CS 313 Lecture 31

Prolog:
  cut
  negation
Prolog cut: !

• Cut is used to prevent backtracking (prune search tree)

\[
G : - \ A, \ B, \ !, \ C, \ D.
\]

\[
G : - \ ...\]

• Once gets past !, solutions for A, B are fixed
• If C, D fails won’t consider other solutions for A, B, or G

• Example: cut.pl
a(1).

a(2).

b(Z) :-
    a(X),
    even(X),
    a(Y),
    Z is X+Y.

b(5).

?- b(Z).
Z = 3 ;
Z = 4 ;
Z = 5.
a(1).
a(2).
b(Z) :- !, a(X), even(X), a(Y), Z is X+Y.
b(5).

?- b(Z).
Z = 3 ;
Z = 4.
a(1).
a(2).
b(Z) :-
a(X), !, 
even(X),
a(Y), 
Z is X+Y.
b(5).

?- b(Z).
false.
a(1).
a(2).
b(Z) :-
a(X),
even(X), !,
a(Y),
Z is X+Y.
b(5).
?- b(Z).
Z = 3 ;
Z = 4.
a(1).

a(2).

b(Z) :-
  a(X),
  even(X),
  a(Y), !,
  Z is X+Y.

b(5).

?- b(Z).
Z = 3.
a(1).
a(2).
b(Z) :-
  a(X),
  even(X),
  a(Y),
  Z is X+Y, !.
b(5).

?- b(Z).
Z = 3.
a(1).
a(2).
b(Z) :-
a(X),
even(X),
a(Y),
Z is X+Y, !.
b(5).

?- b(5).

true
a(1).
a(2).
b(Z) :-
a(X),
even(X),
a(Y), !, 
Z is X+Y.
b(5).

?- b(5).
false
Uses of cuts

1. for efficiency

\[
\begin{align*}
\text{find}(X, \text{node}(X, _, _)) &= . \\
\text{find}(X, \text{node}(N, L, _)) &:: X<N, \text{find}(X, L) . \\
\text{find}(X, \text{node}(N, _, R)) &:: X>N, \text{find}(X, R) .
\end{align*}
\]
Uses of cuts

1. for efficiency

\[
\text{find}(X, \text{node}(X, _, _)) :- !.
\]
\[
\text{find}(X, \text{node}(N, L, _)) :- X<N, !, \text{find}(X, L).
\]
\[
\text{find}(X, \text{node}(N, _, R)) :- X>N, \text{find}(X, R).
\]

• doesn’t affect search tree: *green cut*

• can also use to prevent search after reaching base case
Uses of cuts

2. to purposely restrict backtracking

- Ex: pete1a.pl – only find one solution for each problem

- affects search tree: red cut
Negation as failure

• Can use cut to define \textit{not}

\texttt{not(X) :- X, !, fail.}
\texttt{not(_).}