CSC 442 Project 2 - Automated Reasoning

Name : Tushar Kumar
NetId : tusharku

October 27, 2018

Abstract

In this project I have implemented two inference methods for Propositional logic and demonstrated their implementation on six sample problems of reasoning. The two methods that I have used are Enumeration by truth table and Davis-Putnam-Logemann-Loveland(DPLL henceforth) algorithm. I have also built a converter to convert logical expressions that I have coded for the sample problems into conjunctive normal form(CNF).

1 Introduction

Propositional logic is a mechanism of representing our knowledge or belief about the world and it consists of complex sentences built out of literals and connectives which evaluate to either true or false.

Our goal is to build a program which given a knowledge base(a set of statements which capture our knowledge about the world) and given a query, can compute whether that query can be answered or inferred based on the provided knowledge base or not.

2 Key Implementation Details

a) **Implemented Truth table enumeration** method for completing Part I of project requirement

b) **Implemented DPLL inference method** for completing Part II of project requirement

c) **Built a CNF Converter** to convert all propositional logic statement to CNF form

d) **Solved all 4 required reasoning problems** as per project requirement using both methods.

e) **Solved 2 problems(5,6) for extra credit** using both methods.

f) **Compared the two methods on two statistic : Time, Number of states visited in the tree of possible assignments**(partial or complete) by each method for each problem. The latter is measured by the number of recursion calls.

3 Solution to Problems

I have solved 6 out of 8 given problems. I am just providing the conclusions that I reached on those problems after running inference algorithms on them. We will delve into each problem in later sections. This is just a summary of solutions.

a) Modus Ponens($P => Q, P| = Q$)
It can be inferred that ***Q is indeed entailed***.

b) Wumpus World (Simple)
It can be inferred that ***there is no Pit at [1,2]***.

c) Horn Clauses (Russell & Norvig)
It can be inferred and hence proved that ***unicorn is magical and horned***.
It cannot be inferred and hence ***cannot be proved that unicorn is mythical or not***

d) (a) Liars and Truth-tellers-a (OSSMB 82-12)
It can be inferred that ***Cal is truthful***.
It can be inferred that ***Amy and Bob are liars***.

(b) Liars and Truth-tellers-b (OSSMB 83-11)
It can be inferred that ***Amy is truthful***.
It can be inferred that ***Cal and Bob are liars***.

e) More Liars and Truth-tellers (adapted from JRM14 392)
It can be inferred that ***Jay and Kay are truthful***.
It can be inferred that ***rest all are liars***.

f) (a) The Doors of Enlightenment (from CRUX 357) : Smullyan's problem
It can be inferred that ***philosopher should choose X***

(b) The Doors of Enlightenment (from CRUX 357) : Liu's problem
It can be inferred that ***philosopher has heard enough to know that X is a good door***

4 Design

The basic design can be understood from the Class Diagram 1. In order to be able to have same structure for any kind of reasoning problem, I created an interface which would provide the necessary contract for any problem which needs to be solved by inference.
In order to store sentence, the most appropriate data structure that I could think of was a

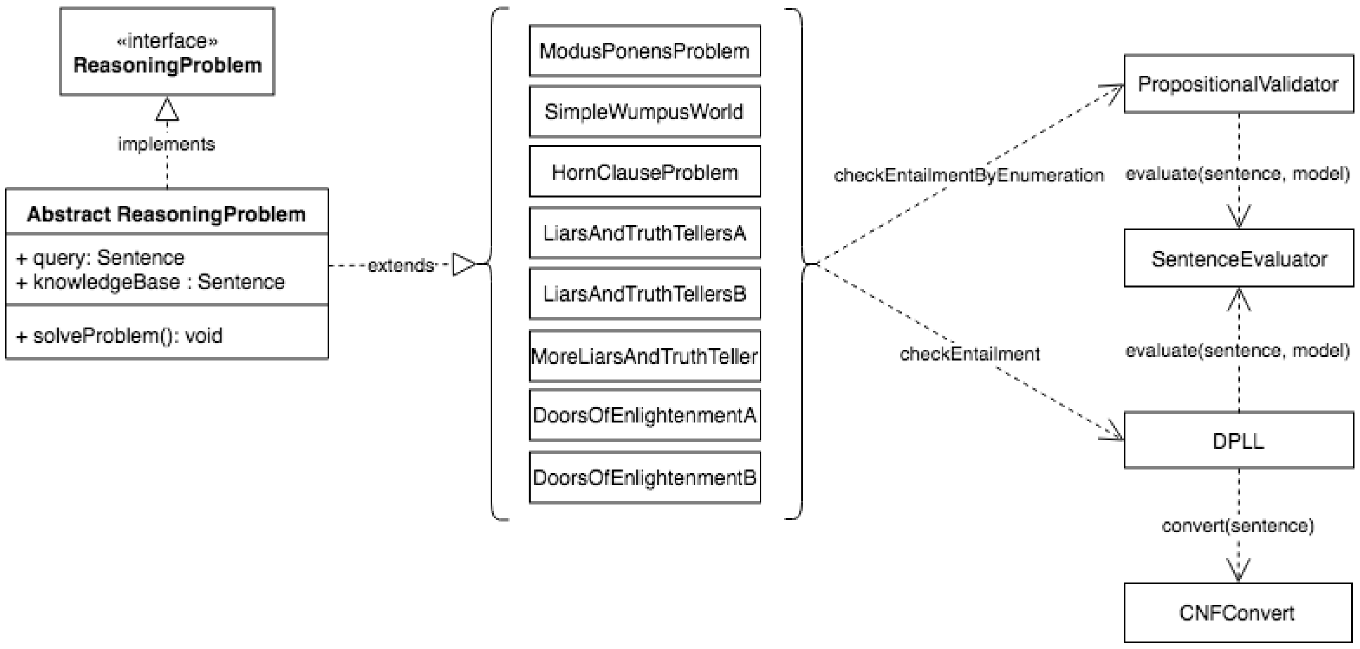


Figure 1: Class Diagram of Automated Reasoning Project

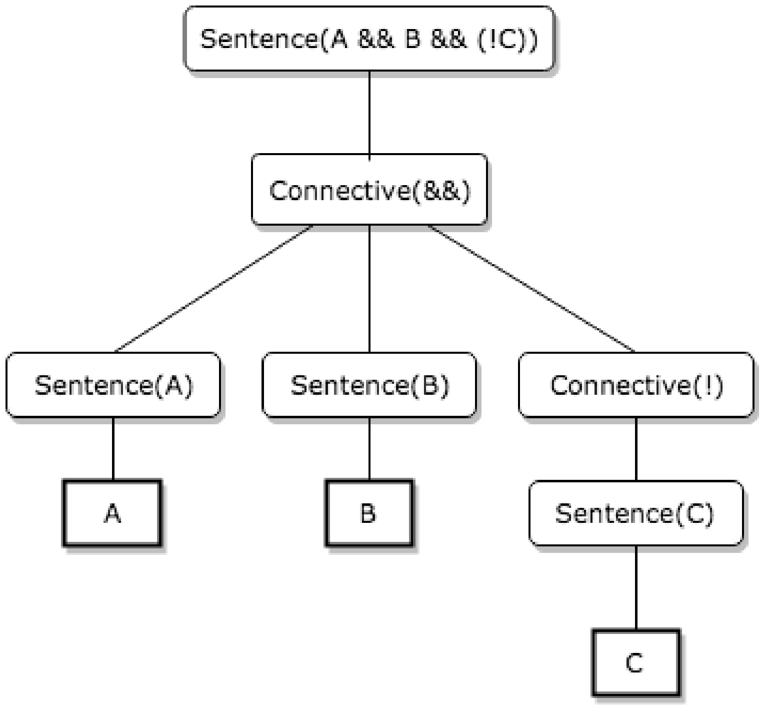


Figure 2: Sample Expression tree for a sentence

N-ary tree which is what is the Sentence class. Figure 2 denotes the way an expression tree is created for a sample statement. Each node can either be a connective($And - \&\&, Or - \|, Not - !, Implies - \; =>, Iff - \; <=>$) or can be a literal symbol in which case it will also be again represented as a sentence. Each node can either be a symbol or can be a connective, connecting N children of same Sentence class. In order to use DPLL algorithm, I also constructed a CNF Converter which given any propositional statement converts it into a conjuction of clauses.

4.1 Key Design and High Level Implementation Details

a) Interface to handle any kind of reasoning problem

b) Built a sentence class that captures expression as a N-ary tree consisting of literals as leaf nodes

c) Added ability to print an expression given the tree built.

d) Symbols used for connectives in program :: AND is represented by &&, OR = represented by ||, Implies is represented by =>, BiConditional is represented by <=> , NOT is represented by !

e) No restriction on number of children that ’OR’ and ’AND’ expressions can have.

f) Built a CNF converter to convert any statement to CNF.

g) Implemented DPLL algorithm for inferencing.

h) Solved six of the eight given problems.

i) Logged time taken and recursion stack size for both enumeration and DPLL inference method

j) Created Enum for Question types.

k) Throwing descriptive errors for bad command line arguments.

