CSCE625: Artificial Intelligence

Final Exam

Minjie Lu, UIN: 922003096

Option1: First-Order Logic Engine

Description About The System

A. Environment:

Python 2.7.6
PLY-3.8 is installed.

B. Structure:

folresolution.py is the main program to run.
SentenceParser.py and AtomicSentenceParser.py are auxiliary programs.

C. Format of input for knowledge base and query:

1. Variables are represented by single or combination of lowercase letters. Constants are represented by single of uppercase letters. Predicates and functions must be a mixture of uppercase and lowercase letters. For example:

Variables: x, y, z...
Constants: WEST, NONO...
Functions: Missile, Weapon...

2. Universal quantifiers are represented by 'ForAll'. Existential quantifier are represented by 'Exist'. Variables mentioned by these quantifier are enclosed by braces. For example:

ForAll{x,y,z}
Exist{u,v}

3. Logic connectives are represented as the list below:

Not: not(Criminal(WEST))
And: &
Or: |
Implies: =>
If and only if: => **(This connective has not been included in my program yet.)**

4. Following are some examples for predicate:

American(x)
Sells(WEST,x,NONO)

5. Following are some examples for functions:

Father(x)
Parent(WEST) **(I haven't tried examples with functions yet)**
'Fa', 'Ga', 'Ha', 'Ia', 'Ja', 'Ka' are reserved for Skolem functions

6. Grouped elements should be enclosed by brackets. For example:

ForAll{x} [ForAll{y} Animal(y) => Loves(x,y)] => [Exist{y} Loves(y,x)]

7. There can't be space within a single atomic sentence. For example:

Wrong: Loves(x, y) Right: Loves(x,y)
Wrong: not(Loves(x,y)) Right: not(Loves(x,y))

8. There should be ENTER between sentences. Then a complete Knowledge base(KB) should be like the following example:

ForAll{x,y,z} American(x) & Weapon(y) & Sells(x,y,z) & Hostile(z) => Criminal(x)
ForAll{x} Missile(x) & Owns(NONO,x) => Sells(WEST,x,NONO)
ForAll{x} Enemy(x,AMERICAN) => Hostile(x)
ForAll{x} Missile(x) => Weapon(x)
Owns(NONO,MI)
American(WEST)
Missile(MI)
Enemy(NONO,AMERICAN)

Query can be like:

Criminal(WEST)

D. Operations:

You can choose to apply a complete or incomplete resolution.
For complete resolution, the first iteration of resolution will be carried out among all the clauses of the KB themselves and between these clauses and the negated query. For incomplete resolution, the first iteration will be only carried out between the clauses in KB and the negated query.

I prepared two KBs. One in the file 'Knowledge_base.txt'. The other in the file 'KB_Enemy.txt'. Now the program is referred to 'KB_Enemy.txt' which is the first example on the text book. **It takes a long time to do the resolution even if you choose incomplete resolution.** But after 9 steps, it will tell 'Criminal(WEST)' is true which is the same as shown on the textbook.
To change the KB. The file name in line 404 of 'folresolution.py' should be changed to 'Knowledge_base'. This is a simple KB and queries like 'Likes(PROFESSOR,WEST)' and 'not(Likes(PROFESSOR,WEST))' can be asked.

E. Features and limitations of my program:

Features:

1. Conversion of a sentence to CNF is implemented according to the procedures in the textbook. Elimate implications => Move not inwards => Standardize variables => Skolemization => Drop Universal quantifiers => Distribute or over and [1, p.346-347]
These procedures are done in a tree for which leaves are complete atomic sentences.
2. For resolution, unification and factoring are both implemented according to rules. When unification happened, variables will be standardized between sentences.[1, p.347]
These procedures are done in a tree for which leaves are variables and constants.
3. If a query is directly against KB, a 'False' will be returned directly.

Limitations:

1. Since I didn't planned carefully at first, I have to import the PLY library twice in my program. So I can't make an iteration to constantly ask queries during a single run. Further improvement can include a modify of this disadvantage.
2. I haven't tried to include 'Equality' and 'If and only if' in my program yet.
3. I should try to implement resolution strategies such as unit preference, set of support, input resolution and subsumption to increase efficiency. [1, p.355-356] Otherwise, we can calculate resolutions happen in each iteration is in the order of O(n^k), where n is the size of the KB and k actually has a Fibonacci relation to iteration times. This is calculated by supposing each pair of old clause and newly generated clause can result in a resolution. Among these strategies, subsumption is very necessary and clear to be implemented.
What's more, it might be possible to program resolution problems as search problems

References

Stuart J. Russell and Peter Norvig. *Artificial Intelligence*: A Modern Approach. Third Edition. Upper Saddle River, NJ, U.S.A.: Prentice-Hall, Inc., 2008.