tree can be considered theorems that follow from these axioms

DCG Example

```
sentence --> noun_phrase, verb_phrase.
noun_phrase --> det, noun.
verb_phrase --> verb, noun_phrase.
det --> [the].
det --> [a].
noun --> [cat].
noun --> [bat].
verb --> [eats].

?- sentence(X,[]).
```

DCG

- Not only context-free grammars
- Context-sensitive grammars can also be expressed with DCGs, by providing extra arguments

```
s --> symbols(Sem,a), symbols(Sem,b), symbols(Sem,c).
symbols(end,_) --> [].
symbols(s(Sem),S) --> [S], symbols(Sem,S).
```

DCG

```
sentence --> pronoun(subject), verb_phrase.
verb_phrase --> verb, pronoun(object).
pronoun(subject) --> [he].
pronoun(subject) --> [she].
pronoun(object) --> [him].
pronoun(object) --> [her].
verb --> [likes].
```

Parsing with DCGs

sentence(s(NP,VP)) --> noun_phrase(NP), verb_phrase(VP).

noun_phrase(np(D,N)) --> det(D), noun(N).

verb_phrase(vp(V,NP)) --> verb(V), noun_phrase(NP).

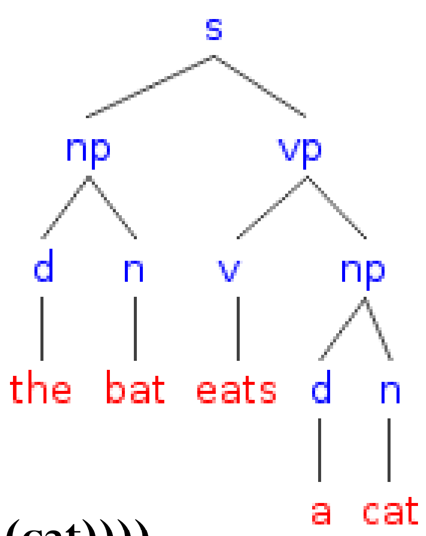
det(d(the)) --> [the].

det(d(a)) --> [a].

noun(n(bat)) --> [bat].

noun(n(cat)) --> [cat].

verb(v(eats)) --> [eats].



?- sentence(Parse_tree, [the,bat,eats,a,cat], []).

Parse_tree = s(np(d(the),n(bat)),vp(v(eats),np(d(a),n(cat))))

```
s --> np, vp.
np --> det, n.
vp --> tv, np.
vp --> v.
det --> [the].
det --> [a].
det --> [every].
n --> [man].
n --> [woman].
n --> [park].
tv --> [loves].
tv --> [likes].
v --> [walks].
```

```
| ?- s([a,man,loves,the,woman],[]).
yes
| ?- s([every,woman,walks],[]).
yes
| ?- s([a,woman,likes,the,park],[]).
yes
| ?- s([a,woman,likes,the,prak],[]).
no
```

Cut (logic programming)

- Cut (! in Prolog) is a goal which always succeeds, but cannot be backtracked past
- **Green cut**

  gamble(X) :- gotmoney(X),!.

  gamble(X) :- gotcredit(X), \+ gotmoney(X).

  - **cut says “stop looking for alternatives”**
  - by explicitly writing \+ gotmoney(X), it guarantees that the second rule will always work even if the first one is removed by accident or changed
- **Red cut**

  gamble(X) :- gotmoney(X),!.

  gamble(X) :- gotcredit(X).