
Supplementary slides on PROLOG

These slides cover Prolog's "cut".

References:

- Clocksin and Mellish, sections 4.2–4.4

©Diane Horton 2000

1

Example with Cut

```
% ----- allListsHave -----
% allListsHave ( List, Key ) returns true iff every element
% of List that is a list, contains Key.

% broken version:

broken( [First|Rest], Key ) :-
    is_list(First),
    member(Key, First),
    broken( Rest, Key ).

broken( [_|Rest], Key ) :-
    broken( Rest, Key ).

broken( [], _ ).

% Working version (with cut):

allListsHave( [First|Rest], Key ) :-
    is_list(First), !,      % A cut here fixes the prob.
    member(Key, First),
    allListsHave( Rest, Key ).

allListsHave( [_|Rest], Key ) :-
    allListsHave( Rest, Key ).

allListsHave( [], _ ).
```

2

Trace without cut

```
| ?- broken( [[a,b], p, [c,d,e]], a).
(0) Call: broken([[a,b],p,[c,d,e]],a) ?
(1) Call: is_list([a,b]) ?
(1) Exit: is_list([a,b]) ?
(2) Call: member(a,[a,b]) ?
(2) Exit: member(a,[a,b]) ?
(3) Call: broken([p,[c,d,e]],a) ?
(4) Call: is_list(p) ?
(4) Fail: is_list(p) ?
(5) Call: broken([[c,d,e]],a) ?
(6) Call: is_list([c,d,e]) ?
(6) Exit: is_list([c,d,e]) ?
(7) Call: member(a,[c,d,e]) ?
...

(7) Fail: member(a,[c,d,e]) ?
(6) Redo: is_list([c,d,e]) ?
(6) Fail: is_list([c,d,e]) ?
(11) Call: broken([],a) ?
(11) Exit: broken([],a) ?
(5) Exit: broken([[c,d,e]],a) ?
(3) Exit: broken([p,[c,d,e]],a) ?
(0) Exit: broken([[a,b],p,[c,d,e]],a) ?

yes
[trace]
```

3

Trace with cut

```
| ?- allListsHave( [[a,b], p, [c,d,e]], a).
(0) Call: allListsHave([[a,b],p,[c,d,e]],a) ?
(1) Call: is_list([a,b]) ?
(1) Exit: is_list([a,b]) ?
(2) Call: member(a,[a,b]) ?
(2) Exit: member(a,[a,b]) ?
(3) Call: allListsHave([p,[c,d,e]],a) ?
(4) Call: is_list(p) ?
(4) Fail: is_list(p) ?
(5) Call: allListsHave([[c,d,e]],a) ?
(6) Call: is_list([c,d,e]) ?
(6) Exit: is_list([c,d,e]) ?
(7) Call: member(a,[c,d,e]) ?
...

(8) Fail: member(a,[d,e]) ?
(7) Fail: member(a,[c,d,e]) ?
(5) Fail: allListsHave([[c,d,e]],a) ?
(3) Fail: allListsHave([p,[c,d,e]],a) ?
(2) Redo: member(a,[a,b]) ?
(11) Call: member(a,[b]) ?
(12) Call: member(a,[]) ?
(12) Fail: member(a,[]) ?
(11) Fail: member(a,[b]) ?
(2) Fail: member(a,[a,b]) ?
(0) Fail: allListsHave([[a,b],p,[c,d,e]],a) ?

no
[trace]
```

4

Another example

```
foo(X) :- a(X).  
foo(X) :- bar(X).
```

```
a(X) :- b(X), c(X), !, d(X), e(X).  
a(X) :- p(X).
```

```
b(1).  
b(2).  
c(3).  
b(4).  
c(4).  
d(4).  
b(5).  
c(5).  
d(5).  
e(5).
```

```
p(22).
```

```
bar(44).
```

5

```
(0) Redo: foo(44) ?  
(8) Redo: bar(44) ?  
(8) Fail: bar(_64) ?  
(0) Fail: foo(_47) ?
```

```
no
```

7

Trace with cut

```
| ?- foo(X).  
  (0) Call: foo(_47) ?  
  (1) Call: a(_64) ?  
  (2) Call: b(_69) ?  
  (2) Exit: b(1) ?  
  (3) Call: c(1) ?  
  (3) Fail: c(1) ?  
  (2) Redo: b(1) ?  
  (2) Exit: b(2) ?  
  (4) Call: c(2) ?  
  (4) Fail: c(2) ?  
  (2) Redo: b(2) ?  
  (2) Exit: b(4) ?  
  (5) Call: c(4) ?  
  (5) Exit: c(4) ?  
  (6) Call: d(4) ?  
  (6) Exit: d(4) ?  
  (7) Call: e(4) ?  
  (7) Fail: e(4) ?  
  (6) Redo: d(4) ?  
  (6) Fail: d(4) ?  
  (1) Fail: a(_64) ?  
  (8) Call: bar(_64) ?  
  (8) Exit: bar(44) ?  
  (0) Exit: foo(44) ?
```

```
X = 44;
```

```
continued ...
```

6

Without cut

```
| ?- foo(X).
```

```
X = 5;
```

```
X = 22;
```

```
X = 44;
```

```
no
```

8

Yet another example with cut

```
1) isa-mother(X) :- female(X),parent(X,_).
2) isa-father(X) :- male(X),parent(X,_).

3) isa(X) :- female(X),parent(X,_), !.

4) top(X) :- isa(X).

5) top2(X):- female(X), isa2(X).
6) isa2(X):- parent(X,_), !.

7) top3(X):- female(X), isa3(X).
8) isa3(X):- !, parent(X,_).

9) parent(fred,sue).
10) parent(janet,sue).
11) parent(fred,tim).
12) parent(janet,tim).
13) parent(diane,william).
14) parent(cathy,kit).

15) male(fred).
16) female(janet).
17) female(diane).
18) female(cathy).
```

```
| ?- isa-mother(X).
X = janet;
X = janet;
X = diane;
X = cathy;
no

| ?- isa(X).
X = janet;
no

| ?- top(X).
X = janet;
no

| ?- top2(X).
X = janet;
X = diane;
X = cathy;
no

| ?- top3(X).
X = janet;
X = janet;
X = diane;
X = cathy;
no
```