CENG 424 - Logic for Computer Sciences
Fall 2018

Middle East Technical University
Uluç Saranlı

Homework 5 - Propositional Theorem Proving

Assigned - Dec 30, 2018, **Due - Dec 6, 2018**

1 Certifying Theorem Prover for Propositional Logic

We have gone over a simple, incomplete theorem prover for propositional logic during class hours. In this homework, you are expected to extend this theorem prover to construct and return a proof term during proof search (i.e. "certify" the proof). To this end, you are expected to implement the following predicate goal in prolog:

```
provecert_e1234567(A, M)
```

which, provided a propositional formula `A`, is expected to unify `M` with the associated proof term if `A` is provable. This goal should fail if the provided formula is not provable. For example, the following is an example of expected behavior:

```
?- provec((a&b=>c)=>a=>b=>c, M).
M = fn(u0, fn(u1, fn(u2, app(u0, pair(u1, u2))))) .
```

You should submit your solution within a file `provecert_e1234567.prolog` (with your own student number), separate from the `pl_prover.prolog` file provided with the homework. Make sure to include **generous** documentation in your prolog code, explaining how and why your implementation works.

2 Using your Automated Theorem Prover

Use your theorem prover implementation from part 1 above to provide proof terms for the following propositions:

(a) $p \supset (B \supset C) \supset ((p \supset B) \supset C)$

(b) $((A_1 \supset (A_2 \supset B)) \supset C) \supset (((A_1 \wedge A_2) \supset B) \supset C)$

(c) $(B \supset C) \supset ((\top \supset B) \supset C)$

(d) $((A_1 \supset B) \supset ((A_2 \supset B) \supset C)) \supset (((A_1 \vee A_2) \supset B) \supset C)$

(e) $C \supset ((\bot \supset B) \supset C)$

(f) $((A_2 \supset B) \supset (A_1 \supset A_2)) \supset (B \supset C) \supset (((A_1 \supset A_2) \supset B) \supset C)$

Note that these propositions correspond to sequents from the second part of your previous homework and the associated proof terms can be used in implementing the constraction-free theorem prover.

Make sure to check your proof terms with `check(A, M)` before including them in your report. We will also be checking that these proof terms are indeed generated by your own theorem prover. Submit proof terms for these propositions in a file `e1234567_hw5_q2.txt`.

3 Bonus: Contraction-free, Complete Theorem Prover

Implement a certifying theorem prover based on the contraction free rules listed in Figure 1 of the lecture notes on theorem proving posted on the course site (Section 4). Your implementation should provide a goal in the following form

```
g4ipc_e1234567(A, M)
```

Include as many negative examples in your prover as possible (i.e. unprovable propositions) to test out your implementation. Note, also, that you will need to find/implement a way in which you can identify atomic propositions since some of the rules in the new calculus require restricting them certain components to be atomic propositions. You should submit your solution within a file `g4ipc_e1234567.prolog`.

Submission

Your submission should include a ZIP file `hw5_e1234567.zip` including your submission files for the above questions. Late submissions will be penalized with $10n^2$ points where n is the number of late days, rounded up.