PROLOG: CUT & NEGATION

Comp3031 Lab 09
Fall 2014

JIA Xiaoying, SU Zhiyang

Cut

- Its syntax is simply “!”
- It may appear as a literal in the body of a goal or clause.
  - It is treated by Prolog as if it’s logically true
  - But it has side-effects

Position of !

q:- p1,p2…pn, !, r1,r2…rm

An example

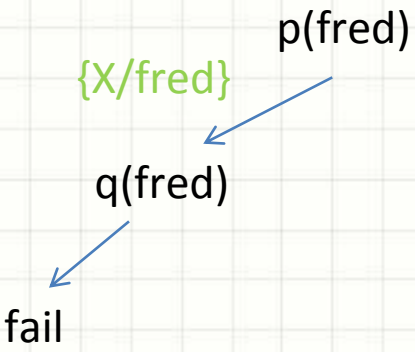
- Example:
  - p(X):-q(X).
  - p(X):-r(X,Y),!,s(Y).
  - p(X):-t(X).
- This program has three clauses, and Prolog will try them top to bottom (in the order they are written).

An example

- Suppose the program has the following four facts,
  - r(fred,b).
  - t(fred).
  - q(b).
  - s(a).
- Let's try to draw the search tree of query p(fred).

An example(cont.)

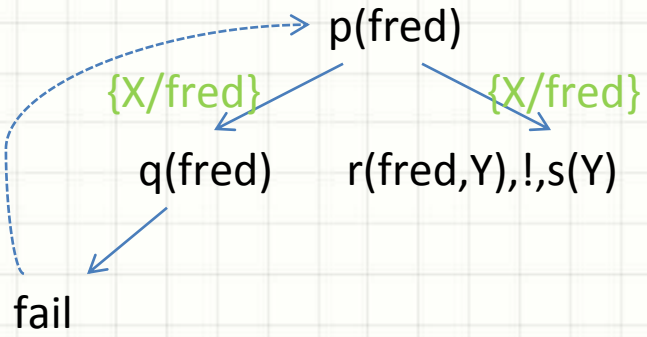
Goal:
p(fred)



Program:
r(fred,b).
t(fred).
q(b).
s(a).
p(X):-q(X).
p(X):-r(X,Y),!,s(Y).
p(X):-t(X).

An example(cont.)

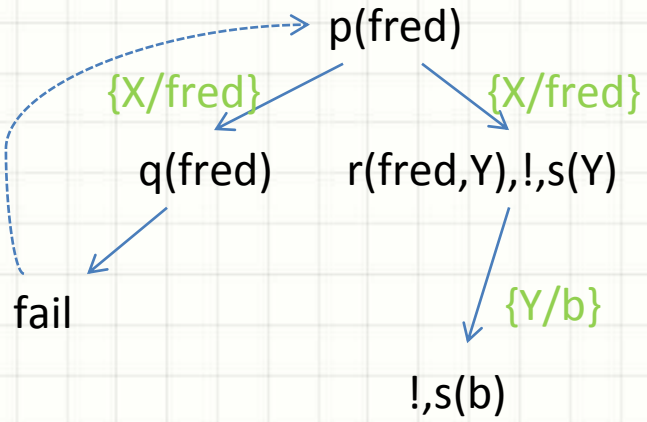
Goal:
p(fred)



Program:
r(fred,b).
t(fred).
q(b).
s(a).
p(X):-q(X).
p(X):-r(X,Y),!,s(Y).
p(X):-t(X).

An example(cont.)

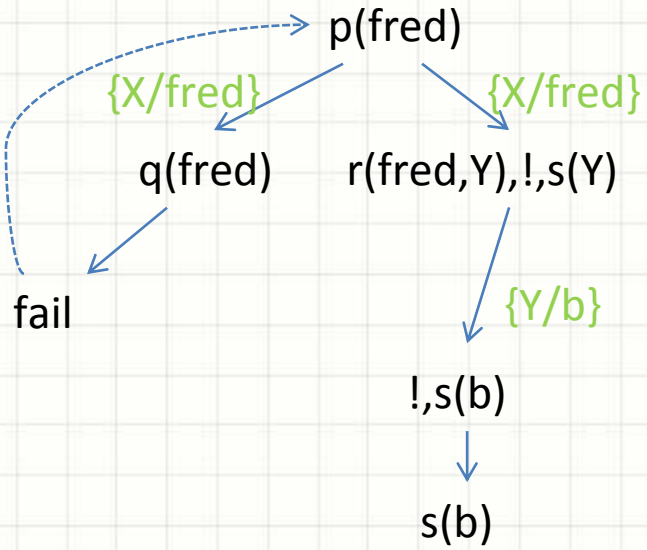
Goal:
p(fred)



Program:
r(fred,b).
t(fred).
q(b).
s(a).
p(X):-q(X).
p(X):-r(X,Y),!,s(Y).
p(X):-t(X).

An example(cont.)