5 Implementation

I have implemented the truth table enumeration method and DPLL algorithm for inference. I have also built a CNF converter to have the statements converted to CNF for DPLL. Before going over the details and solution of each problem, I would be talking about the CNF converter and two kinds of inference methods I used.

5.1 CNF Converter

CNFConverter, as the name suggests does exactly that, converting a statement into CNF form. The overall mechanism that I use is pretty much a divide and conquer method. I divide the statement into substatements and convert each of them into CNF and then just join them appropriately based on the connectives they had. It took a lot of iterations to do it correctly, because it took me time to realize that I must first convert all implications and biconditional expressions before using the divide and conquer approach. Here is the details on how exactly I implemented it:

a) Convert BiConditional expressions to implications

b) Convert implications $A => B\ to\ !A\ ||\ B$

c) Push negation and handle these three cases:

   (a) $!!A$ becomes A
   (b) For A && B call pushNegation($!A$) and pushNegation($!B$) and combine the result by Or.
   (c) For A || B call pushNegation($!A$) and pushNegation($!B$) and combine the result by &&.

d) If the main connective is And then convert each connected sentence by calling the same method(for each subproblem) and combine the output by And

e) If the main connective is OR then handle the four scenarios appropriately

   (a) (A or B) Or (C or D)
   (b) (A and B) Or (C and D)
   (c) (A and B) Or (C or D)
   (d) (A or B) Or (C and D)

f) Extract all sentences linked together by AND, and these are the CNF clauses.

5.2 Truth Table Enumeration

The truth table enumeration method is pretty much what was discussed in the classes. Take symbols one by one and keep on branching out with one branch for true value of that symbol and another for false value of that symbol. Once all symbols are assigned, I use Sentence evaluator to check if there is any case where knowledge base is true but the query is false. If there is any such case then the query is not entailed by the KB, otherwise it indeed is.

5.3 DPLL

The DPLL algorithm is implemented from the pseudo code given in AIMA. In the worst case scenario DPLL will eventually just assign symbols one by one, but it tries to find an "intelligent" order of assignments such that if the query was indeed unsatisfiable then we get to that result very quickly. I check the unsatisfiability of (KB !Query). The brief steps are same as the one mentioned in book:

a) Find if all clauses evaluate to true

b) Find if any one clause evaluates to false

c) Get pure symbol and assign that an appropriate value to make that clause as true and then recurse after removing it from symbol list and adding assignment to model

d) Get unit clause if any and assign that an appropriate value to make that clause as true and then recurse after removing it from symbol list and adding assignment to model

e) If neither a pure symbol neither a unit clause was found then just assign then just pickup one symbol from list and create two branches where for one its assigned as truth and other is assigned as false. Return true if any one branch is true else return false.

6 Problems

I will now be talking about all the problems which I solved using both the methods one by one. I have attached the snapshot of the result of solving that problem using both methods. I also compare the time taken for solving using both of them.

6.1 Modus Ponens

$$\{P, P => Q\} \mid = Q$$

6.1.1 CNF Form generated with KB and negation of query

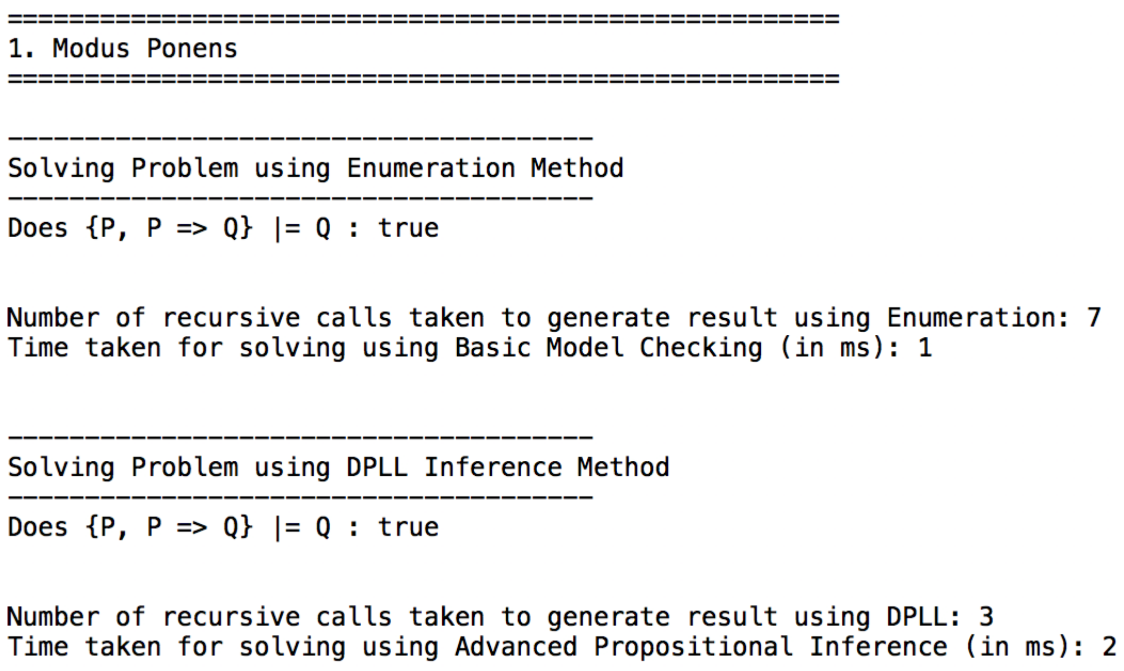
$$P$$

$$((!P) \mid\mid Q)$$

$$(!Q)$$

6.1.2 Output of Program

Figure 3 provides the output for running inference for Modus Ponens problem. Since modus ponens is a sound rule its no surprise the the result of both the methods is obviously True. Both the methods take about the same time(which is small). Do note that the time taken for DPLL does not include the time taken for converting the statement to CNF. It includes on the inference time.



```
======================================================
1. Modus Ponens
======================================================

--------------------------------------
Solving Problem using Enumeration Method
--------------------------------------
Does {P, P => Q} |= Q : true


Number of recursive calls taken to generate result using Enumeration: 7
Time taken for solving using Basic Model Checking (in ms): 1


--------------------------------------
Solving Problem using DPLL Inference Method
--------------------------------------
Does {P, P => Q} |= Q : true


Number of recursive calls taken to generate result using DPLL: 3
Time taken for solving using Advanced Propositional Inference (in ms): 2
```

Figure 3: Output of executing inference for Modus Ponens problem

6.2 Wumpus World (Simple)

Prove whether P1,2 is true or not (that is, whether there is a pit at location [1,2]).

6.2.1 CNF Form generated with KB and negation of query

KB Expression
Generated using code

$((!P11)\ \&\&\ (B11 <=> (P12||P21))\ \&\&\ (B21 <=> (P11||P22||P31))\ \&\&\ (!B11)\ \&\&\ B21)$

Query statements

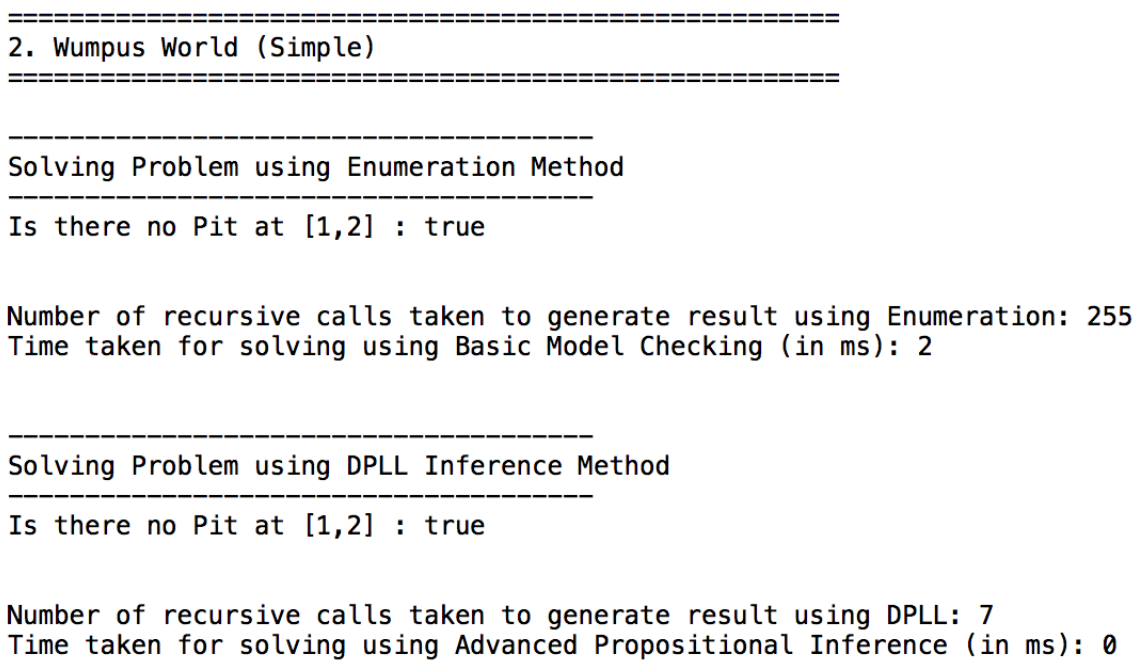
!P12 (Is there no pit at 1,2)

