Assignment 7
Propositions as Types

15-814: Types and Programming Languages
Frank Pfenning

Due Tuesday, November 5, 2019

1 Propositions as Types

Task 1 (L14.1, 15 points) One proposition is *more general* than another if we can instantiate the propositional variables in the first to obtain the second. For example, $A \supset (B \supset A)$ is more general than $A \supset (\bot \supset A)$ (with $[\bot/B]$), $(C \wedge D) \supset (B \supset (C \wedge D))$ (with $[C \wedge D/A]$, but not more general than $C \supset (D \supset E)$.

For each of the following proof terms, give the most general proposition proved by it. (We are justified in saying "*the most general*" because the most general proposition is unique up to the names of the propositional variables.)

1. $\lambda u.\, \lambda w.\, \lambda k.\, w\,(u\,k)$
2. $\lambda w.\, \langle\!|(\lambda u.\, w\,(\ell \cdot u)), (\lambda k.\, w\,(r \cdot k))|\!\rangle$
3. $\lambda x.\, (\mathsf{fst}\, x)\,(\mathsf{snd}\, x)\,(\mathsf{snd}\, x)$
4. $\lambda x.\, \lambda y.\, \lambda z.\, (x\,z)\,(y\,z)$

Task 2 (L14.2, 15 points) Write out a proof term for each of the following propositions. As you know from this lecture, this is the same as writing a program of the translated type in our program language without the use of fixed points.

1. $(A \wedge (A \supset \bot)) \supset B$
2. $(A \vee (A \supset \bot)) \supset (((A \supset \bot) \supset \bot) \supset A)$

2 Parametric Polymorphism

Task 3 (L15.1, 10 points) Find closed types τ and σ such that

$$\cdot\,;\cdot \vdash \lambda x.\, x\,[\tau]\,x : \sigma$$

Task 4 (L15.3, 20 points) For each of the following potential isomorphisms, fill in the missing entry and write down properly typed candidate functions *Forth* and *Back* to witness an isomorphism. You do not need to prove the isomorphism property. On the left side of each candidate isomorphism, we have a type with only universal quantification and function types. On the right side we have a type using any of the type constructors from this course (functions, eager products, lazy products, unit, sum, recursive types, lazy products) but *not* universally quantified types. We have filled in the first line for you, and you can find the *Forth* and *Back* functions in Section L15.4 (no need to repeat them).

	$\forall\alpha.\,\alpha \to \alpha \to \alpha$	$\cong$	$1 + 1$
(1)	$\forall\alpha.\,\alpha \to \alpha$	$\cong$	
(2)	$\forall\alpha.\,\alpha$	$\cong$	
(3)		$\cong$	$\rho\alpha.\,(\texttt{e} : 1) + (\texttt{b0} : \alpha) + (\texttt{b1} : \alpha)$
(4)		$\cong$	$\mathit{nat} \times \mathit{nat}$

The type $\mathit{nat} = \rho\alpha.\,(\texttt{zero} : 1) + (\texttt{succ} : \alpha)$. You may use the functions from Section L15.4 in your solution to Part 4.