Goal:
p(fred)

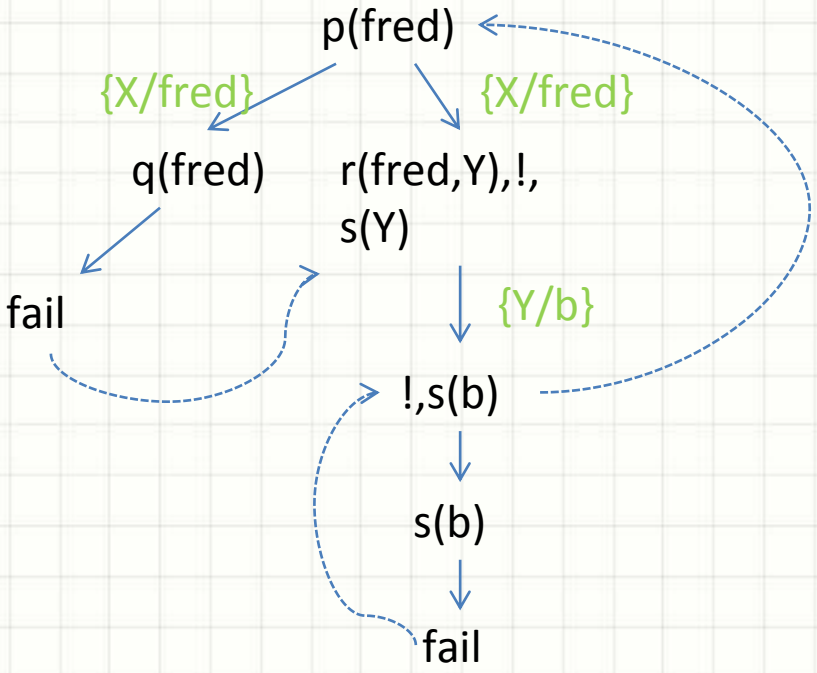


Program:
r(fred,b).
t(fred).
q(b).
s(a).
p(X):-q(X).
p(X):-r(X,Y),!,s(Y).
p(X):-t(X).

An example(cont.)

Goal:
p(fred)

Program:
r(fred,b).
t(fred).
q(b).
s(a).
p(X):-q(X).
p(X):-r(X,Y),!,s(Y).
p(X):-t(X).



An example(cont.)

- Notice that although the last clause,
  - p(X):-t(X).

  could prove our goal, it is not reached as there is a cut.

Why is cut helpful?

- Save space
  - Discarding alternatives to the current derivation can allow Prolog to reclaim the storage space that is used to save them.
- Save time
  - It may be that we know these alternatives cannot lead to a solution, so discarding them does not affect the solution.

Cut-free Example

- s(X,Y) :- q(X,Y).
- s(0,0).
- q(X,Y) :- i(X), !, j(Y).
- %what if q(X,Y) :- i(X),j(Y),!.
- i(1).
- i(2).
- j(1).
- j(2).
- j(3).

Query: s(X,Y).

Negation

- not(X), or written as \+ (X)
  - It doesn't mean that X is false
  - Just means that X cannot be proved true
- It's easily implemented using cut:
  - not(X) :- X, !, fail.
  - not(_).

Why is it useful?

- Rules in real life have many exceptions.
- For example:
  - Vincent enjoys burgers expect Big Kahuna burgers.
  - enjoys(vincent,X) :- burger(X),
    \+ big_kahuna_burger(X).

Exercise example

- The following query *member(Y,[[1,2],[3,4]]),member(X,Y).* returns X=1, X=2, X=3, X=4 successively. Now use cuts to rewrite the query such that it will return:
  - X=1 only
  - X=1, X=2 successively
- The member relation is defined as follows.
  - member(X, [X | Y]).
  - member(X, [Y | Z]) :- member(X, Z).

Exercise example(answer)

- To return X=1 only, we need to add the cut:
  - *?- member(Y,[[1,2],[3,4]]),member(X,Y),!.*
- To return X=1, X=2 successively, we need to add the cut:
  - *?- member(Y,[[1,2],[3,4]]),!,member(X,Y).*

Exercise example

- Write a Prolog predicate mymerge to merge two sorted lists using cut.
  - mymerge(L1, [], L1) :- !.
  - mymerge([], L2, L2).
  - mymerge([X1 | L1], [X2 | L2], [X1 | L]) :- X1 < X2, !, mymerge(L1, [X2 | L2], L).
  - mymerge([X1 | L1], [X2 | L2], [X2 | L]) :- mymerge([X1 | L1], L2, L).

Exercise example

- Suppose you have a prolog program containing only the following three sentences:
  - q(a).
  - p(X):-!,q(X).
  - p(b).
- Check the answer for p(X). How can you modify it so that it returns X = b, and X = a successively?

Exercise example(answer)

- Yes, modify it into:
  - q(a).
  - p(b).
  - p(X):-!,q(X).

Exercise

- Write a Prolog program to generate all prime numbers less than 100:
  - ?- prime100(X).
  - X = 2;
  - X = 3;
  - ….