Report Assignment #3: Theorem Prover

Authors: Simon Bilgeri (422852), Rui Lopes (31689) – Group 29

Python Version: 3.4.2

Code usage: python prove_theorem.py <path to input file>, default: test4.txt

In a first step, the sentences are read from an input file and converted to CNF using the prop2cnf tool developed in the second assignment. The last sentence read is the sentence to prove and thus it is negated before conversion. In case, the sentences are already provided in CNF, the tool detects it and assumes that the sentence to prove is already negated and provided with the set of clauses.

After conversion, resolution is used to introduce new clauses and to find a contradiction (empty clause). To do so, an extended unit preference strategy is implemented. Not only the single literal clauses are considered first, but the clauses are ordered by their number of literals (from small to large). In that way, smaller clauses are resolved first, which are more likely to lead to an empty clause. Every time a new clause is added, the clauses are sorted again and the algorithm restarts the resolving.

A clause is only added if it is not already in the knowledge base (KB). Furthermore, simplifications like factoring are used. That means that (B|B) is simplified to (B). Additionally, a clause containing a tautology is not added to the KB. As the simplifications are similar to the ones for the CNF conversion, the same function is used.

Like in the second assignment, for better readability, the function *show_nice_format(sentence)* converts the input format in a more convenient way.

In the menu, it is possible to show all input sentences in propositional logic or already converted to CNF. With the third menu entry, the proof and all intermediate steps and clauses are provided. All the clauses and the result is written to the file "Proof.txt".

In order to test the theorem prover, the test sets "test1-5.txt" are applied.

- test1.txt: Basic check if the prover can infer an implication, and also if the converting works
- test2.txt: Check if the algorithm gets rid of duplicate literals and whether it stops if no empty clause can be derived
- test3.txt: Check if a tautology is detected and not added to the set of clauses
- test4.txt: Known example of the unicorn, which is proved to be magical. Here the unit clause preference is shown. Before resolving (c0,c2), which would lead to a large clause, the single literal (c6) with (c5) is used, which yields another single literal. This speeds up the resolution process.
- test5.txt: This last test shows, that it is not possible to prove that the unicorn is immortal, as no more clauses can be generated.

All in all, it can be stated, that it is possible to check if a sentence in propositional logic is inferred by a KB in CNF with the developed theorem prover. Nevertheless, in practice, e.g. actual robots, propositional logic is too static. First order logic is more flexible and can be used to distinguish different situations and actions. Therefore, the algorithm should be extended to first order logic, before using it for an actual application.