CNF

(!P11), ((!B11) || (P12||P21)), ((!P12)|| B11), ((!P21) ||B11), ((!B21) || ((P11||P22)||P31)), ((!P11) || B21), ((!P22) || B21), ((!P31) || B21), (!B11), B21, (P12)

6.2.2 Output of Program

Figure 4 provides the output for running inference for wumpus world problem. Both the methods conclude that there is indeed no pit at 1,2. DPLL concludes that there being a Pit at 1,2 is unsatisfiable with the KB, hence there must not be a pit at 1,2.



```
======================================================
2. Wumpus World (Simple)
======================================================

--------------------------------------
Solving Problem using Enumeration Method
--------------------------------------
Is there no Pit at [1,2] : true


Number of recursive calls taken to generate result using Enumeration: 255
Time taken for solving using Basic Model Checking (in ms): 2


--------------------------------------
Solving Problem using DPLL Inference Method
--------------------------------------
Is there no Pit at [1,2] : true


Number of recursive calls taken to generate result using DPLL: 7
Time taken for solving using Advanced Propositional Inference (in ms): 0
```

Figure 4: Output of executing inference for Wumpus world simple problem

6.3 Horn Clauses (Russell & Norvig)

Can we prove that the unicorn is mythical?
Can we prove that the unicorn is magical?
Can we prove that the unicorn is horned?

6.3.1 CNF Form generated with KB

KB Expression

$((U\_Mythical => (!U\_Mortal))\ \&\&\ ((!U\_Mythical) => (U\_Mortal\&\&U\_Mammal))$
$\&\&(((!U\_Mortal)||U\_Mammal) => U\_Horned)\ \&\&\ (U\_Horned => U\_Magical))$

Query statements

U_Mythical (Unicorn is mythical)
!U_Mythical (Unicorn is NOT mythical)
U_Magical (Unicorn is magical)
U_Horned (Unicorn is horned)

CNF

((!U_Mythical) || (!U_Mortal)), (U_Mythical || U_Mortal), (U_Mythical ||U_Mammal), (U_Mortal || U_Horned), ((!U_Mammal) || U_Horned), ((!U_Horned) || U_Magical)

6.3.2 Output of Program

A false value by inference does not mean that we can conclude the opposite fact. Hence I tried with all the cases and then am providing the output of cases which indeed can be inferred from the KB. ***Above every statement which I cannot prove as true, I provide a model where we inferred that KB was true but query was false.*** Figure 5 provides the output for running inference for wumpus world problem. I test the queries one by one and the time mentioned is the total time taken for inference on all queries. Here's what we can conclude:

We CANNOT prove whether the the unicorn is mythical or not
We can prove that the unicorn is magical
We can prove that the unicorn is horned

6.4 Liars and Truth-tellers-a (OSSMB 82-12)

What can you conclude about the truthfulness of each?

6.4.1 CNF Form generated with KB

In all the truth statements, we need to have biconditional because if we know that what the person said is true then we can conclude that person is truthful and vice versa, if we know that what they said is false, then we can conclude they are a liar.

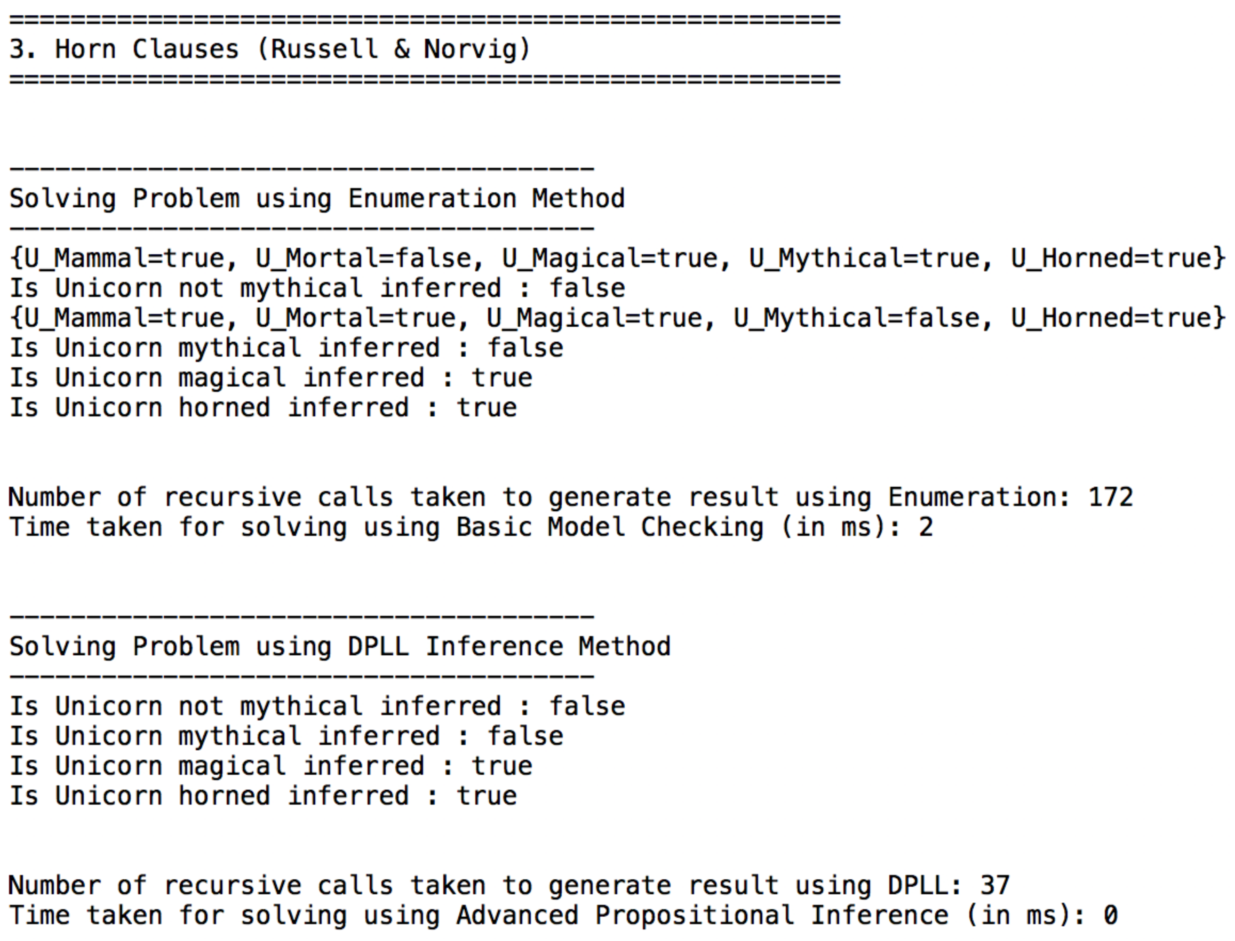
KB Expression

$((Amy <=> (Cal \&\& Amy)) \&\& (Bob <=> (!Cal)) \&\& (Cal <=> (Bob || (!Amy))))$

Query statements

Amy (Is Amy truthful)
!Amy (Is Amy dishonest)
Cal (Is Cal truthful)
Bob (Is Bob truthful)
!Bob (Is Bob dishonest)

CNF



```
======================================================
3. Horn Clauses (Russell & Norvig)
======================================================


--------------------------------------
Solving Problem using Enumeration Method
--------------------------------------
{U_Mammal=true, U_Mortal=false, U_Magical=true, U_Mythical=true, U_Horned=true}
Is Unicorn not mythical inferred : false
{U_Mammal=true, U_Mortal=true, U_Magical=true, U_Mythical=false, U_Horned=true}
Is Unicorn mythical inferred : false
Is Unicorn magical inferred : true
Is Unicorn horned inferred : true


Number of recursive calls taken to generate result using Enumeration: 172
Time taken for solving using Basic Model Checking (in ms): 2


--------------------------------------
Solving Problem using DPLL Inference Method
--------------------------------------
Is Unicorn not mythical inferred : false
Is Unicorn mythical inferred : false
Is Unicorn magical inferred : true
Is Unicorn horned inferred : true


Number of recursive calls taken to generate result using DPLL: 37
Time taken for solving using Advanced Propositional Inference (in ms): 0
```

