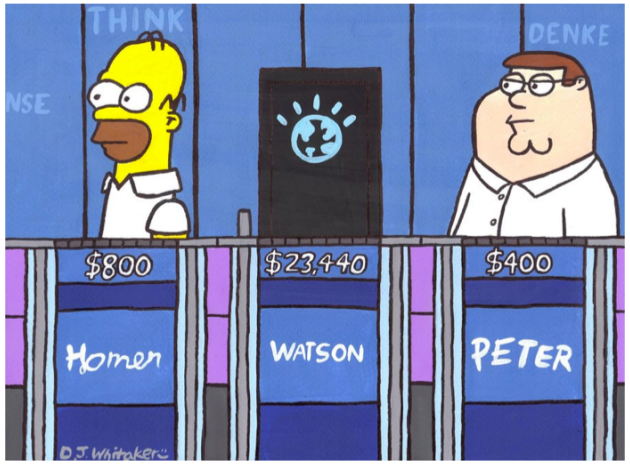
Prolog and declarative programming, pt 2



Prolog examples - member

```
member(X,[X|R]).
member(X,[Y|R]) :- member(X,R).
```

Prolog examples - member

```
?- member(2,[1,2,3]).
Yes

?- member(X,[1,2,3]).
X = 1 ;
X = 2 ;
X = 3 ;
No
```

```
?- member([3,Y], [[1,a],[2,m],[3,z],[4,v],[3,p]]).
Y = z ;
Y = p ;
No

?- member(X,[23,45,67,12,222,19,9,6]),
   Y is X*X,
   Y < 100.
X = 9    Y = 81 ;
X = 6    Y = 36 ;
No
```

Prolog examples - takeout

```
takeout(X,[X|R],R).
takeout(X,[F|R],[F|S]) :- takeout(X,R,S).
```

Prolog examples - takeout

```
?- takeout(X,[1,2,3],L).
X=1  L=[2,3] ;
X=2  L=[1,3] ;
X=3  L=[1,2] ;
No
```

```
?- takeout(3,W,[a,b,c]).
W = [3,a,b,c] ;
W = [a,3,b,c] ;
W = [a,b,3,c] ;
W = [a,b,c,3] ;
No
```

Prolog examples - reverse

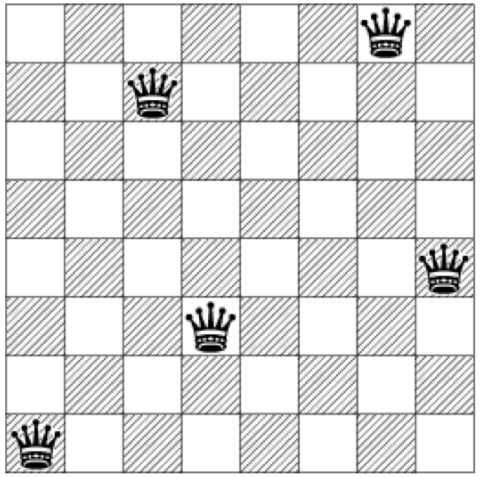
```
reverse([X|Y],Z,W) :- reverse(Y,[X|Z],W).
reverse([],X,X).
```

Prolog examples

```
perm([X|Y],Z) :- perm(Y,W), takeout(X,Z,W).
perm([],[]).
```

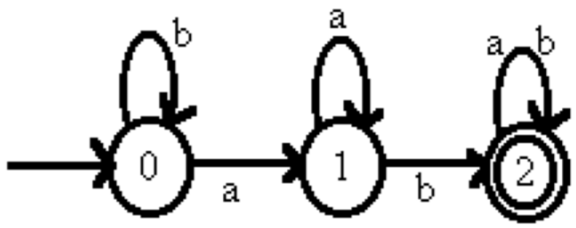
Chess queens

```
solve(P) :-
      perm([1,2,3,4,5,6,7,8],P),
      combine([1,2,3,4,5,6,7,8],P,S,D),
      all_diff(S),
      all_diff(D).

combine([X1|X],[Y1|Y],[S1|S],[D1|D]) :-
      S1 is X1 +Y1,
      D1 is X1 - Y1,
      combine(X,Y,S,D).
combine([],[],[],[]).

all_diff([X|Y]) :-  \+member(X,Y),
all_diff(Y).
all_diff([X]).
```



DFA checker

Accepts the language b*aa*b(a,b)*




```
parse(L) :- start(S),
            trans(S,L).

trans(X,[A|B]) :-
      delta(X,A,Y),    /*  X ---A---> Y */
      write(X),
      write('  '),
      write([A|B]),
      nl,
      trans(Y,B).

trans(X,[]) :-
      final(X),
      write(X),
      write('  '),
      write([]), nl.

delta(0,a,1).
delta(0,b,0).
delta(1,a,1).
delta(1,b,2).
delta(2,a,2).
delta(2,b,2).

start(0).

final(2).
```