Project 2

Automated Reasoning

1. Basic Model Checking

Model Checking is a simple enumeration method of checking for entailment. In this KB the knowledge base entails α, the query. Model Checking is a method of inference by enumeration. The algorithm takes input clauses of Knowledge Base(KB) followed by query.

Input Format: Clause1, Clause2, Clause3, ... |= Query

The algorithm assigns all possible truth – table (model) assignments and checks whether the Query can be inferred.

Output: # When the Query can be inferred

Query can be inferred
Model 1
Model 2
…….
When the Query cannot be inferred
Query cannot be inferred

2. Advanced Propositional Inference

The WalkSAT algorithm for SAT (Satisfiability) checking is implemented as Part 2. WalkSAT is a local search algorithm to solve Boolean satisfiability problems. The input to the algorithm are Boolean clauses that are in conjunctive normal form. The clauses used in this project have been converted to CNF before giving them as input. The algorithm starts by assigning a random value to each variable in the formula. Then checks for satisfiability, if the assignment satisfies all clauses, the algorithm terminates, returning the model (assignment). Otherwise, a variable is flipped. The algorithm keeps on performing these operations until a satisfiable assignment is achieved, or the maximum number of flips are achieved.

The algorithm takes the inputs of clauses & query and requires the maximum number of allowable flips, and the flipping probability. Variable is flipped in the following way, first the WalkSAT algorithm picks an unsatisfied clause, then a variable in that clause is flipped. The clause is picked at random among unsatisfied clauses. The variable is picked that will result in the fewest previously satisfied clauses becoming unsatisfied, with some probability of picking one of the variables at random.

The algorithm may restart with a new random assignment if no solution has been found for too long, as a way of getting out of local minima of numbers of unsatisfied clauses.

In this project the maximum allowable flips are set to 10000 and the probability of random flip is set to 0.5.

Input Format: Clause1, Clause2, Clause3, ... |= Query

The algorithm returns a model if input is satisfiable else returns that Failure

Output: # When the Query is satisfiable
    Satisfiable
    Model

When the Query cannot be inferred
    Failure

When the algorithm returns a "failure" it is either that the sentence is unsatisfiable or that the algorithm could not find a model for the given sentence with the set maximum number of flips.

3. Representation

In this project the propositional logic sentences are taken in the following format:
Clause1 , Clause2 , Clause3 , ... |= Query

Each variable, symbol are separated by a space

^ - Logical AND
V – Logical OR
-> - Logical Implication
<> - Logical Biconditional
|= is used to separate KB and Query

4.Problems

Both the algorithms take the input propositional sentence and display the output. Once an input is tested the programs ask if there are any more sentences to test, to which “y” has to be pressed to test more sentences and “n” to terminate.

Given below are the solutions obtained from running the programs. The screenshots of both the Enumeration Method and WalkSAT program are also shown.

Modus Ponens:

Q can be inferred

Enumeration Method:

```
Enter the Proposition Sentence to be tested      P , P -> Q |= Q
 Q  can be inferred
Model 1 : {'Q': True, 'P': True}
Time taken =  0.0004718019999998546 s
Enter y to check another sentance or n to exit   y
Enter the Proposition Sentence to be tested      P , P -> Q |= ! Q
 ! Q cannot be inferred
Time taken =  0.0003413330000001498 s
```

WalkSAT:

```
Enter the Proposition Sentence to be tested      P , ! P v Q |= ! Q
Unsatisfiable
Time taken =  1.741035058 s
 Do  you want to check any more sentences (Y/N)         y
Enter the Proposition Sentence to be tested      P , ! P v Q |= Q
Satisfiable
Model =  {'Q': True, 'P': True}
Time taken =  0.0003572019999964482 s
 Do  you want to check any more sentences (Y/N)         n
```

WUMPUS WORLD (SIMPLE):

There is no pit at location [1,2]

