

## The 'fail' predicate

- It's a built in predicate without any argument.
- It always fails. It is an explicit way of forcing backtracking.

### Example:

1.        go :- test, write ("Will you get here?").  
          test :- fail.

goal: go

False

2.        go :- write ("You will get here"), test.  
          test :- fail.

goal: go

You will get here

False

3.        predicates  
          location (string, string)

go

clauses

go :- writef("%-10 %5\n", "CITY", "STATE"),  
      fail.

①

go :- location (City, State),  
writef("%-10 %5\n", City, State),  
fail.

go.

location ("Jackson", "MS").  
location ("Washington", "DC").  
location ("Raleigh", "NC").

Goal: go ←

CITY	STATE
Jackson	MS
Washington	DC
Raleigh	NC

Yes.

(1) If we remove fail ①, the o/p would be

CITY	STATE
Jackson	MS
Washington	DC
Raleigh	NC

Yes.

(2) With no fail in the program, the output is same as (1).

(3) With fail in 1st<sup>go</sup> clause and removing it from the 2nd, the o/p would be

CITY	STATE
Jackson	MS

Yes.

(No variable in the goal.  
So, once a goal succeeds,  
there is no backtracking)

predicates

location(string, string)

go(string, string)

clauses

$go(-, -) :- writef("%-10 %5s\n", "CITY", "STATE"), fail.$

$go(X, Y) :- location(X, Y).$

location("Jackson", "MS").

location("Washington", "DC").

location("Raleigh", "NC").

goal: go(Location, City).

CITY

STATE

Location = Jackson, City = MS

Location = Washington, City = DC

Location = Raleigh, City = NC

3 sol<sup>n</sup>

If we remove fail in the first go clause, then the answer is:

Location = —, City = —,

+ original 3 solutions

---

4 Solutions



predicates

location (string, string)

go

chkstate (string)

clauses

go :- writef("%-10 %5\n", "CITY",  
"STATE"), fail.

go :- location (City, State), chkstate (State),  
writef("%-10 %2\n", City, State), fail.

go.

chkstate ("DC") :- fail.

chkstate (-).

location ("Jackson", "MS").

location ("Washington", "DC").

location ("Raleigh", "NC").

Goal: go.

CITY	STATE
Jackson	MS
Washington	DC
Raleigh	NC
Yes	

To print only states MS & NC, (without DC),

## The Cut (!)

- Its symbol is '!'.
- Its a built in predicate without any argument.
- It always succeeds.
- It is used for 'preventing backtracking'.
- It is mainly used for :
  - i) Reducing the search space.
  - ii) Retaining values of variables once they are bound.

go :- premise 1, premise 2, !, premise 3, premise 4.

'Cut' acts as a fence. All possibilities in premise 3 & 4 could be tried out. Whether it succeeds or fails, no more go clause or premise 1 or premise 2 can be tried.

chkstate ("DC") :- !, fail.

chkstate (-).

'!' will not allow PROLOG to try alternate clause.

Thus, go clause will fail with City & State variable of location bound to "Washington" and "DC". So, PROLOG will try with alternate values of these variables.

(6)

{ we can also get the desired result  
like :

chkstate("MS").

chkstate("NC").

Thus, for chkstate("DC"), there is no  
matching and it fails.

We don't want to do that.

}

predicates

l(string, string)

go

cs(string)

clauses

go :- writef("%-10 %5\n", "CITY", "STATE"), fail.

go :- l(City, State), cs(State), writef("%-10 %5\n",  
City, State),  
fail.

go.

l("JK", "MS").

l("WS", "DC").

l("RL", "NC").

cs("DC") :- !.

cs(\_).



Goal: go

O/p

CITY	STATE
JK	MS
WS	DC
RL	NC

Yes.

{ ∴ cs("DC") would succeed with "!" as a condition. So, first two cities would be printed. Now  $\lambda(\text{City}, \text{State})$  would be tried for the 3rd possibility also.

}

Imp. cs(State) would not be tried out any more once "!" comes. But, once it succeeds or fails, other possibilities to the left of ~~ent~~ premise in which '!' is present could be tried.

