PA 17 Ketaki Patil

# AI Assignment no. 3

* **Title**: Implementation of solution of constraint satisfaction problem like SEND + MORE = MONEY or CROSS + ROADS = DANGER

* **Aim**: Solve constraint satisfaction problem like SEND+MORE = MONEY or CROSS + ROADS = DANGER

* **Objective**: To study constraint satisfaction method and solve constraint satisfaction problem such as SEND+MORE = MONEY or CROSS + ROADS = DANGER.

* **Theory**:

- **Constraint satisfaction Method**:

A CSP is a special kind of problem that satisfies some additional structural properties beyond the basic requirements for problems in general. In a CSP, the states are defined by the values of a set of variables and the goal test specifies a set of constraints that the values have to obey.

The state components are:

a) Variables
b) Domain
c) Constraints between variables

The goal is to find a state which satisfies the constraints.

eg: map colosing, N queens etc.

- **BackTracking Search**

It is an algorithmic for solving problems recursively by

trying to build a solution incrementally, one piece at a time, removing those solutions that fail to satisfy the constraints of the problem at any point of time. It is used to find all the possible combination to solve an optimization problem.

- Constraint propagation

Constraint propagation is the general for propagating the complications of a constraint on one variable onto the other variable. The idea of 'arc consistency provides a fast method of constraint propagation that is substantially stronger than forward checking.

★ Input: Initial values for some letters in given problem.
★ Output: Unique values for letters S,E,N,D,M,O,R,Y or C,R,O,S,A,D,N,G,E

★ Algorithm:

① Set each variable as undefined. Empty stack. All variables are future variables.
② Select a future variable as current variable. If it exists delete from FUTURE and stack it. If not, assignment is a solution.
③ Select an unused value for the current variable. If it exists mask the value as used.
If not set current variable as undefined, mask all its values as unused, unstack the variable and add it to FUTURE, If stack is empty there is no solution, if not

go to step 3.

④ Test constraints between past variables and the current one. If they are satisfied, go to step ② if not go to ③.

* Platform : Linux / Windows
* FAQs :

1) What are other constraint satisfaction problems?

→ (i) CSP is a special kind of problem that satisfies some additional structural problems beyond the basic requirement for problem in general.
(ii) State components are variables and domains.
(iii) Goal is to find state that satisfies the constraints.
(iv) Cryptarithmetic puzzles are type of CSP where each digit is replaced for an alphabet. They should be unique and digits from (0 to 9) are used.
(v) Map coloring is another example of CSP.
(vi) N Queens problem, crossword puzzles, sudoku, resource assignment / distribution are other constraint satisfaction problem.

2) What do you mean by constraint propagation?

→ Constraint propagation is a general term for propagating the implications of a constraint of one variable onto other variables.

3) Why backtracking search can be used to solve constraint satisfaction problem?

→ We can build up to solution by searching through the space of partial assignments. The order in which we assign the variables does not matter eventually they all have to be assigned. We can decide on a suitable value for one variable at a time. This is the key idea of backtracking search. If during the process of building, we can immediately reject, all possible ways of extending the current partial assignment.

# AI ASSIGNMENT 3 : CONSTRAINT SATISFACTION PROBLEM

**PA 17 KETAKI PATIL**

## CODE :

```
'''
ASSIGNMENT 3 : Constraint satisfaction problem
PA 17 KETAKI PATIL
BATCH A1
'''
import itertools
import string


def correct_vals(p, puzzle):
    op1, op, op2, e, r = break_puzzle(puzzle.translate(p))

    return eval(op1 + op + op2 + "==" + r)

def break_puzzle(puzzle):
    return tuple(puzzle.upper().split())

def get_unique_letters(puzzle):
    return [i for i in set(''.join(break_puzzle(puzzle))) if i.isalpha()]

def get_starting_letters(puzzle, letters):
    return [i for i in range(len(letters)) if letters[i] == break_puzzle(puzzle)[0][0] or
letters[i] == break_puzzle(puzzle)[2][0] or letters[i] == break_puzzle(puzzle)[4][0]]

def get_valid_permutations(puzzle):
    digits = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']
    letters = get_unique_letters(puzzle)
    critical_indices = get_starting_letters(puzzle, letters)
    poss_perms = []
    for perm in itertools.permutations(digits, len(letters)):
        flag = 0
        for i in critical_indices:
            if perm[i] == '0':
                flag = 1
                break
        if flag == 0:
            poss_perms.append(perm)
    return poss_perms

def solve(puzzle):
    letters = get_unique_letters(puzzle)
    if len(letters) > 10:
        print("INVALID EQUATION : more than one letter maps to same digit")
        return
    for poss in get_valid_permutations(puzzle):
        p = str.maketrans(''.join(letters), ''.join(poss))
        if correct_vals(p,puzzle):
```

# AI ASSIGNMENT 3 : CONSTRAINT SATISFACTION PROBLEM

**PA 17 KETAKI PATIL**

```
            answer = dict(zip(letters, poss))
            #print(answer)
            solved_puzzle = puzzle
            for c in answer:
                x = solved_puzzle.replace(c, answer[c])
                solved_puzzle = x
            print(solved_puzzle)

n=1
while(n==1):
    puzzle=input("Enter the equation : ")
    print(puzzle)
    solve(puzzle)
    print("--------------------------------")
    n=int(input("Do you wish to continue : yes=1\n"))
```

# OUTPUT :

```
PS D:\SEM_9\AI>  d:; cd 'd:\SEM_9\AI'; & 'C:\Users\ketak\AppData\Local\Microsoft\WindowsApps\PythonSoftwareFoundation.Python.3.7_
scode\extensions\ms-python.python-2021.5.842923320\pythonFiles\lib\python\debugpy\launcher' '52461' '--' 'd:\SEM_9\AI\lab3.py'
Enter the equation : BASE + BALL = GAMES
BASE + BALL = GAMES
7483 + 7455 = 14938
--------------------------------
Do you wish to continue : yes=1
1
Enter the equation : SEND + MORE = MONEY
SEND + MORE = MONEY
9567 + 1085 = 10652
--------------------------------
Do you wish to continue : yes=1
1
Enter the equation : CROSS + ROADS = DANGER
CROSS + ROADS = DANGER
96233 + 62513 = 158746
--------------------------------
Do you wish to continue : yes=1
1
Enter the equation : RED + BLUE = COLOR
RED + BLUE = COLOR
--------------------------------
Do you wish to continue : yes=1
1
Enter the equation : ABC + XYZ = LETTERS
ABC + XYZ = LETTERS
--------------------------------
Do you wish to continue : yes=1
0
```