Enumeration Method:

```
Enter y to check another sentance or n to exit   y
Enter the Proposition Sentence to be tested      ! P11 , B11 <> P12 v P21 , B21 <> P11 v P22 v P31 , ! B11 , B21 |= P12
 P12 cannot be inferred
Time taken =  0.014746101999996597 s
Enter y to check another sentance or n to exit   y
Enter the Proposition Sentence to be tested      ! P11 , B11 <> P12 v P21 , B21 <> P11 v P22 v P31 , ! B11 , B21 |= ! P12
 ! P12  can be inferred
Model 1 : {'P22': True, 'P11': False, 'B11': False, 'P21': False, 'P31': True, 'P12': False, 'B21': True}
Model 2 : {'P22': True, 'P11': False, 'B11': False, 'P21': False, 'P31': False, 'P12': False, 'B21': True}
Model 3 : {'P22': False, 'P11': False, 'B11': False, 'P21': False, 'P31': True, 'P12': False, 'B21': True}
Time taken =  0.034454233000005274 s
```

WalkSAT:

```
Enter the Proposition Sentence to be tested      ! P11 , ! B11 v P12 v P21 ^ ! P12 v B11 ^ ! P21 v B11 , ! B21 v P11 v P22 v P31 ^ ! P11 v B21 ^ ! P22 v B21 ^ ! P31 v B21 ,
 ! B11 , B21 |= P12
Unsatisfiable
Time taken =  13.463278438 s
 Do  you want to check any more sentences (Y/N)        y
Enter the Proposition Sentence to be tested      ! P11 , ! B11 v P12 v P21 ^ ! P12 v B11 ^ ! P21 v B11 , ! B21 v P11 v P22 v P31 ^ ! P11 v B21 ^ ! P22 v B21 ^ ! P31 v B21 ,
 ! B11 , B21 |= ! P12
Satisfiable
Model =  {'P11': False, 'P31': False, 'B11': False, 'P21': False, 'P12': False, 'B21': True, 'P22': True}
Time taken =  0.004834382999987952 s
 Do  you want to check any more sentences (Y/N)        n
```

HORN CLAUSES

Unicorn is Magical and Horned, but Mythical cannot be inferred.

Enumeration Method:

```
Enter y to check another sentance or n to exit   y
Enter the Proposition Sentence to be tested      Mythical -> ! Mortal , ! Mythical -> Mortal ^ Mammal , ! Mortal v Mammal -> Horned , Horned -> Magical |= Mythical
 Mythical cannot be inferred
Time taken =  0.002003571000003035 s
Enter y to check another sentance or n to exit   y
Enter the Proposition Sentence to be tested      Mythical -> ! Mortal , ! Mythical -> Mortal ^ Mammal , ! Mortal v Mammal -> Horned , Horned -> Magical |= ! Mythical
 ! Mythical cannot be inferred
Time taken =  0.0022574559999952726 s
Enter y to check another sentance or n to exit   y
Enter the Proposition Sentence to be tested      Mythical -> ! Mortal , ! Mythical -> Mortal ^ Mammal , ! Mortal v Mammal -> Horned , Horned -> Magical |= Magical
 Magical  can be inferred
Model 1 : {'Magical': True, 'Horned': True, 'Mortal': True, 'Mythical': False, 'Mammal': True}
Model 2 : {'Magical': True, 'Horned': True, 'Mortal': False, 'Mythical': True, 'Mammal': True}
Model 3 : {'Magical': True, 'Horned': True, 'Mortal': False, 'Mythical': True, 'Mammal': False}
Time taken =  0.007340787000039566 s
Enter y to check another sentance or n to exit   y
Enter the Proposition Sentence to be tested      Mythical -> ! Mortal , ! Mythical -> Mortal ^ Mammal , ! Mortal v Mammal -> Horned , Horned -> Magical |= Horned
 Horned  can be inferred
Model 1 : {'Magical': True, 'Horned': True, 'Mortal': True, 'Mythical': False, 'Mammal': True}
Model 2 : {'Magical': True, 'Horned': True, 'Mortal': False, 'Mythical': True, 'Mammal': True}
Model 3 : {'Magical': True, 'Horned': True, 'Mortal': False, 'Mythical': True, 'Mammal': False}
Time taken =  0.010886704999990116 s
```

