TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING
PULCHOWK CAMPUS

A
LAB REPORT
ON

PROLOG - II: CONSTRAINT PROGRAMMING

SUBMITTED BY:

Diwas Adhikari **(076BEI014)**

SUBMITTED TO:

Department of Electronics & Computer Engineering
Pulchowk Campus, IOE

LAB NUMBER

2

SUBMISSION DATE

July 12, 2023

(For course completion of BEI – Artificial Intelligence)

OBJECTIVES

- To study about constraint programming and constraint satisfaction problems.
- To solve puzzles and games associated to constraint satisfaction and devise solution for it using PROLOG.

THEORY

Constraint programming is a paradigm for solving combinatorial problems that draws on a wide range of techniques from artificial intelligence, computer science and operations research. It is a useful tool in formulating and solving problems that can be defined in terms of constraint among a set of variables. In constraint programming , users declaratively state the constraints on the feasible solutions for a set of decision variables. Real world problems are defined in terms of some variables that bear some constraints. Finding a set of variables that are within the constraints given, is a solution to that problem.

Let us consider a problem, that can be represented by some relations of the variables x, y and z. We have a domain Dx, Dy, Dz from where the variables can take a value. The constraint is given by a set C and may have a number of constraints C1, C2, C3, etc. where each of them relates to some or all of the variables x, y, z. Now, a solution to the problem is a set from the problem domain $dx \in Dx$, $dy \in Dy$ and $dz \in Dz$ for which all the constraints of set C are satisfied.

Crypto-arithmetic Problem

Crypto-arithmetic problem is a constraint satisfaction problem in which the game revolves around digits and their unique replacement with alphabets or other symbols. The constraints which this problem follows during the conversion are as follows.

1. A number 0-9 is assigned to a particular alphabet.
2. Each different alphabet has a unique number.
3. All the same alphabets have the same numbers.
4. The numbers should satisfy arithmetic operations similar to any normal number.

For example: <u>SEND+MORE=MONEY</u>
We have to assign values to the individual alphabets in such a way the arithmetic rules are adhered. A trivial solution will be to assign all alphabets with zeros but, we have a constraint that no two alphabets should be assigned with the same number.

C4	C3	C2	C1	
	S	E	N	D
+	M	O	R	E
M	O	N	E	Y

We must define constraints to check that no two alphabets should be equal i.e. assigned to the same number. The domain for the alphabets along with numeric constaints for the sum and carry-over is given by:

$$S, E, N, D, M, O, R, Y \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}.$$
$$The\ constraints\ are:$$
$$D + E = Y + 10C1$$
$$N + R + C_1 = E + 10C_2$$
$$E + O + C_2 = N + 10C_3$$
$$S + M + C_3 = 0 + 10C_4$$
$$M = C_4$$
$$C_1, C_2, C_3, C_4 \in \{0, 1\}$$

Eight Queens Problem

The eight queens problem is also a constraint satisfaction problem. The goal is to place eight queens in any of the 64 available squares of the chess board without them attacking each other. As a result, the problem may be phrased using the variables x1 , x2, x3 , x4 , x5, x6, x7, x8 and y1 , y2 , y3, y4 , y5, y6 , y7 , y8 where the x1 represent the rows and y1 represent the columns. Assigning values for x and y such that the requirement is satisfied is a solution to this problem.

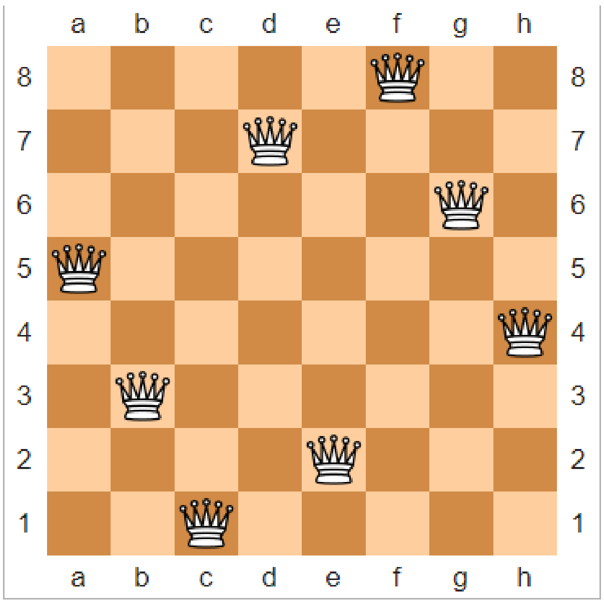
The problem can be formulated as P = { (x1, y1) , ..., (x8 , y8) }.

