UPPSALA UNIVERSITET
Matematiska institutionen
Erik Palmgren

EXERCISES 3
Tillämpad Logik, ht-08
2008-11-06

Exercises 3, Applied Logic

CLTT refers to the March 2004 version of the handout *Constructive Logic and Type Theory,* by Erik Palmgren. (Available on the webpage of the course.)

1. * Exercise 1.1 in CLTT.

2. The following lemma is an example of a non-constructive result which is often used (implicitly) in mathematical analysis (and in computer science!) when reasoning about infinite processes.

   (König's lemma). A finite string over the alphabet $\{l, r\}$ is regarded as describing a path in a binary tree, starting from the root. Suppose that P is an infinite set of such paths. Show that there is an infinite string
   $$d_1 d_2 d_3 \cdots$$
   such that for every n, the string $d_1 d_2 \cdots d_n$ is an initial segment of some path in P.

   Suppose that there is an algorithm which decides whether a finite path $s \in \{l, r\}^*$ is in P. Is there any hope to find an algorithm which for each index i computes the value of $d_i \in \{l, r\}$? Discuss.

3. Exercise 2.1 in CLTT.

4. Exercise 2.2 in CLTT.

5. * The *Ackermann function* $\mathsf{ack} : \mathrm{N} \to \mathrm{N} \to \mathrm{N}$ can be defined by the following recursion equations
   $$\begin{aligned} \mathsf{ack}(0)(n) &= \mathsf{S}(n) \\ \mathsf{ack}(\mathsf{S}(m))(0) &= \mathsf{ack}(m)(\mathsf{S}(0)) \\ \mathsf{ack}(\mathsf{S}(m))(\mathsf{S}(n)) &= \mathsf{ack}(m)(\mathsf{ack}(\mathsf{S}(m))(n)) \end{aligned}$$

   It can be shown that the Ackermann function grows faster than any primitive recursive function. Show that it nevertheless may be defined in Gödel's system T (or the typed lambda calculus described in Chapter 2 of CLTT) with the help of the recursion operator rec. [Hint: expand the third line of the definition.]

6. Exercises 3.1 (a-c,e) in CLTT.

7. Exercises 3.1 (h,j) in CLTT.

8. * Exercises 3.1 (d,f,i,k) in CLTT.

9. * Exercise 3.2 in CLTT.

10. Exercise 3.3.(a)

11. Do selected parts of Exercise 4.1 in CLTT (compare with Ex 3.1).

12. Three variants of the induction scheme for natural numbers:

$$\begin{aligned} &\text{(IND)} && A(0) \wedge (\forall x)(A(x) \to A(\mathsf{S}(x))) \to (\forall x)A(x) \\ &\text{(C-IND)} && (\forall x)[(\forall y)(y < x \to B(y)) \to B(x)] \to (\forall x)B(x) \\ &\text{(LNP)} && \neg(\forall x)C(x) \to (\exists x)[\neg C(x) \wedge (\forall y)(y < x \to C(y))] \end{aligned}$$

Here $y < x$ is defined as the formula $(\exists z)(y + \mathsf{S}(z) = x)$. These induction principles are all equivalent in Peano arithmetic with classical logic.

(a) Prove (C-IND) from (IND) using only intuitionistic logic and the assumptions

(H1) $(\forall y)\neg(y < 0)$,
(H2) $(\forall x)(\forall y)(y < \mathsf{S}(x) \leftrightarrow y = x \vee y < x)$,
(H3) $(\forall x)(x < \mathsf{S}(x))$,
(H4) $(\forall x)(\forall y)(x = y \wedge P(x) \to P(y))$, where $P(\cdot)$ is a formula.

Hint: consider some $A(x)$ of the form $(\forall y)(y < x \to \cdots)$.

(b) Prove the Least Number Principle (LNP) from (C-IND) using only classical logic.