WalkSAT:

```
:\Users\aniru\Desktop\UoR\Courses\FALL 2018\Artificial Intelligence\Project 2 due 22.10.2018\Project 2>WalkSATFin.py
nter the Proposition Sentence to be tested      ! Mythical v ! Mortal , Mythical v Mortal ^ Mythical v Mammal , Mortal v Horned ^ ! Mammal v Horned , ! Horned v Magical |=
! Mythical
atisfiable
odel =  {'Magical': True, 'Mythical': False, 'Mortal': True, 'Horned': True, 'Mammal': True}
ime taken =  0.0006435269999993665 s
Do  you want to check any more sentences (Y/N)         y
nter the Proposition Sentence to be tested      ! Mythical v ! Mortal , Mythical v Mortal ^ Mythical v Mammal , Mortal v Horned ^ ! Mammal v Horned , ! Horned v Magical |=
Mythical
atisfiable
odel =  {'Magical': True, 'Mythical': True, 'Mortal': False, 'Horned': True, 'Mammal': False}
ime taken =  0.0021439129999993867 s
Do  you want to check any more sentences (Y/N)         y
nter the Proposition Sentence to be tested      ! Mythical v ! Mortal , Mythical v Mortal ^ Mythical v Mammal , Mortal v Horned ^ ! Mammal v Horned , ! Horned v Magical |=
! Magical
nsatisfiable
ime taken =  5.391842057999998 s
Do  you want to check any more sentences (Y/N)         y
nter the Proposition Sentence to be tested      ! Mythical v ! Mortal , Mythical v Mortal ^ Mythical v Mammal , Mortal v Horned ^ ! Mammal v Horned , ! Horned v Magical |=
Magical
atisfiable
odel =  {'Magical': True, 'Mythical': False, 'Mortal': True, 'Horned': True, 'Mammal': True}
ime taken =  0.009086240999998552 s
Do  you want to check any more sentences (Y/N)         y
nter the Proposition Sentence to be tested      ! Mythical v ! Mortal , Mythical v Mortal ^ Mythical v Mammal , Mortal v Horned ^ ! Mammal v Horned , ! Horned v Magical |=
! Horned
nsatisfiable
ime taken =  6.225634500000005 s
Do  you want to check any more sentences (Y/N)         y
nter the Proposition Sentence to be tested      ! Mythical v ! Mortal , Mythical v Mortal ^ Mythical v Mammal , Mortal v Horned ^ ! Mammal v Horned , ! Horned v Magical |=
Horned
atisfiable
odel =  {'Magical': True, 'Mythical': False, 'Mortal': True, 'Horned': True, 'Mammal': True}
ime taken =  0.008806262999996761 s
Do  you want to check any more sentences (Y/N)         n
```

4. LIARS AND TRUTH-TELLERS

a. Amy is a liar (Amy cannot be inferred and ! Amy can be inferred)
   Bob is a liar (Bob cannot be inferred and ! Bob can be inferred)
   Cal tells the truth (Cal cannot be inferred)

b. Amy tells the truth (Amy can be inferred)
   Bob is a liar (Bob cannot be inferred and ! Bob can be inferred)
   Cal is a liar (Cal cannot be inferred and ! Cal can be inferred)

Enumeration Method:

a.

```
Enter the Proposition Sentence to be tested      Amy <> Cal ^ Amy , Bob <> ! Cal , Cal <> Bob v ! Amy |= Amy
 Amy cannot be inferred
Time taken =  0.0006675050000000127 s
Enter y to check another sentance or n to exit   y
Enter the Proposition Sentence to be tested      Amy <> Cal ^ Amy , Bob <> ! Cal , Cal <> Bob v ! Amy |= ! Amy
 ! Amy  can be inferred
Model 1 : {'Cal': True, 'Bob': False, 'Amy': False}
Time taken =  0.0016668220000006784 s
Enter y to check another sentance or n to exit   y
Enter the Proposition Sentence to be tested      Amy <> Cal ^ Amy , Bob <> ! Cal , Cal <> Bob v ! Amy |= Bob
 Bob cannot be inferred
Time taken =  0.0007993830000003754 s
Enter y to check another sentance or n to exit   y
Enter the Proposition Sentence to be tested      Amy <> Cal ^ Amy , Bob <> ! Cal , Cal <> Bob v ! Amy |= ! Bob
 ! Bob  can be inferred
Model 1 : {'Cal': True, 'Bob': False, 'Amy': False}
Time taken =  0.0013491129999998464 s
Enter y to check another sentance or n to exit   y
Enter the Proposition Sentence to be tested      Amy <> Cal ^ Amy , Bob <> ! Cal , Cal <> Bob v ! Amy |= Cal
 Cal  can be inferred
Model 1 : {'Cal': True, 'Bob': False, 'Amy': False}
Time taken =  0.001353344999998285 s
Enter y to check another sentance or n to exit   n
```

b.

```
Enter the Proposition Sentence to be tested      Amy <> ! Cal , Bob <> Amy ^ Cal , Cal <> Bob |= Amy
 Amy  can be inferred
Model 1 : {'Bob': False, 'Cal': False, 'Amy': True}
Time taken =  0.0011058080000001524 s
Enter y to check another sentance or n to exit   y
Enter the Proposition Sentence to be tested      Amy <> ! Cal , Bob <> Amy ^ Cal , Cal <> Bob |= Bob
 Bob cannot be inferred
Time taken =  0.0008360560000006956 s
Enter y to check another sentance or n to exit   y
Enter the Proposition Sentence to be tested      Amy <> ! Cal , Bob <> Amy ^ Cal , Cal <> Bob |= ! Bob
 ! Bob  can be inferred
Model 1 : {'Bob': False, 'Cal': False, 'Amy': True}
Time taken =  0.0010359899999983213 s
Enter y to check another sentance or n to exit   y
Enter the Proposition Sentence to be tested      Amy <> ! Cal , Bob <> Amy ^ Cal , Cal <> Bob |= Cal
 Cal cannot be inferred
Time taken =  0.0008466339999984029 s
Enter y to check another sentance or n to exit   y
Enter the Proposition Sentence to be tested      Amy <> ! Cal , Bob <> Amy ^ Cal , Cal <> Bob |= ! Cal
 ! Cal  can be inferred
Model 1 : {'Bob': False, 'Cal': False, 'Amy': True}
Time taken =  0.0010201209999962657 s
Enter y to check another sentance or n to exit   n
```

WalkSAT:

a.

```
Enter the Proposition Sentence to be tested      ! Amy v ! Cal , ! Bob v ! Cal ^ Bob v Cal , ! Cal v Bob v ! Amy ^ Cal v ! Bob ^ Cal v Amy |= Amy
Unsatisfiable
Time taken =  3.8426244449999998 s
 Do  you want to check any more sentences (Y/N)         y
Enter the Proposition Sentence to be tested      ! Amy v ! Cal , ! Bob v ! Cal ^ Bob v Cal , ! Cal v Bob v ! Amy ^ Cal v ! Bob ^ Cal v Amy |= ! Amy
Satisfiable
Model =  {'Bob': False, 'Amy': False, 'Cal': True}
Time taken =  0.0006392949999991515 s
 Do  you want to check any more sentences (Y/N)         y
Enter the Proposition Sentence to be tested      ! Amy v ! Cal , ! Bob v ! Cal ^ Bob v Cal , ! Cal v Bob v ! Amy ^ Cal v ! Bob ^ Cal v Amy |= Bob
Unsatisfiable
Time taken =  3.9557613430000025 s
 Do  you want to check any more sentences (Y/N)         y
Enter the Proposition Sentence to be tested      ! Amy v ! Cal , ! Bob v ! Cal ^ Bob v Cal , ! Cal v Bob v ! Amy ^ Cal v ! Bob ^ Cal v Amy |= ! Bob
Satisfiable
Model =  {'Bob': False, 'Amy': False, 'Cal': True}
Time taken =  0.001334657000001016 s
 Do  you want to check any more sentences (Y/N)         y
Enter the Proposition Sentence to be tested      ! Amy v ! Cal , ! Bob v ! Cal ^ Bob v Cal , ! Cal v Bob v ! Amy ^ Cal v ! Bob ^ Cal v Amy |= Cal
Satisfiable
Model =  {'Bob': False, 'Amy': False, 'Cal': True}
Time taken =  0.0008385240000023941 s
 Do  you want to check any more sentences (Y/N)         y
Enter the Proposition Sentence to be tested      ! Amy v ! Cal , ! Bob v ! Cal ^ Bob v Cal , ! Cal v Bob v ! Amy ^ Cal v ! Bob ^ Cal v Amy |= ! Cal
Unsatisfiable
Time taken =  4.9136013639999945 s
 Do  you want to check any more sentences (Y/N)         n
```

