Middle East Technical University — Department of Computer Engineering

CENG462

Artificial Intelligence

Spring 2022-2023

Midterm

Due: May 16th, 2023, 16:00 (4 P.M.)

first order predicate logic; inference; theorem proving; resolution refutation; set of support

Problem Definition

In this exam, you are going to implement a function called `theorem_prover` in python as a theorem prover for First Order Predicate Logic by using *Resolution Refutation* technique and *Set of Support* strategy with *Breadth-First* order. This function gets two lists of clauses, namely the list of base clauses and the list of clauses obtained from the negation of the theorem.

Your program has to eliminate

- tautologies
- subsumptions

The function detects whether the theorem is derivable or not. Then, it returns the result as a tuple. First element of this tuple will be **"yes"** or **"no"** according to derivability of the theorem. The second element will be:

- **list of resolutions** which contribute to the proof of the theorem, if it is derivable.
- **an empty list**, if the theorem is not derivable.

Specifications

- By the convention we follow in FOPL; variables, predicate and function names start with with a lower case letter, while the constants with an upper case letter.
- In the given clauses; "+" and "∼" signs will be used for disjunction and negation respectively.
- Each resolution in the return value must be given in "$< clause1 > \$ < clause2 > \$ < resolvent >$" form **without using any space character**.
- The "empty" string must be used for empty clause in the return value.

Sample Function Calls

```
>>> theorem_prover(["p(A,f(t))", "q(z)+~p(z,f(B))", "~q(y)+r(y)"],["~r(A)"])
('yes', ['~r(A)$~q(y)+r(y)$~q(A)', '~q(A)$q(z)+~p(z,f(B))$~p(A,f(B))', '~p(A,f(B))$p(A,f(t))$empty'])

>>> theorem_prover(["p(A,f(t))", "q(z)+~p(z,f(B))", "q(y)+r(y)"],["~r(A)"])
('no', [])
```

Regulations

1. **Deadline:** The deadline for this homework is strict and no late submissions will be accepted. Submission deadlines are **not** subject to postponement.
2. **We have zero tolerance policy for cheating.** People involved in cheating (any kind of code sharing or codes taken from internet included) will be punished according to the university regulations.
3. **Programming Language:** Your code should be written in Python3. Your submission will be tested in inek machines. So make sure that it can be executed on them as follows.

```
>>> import e1234567_mt
>>> e1234567_mt.theorem_prover(...)
```

4. **Implementation:** You have to write your program by only using the functions in standard module of python. Namely, you **cannot** import any module in your program.
5. **Evaluation:** Your program will be evaluated automatically using "black-box" technique so make sure to obey the specifications. A reasonable timeout will be applied according to the complexity of test cases. This is not about the code efficiency, its only purpose is avoiding infinite loops due to an erroneous code.
6. **Submission:** Submission will be done via OdtuClass. You should upload a **single** python file named `<your_student_id>_mt.py`, e.g., `e1234567_mt.py`.