Lab7

October 20, 2021

106119029
Lab 6 AI/ML

Write a Program to Implement the following Scenario using 3 SAT algorithm. Alice recently started to work for a hardware design company and as a part of her job, she needs to identify defects in fabricated integrated circuits. An approach for identifying these defects boils down to solving a satisfiability instance. She needs your help to write a program to do this task. Input The first line of input contains a single integer, not more than 5, indicating the number of test cases to follow. The first line of each test case contains two integers n and m where 1 n 20 indicates the number of variables and 1 m 100 indicates the number of clauses. Then, m lines follow corresponding to each clause. Each clause is a disjunction of literals in the form Xi or ~Xi for some 1 i n, where ~Xi indicates the negation of the literal Xi. The "or" operator is denoted by a 'v' character and is separated from literals with a single space. Output For each test case, display satisfiable on a single line if there is a satisfiable assignment; otherwise display unsatisfiable. Sample Input 2

33
X1 v X2
~X1
~X2 v X3
35
X1 v X2 v X3
X1 v ~X2
X2 v ~X3
X3 v ~X1
~X1 v ~X2 v ~X3
Sample Output
satisfiable
unsatisfiable
Google Colab Link

[ ]:
```
def get_reverse(literal: str):
    if literal[0] == '~':
        return literal[1:]
    else:
        return '~' + literal
```

[ ]:
```
def is_satisfiable(n_vars, clauses):
    candidates = {frozenset()}
    for clause in clauses:
```

```
        temp = set()
        for s in candidates:
            for literal in clause:
                if get_reverse(literal) not in s:
                    temp.add(s | {literal})
        candidates = temp
        if len(candidates) == 0:
            return False
    return True
```

[ ]:
```
def load_case():
    n_vars, n_clauses = input().split()
    clauses = [input().strip().split(' v ')
               for _ in range(int(n_clauses))]
    return int(n_vars), clauses
```

[ ]:
```
def main():
    num_cases = int(input())
    outputs = []
    for _ in range(num_cases):
        n_vars, clauses = load_case()
        result = is_satisfiable(n_vars, clauses)
        if result:
            outputs.append("satisfiable")
        else:
            outputs.append("unsatisfiable")
    for out in outputs:
        print(out)
```

[ ]:
```
main()
```

```
2
3 3
X1 v X2
~X1
~X2 v X3
3 5
X1 v X2 v X3
X1 v ~X2
X2 v ~X3
X3 v ~X1
~X1 v ~X2 v ~X3
satisfiable
unsatisfiable
```

[17]:

[ ]: