Project 2: Automated Reasoning

1. Key data structures

(a) Formula

Formulas are represented as binary trees. The basic formula tree node is designed as below. Class Formula represents a literal and is also a super class of conjunction, disjunction, implication and biconditional.

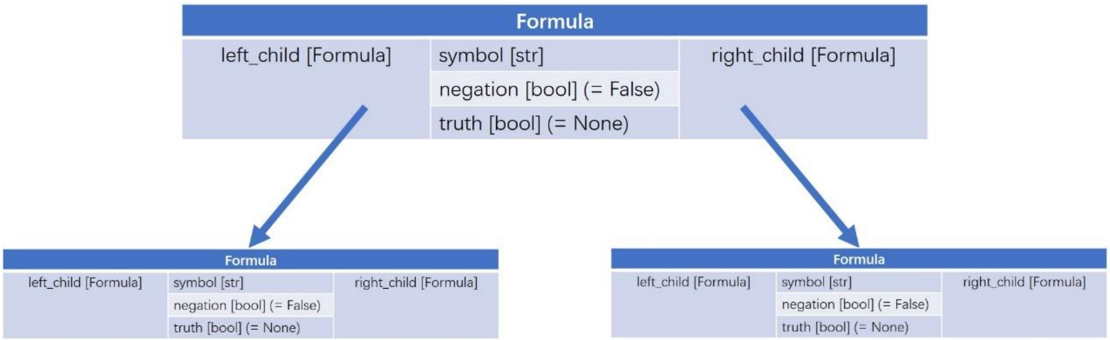

Figure 1: Formula Structure

The element ‘symbol’ stores a string of proposition symbol. The element ‘negation’ indicates whether it is a negation of an atomic formula. For example, symbol = ‘P’, negation = True refers to ‘¬P’. The element ‘truth’ stores the truth value of the formula that a node represents.

Formula	
is_leaf()	Return True if it is a leaf node in a tree
get_leaf_symbols(sym_list)	Populate a list with symbols(str) stored in the leaf nodes of a tree
assign(model)	Assign truth value to each literal according to a given model
models()	Return the truth value of the formula that a node represents

Figure 2: Main Methods in Formula

We also rewrite __str__ in the Formula class. The output of print() will be “[SYM], [NEG]”.

Class Conjunction, Disjunction, Implication and Biconditional inherit class Formula and override method models() in class Formula to implement different propositional logic operations according to the logical connectives. The symbol of instance of class Conjunction, Disjunction, Implication and Biconditional is automatically set as ‘^’, ‘V’, ‘=>’ and ‘<=>’ respectively.

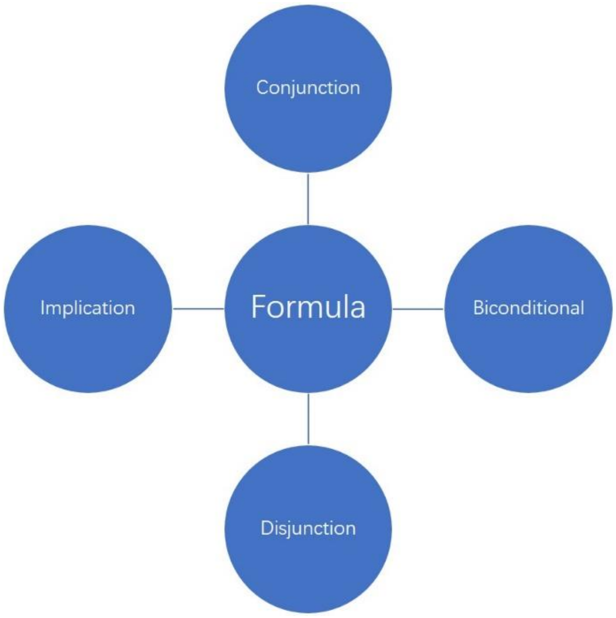

Figure 3: Class Hierarchy

(b) Model
We use dictionary in Python to represent models. A collection of pairs of [proposition symbol: truth value] represents a possible model.