So it can be clearly seen that the domains for Xi and Yi are Dx = { 1, 2, 3, 4, 5, 6, 7, 8 } and Dy = { 1, 2, 3, 4, 5, 6, 7, 8 } respectively.

The constraints are :

1. No two queens should be in the same row. [yi ≠ yj for i = 1 to 8; j = 1 to 8; i ≠ j]
2. No two queens should be in the same columns. [xi ≠ xj for i = 1 to 8; j = 1 to 8; i ≠ j]
3. There should not be two queens placed on same diagonal. [(yi –yj) ≠ ± (xi –xj)]

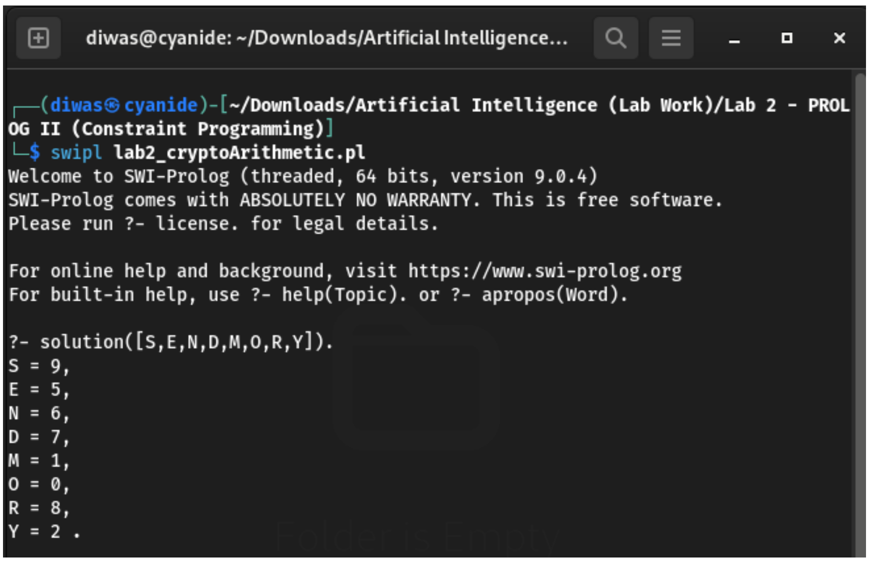
A solution to this problem is an instance of P where in the above mentioned constraints are satisfied which is observable in the figure below:



PROLOG CODE & OBSERVATION

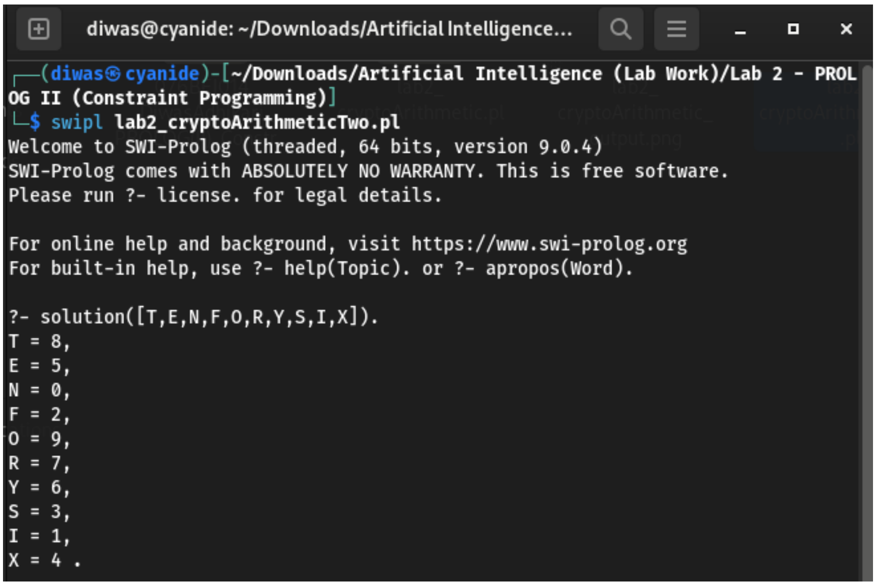
1) Crypto-arithmetic Problem [<u>SEND + MORE = MONEY</u>]

```
solution([]).
solution([S,E,N,D,M,O,R,Y]):-
C4 is 1, % C4 must be 1 for ‘M’ to be represented.
member(C1,[0,1]),
member(C2,[0,1]),
member(C3,[0,1]),
% C1, C2, C3 will have values 0 or 1.
member(E,[0,1,2,3,4,5,6,7,8,9]),
member(N,[0,1,2,3,4,5,6,7,8,9]),
member(D,[0,1,2,3,4,5,6,7,8,9]),
member(M,[0,1,2,3,4,5,6,7,8,9]),
member(O,[0,1,2,3,4,5,6,7,8,9]),
member(R,[0,1,2,3,4,5,6,7,8,9]),
member(Y,[0,1,2,3,4,5,6,7,8,9]),
member(S,[0,1,2,3,4,5,6,7,8,9]),
% S, E, N, D, M, O, R, Y will have values between 0 and 9.
% The values of S, E, N, D, M, O, R, Y must not be equal.
S=\=E, S=\=N, S=\=D, S=\=M, S=\=O, S=\=R, S=\=Y,
E=\=N, E=\=D, E=\=M, E=\=O, E=\=R, E=\=Y,
N=\=D, N=\=M, N=\=O, N=\=R, N=\=Y,
D=\=M, D=\=O, D=\=R, D=\=Y,
M=\=O, M=\=R, M=\=Y,
O=\=R, O=\=Y,
R=\=Y,
% Constraints for solution
D+E =:= Y+10*C1,
N+R+C1 =:= E+10*C2,
E+O+C2 =:= N+10*C3,
S+M+C3 =:= O+10*C4,
M is C4.
member(X, [X|_]).
member(X, [_|Z]):-
member(X,Z).
```