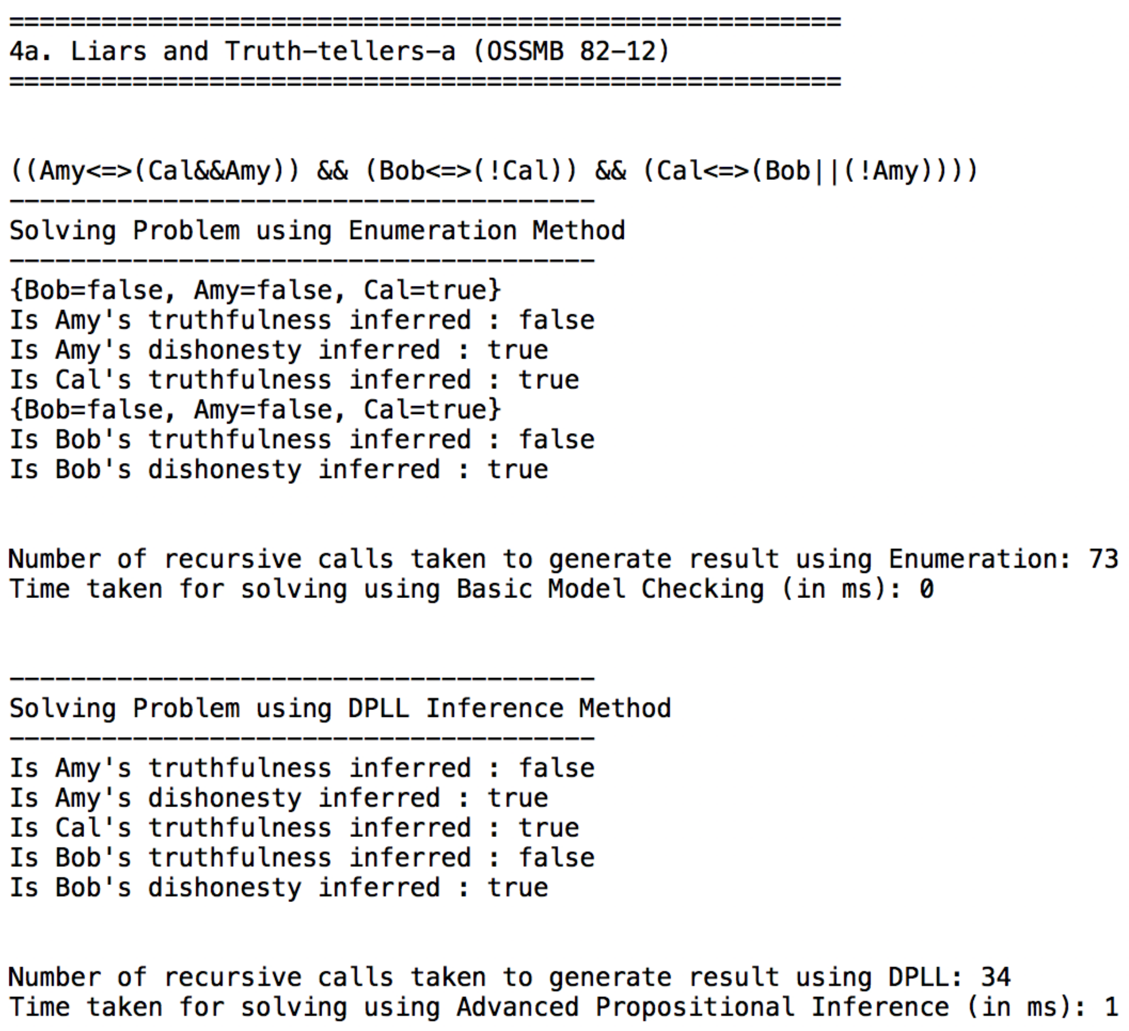
Figure 5: Output of executing inference for HornClause problem

((!Amy) || Cal) , ((!Amy) || Amy) , (((!Cal)||(!Amy)) || Amy) , ((!Bob) || (!Cal)) , (Cal || Bob) , ((!Cal) || (Bob||(!Amy))) , ((!Bob) || Cal) , (Amy || Cal)

6.4.2 Output of Program

A false value by inference does not mean that we can conclude the opposite fact. Hence I tried with all the cases and then am providing the output of cases which indeed can be inferred from the KB. Figure 6 provides the output for running inference for Liars and truth tellers a.) problem. Here's what we can conclude:

We can conclude that Amy is a liar
We can conclude that Cal is truthful
We can conclude that Bob is a liar



```
=====================================================
4a. Liars and Truth-tellers-a (OSSMB 82-12)
=====================================================


((Amy<=>(Cal&&Amy)) && (Bob<=>(!Cal)) && (Cal<=>(Bob||(!Amy))))
--------------------------------------
Solving Problem using Enumeration Method
--------------------------------------
{Bob=false, Amy=false, Cal=true}
Is Amy's truthfulness inferred : false
Is Amy's dishonesty inferred : true
Is Cal's truthfulness inferred : true
{Bob=false, Amy=false, Cal=true}
Is Bob's truthfulness inferred : false
Is Bob's dishonesty inferred : true


Number of recursive calls taken to generate result using Enumeration: 73
Time taken for solving using Basic Model Checking (in ms): 0


--------------------------------------
Solving Problem using DPLL Inference Method
--------------------------------------
Is Amy's truthfulness inferred : false
Is Amy's dishonesty inferred : true
Is Cal's truthfulness inferred : true
Is Bob's truthfulness inferred : false
Is Bob's dishonesty inferred : true


Number of recursive calls taken to generate result using DPLL: 34
Time taken for solving using Advanced Propositional Inference (in ms): 1
```

Figure 6: Output of executing inference for Liars And Truth Tellers a.) problem

6.5 Liars and Truth-tellers-b (OSSMB 83-11)

What can you conclude about the truthfulness of each?

6.5.1 CNF Form generated with KB

KB Expression

$((Amy <=> (!Cal))\ \&\&\ (Bob <=> (Amy\&\&Cal))\ \&\&\ (Cal <=> Bob))$

Query statements

Amy (Is Amy truthful)
Cal (Is Cal truthful)

!Cal (Is Cal dishonest)
Bob (Is Bob truthful)
!Bob (Is Bob dishonest)

CNF

((!Amy) || (!Cal)) , (Cal || Amy) , ((!Bob) || Amy) , ((!Bob) || Cal) , (((!Amy)||(!Cal)) || Bob) , ((!Cal) || Bob) , ((!Bob) || Cal)

6.5.2 Output of Program

A false value by inference does not mean that we can conclude the opposite fact. Hence I tried with all the cases and then am providing the output of cases which indeed can be inferred from the KB. Figure 7 provides the output for running inference for Liars and truth tellers b.) problem. Here's what we can conclude:

We can conclude that Amy is truthful
We can conclude that Cal is a liar
We can conclude that Bob is a liar

6.6 More Liars and Truth-tellers (adapted from JRM14 392)

Who are liars and who are truth-tellers?

Query statements

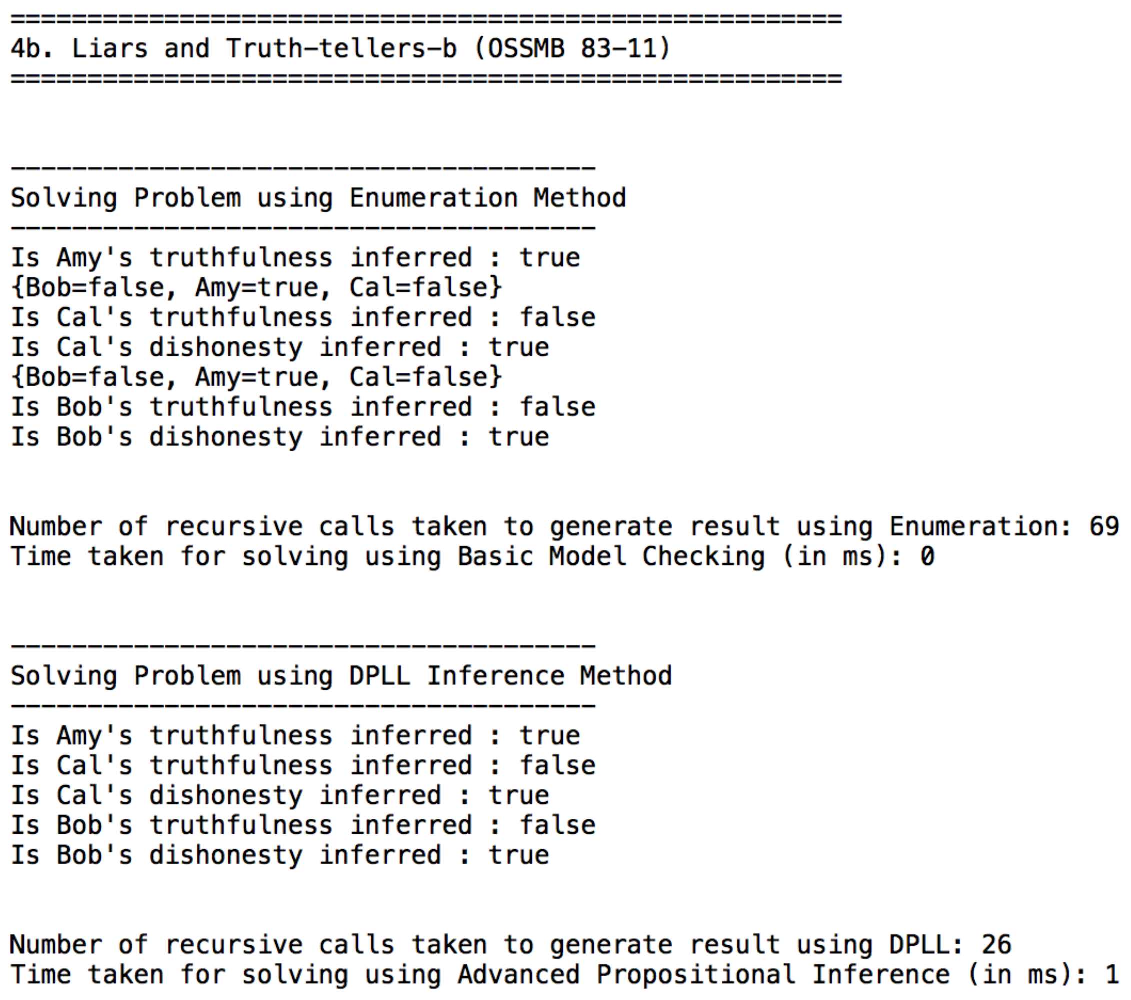
X (Is X truthful) X ∈ {Amy, Hal, Ida, Bob, Lee, Gil, Dee, Eli, Fay, Cal, Jay, Fay}
!X (Is X dishonest) X ∈ {Amy, Hal, Ida, Bob, Lee, Gil, Dee, Eli, Fay, Cal}

