Yet Another Example

main :- a.
main.
a :- b(B,C), !, c(C,D), d(D,B).
a :- e(B,C), f(C,D), g(D,B).
a :- h(B,B).

When hit the cut, removes ability to backtrack to other solutions in clause prior to backtrack.

Depth First Search Example
path(Y ,[Y | Path], [Y | Path]).
path(Y ,[Top | Rest], NewPath) :-
  connectedto(Top,Z), +member(Z,Rest), !,
  path(Z,Y ,[Z,Top | Path], NewPath).

?- path(16, [1], Path).

What is the effect of the above cut?

Cuts (Prolog book 4.3–4.4)

• Backtracking point: where Prolog can pursue different proof
  • Can control if backtracking points are kept
  • Removes alternatives seen so far from current goal

• Depth First Search Example

?- path(16, [1], Path).

What is the effect of the above cut?
Conditions

• Problem: print a message whether argument is an empty list, non-empty list, atom or other

\[ \text{print}([\]) \] :- write('argument is an empty list').
\[ \text{print}([|]) \] :- write('argument is a non-empty list').
\[ \text{print}(A) \] :- atom(A), write('argument is an atom').
\[ \text{print}(() \text{ or }) \] :- write('argument is other').

• Will it work right if we backtrack through code for \text{print}([])?

• How can we change it to work with \text{'\textasciitilde'}\text{'\textdagger'}?

• How can we change it to work with \text{'!'}?

Red cut • Can be used to implement an else more efficiently

- Guard against unwanted solutions by disallowing backtracking after first solution found.

Green cuts • Green cut:

- If you know you are not throwing away possible solutions
- Adding cut doesn't change behavior of program
- Just makes it more efficient
- Doesn't waste time to do fruitless check of alternatives
- Doesn't store choice points

\[ \text{print}([\]) \] :\!
\[ \text{write}('argument is an empty list'). \]
\[ \text{print}([|]) \] :\!
\[ \text{write}('argument is a non-empty list'). \]
\[ \text{print}(A) \] :\!
\[ \text{atom}(A), \text{write}('argument is an atom'). \]
\[ \text{print}(() \text{ or }) \] :\!
\[ \text{write}('argument is other'). \]

And another Example

• Problem: print a message about the type of argument

\[ \text{main} \] :- \text{print}([\]), fail.

• What will happen if we backtrack through this code?

• How can we make code more efficient?
More Prolog

- Strings
  - List of ASCII characters
- Printing strings
  - Use `put` to print a character
  - How do you print a string?