

TYPES AND λ -CALCULUS

Problem Sheet 1

Questions 2 and 3 will be marked if submitted on time.

You will need to use other rules from Appendix A of the notes in addition to those that we discussed in the lectures.

1. Consider the following proof (annotated with circled numbers) of:

$$\forall n m \in \mathbb{N}. n \leq m \Rightarrow \exists x. m = n + x$$

In this proof, we silently assume the basic facts about arithmetic, but the only thing we know about \leq is its definition:

$$\forall p. 0 \leq p \tag{1}$$

$$\forall p q. (p + 1) \leq q \text{ iff } \exists q'. q = q' + 1 \wedge p \leq q' \tag{2}$$

Proof. The proof is by induction on n .

- When $n = 0$, we argue as follows. Let $m \in \mathbb{N}$. ① Suppose $0 \leq m$. Then let the witness x be m . Then the goal $m = 0 + x$ is just $x = 0 + x$ which is true by arithmetic.
- When n is of shape $k + 1$, we assume the induction hypothesis. Let $m \in \mathbb{N}$ and suppose $k + 1 \leq m$. We can apply the definition of less-than, clause (2), from left to right to obtain some q' such that (i) $m = q' + 1$ and (ii) $k \leq q'$. ② Then we can apply the induction hypothesis to (ii) to obtain some x' such that (iii) $q' = k + x'$. Then let the witness to the goal also be x' . ③ It follows from (i) that this is just $q' + 1 = k + 1 + x'$; and by (iii), this becomes $k + x' + 1 = k + 1 + x'$ which is true by basic arithmetic.

□

Note that we often apply forwards rules implicitly in this proof, and this is typical.

- (a) What is the induction hypothesis in the second case of the proof?

(b) What is the state of the proof at each position ①, ② and ③?

** 2. Note that, by the conventions of logic, $A \Rightarrow B \Rightarrow C$ is a shorthand for $A \Rightarrow (B \Rightarrow C)$ and conjunction binds tighter than implication, so $A \wedge B \Rightarrow C$ means $(A \wedge B) \Rightarrow C$.

Give proofs of the following. I recommend you keep track of the proof state on a scrap of paper as you complete the proof, but you need not submit this.

- (a) $\neg A \Rightarrow A \Rightarrow B$
- (b) $(A \wedge B \Rightarrow C) \Rightarrow A \Rightarrow B \Rightarrow C$
- (c) $\neg(A \wedge \neg A)$
- (d) $(A \Rightarrow B) \Rightarrow (B \Rightarrow C) \Rightarrow A \Rightarrow C$
- (e) $\neg A \wedge \neg B \Rightarrow \neg(A \vee B)$

** 3. The following build on top of each other. I recommend you keep track of the proof state on a piece of paper as you build your proof.

- (a) Prove $(A \vee B) \wedge \neg B$ implies A .
- (b) Prove $\forall nm \in \mathbb{N}. n + m = 0 \Rightarrow m = 0$. Induction is not necessary. You may use the following two theorems of arithmetic:

$$(i) \forall p \in \mathbb{N}. p = 0 \vee (\exists q \in \mathbb{N}. p = q + 1)$$

$$(ii) \forall nm \in \mathbb{N}. n + m = 0 \Rightarrow \neg(\exists q \in \mathbb{N}. m = q + 1)$$

The second of these is equivalent to Lemma 1.1 from the notes.

- (c) Prove $\forall nm \in \mathbb{N}. n + m = 0 \Rightarrow n * m = 0$. Induction is not necessary. Multiplication on natural numbers can be defined as follows:

$$p * 0 = 0 \tag{3}$$

$$p * (q + 1) = p + (p * q) \tag{4}$$

4. Recall the grammar for regular expressions given in the notes. We define substitution on regular expressions. For any regular expressions R and S over alphabet Σ and any $a \in \Sigma$, we define $R[S/a]$, the result of replacing every a in R by S ,

by recursion on the structure of R :

$$\begin{aligned}
a[S/a] &= S \\
b[S/a] &= b && \text{if } a \neq b \\
\epsilon[S/a] &= \epsilon \\
(R_1 \cdot R_2)[S/a] &= (R_1[S/a]) \cdot (R_2[S/a]) \\
(R_1 + R_2)[S/a] &= (R_1[S/a]) + (R_2[S/a]) \\
(R_1^*)[S/a] &= (R_1[S/a])^*
\end{aligned}$$

For example:

$$((ab^*)^* + bba)[(a+b)/a] = ((a+b)b^*)^* + bb(a+b)$$

Consider the following partial proof of the statement:

Suppose S is a regex over Σ and $a \in \Sigma$. Then for all regexes R , $R[S/a]$ is a valid regex.

Proof. Suppose (i) S is a regex and (ii) $a \in \Sigma$. We prove $\forall R. R[S/a]$ is a regex, by induction on R .

- When R is an arbitrary letter, say c , the goal is to show $c[S/a]$ is a regex. There are two possibilities:
 - If c is a , the letter we are replacing, then the definition of substitution gives us that $c[S/a] = S$. We have from (i) that S is a regex.
 - If c is different from a , then the definition gives us that $c[S/a] = c$ a single letter. By the grammar defining regexes, every single letter is itself a regex.

Since the result holds in both cases, we conclude that it holds for any letter c .

- When R is a concatenation $(R_1 \cdot R_2)$, we may assume the induction hypotheses:

(IH1) ??

(IH2) ??

The goal is to show that $(R_1 \cdot R_2)[S/a]$ is a regex. It follows from (IH1) and (IH2), by the grammar for regexes, that $(R_1[S/a] \cdot R_2[S/a])$ is a regex. By definition of substitution, $(R_1[S/a] \cdot R_2[S/a]) = (R_1 \cdot R_2)[S/a]$, so we have that this is a regex, which was our goal.

□

- (a) What are the two induction hypotheses (IH1) and (IH2)?
- (b) Complete the proof (like most induction proofs, it is very repetitive).