b.

```
Enter the Proposition Sentence to be tested      ! Amy v ! Cal ^ Amy v Cal , ! Bob v Amy ^ ! Bob v Cal ^ Bob v ! Amy v ! Cal , ! Cal v Bob ^ ! Bob v Cal |= Amy
Satisfiable
Model =  {'Cal': False, 'Bob': False, 'Amy': True}
Time taken =  0.001861113999999997 s
 Do  you want to check any more sentences (Y/N)         y
Enter the Proposition Sentence to be tested      ! Amy v ! Cal ^ Amy v Cal , ! Bob v Amy ^ ! Bob v Cal ^ Bob v ! Amy v ! Cal , ! Cal v Bob ^ ! Bob v Cal |= ! Amy
Unsatisfiable
Time taken =  5.042683615000001 s
 Do  you want to check any more sentences (Y/N)         y
Enter the Proposition Sentence to be tested      ! Amy v ! Cal ^ Amy v Cal , ! Bob v Amy ^ ! Bob v Cal ^ Bob v ! Amy v ! Cal , ! Cal v Bob ^ ! Bob v Cal |= Bob
Unsatisfiable
Time taken =  5.354949125000001 s
 Do  you want to check any more sentences (Y/N)         y
Enter the Proposition Sentence to be tested      ! Amy v ! Cal ^ Amy v Cal , ! Bob v Amy ^ ! Bob v Cal ^ Bob v ! Amy v ! Cal , ! Cal v Bob ^ ! Bob v Cal |= ! Bob
Satisfiable
Model =  {'Cal': False, 'Bob': False, 'Amy': True}
Time taken =  0.0017845960000002492 s
 Do  you want to check any more sentences (Y/N)         y
Enter the Proposition Sentence to be tested      ! Amy v ! Cal ^ Amy v Cal , ! Bob v Amy ^ ! Bob v Cal ^ Bob v ! Amy v ! Cal , ! Cal v Bob ^ ! Bob v Cal |= Cal
Unsatisfiable
Time taken =  5.495401869999995 s
 Do  you want to check any more sentences (Y/N)         y
Enter the Proposition Sentence to be tested      ! Amy v ! Cal ^ Amy v Cal , ! Bob v Amy ^ ! Bob v Cal ^ Bob v ! Amy v ! Cal , ! Cal v Bob ^ ! Bob v Cal |= ! Cal
Satisfiable
Model =  {'Cal': False, 'Bob': False, 'Amy': True}
Time taken =  0.001027879999995207 s
 Do  you want to check any more sentences (Y/N)         n
```

5. Conclusion

As one can see WalkSAT algorithm is less computationally costly than the enumeration method. But when a query is unsatisfiable the WalkSAT algorithm takes a lot of time when compared to enumeration method.

The ability of the WalkSAT algorithm to find a model depends on the maximum allowable flips and the probability of flipping (randomly), the chances of finding the model increase as we increase the maximum allowable flips and the probability of flipping.

6. REFERENCES:

1. Russell, S. and Norvig, P., 2010. *Artificial Intelligence: A Modern Approach* 3rd ed.,

2. WalkSAT. *En.wikipedia.org*.

SAI ANIRUDH REDDY AVILALA

UR ID: 31428149