Eric Smrkovsky
CSci 117
Lab 11
11/4/2019

1) Loop one:
Q1::1
Q2::3#8
Q3::2,4#8
Q4::2#3,5#8
Q5::2#4,6#8
Q6::2#5,7#8
Q7::2#6,8
Q8::2#7

Loop two:
Q1::1
Q2::5
Q3::2,7#8
Q4::2,6,8
Q5::3#4,6#7
Q6::2#4,7#8
Q7::2#4,6,8
Q8::2#4,6#7

The propagator **Q0 #\= Q** prevents any value for Q from being in the same column that **Q0** is in.
The propagator **abs(Q0 - Q) #\= D0** prevents any queen from being in the diagonal row. Example: if the queen is in column 2, row 5 of the previous then the next queen can't be in row 3 column 6 because abs(1-6) is equal to 5. It could be in row 3 column 7 because abs(1-7) does **not** equal 5.
The propagator **D1 #= D0 + 1** simply prevents any queen from being in the column that is represented by **D0**.

2)

```
sudoku(Rows) :-
        length(Rows, 9),                              #1
        maplist(same_length(Rows), Rows),             #2
        append(Rows, Vs),                             #3
        Vs ins 1..9,                                  #4
        maplist(all_distinct, Rows),                  #5
        transpose(Rows, Columns),                     #6
        maplist(all_distinct, Columns),               #7
        Rows = [A,B,C,D,E,F,G,H,I],                   #8
        blocks(A, B, C),                              #9
        blocks(D, E, F),                              #10
        blocks(G, H, I).                              #11

blocks([], [], []).                                   #12
blocks([A,B,C|Bs1], [D,E,F|Bs2], [G,H,I|Bs3]) :-      #13
        all_distinct([A,B,C,D,E,F,G,H,I]),            #14
        blocks(Bs1, Bs2, Bs3).                        #15
```

#1 The length of the Rows will be 9 (for now unbound variables).
#2 Each of the unbound variables from #1 will be a list of size 9 (for now unbound variables) like a 9 by 9 array.
#3 Takes each list and appends them together into one lone list of size 81 called Vs.
#4 Each variable in the list Vs will be a number from 1 to 9.
#5 Each of the sub-lists of Rows will have distinct variables within each sub-list.
#6 Create a new list Cols that is the same structure as Rows.
#7 Each of the sub lists of Rows will have distinct variables within each sub-list.
#8 Assign variable names for each of the 9 sub-lists Of Rows.
#9 Call blocks function with first three sub-lists of Rows.
#10 Call blocks function with next three sub-lists of Rows.
#11 Call blocks function with last three sub-lists of Rows.
#12 Base case, program has reached an end.
#13 Breaks up each sub-list of Rows that was passed into the function into it's own list with the first three elements given variables and a tail for each broken up list.
#14 Each of the first three elements must be distinct within their own list.
#15 Recursive call with the tails until base case is reached.

Eric Smrkovsky
CSci 117
Lab 11
11/4/2019

3A) A says: "Two of us are a knaves." B says: "A is a knave or C is a knight."

```
example_knights(6, [A,B,C]) :-
        sat(A=:=card([2],[~A,~B,C])),
        sat(B=:=(~A + C)).
```

Result: A = C, C = 0, B = 1

3B) A says C is a knave and B is a knight. B agrees with A. C says A and B are both Knaves.

```
example_knights(7,[A,B,C]) :-
        sat(A=:= (~C * B)),
        sat(B=:= (~C * B)),
        sat(C=:= (~A * ~B)).
```

Output: **A** = B, *sat*(B=\=C)

3C) A says: "A or C are not the same as B or D, or E is a knight."

D says: "I agree with A or I know that E is the same as me."

E says: "A is a Knight and B is a Knave."

B says: "C can't be the same as D or E."

C is a knight.

```
example_knights(7, [A,B,C,D,E]) :-
    sat(A=:=((A + C)=\=(B + D) + E)),
    sat(D=:=((A + C)=\=(B + D) + (E=:=D))),
    sat(E=:=(A * ~B)),
    sat(B=:=(C=\=(D + E))),
    sat(C=:=1).
```

Result: A = D, D = E, E = 0, B = C, C = 1

Eric Smrkovsky
CSci 117
Lab 11
11/4/2019

4)

```
    :- use_module(library(clpfd)).
    crypt1([H1|L1],[H2|L2],[H3|L3],L4,R1,R2,R3) :-

% Give constraints on variable values in L4
        L4 ins 0..9,
% Heads cannot have value 0
        H1 #\= 0,
        H2 #\= 0,
        H3 #\= 0,
% Variable values are distinct in L4
        all_distinct(L4),
% Reverse the 3 input words
        reverse([H1|L1],R1),
        reverse([H2|L2],R2),
        reverse([H3|L3],R3),
% Call to helper function that does the sum with
%reversed words one iteration at a time
        helpr(R1,R2,R3,0,1).
        helpr([],[],[X],C,_) :- X #= C.
        helpr([RH1|RT1],[RH2|RT2],[RH3|RT3],C,E) :-
            C2 #= div(RH1+RH2+C,10),
            Carry #= 10*C2,
            Digit #= mod(RH1+RH2+C,10),
            E*(RH1+RH2+C) #= E*(Digit+Carry),
            Digit #= RH3,
            E2 #= E*10,
            helpr(RT1,RT2,RT3,C2,E2).
```

Eric Smrkovsky
CSci 117
Lab 11
11/4/2019

4b) Backwards lists are the R1, R2, R3 variables for easy checking.

[H,O,M,E],[R,O,M,E],[P,O,E,M,S]

R1 = [4, 0, 2, 9],
R2 = [4, 0, 2, 3],
R3 = [8, 0, 4, 2, 1],

E = 4,
H = 9,
M = 0,
O = 2,
P = 1,
R = 3,
S = 8

[M,A,L,E],[P,L,A,N],[L,O,N,E,R]

R1 = [7, 1, 5, 2],
R2 = [6, 5, 1, 8],
R3 = [3, 7, 6, 0, 1]

A = 5,
E = 7,
L = 1,
M = 2,
N = 6,
O = 0,
P = 8,
R = 3,

[C,R,O,S,S],[R,O,A,D,S],[D,A,N,G,E,R]

R1 = [3, 3, 2, 6, 9],
R2 = [3, 1, 5, 2, 6],
R3 = [6, 4, 7, 8, 5, 1],

A = 5,
C = 9,
D = 1,
E = 4,
G = 7,
N = 8,
O = 2,
R = 6,
S = 3

Eric Smrkovsky
CSci 117
Lab 11
11/4/2019

Extra Credit)

```
% render solutions nicely.
:- use_rendering(chess).

queens(N,Qs) :- makeList(N,Qs), placeQueens(N,Qs,_,_).

% could have used length(Qs,N) for makeList
makeList(0,[]).
makeList(N,[_|T]) :- Ndec is N-1, makeList(Ndec,T).

placeQueens(0,_,_,_) :- !.
placeQueens(N,Cs,Us,[_|Ds]) :- N>0, Ndec is N-1,
                               placeQueens(Ndec,Cs,[_|Us],Ds),
                               placeQueen(N,Cs,Us,Ds).

placeQueen(N,[N|_],[N|_],[N|_]).
placeQueen(N,[_|Cs2],[_|Us2],[_|Ds2]) :- placeQueen(N,Cs2,Us2,Ds2).
```