Principles of Knowledge Representation and Reasoning

B. Nebel, S. Wölfl, J.Hué — University of Freiburg
Winter Semester 2012/2013 — Department of Computer Science

Exercise Sheet 2

Due: November 7th, 2012

Exercise 2.1 (Predicate Logic, 2+2+2)

(a) Classify the following expressions as terms, ground terms, atoms, formulae, sentences, or statements in meta language. If there is more than one possibility for an expression please list them all. The usage of symbols complies with the convention introduced with the syntax of predicate logic.

(a) $P(x, a)$
(b) $g(a, b)$
(c) $\mathcal{I} \models P(a, f(b))$
(d) $\mathcal{I}, \alpha \models P(f(x), f(a))$
(e) $g(f(a), y)$
(f) $Q(b)$ is satisfiable.

(g) $\forall x(P(x, y) \rightarrow Q(x)) \vee \neg P(y, x)$
(h) $\forall x \forall y(P(x, y) \wedge Q(x) \vee P(f(y), x))$
(i) $\forall x(\exists y(P(x, y) \wedge Q(x)) \vee P(x, y))$
(j) $Q(a) \vee P(a, b) \equiv P(b, a) \vee Q(b)$

(b) Consider the following theory:

$$\Theta = \left\{ \begin{array}{l} \forall x \neg P(x, x) \\ \forall x \forall y \forall z(P(x, y) \wedge P(y, z) \rightarrow P(x, z)) \\ \forall x \forall y(P(x, y) \vee x = y \vee P(y, x)) \\ \forall x \exists y P(x, y) \\ \neg \exists y P(y, a) \end{array} \right\}$$

Specify an interpretation $\mathcal{I} = \langle \mathcal{D}, \cdot^{\mathcal{I}} \rangle$ with $\mathcal{I} \models \Theta$ (with proof). Does Θ have a model that is defined on a finite domain D?

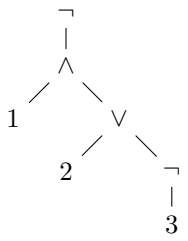
(c) Proof by structural induction over terms and formulae: If ϕ is a formula, $\mathcal{I}$ an interpretation, and α, α' variable maps over $\mathcal{I}$ such that $\alpha(x) = \alpha'(x)$ for each variable x that occurs free in ϕ, then

$$\mathcal{I}, \alpha \models \phi \iff \mathcal{I}, \alpha' \models \phi.$$

Exercise 2.2 (Project: Handling Propositional Formulae, 1+4+1)

The aim of this exiercise is to write a small program that parses propositional logic formulae, translates them to CNF, and decides their satisfiability by using an existing satisfiability solver. You can use any programming language you like

(given that it is usable under Ubuntu 12). Source code **must be submitted** on time to: `westpham@informatik.uni-freiburg.de`.
The input formulae for this project are available on the exercise subpage of the lecture website.[1] We restrict ourselves to postfix notation[2] to keep the formula parsing simple. In these formulae propositional variables are written as (non-zero) positive integer numbers. Only the following propositional connectives are used: `not` (unary), `or` (binary), `and` (binary). After parsing, a formula is internally represented as a binary tree (Figure 1 gives an example) and must be converted to CNF.



Postfix notation: `3 not 2 or 1 and not`

Formula: $\neg\big(1 \wedge (2 \vee \neg 3)\big)$

NNF: $\neg 1 \vee (\neg 2 \wedge 3)$

CNF: $(\neg 1 \vee \neg 2) \wedge (\neg 1 \vee 3)$

Figure 1: Example for postfix notation, formula, CNF, binary tree.

For evaluating CNF formulae you can use an existing propositional satisfiability solver (SAT solver), e.g., the MiniSat solver `http://minisat.se/`. Virtually all SAT solvers accept as input the simple DIMACS format:

```
p cnf 5 2
1 -2 3 0
-1 2 5 4 0
```

The first line specifies that it is a CNF problem (`p cnf`) and gives the number of variables and clauses (in this case 5 variables; atoms $1, \ldots, 5$ and 2 clauses). Each of the following lines specifies one clause: positive integers represent positive literals, negative integers represent negative literals with 0 terminating the clause/line.

(a) Write a parser for the given postfix format that generates a binary tree for a given formula.

(b) Write a function to convert an arbitrary formula to CNF using the binary tree representation.

(c) Write a function to output the CNF in DIMACS format and test the satisfiability of the formulae `formula_1.kb, formula_2.kb, formula_3.kb`. Which of these formulae are satisfiable?

Hint: The `test_cases` folder contains some formulae for testing.

Your program should take as argument one filename to read the formula from and output the CNF in DIMACS.

[1] `http://www.informatik.uni-freiburg.de/~ki/teaching/ws1213/krr/`
[2] `http://en.wikipedia.org/wiki/Postfix_notation`