2) Crypto-arithmetic Problem [TEN + TEN + FORTY = SIXTY]

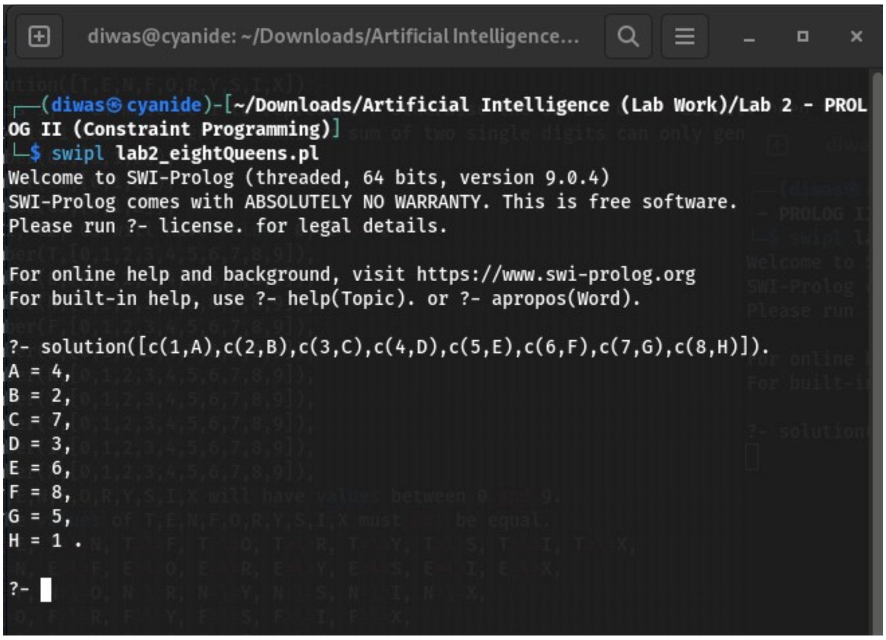
```
solution([]).
solution([T,E,N,F,O,R,Y,S,I,X]):-
% These following values need to be as follows due to no other possibility.
N is 0, E is 5, C4 is 1,
% C4 is 1 because even the maximum sum of two single-digits can only generate
a carry of 1 & C4 must be 1 so that 'F' and 'S' remains unequal..
member(F,[1,2,3,4,6,7,8,9]),
member(O,[1,2,3,4,6,7,8,9]),
member(R,[1,2,3,4,6,7,8,9]),
member(T,[1,2,3,4,6,7,8,9]),
member(Y,[1,2,3,4,6,7,8,9]),
member(S,[1,2,3,4,6,7,8,9]),
member(I,[1,2,3,4,6,7,8,9]),
member(X,[1,2,3,4,6,7,8,9]),
% C1, C2, C3 will have values 0, 1 or 2 [Maximum sum of 3-digits can generate a
carry upto 2.]
member(C1,[0,1,2]), member(C2,[0,1,2]), member(C3,[0,1,2]),
% T,E,N,F,O,R,Y,S,I,X will have values between 0 and 9.
% The values of T,E,N,F,O,R,Y,S,I,X must not be equal.
T=\=E, T=\=N, T=\=F, T=\=O, T=\=R, T=\=Y, T=\=S, T=\=I, T=\=X,
E=\=N, E=\=F, E=\=O, E=\=R, E=\=Y, E=\=S, E=\=I, E=\=X,
N=\=F, N=\=O, N=\=R, N=\=Y, N=\=S, N=\=I, N=\=X,
F=\=O, F=\=R, F=\=Y, F=\=S, F=\=I, F=\=X,
O=\=R, O=\=Y, O=\=S, O=\=I, O=\=X,
R=\=Y, R=\=S, R=\=I, R=\=X,
Y=\=S, Y=\=I, Y=\=X,
S=\=I, S=\=X,
I=\=X,
% Constraints for solution
2*N+Y =:= Y+10*C1,
2*E+T+C1 =:= T+10*C2,
2*T+R+C2 =:= X+10*C3,
O+C3 =:= I+10*C4,
F+C4 =:= S.
member(X, [X|_]).
member(X, [_|Z]):-
member(X,Z).
```