2. Performance
(1) Modus Ponens
Model-Checking and Resolution algorithm both return True.

```
Modus Ponens test
Knowledge base:
 P=>Q
 P
Query: Q
Ans with model checking: True
Ans with resolution: True
```

Figure 4: Results of Modus Ponens Test

(2) Wumpus World
Model-Checking and Resolution algorithm both return False.

```
Wumpus World test
Knowledge base:
 !P(1,1) ^ !B(1,1) ^ B(2,1)
 B(1,1) <=> (P(1,2) V P(2,1))
 B(2,1) <=> (P(1,1) V P(2,2) V P(3,1))
Query: P(1,2)

Ans with model checking: False
Ans with resolution: False
```

Figure 5: Results of Wumpus World Test

(3) Horn Clauses

Both methods return “MAYBE” for question (a), True for (b) and True for (c).

```
Horn Clauses test.
Knowledge base:
(MYTHICAL => IMMORTAL) If the unicorn is mythical, then it is immortal;
(!MYTHICAL => MAMMAL) but if it is not mythical, then it is a mortal mammal;
((IMMORTAL V MAMMAL) => HORNED) If the unicorn is either immortal or a mammal, then it is horned;
(HORNED => MAGICAL) The unicorn is magical if it is horned.
Query: MYTHICAL, MAGICAL, HORNED

(1) Can we prove that the unicorn is mythical?
 -Ans with model checking MAYBE
 -Ans with resolution MAYBE

(2) Can we prove that the unicorn is mythical?
 -Ans with model checking True
 -Ans with resolution True

(3) Can we prove that the unicorn is mythical?
 -Ans with model checking True
 -Ans with resolution True
```

Figure 6: Results of Horn Clauses Test

(4) The Doors of Enlightenment

For this question, resolution algorithm can only get the answer for (a) when query is X with a certain time. For other situations with different queries, the set of resolvents keeps growing and cannot get a result.

The results of model checking are {True, Maybe, Maybe, Maybe} for (a), {True, Maybe, Maybe, Maybe} for (b).

```
The Doors of Enlightenment test.

(a) Smullyan's problem:
Knowledge base:
A: X is a good door. (A <=> X)
B: At least one of the doors Y or Z is good. (B <=> (Y V Z))
C: A and B are both knights. (C <=> (A ^ B))
D: X and Y are both good doors. (D <=> (X ^ Y))
E: X and Z are both good doors. (E <=> (X ^ Z))
F: Either D or E is a knight. (F <=> (D V E))
G: If C is a knight, so is F. (G <=> (C => F))
H: If G and I (meaning H) are knights, so is A. (H <=> ((G ^ H) => A))
Query: X, Y, Z, W
Query: X
-Ans with model checking: True
-Ans with resolution: True

Query: Y
-Ans with model checking: MAYBE
-Ans with resolution: Run too long. Cannot get answer

Query: Z
-Ans with model checking: MAYBE
-Ans with resolution: Run too long. Cannot get answer

Query: W
-Ans with model checking: MAYBE
-Ans with resolution: Run too long. Cannot get answer
```

```
(b) Liu's problem:
Knowledge base:
A: X is a good door. (A <=> X)
H: If G and I (meaning H) are knights, so is A. (H <=> ((G ^ H) => A))
C: A and ... are both knights. (C <=> (A ^ ?))
G: If C is a knight, then ... (G <=> (C => ?))
Query: X, Y, Z, W
Query: X
-Ans with model checking: True
-Ans with resolution: Run too long. Cannot get answer

Query: Y
-Ans with model checking: MAYBE
-Ans with resolution: Run too long. Cannot get answer

Query: Z
-Ans with model checking: MAYBE
-Ans with resolution: Run too long. Cannot get answer

Query: W
-Ans with model checking: MAYBE
 -Ans with resolution: Run too long. Cannot get answer
```

Figure 7: Results of The Door of Enlightment Test

3. Member contributions

Jiapeng Li:

(1) Design a binary tree to represent well-formed formulas;

(2) Implement a model checking theorem prover using truth-table enumeration method;

(3) Apply the model checking theorem prover to four selected problems and test the performance of the theorem prover.

Zeyi Pan:
Implement resolution algorithm in theorem proving which uses proof by reputation, gets correct resolvents, achieves factoring and gets the answer for entailment for most questions.