6.6.1 CNF Form generated with KB

((!Amy) || Hal) , ((!Amy) || Ida) , (((!Hal)||(!Ida)) ||Amy) , ((!Bob) || Amy) , ((!Bob) || Lee) , (((!Amy)||(!Lee)) || Bob) , ((!Cal) || Bob) , ((!Cal) || Gil) , (((!Bob)||(!Gil)) || Cal) , ((!Dee) || Eli) , ((!Dee) || Lee) , (((!Eli)||(!Lee)) || Dee) , ((!Eli) || Cal) , ((!Eli) || Hal) , (((!Cal)||(!Hal)) || Eli) , ((!Fay) || Dee) , ((!Fay) || Ida) , (((!Dee)||(!Ida)) || Fay) , ((!Gil) || (!Eli)) , ((!Gil) || (!Jay)) , ((Eli||Jay) || Gil) , ((!Hal) || (!Fay)) , ((!Hal) || (!Kay)) , ((Fay——Kay) || Hal) , ((!Ida) || (!Kay)) , ((!Ida) || (!Gil)) , ((Kay||Gil) || Ida) , ((!Jay) || (!Amy)) , ((!Jay) || (!Cal)) , ((Amy||Cal) || Jay) , ((!Kay) || (!Dee)) , ((!Kay) || (!Fay)) , ((Dee||Fay) || Kay) , ((!Lee) || (!Bob)) , ((!Lee) || (!Jay)) , ((Bob||Jay) || Lee)

6.6.2 Output of Program

A false value by inference does not mean that we can conclude the opposite fact. Hence I tried with all the cases and then am providing the output of cases which indeed can be



```
=====================================================
4b. Liars and Truth-tellers-b (OSSMB 83-11)
=====================================================


--------------------------------------
Solving Problem using Enumeration Method
--------------------------------------
Is Amy's truthfulness inferred : true
{Bob=false, Amy=true, Cal=false}
Is Cal's truthfulness inferred : false
Is Cal's dishonesty inferred : true
{Bob=false, Amy=true, Cal=false}
Is Bob's truthfulness inferred : false
Is Bob's dishonesty inferred : true


Number of recursive calls taken to generate result using Enumeration: 69
Time taken for solving using Basic Model Checking (in ms): 0


--------------------------------------
Solving Problem using DPLL Inference Method
--------------------------------------
Is Amy's truthfulness inferred : true
Is Cal's truthfulness inferred : false
Is Cal's dishonesty inferred : true
Is Bob's truthfulness inferred : false
Is Bob's dishonesty inferred : true


Number of recursive calls taken to generate result using DPLL: 26
Time taken for solving using Advanced Propositional Inference (in ms): 1
```

Figure 7: Output of executing inference for Liars And Truth Tellers b.) problem

inferred from the KB. Figure 8 provides the output for running inference for Liars and truth tellers b.) problem. Here's what we can conclude:

Only Jay and Kay are truthful
Everybody else is a liar

6.7 The Doors of Enlightenment (from CRUX 357) : Smullyan's problem

Which door should the philosopher choose?

6.7.1 CNF Form generated with KB

KB Expression

$((X||Y||Z||W)$ && $(A <=> X)$ && $(B <=> (Y||Z))$ && $(C <=> (A\&\&B))$ &&
$(D <=> (X\&\&Y))$ && $(E <=> (X\&\&Z))$ && $(F <=> (D||E))$ && $(G <=> (C =>$
$F))$ &&
$(H <=> ((G\&\&H) => A)))$

Query statements

X (Is X a good door)
Y (Is Y a good door)
Z (Is Z a good door)
W (Is W a good door)

CNF

(((X||Y)||Z) || W) , ((!A) || X) , ((!X) || A) , ((!B) || (Y——Z)) , ((!Y) || B) , ((!Z) || B) , ((!C) || A) , ((!C) || B) , (((!A)||(!B)) || C) , ((!D) || X) , ((!D) || Y) , (((!X)||(!Y)) || D) , ((!E) || X) , ((!E) || Z) , (((!X)||(!Z)) || E) , ((!F) || (D——E)) , ((!D) || F) , ((!E) || F) , ((!G) || ((!C)||F)) , (C || G) , ((!F) || G) , ((!H) || (((!G)||(!H))||A)) , (G || H) , (H || H) , ((!A) || H)

6.7.2 Output of Program

After running the inference methods on the given KB and querying for all possible doors only door X is good is a statement that can be inferred from KB. Figure 9 provides the output for running inference on Doors of enlightenment Smullyan's problem. Here's what we can conclude:

Philosopher should choose door X

6.8 The Doors of Enlightenment (from CRUX 357) : Liu's problem

Prove that he had heard enough to make a decision.

6.8.1 CNF Form generated with KB

This is a tricky question, on first glance it seems that there is no way any knowledge can be gained from the statement "If C is a knight", but one must realize that if by any chance C is not a knight, then the implication will always end up being true. That means, if we know that C is indeed not a knight, we

can infer that G is a knight as he is telling the truth. Since premise being false, makes the implication always true, which means its a statement that DOES indeed provide some information..

In order to build an expression for what philosopher heard from G "If C is a knight,...", I have assumed the conclusion or consequent of this implication clause to be "Whatever". This is because when we will give whatever a true value and when we will give it a false value, in both cases once we add that to KB, we will see that no matter what "Whatever" is, philosopher can still choose the door.

Expression using whatever as false

$((X||Y||Z||W)$ && $(A <=> X)$ && $(H <=> ((G$ && $H) => A))$ && $(C <=> (A$ && $(B||C||D||E||F||G||H)))$ && $(!Whatever)$ && $(G <=> (C => Whatever)))$

CNF Form

(((X||Y)||Z) || W) , ((!A) || X) , ((!X) || A) , ((!H) || (((!G)||(!H))||A)) , (G || H) , (H || H) , ((!A) || H) , ((!C) || A) , ((!C) || ((((((B||C)||D)||E)||F)||G)||H)) , (((!A)||(!B)) || C) , (((!A)||(!C)) || C) , (((!A)||(!D)) || C) , (((!A)||(!E)) || C) , (((!A)||(!F)) || C) , (((!A)||(!G)) || C) , (((!A)||(!H)) || C) , (!Whatever) , ((!G) || ((!C)||Whatever)) , (C || G) , ((!Whatever) || G)

6.8.2 Output of Program

After running the inference methods on the given KB and querying for all possible doors only door X is good is a statement that can be inferred from KB. Also this statement is inferred if whatever is true is added to KB or whatever is false is added to kb. Figure 10 provides the output for running inference on Doors of enlightenment Liu's problem with whatever being false added to KB. I also changed it Whatever being true and added it to KB, Figure 11, and both the cases lead to the same inference and Whatever could only either be false or true. Here's what we can conclude:

