Introduction to Artificial Intelligence

Exercise Sheet 6

Handed out: 25.01.2021 — Prof. Dr. Susanne Biundo-Stephan
Due: 08.02.2021 — Conny Olz

Exercise 1 (2.5 points)

Consider the proposition symbols A, B, C, D and E. Which of the following are correct?

a) $\mathit{False} \models \mathit{True}$

b) $\mathit{True} \models \mathit{False}$

c) $(A \wedge B) \models (A \Leftrightarrow B)$

d) $(A \Leftrightarrow B) \models A \vee B$

e) $(A \Leftrightarrow B) \models \neg A \vee B$

f) $(A \vee B) \wedge (\neg C \vee \neg D \vee E) \models (A \vee B \vee C) \wedge (B \wedge C \wedge D \Rightarrow E)$

g) $(A \vee B) \wedge (\neg C \vee \neg D \vee E) \models (A \vee B) \wedge (\neg D \vee E)$

h) $(A \vee B) \wedge \neg(A \Rightarrow B)$ is satisfiable

i) $(A \Leftrightarrow B) \wedge (\neg A \vee B)$ is satisfiable

j) $(C \vee (\neg A \wedge \neg B)) \equiv ((A \Rightarrow C) \wedge (B \Rightarrow C))$

Hand in your results in Moodle. As usual, please submit only **one** solution per group.

Exercise 2 (8 points)

Implement a program that generates resolution proofs for propositional logic. Slide 51 of chapter "Propositional logic" describes such a resolution prover. PL-Resolution′ (given in Algorithm 1) corresponds to the algorithm PL-Resolution from the lecture, but makes use of the separately callable submethod IsInconsistent (shown in Algorithm 2).

Algorithmus 1: PL-Resolution′(KB, α)

Input : KB, the knowledge base; a sentence in propositional logic
α, the query; a sentence in propositional logic
Output: *true* if $KB \models \alpha$, *false*, otherwise

```
1 clauses_KB ← the set of clauses representing the KB in CNF;
2 clauses_¬α ← the set of clauses representing ¬α in CNF;
3 return IsInconsistent(clauses_KB ∪ clauses_¬α)
```

Implement the program IsInconsistent (only Algorithm 2), i.e. you don't have to take care of the conversion to clause form and the negation of the query. Your program receives a set of clauses from the standard input.

Introduction to Artificial Intelligence

Exercise Sheet 6

Handed out: 25.01.2021 — Prof. Dr. Susanne Biundo-Stephan
Due: 08.02.2021 — Conny Olz

Algorithmus 2: IsInconsistent($clauses_1, clauses_2$)

Input : $clauses_{all}$, a set of clauses in CNF

```
1 clauses_new ← ∅;
2 while true do
3     for each C_i, C_j ∈ clauses_all do
4         resolvents ← PL-Resolve(C_i, C_j) ;        // computes the resolvents of C_i and C_j
5         if □ ∈ resolvents then return true;
6         clauses_new ← clauses_new ∪ resolvents;
7     if clauses_new ⊆ clauses_all then return false;
8     clauses_all ← clauses_all ∪ clauses_new;
```

The set of clauses is encoded using the DIMACS-format. It begins with an arbitrary number of lines having `c` as the first letter. Lines marked in this way are comments and can be skipped. Next follows a line with the structure `p cnf` $|atom|$ $|clauses|$, where $|atom|$ and $|clauses|$ are numbers specifying the exact number of atoms and clauses that are used in the set of clauses. This is followed by $|clauses|$ lines, each describing a clause. Atoms are represented by the numbers from 1 to $|atom|$, the negation of the atom i by the number $-i$. A clause is a sequence of different numbers in $\{i \mid i \in \mathbb{Z}, -|atom| \leq i \leq |atom|, i \neq 0\}$. Every sequence is terminated by a 0 (which does not belong to the clause).

The following input encodes the set of clauses $\{\{\neg a, b\}, \{a, \neg b, c\}, \{\neg b\}, \{a\}\}$

```
c This is a comment.
c
p cnf 3 4
-1 2 0
1 -2 3 0
-2 0
1 0
```

Like Algorithm 2, your program should output either “true” or “false” to the standard output.