→ If we remove "fail" from 2nd go clause, then the O/p would be:

CITY	STATE
JK	MS

Here chkstate clauses are:

chkstate(State) :- 1

→ 8 Let's change 2nd go clause & check state clauses: ⑧

go :- l ( city, state ), cs ( state ), writef ( " " )  
!

cs ( "DC" ) :- !, fail.

cs ( - ).

Output : CITY STATE

JK MS

Yes

( As no more "go" clauses can be tried ).

→ With 2nd go clause as :

go :- l ( city, state ), cs ( state ), writef ( " " )  
!, fail.

cs ( "DC" ) :- !, fail.

cs ( - ).

O/p

CITY

STATE

JK

MS

No

( ∵ No more go clauses can be tried ).



→ Let's check output for the following "go" clause:

go :- writet(" "), fail.

go :- l(City, State), !, cs(State),  
writet(" ").

go.

chkstate("DC") :- !, fail.

chkstate(\_).

Goal: go

CITY	STATE
JK	MS
Yes	

The O/p is so because '!' in the "go" clause is different from the cut in the chkstate clause. In all previous examples, cut was present in the chkstate clause, whereas it is present here in the go clause. So, no more possibility in the go clause can be tried.

→ If we remove our go clause is

go :- l(City, State), !, cs(State), writet(" ")  
fail.

O/P

10

CITY STATE  
JK MS  
NO

$a(0).$

Change ①  $a(x) :- \text{!}, b(x), \text{!}$

$b(1).$

$b(2).$

Change ②  $b(x) :- c(x), \boxed{\text{!}}$

$b(3).$

$c(4).$

$c(5).$

$c(x) :- \text{!}, d(x).$

$c(6).$

$d(7).$

Goal: 1)  $d(x)$

$x = 7$

1 solution.

4) Goal:  $a(x)$

$x = 0$

$x = 1$

$x = 2$

$x = 4$

4 solution.

2) Goal:  $c(x)$

$x = 4$

$x = 5$

$x = 7$

3 sol<sup>n</sup>

① If we change  $a(x)$  as:   
  $\nearrow$  No change in 2

$a(x) :- b(x), \text{!}$

Then o/p for goal  $a(x)$  is:

$x = 0$

$x = 1$

2 sol<sup>n</sup>

3) Goal:  $b(x).$

$x = 1$

$x = 2$

$x = 4$

3 sol<sup>n</sup>

② If ~~o/p~~  $b(x)$  is changed as   
  $\nearrow$  No change in (1)

$b(x) :- c(x)$  (i.e. cut is removed)

O/p is for the goal  $a(x)$  is:



$a(0)$

$a(x) :- !, b(x).$

$b(1).$

$b(2).$

$b(x) :- c(x), !, \text{fail.} \quad /* \text{left has changed} */$

$b(3).$

$c(4).$

$c(5).$

$c(x) :- !, d(x).$

$c(6).$

goal:  $b(x)$

$x = 1$

$x = 2$

2 sol<sup>n</sup>s

[ "!" in the  $b(x)$  clause does not allow  
Other  $b(x)$  clause to be tried out ]

→ New  $b(x)$  clause is:

$b(x) :- !, c(x), \text{fail.}$

Then o/p is same for the same reason.

$a(0).$

$a(x) :- !, b(x).$

$b(1).$

$b(2).$

$b(x) :- c(x) \quad /* \text{ This has changed } */$

$b(3).$

$c(4).$

$c(5).$

$c(x) :- !, d(x), \text{fail.}$

$c(6).$

$d(7).$

goal:  $b(x).$

$x = 1$

$x = 2$

$x = 4$

$x = 5$

$x = 3$ , 5th soln

5 solution

If we change the  $c(x)$  clause as (other things unchanged)

$c(x) :- \text{fail}, !, d(x).$

Then the o/p is for the goal  $b(x)$ :

$x = 1$

$x = 2$

$x = 4$

$x = 5$

$x = 6$

$x = 3$