Philosopher should choose door X

7 DPLL vs Truth Table Enumeration

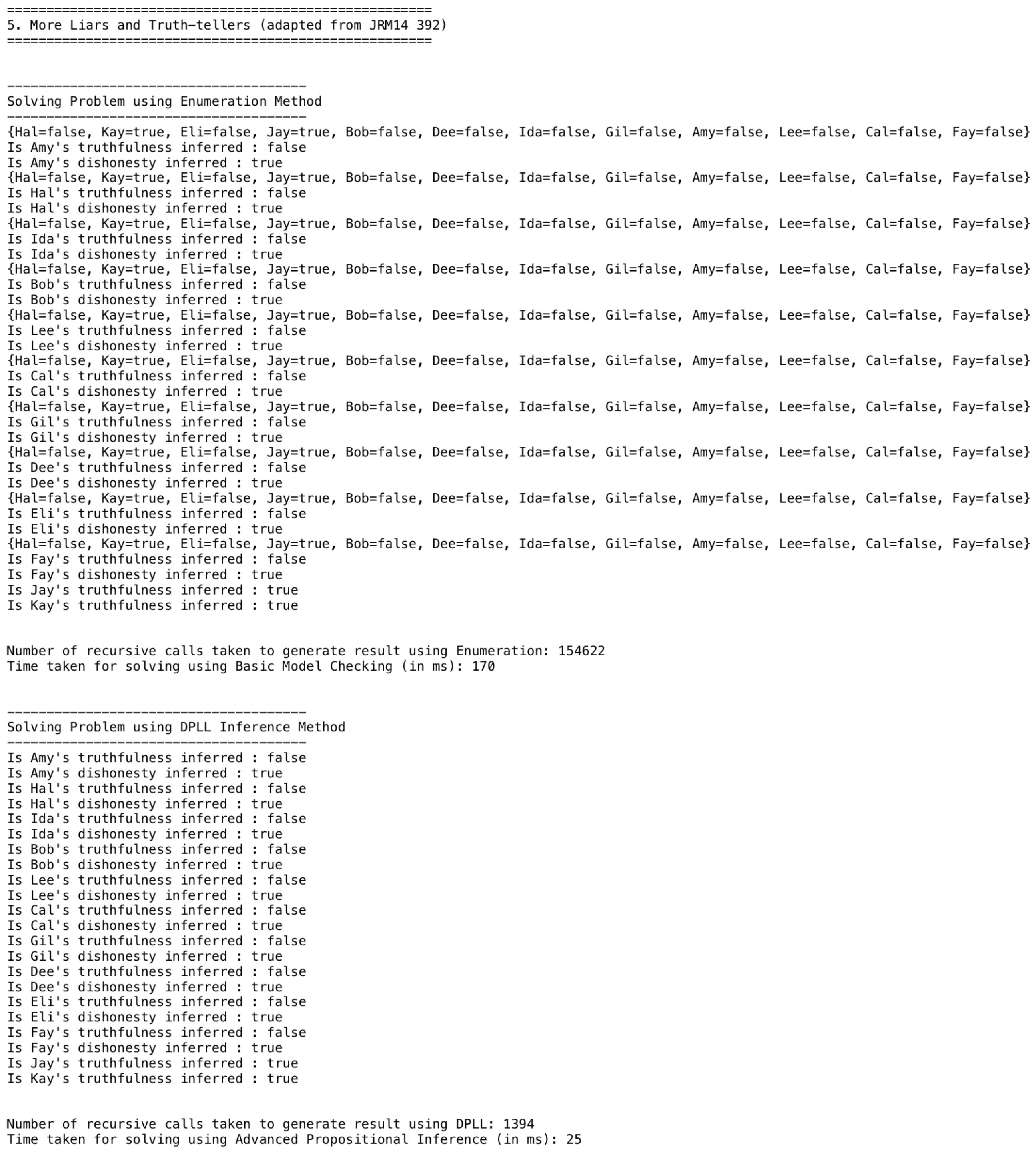
There are couple of insights that I could garner while executing inference through both of these methods as part of this project.

7.1 Time taken for Inference

Whenever the problem has a lot of variables or symbols involved in it like problems 5,6, we can see that enumeration technique takes much longer than the DPLL technique. The reason being that because of the right choices that DPLL makes, its able to reach to a decision much quickly than naive enumeration technique. I have plotted the effect of inference method on time in Figure 12 One might see different timings taken for the first program they run, but that is only because the loading of static class takes time for the first time and hence the timings would be a bit off for the first problem when you run from terminal vs when you run from IDE(where class is loaded already).

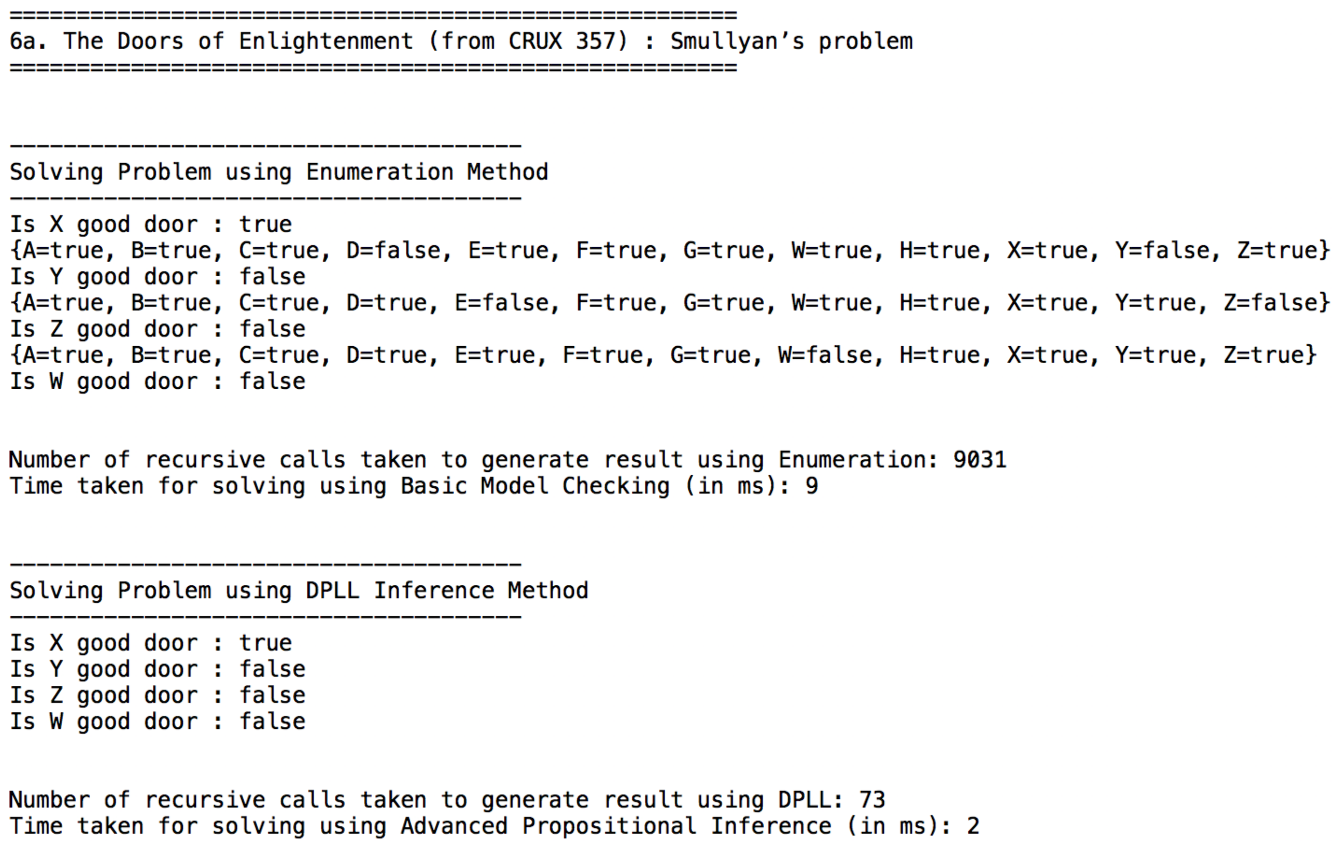
7.2 Number of recursion calls used for Inference

When the problem deals with a lot of variables or symbols, another thing to notice is that the striking difference between recursion call stack size. The number of recursion calls in DPLL are much much smaller than that using enumeration. Enumeration travels the entire search tree of possible model assignments before reaching a conclusion. Though the And logical expression in return statement will short circuit some evaluations but nevertheless the number still is huge. DPLL on the other hand through its intelligent order of picking symbols to assign a value, ends up searching a smaller portion of assignment tree. I have plotted the effect of inference method on number of recursion calls in Figure 13



