Technische Universität München
Institut für Informatik
Prof. Tobias Nipkow, Ph.D.
Alexander Krauss, Sascha Böhme

WS 2008/09
28. 11. 2008

Logic

Exercise Sheet 7

Exercise 7.1 Characterizing models with formulas

Find a satisfiable formula p, such that $M \vDash p$ implies

a) $|D_M| \geq 2$

b) $|D_M| \geq 3$

c) D_M is infinite.

Show that any formula that has a model of size n (that is, $|D_M| = n$,) also has a model of size $n + 1$.

Solution

a) $\exists xy.P(x) \wedge \neg P(y)$

b) $\exists xyz.P(x) \wedge \neg P(y) \wedge \neg Q(x) \wedge Q(y) \wedge \neg Q(z)$

c) $(\forall xyz.P(x,y) \wedge P(y,z) \Rightarrow P(x,z)) \wedge (\forall x.\neg P(x,x)) \wedge (\forall x.\exists y.P(x,y))$
The formula expresses that P is a strict partial order (transitive and irreflexive), and that for each element there is a larger one.

If we have a model M of size n, then we can always extend it by choosing an arbitrary element $a \in D_M$ and adding a new element $*$ to the domain. The interpretations of functions and relations are extended in such a way that $*$ is treated just like the existing element a.

More explicitly, we define $D_{M'} = D_M \cup \{*\}$, and define an operation $norm : D_{M'} \to D_M$ with $norm(*) = a$ and $norm(x) = x$ for all $x \in D_M$. Then, $P_{M'}(x_1, \ldots, x_n) = P_M(norm(x_1), \ldots, norm(x_n))$ and $f_{M'}(x_1, \ldots, x_n) = f_M(norm(x_1), \ldots, norm(x_n))$.

This property (the upward Löwenheim-Skolem-Theorem, cf. Thm. 3.50) implies that we cannot express upper bounds on the sizes of the models in a logical formula.

Exercise 7.2 Composition and substitution

The following statement about composition and substitution into terms is not well-formed:

$$\texttt{tsubst}\ i \circ \texttt{tsubst}\ j\ = \texttt{tsubst}\ (i \circ j)$$

What is the problem? (Hint: Consider the types of i and j.)

Correct the statement and then prove it by induction.

Solution

The functions i and j are substitutions and map variables to terms, so their composition is not well-defined since terms are not variables. The correct statement is

$$\texttt{tsubst}\ i \circ \texttt{tsubst}\ j\ = \texttt{tsubst}\ (\texttt{tsubst}\ i \circ j)\,.$$

We prove the equality between functions by showing that they return the same result on every input, i.e.,

$$\texttt{tsubst}\ i\ (\texttt{tsubst}\ j\ t)\ = \texttt{tsubst}\ (\texttt{tsubst}\ i \circ j)\ t \quad \text{for all } t\,.$$

The proof is a routine induction on the structure of t:

Case $t = v$: Then $\texttt{tsubst}\ i\ (\texttt{tsubst}\ j\ v) = \texttt{tsubst}\ i\ (j\ v) = \texttt{tsubst}\ (\texttt{tsubst}\ i \circ j)\ v$.

Case $t = f(t_1, \ldots, t_n)$:

$$\begin{aligned}
&\texttt{tsubst}\ i\ (\texttt{tsubst}\ j\ (f(t_1, \ldots, t_n))) \\
&= \texttt{tsubst}\ i\ (f(\texttt{tsubst}\ j\ t_1, \ldots, \texttt{tsubst}\ j\ t_n)) \\
&= f(\texttt{tsubst}\ i\ (\texttt{tsubst}\ j\ t_1), \ldots, \texttt{tsubst}\ i\ (\texttt{tsubst}\ j\ t_n)) \\
&= f(\texttt{tsubst}\ (\texttt{tsubst}\ i \circ j)\ t_1, \ldots, \texttt{tsubst}\ (\texttt{tsubst}\ i \circ j)\ t_n) && \text{(by ind.hyp.)} \\
&= \texttt{tsubst}\ (\texttt{tsubst}\ i \circ j)\ (f(t_1, \ldots, t_n))
\end{aligned}$$

Exercise 7.3 Skolemization

Skolemize the following formulas, putting them into prenex normal form if necessary.

a) $\forall x.\exists y.\forall z.\exists w.\neg P(a, w) \vee Q(f(x), y)$

b) $(\forall x.\exists y.P(x, g(y, f(x)))) \vee \neg Q(z) \vee \neg(\forall x.R(x, y))$

Solution

We denote the Skolem function arising from a variavle x by s_x:

a) $\forall xz.\neg P(a, s_w(x, z)) \vee Q(f(x), s_y(x))$

b)

$$\begin{aligned}
&(\forall x.\exists w.P(x, g(w, f(x)))) \vee \neg Q(z) \vee \neg(\forall v.R(v, y)) && \text{(renamed bound variables)} \\
\equiv\ &\exists v.\forall x.\exists w.P(x, g(w, f(x))) \vee \neg Q(z) \vee \neg R(v, y) && \text{(pulled out quantifiers)} \\
\rightsquigarrow\ &\forall x.P(x, g(s_w(x), f(x))) \vee \neg Q(z) \vee \neg R(s_v, y) && \text{(skolemized)}
\end{aligned}$$

Exercise 7.4

Consider the following alternative definition for `substq`, the quantifier-part of substitution:

```
let substq i quant x p =
  let x' = if mem x (fv p)
           then variant x (fv p) else x in
  quant x' (subst ((x |-> Var x') i) p);;
```

Find a counterexample, where this function leads to a variable capture in a substitution.

Solution

A counterexample is the substitution $i = \{x \mapsto y\}$ and the formula $\forall y.P(x)$ (hence `quant` $= \forall$, `x` $= y$ and `p` $= P(x)$. Then the substitution produces the wrong result $\forall y.P(y)$.

Exercise 7.5 Nameless representation of bound variables

a) Two formulas are called α-equivalent, if they differ only in the naming of bound variables.

   Write an OCaml function that checks if two formulas are α-equivalent.

Using an alternative representation for bound variables, we can enforce that α-equivalent formulas have the same representation:

Instead of giving names to bound variables, we write them uniformly as b_i, where i is a non-negative integer called a *deBruijn-index*. The index serves as a relative address that tells us how many quantifiers we have to skip (when moving up the term structure), until we hit the quantifier that binds the variable. The quantifiers themselves no longer carry a variable name.

Example 1: The nameless formula $\forall\forall P(b_1, b_0)$ represents $\forall x.\exists y.P(x, y)$ or, equivalently, $\forall z.\exists w.P(z, w)$.

Example 2: The formula $\forall x.P(x) \wedge (\exists y.Q(x, y))$ is written $\forall(P(b_0) \wedge \exists(Q(b_1, b_0)))$. Note that the same variable may occur as different indices in different positions.

b) Implement the nameless representation for formulas in OCaml, and define a translation from the named to the nameless representation.

c) Implement a substitution function that works on the nameless representation.