Project 2 Automated Reasoning

1. Preparation for CNF

Note: for convenience I use “|” represent “or”, “&” represent “and”, “>”and”<” represent “entail” and “=” represent “equal”

Problem 1: **{p, p>q}** to prove **q**. in this case I transform **p>q** into **-p|q**, after that the CNF format is **["p","-p|q"]**, then regard it as knowledge base then try to prove q.

Problem 2: **{-p11, b11=p12|p21, b21=p11|p22|p31, -b11, b21}** to prove **p12**.

First:

b11=p12|p21 transform to b11>p12|p21 and b11<p12|p21,

b21=p11|p22|p31 equal to b21>p11|p22|p31 and b21<p11|p22|p31

then:

transform to -b11|p12|p21 and b11|-p12 and -p21|b11 and -b21|p11|p22|p31

in this case I transform them into **["-p11","-b11|p12|p21","-b21|p11|p22|p31","-b11","b21","b11|-p12","b11|-p21"]**, then regard it as knowledge base then try to prove p12.

Problem 3: first to do is figure out how to transform the sentence into logic sentence. I use the abbreviation to represent the variable, “my”-“mythical”,”mo”-“motal”,”ma”-“mammal”,”h”-hourned”,”mg”-“magical”

As descript logic sentence should be “my>-mo”,”-my>mo&ma”,”-mo|ma>h”,”h>mg”

then:

"-my|-mo","my|mo","my|ma","mo|h","-ma|h","-h|mg"

then regard it as knowledge base then try to prove "my","mg","h", if get “my is truth” means “my” is truth. Otherwise “my is false” means my cannot be sure.

Problem 4: first to do is figure out how to transform the sentence into logic sentence. I use the abbreviation to represent the variable, “a”-“amy”,”b”-“bob”,”c”-“cal”

As descript for (a) logic sentence should be “a = c&a”,”b=-c”,”c=b&-a”

then:

can get "b|c","-b|-c","-a","-c|b|-a","c|-b","a|c"

As descript for (b) logic sentence should be “a = -c”,”b=a&c”,”c=b”

then:

can get "b|-c","-b|c","-a|b|-c","-b|a","-c|-a","a|c"

then regard it as knowledge base then try to prove "a","b","c", if get “a is truth” means “a” tell truth. Otherwise “a is false” means “a” tell lies.

Problem 5: first to do is figure out how to transform the sentence into logic sentence. I use the abbreviation to represent the variable, “a”-“amy”,”b”-“bob”,”c”-“cal”,”d”-“dee”,”e”-

“eli”,”f”-“fay”,”g”-“gil”,”h”-“hal”,”I”-“ida”,”j”-“jay”,”k”-“kay”,”l”-“lee”

As descript logic sentence should be “a= h&i”,”b=a&l”,”c=b&g”,”d=e&l”,”e=c&h”,”f=d&i”,”g=-e&-j”,”h=-f&-k”,”i=-g&-k”,”j=-a&-c”, ”k=-d&-f”, ”l=-b&-j”

then:

can get "b|-a|-l","-b|a","-b|l","c|-b|-g","-c|b","-c|g","d|-e|-l","-d|e","-d|l","f|-d|-i","-f|d","-f|i","e|-c|-h","-e|c","-e|h","g|e|j","-g|-j","-g|-e","h|f|k","-h|-f","-h|-k","i|g|k","-i|-g","-i|-k","j|a|c", "-j|-a", "-j|-c","k|d|f","-k|-d","-k|-f","l|b|j","-l|-b","-l|-j"

then regard it as knowledge base then try to prove "a","b","c","d","e","f","g","h","i","j","k","l", if get “a is truth” means “a” tell truth. Otherwise “a is false” means “a” tell lies.