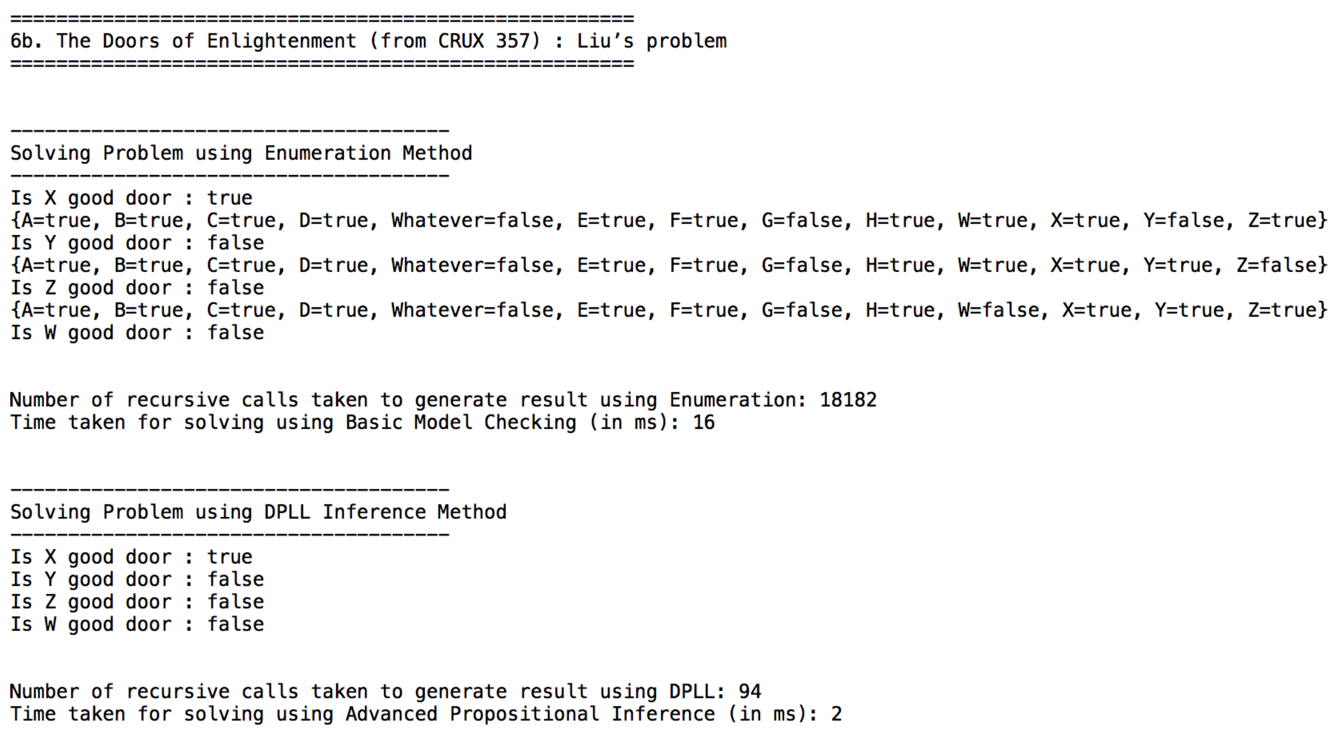
```
======================================================
5. More Liars and Truth-tellers (adapted from JRM14 392)
======================================================


--------------------------------------
Solving Problem using Enumeration Method
--------------------------------------
{Hal=false, Kay=true, Eli=false, Jay=true, Bob=false, Dee=false, Ida=false, Gil=false, Amy=false, Lee=false, Cal=false, Fay=false}
Is Amy's truthfulness inferred : false
Is Amy's dishonesty inferred : true
{Hal=false, Kay=true, Eli=false, Jay=true, Bob=false, Dee=false, Ida=false, Gil=false, Amy=false, Lee=false, Cal=false, Fay=false}
Is Hal's truthfulness inferred : false
Is Hal's dishonesty inferred : true
{Hal=false, Kay=true, Eli=false, Jay=true, Bob=false, Dee=false, Ida=false, Gil=false, Amy=false, Lee=false, Cal=false, Fay=false}
Is Ida's truthfulness inferred : false
Is Ida's dishonesty inferred : true
{Hal=false, Kay=true, Eli=false, Jay=true, Bob=false, Dee=false, Ida=false, Gil=false, Amy=false, Lee=false, Cal=false, Fay=false}
Is Bob's truthfulness inferred : false
Is Bob's dishonesty inferred : true
{Hal=false, Kay=true, Eli=false, Jay=true, Bob=false, Dee=false, Ida=false, Gil=false, Amy=false, Lee=false, Cal=false, Fay=false}
Is Lee's truthfulness inferred : false
Is Lee's dishonesty inferred : true
{Hal=false, Kay=true, Eli=false, Jay=true, Bob=false, Dee=false, Ida=false, Gil=false, Amy=false, Lee=false, Cal=false, Fay=false}
Is Cal's truthfulness inferred : false
Is Cal's dishonesty inferred : true
{Hal=false, Kay=true, Eli=false, Jay=true, Bob=false, Dee=false, Ida=false, Gil=false, Amy=false, Lee=false, Cal=false, Fay=false}
Is Gil's truthfulness inferred : false
Is Gil's dishonesty inferred : true
{Hal=false, Kay=true, Eli=false, Jay=true, Bob=false, Dee=false, Ida=false, Gil=false, Amy=false, Lee=false, Cal=false, Fay=false}
Is Dee's truthfulness inferred : false
Is Dee's dishonesty inferred : true
{Hal=false, Kay=true, Eli=false, Jay=true, Bob=false, Dee=false, Ida=false, Gil=false, Amy=false, Lee=false, Cal=false, Fay=false}
Is Eli's truthfulness inferred : false
Is Eli's dishonesty inferred : true
{Hal=false, Kay=true, Eli=false, Jay=true, Bob=false, Dee=false, Ida=false, Gil=false, Amy=false, Lee=false, Cal=false, Fay=false}
Is Fay's truthfulness inferred : false
Is Fay's dishonesty inferred : true
Is Jay's truthfulness inferred : true
Is Kay's truthfulness inferred : true


Number of recursive calls taken to generate result using Enumeration: 154622
Time taken for solving using Basic Model Checking (in ms): 170


--------------------------------------
Solving Problem using DPLL Inference Method
--------------------------------------
Is Amy's truthfulness inferred : false
Is Amy's dishonesty inferred : true
Is Hal's truthfulness inferred : false
Is Hal's dishonesty inferred : true
Is Ida's truthfulness inferred : false
Is Ida's dishonesty inferred : true
Is Bob's truthfulness inferred : false
Is Bob's dishonesty inferred : true
Is Lee's truthfulness inferred : false
Is Lee's dishonesty inferred : true
Is Cal's truthfulness inferred : false
Is Cal's dishonesty inferred : true
Is Gil's truthfulness inferred : false
Is Gil's dishonesty inferred : true
Is Dee's truthfulness inferred : false
Is Dee's dishonesty inferred : true
Is Eli's truthfulness inferred : false
Is Eli's dishonesty inferred : true
Is Fay's truthfulness inferred : false
Is Fay's dishonesty inferred : true
Is Jay's truthfulness inferred : true
Is Kay's truthfulness inferred : true


Number of recursive calls taken to generate result using DPLL: 1394
Time taken for solving using Advanced Propositional Inference (in ms): 25
```