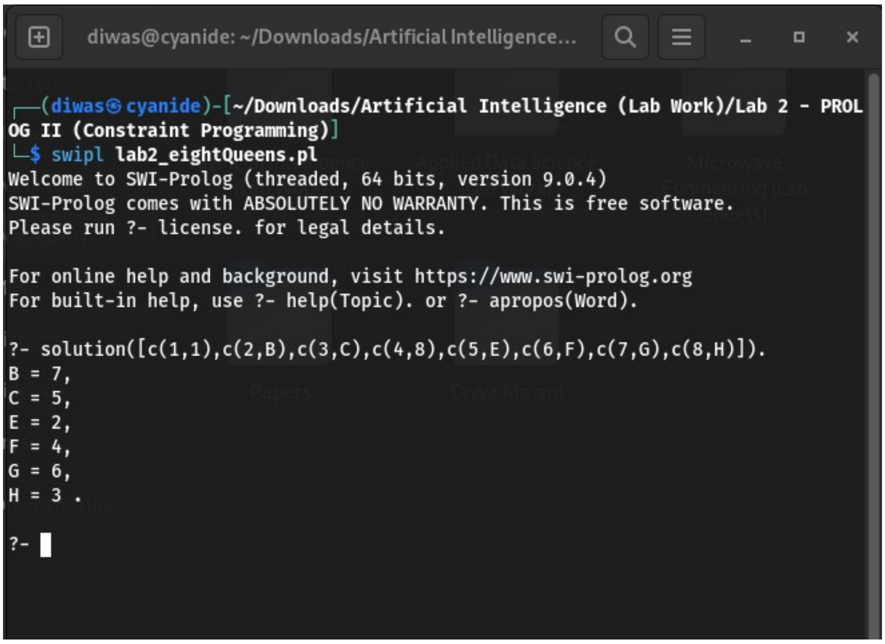
```
diwas@cyanide: ~/Downloads/Artificial Intelligence...
┌──(diwas㉿cyanide)-[~/Downloads/Artificial Intelligence (Lab Work)/Lab 2 - PROL
OG II (Constraint Programming)]
└─$ swipl lab2_cryptoArithmeticTwo.pl
Welcome to SWI-Prolog (threaded, 64 bits, version 9.0.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- solution([T,E,N,F,O,R,Y,S,I,X]).
T = 8,
E = 5,
N = 0,
F = 2,
O = 9,
R = 7,
Y = 6,
S = 3,
I = 1,
X = 4 .
```

3) Eight Queens Problem

```
solution([]).
solution([c(X,Y)|Others]):-
solution(Others),
member(Y,[1,2,3,4,5,6,7,8]),
noattack(c(X,Y),Others).
noattack(_,[]).
noattack(c(X,Y),[c(X1,Y1)|Others]):-
Y =\= Y1,
Y1-Y =\= X1-X,
Y1-Y =\= X-X1,
noattack(c(X,Y),Others).
member(X,[X|_]).
member(X,[_|Z]):-
member(X,Z).
```



<u>Queens Placed Explicitly</u>



DISCUSSION & CONCLUSION

Hence, PROLOG is well-suited for solving constraint satisfaction problems. Constraint programming problems are problems that involve finding a solution that satisfies a set of constraints. PROLOG's declarative nature makes it easy to express constraints, and its backtracking search algorithm makes it efficient at finding solutions. If we ease the constraints by supplying some initial conditions or possibilities associated to the solution, the problems can be solved much faster.

SUBMITTED BY:

Diwas Adhikari
076BEI014 – (Roll Number: 14)
Bachelor in Electronics & Communication Engineering (BEI)
076bei014.diwas@pcampus.edu.np
lunaticoda123@gmail.com