Problem 6: first to do is figure out how to transform the sentence into logic sentence. As descript for (a) logic sentence should be “a =x”,”b=y|z”,”c=b&a”, ”d=x&y”, ”e=x&z”, ”f=d&e”,”g=c>f”, ”h=g&h>a”

then:

can get "a|-x","-a|x","b|-y","b|-z","-b|y|z","c|-a|-b","-c|a","-c|b","d|-x|-y","-d|x","-d|y","e|-x|-z","-e|x","-e|z","-f|d|e","f|-d","f|-e","g|-f","g|c","-g|-c|f","h|-g","-g|a","x|y|z|w"

As descript for (b) logic sentence should be “a =x”,”c=?&a”,”g=c>??”, ”h=g&h>a” (use ? and ?? represent some variable)

then:

can get "a|-x","-a|x","h|-g","-g|a","g|-??","g|c","-g|-c|??","c|-a|-?","-c|a","-c|?","x|y|z|w"

then regard it as knowledge base then try to prove "x","y","z","w", if get “x is truth” means “x” is good door. Otherwise “x is false” means “x” is bad door or “x” cannot be sure.

1. Test

In this algorithm, it functions like main, and all the interaction happens here. It defines the knowledge base and variable needed to be proved.

2. KB class

1. Model checking

For initial function, It is used to design a structure of knowledge base. In this class, at first it will initial some variables. There will be a kb which includes all the original CNF clauses. Then use the table to save all variables appearing in the CNF.

There is only one difference between two implements in the kb class, which is add a unprove in the DPLL algorithm and it is used to save unproved clauses.

For the tell function, it will parser all the input sentence and then add them into knowledge base and table.

For ask function, it takes different strategies to realize reasoning. In problem 1-6(mc), it uses model checking implement. And use a binary number to represent all the model. For example, from “1111” to “0000”, so if the model satisfies all the sentence in kb, it will check whether the unproved variable is satisfied and terminal will display message.

For check function, it is used to check whether the model is fine. It will check all the clauses in kb, if one of them are not satisfied, the kb will be false then the model is abandoned.

For parser function, it transforms the CNF into pure variable and variable may contained negation. Then add them into kb or table.

2. DPLL

For the strategy, it will create a tree and every node in the tree represent a variable and their left node is assigned a 0(false) for the variable and right node is assigned 1(true) for the variable. In every node, there will be an unprove variable to represent the sentences in kb which are not proved yet. The bool is a string which represent the model like”0011”. If every sentence in kb has been proved the flag value will be 1.
In this algorithm, the tell function and parser function are the same. There are the differences:

1. In initial function, it adds a model variable to save the satisfied model
2. There is an additional node class called dpll_tree which used to create tree node.
3. In ask function, it will directly create the whole tree. After creating, all the possible model will be in model variable.
4. The findclause function is used for find the clauses can be prove after assigning a new value to variables. The input unprove represents the node’s parents unproved kb. The value is assignment (0 or 1). Var represent the variable assigned. It only check the clauses including the var.
5. The createtree function is aimed to recursively create the whole tree. Index represents the place of variable in table, which can used for counting. Temp is unproved clauses and var means the variable string, parent is the parent node, number represent the assignment.
6. After creating the whole tree, all the models have been added into model, then in test, just check if every model possible model can make the unproved variable true, it will get the solution.

3. Result

For model checking:

```
which problem do you want to solve(please type 1-6):
1
which subproblem do you want to solve(please type 1-2):
1
q is truth!!
it is done!!!
```

```
which problem do you want to solve(please type 1-6):
2
which subproblem do you want to solve(please type 1-2)
1
p12 is false!!
it is done!!!
```

```
a is false!!
b is false!!
c is false!!
d is false!!
e is false!!
f is false!!
g is false!!
h is false!!
i is false!!
j is truth!!
k is truth!!
l is false!!
it is done!!!
```

For DPLL:

```
which problem do you want to solve(please type 1-6):
1
which subproblem do you want to solve(please type 1-2):
1
q is truth!!
```

```
which problem do you want to solve(please type 1-6):
2
which subproblem do you want to solve(please type 1-2):
1
p12 is false!!
```

```
a is false!!
b is false!!
c is false!!
d is false!!
e is false!!
f is false!!
g is false!!
h is false!!
i is false!!
j is truth!!
k is truth!!
l is false!!
```

Note

At the end of kb file, I write many functions which try to transform a normal clauses into CNF. The revised function is to do this. For revised_negaton function, revise_dbbrakets and so on, they are used to parser the “&”,”=”,”<>”. Final two functions-cnf and distr are most important functions. They can recursively parser the whole sentence like “a|(b&c)”,but in the end I think it is too complicated for me, I failed in handling the brackets. I cant resolve the extra brackets and will display a wrong sentence.