Figure 8: Output of executing inference for More Liars And Truth Tellers problem



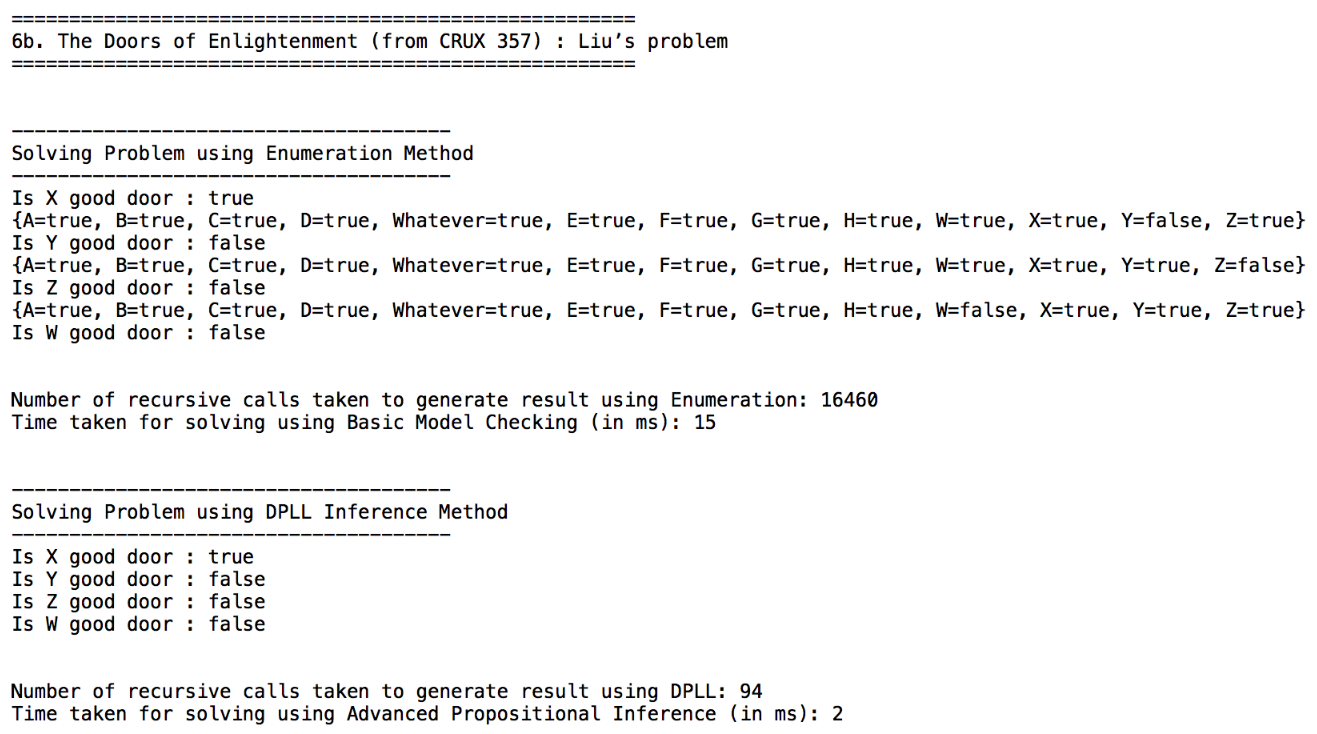
```
=======================================================
6a. The Doors of Enlightenment (from CRUX 357) : Smullyan's problem
=======================================================


--------------------------------------
Solving Problem using Enumeration Method
--------------------------------------
Is X good door : true
{A=true, B=true, C=true, D=false, E=true, F=true, G=true, W=true, H=true, X=true, Y=false, Z=true}
Is Y good door : false
{A=true, B=true, C=true, D=true, E=false, F=true, G=true, W=true, H=true, X=true, Y=true, Z=false}
Is Z good door : false
{A=true, B=true, C=true, D=true, E=true, F=true, G=true, W=false, H=true, X=true, Y=true, Z=true}
Is W good door : false


Number of recursive calls taken to generate result using Enumeration: 9031
Time taken for solving using Basic Model Checking (in ms): 9


--------------------------------------
Solving Problem using DPLL Inference Method
--------------------------------------
Is X good door : true
Is Y good door : false
Is Z good door : false
Is W good door : false


Number of recursive calls taken to generate result using DPLL: 73
Time taken for solving using Advanced Propositional Inference (in ms): 2
```

Figure 9: Output of executing inference for Doors of enlightenment - Smullyan's problem



```
=====================================================
6b. The Doors of Enlightenment (from CRUX 357) : Liu’s problem
=====================================================


--------------------------------------
Solving Problem using Enumeration Method
--------------------------------------
Is X good door : true
{A=true, B=true, C=true, D=true, Whatever=false, E=true, F=true, G=false, H=true, W=true, X=true, Y=false, Z=true}
Is Y good door : false
{A=true, B=true, C=true, D=true, Whatever=false, E=true, F=true, G=false, H=true, W=true, X=true, Y=true, Z=false}
Is Z good door : false
{A=true, B=true, C=true, D=true, Whatever=false, E=true, F=true, G=false, H=true, W=false, X=true, Y=true, Z=true}
Is W good door : false


Number of recursive calls taken to generate result using Enumeration: 18182
Time taken for solving using Basic Model Checking (in ms): 16


--------------------------------------
Solving Problem using DPLL Inference Method
--------------------------------------
Is X good door : true
Is Y good door : false
Is Z good door : false
Is W good door : false


Number of recursive calls taken to generate result using DPLL: 94
Time taken for solving using Advanced Propositional Inference (in ms): 2
```

Figure 10: Inference for Doors of enlightenment - Liu’s problem (Whatever=False)



```
=====================================================
6b. The Doors of Enlightenment (from CRUX 357) : Liu’s problem
=====================================================


--------------------------------------
Solving Problem using Enumeration Method
--------------------------------------
Is X good door : true
{A=true, B=true, C=true, D=true, Whatever=true, E=true, F=true, G=true, H=true, W=true, X=true, Y=false, Z=true}
Is Y good door : false
{A=true, B=true, C=true, D=true, Whatever=true, E=true, F=true, G=true, H=true, W=true, X=true, Y=true, Z=false}
Is Z good door : false
{A=true, B=true, C=true, D=true, Whatever=true, E=true, F=true, G=true, H=true, W=false, X=true, Y=true, Z=true}
Is W good door : false


Number of recursive calls taken to generate result using Enumeration: 16460
Time taken for solving using Basic Model Checking (in ms): 15


--------------------------------------
Solving Problem using DPLL Inference Method
--------------------------------------
Is X good door : true
Is Y good door : false
Is Z good door : false
Is W good door : false


Number of recursive calls taken to generate result using DPLL: 94
Time taken for solving using Advanced Propositional Inference (in ms): 2
```

Figure 11: Inference for Doors of enlightenment - Liu’s problem (Whatever = True)

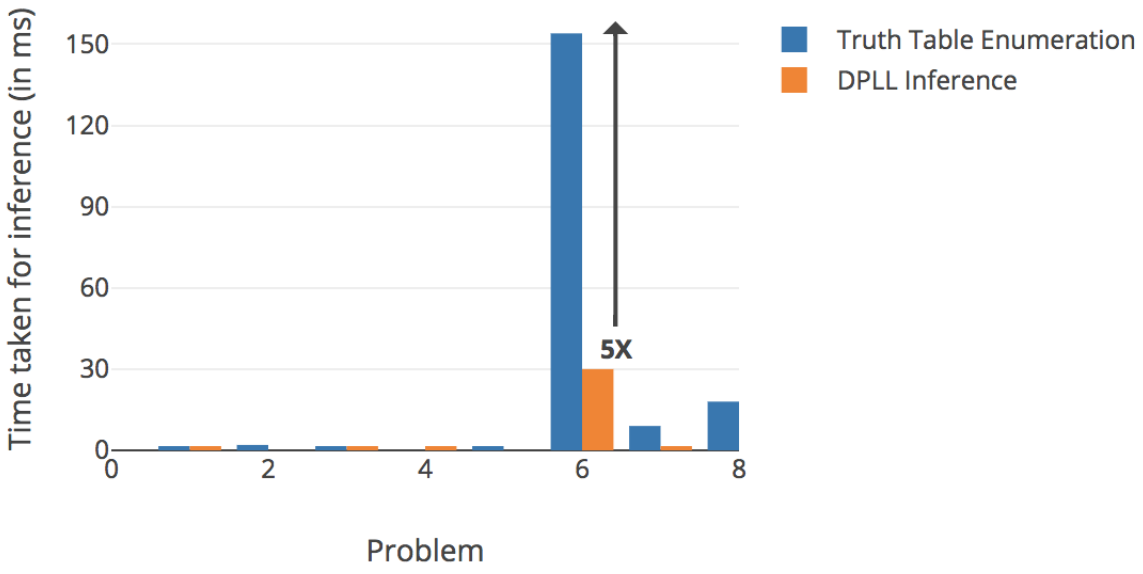


Figure 12: Effect of using DPLL vs Enumeration on time

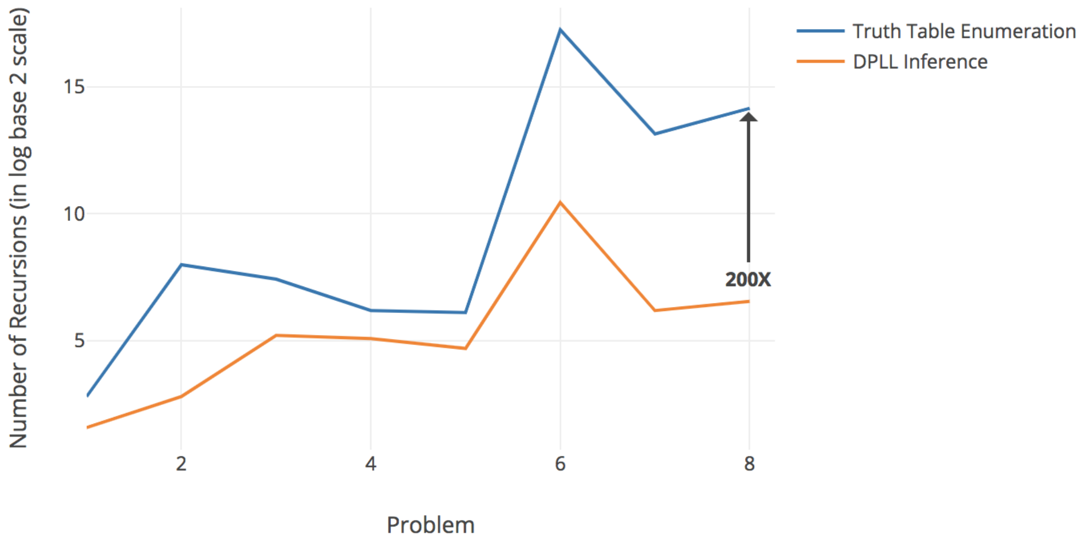


Figure 13: Effect of using DPLL vs Enumeration on number of recursion calls