Exercise solutions – Introduction to Type Theory

Exercise 1 – Types and terms

1. Find a term t which has type $\mathbb{N} \to (\mathbb{N} \to \mathbb{N})$.

   NB: when we say that a term t has type A, we mean that t has type A *in the empty context* unless specified otherwise – that is, the judgment $\vdash t : A$ holds.

2. Find a term t which has type $A \to (A \to A)$ for any type A.

3. Write a derivation tree which shows that $\lambda\,(t : A \times B)\,.\,\langle\,\pi_2(t)\,,\pi_1(t)\,\rangle$ has type $A \times B \to B \times A$ for any two types A, B.

4. ($\star$) Is it possible to find a type A such that the judgment $\vdash \lambda\,t\,.\,t\;t : A$ holds?

Solution

1. One possible such term is $\lambda\,n\,.\,\lambda\,m\,.\,m + n$

$$\dfrac{\dfrac{\dfrac{}{n : \mathbb{N}, m : \mathbb{N} \vdash m : \mathbb{N}} \quad \dfrac{}{n : \mathbb{N}, m : \mathbb{N} \vdash n : \mathbb{N}}}{n : \mathbb{N}, m : \mathbb{N} \vdash m + n : \mathbb{N}}}{\dfrac{n : \mathbb{N} \vdash \lambda\,m\,.\,m + n : \mathbb{N} \to \mathbb{N}}{\vdash \lambda\,n\,.\,\lambda\,m\,.\,m + n : \mathbb{N} \to (\mathbb{N} \to \mathbb{N})}}$$

   Other solutions include $\lambda\,n\,.\,\lambda\,m\,.\,0$, or $\lambda\,n\,.\,\lambda\,m\,.\,m$

2. If the arguments are not integers, we cannot add them. One solution that still works no matter what the type A is $\lambda\,n\,.\,\lambda\,m\,.\,m$

$$\dfrac{\dfrac{\dfrac{}{n : A, m : A \vdash m : A}}{n : A \vdash \lambda\,m\,.\,m : A \to A}}{\vdash \lambda\,n\,.\,\lambda\,m\,.\,m : A \to (A \to A)}$$

   In fact, up to definitional equality, there are only two solutions: $\lambda\,n\,.\,\lambda\,m\,.\,m$ and $\lambda\,n\,.\,\lambda\,m\,.\,n$.

3. We let the syntax guide us – at each step, there is only one reasonable rule to apply

$$\dfrac{\dfrac{\dfrac{\dfrac{}{t : A \times B \vdash t : A \times B}}{t : A \times B \vdash \pi_2(t) : B} \quad \dfrac{\dfrac{}{t : A \times B \vdash t : A \times B}}{t : A \times B \vdash \pi_1(t) : A}}{t : A \times B \vdash \langle\,\pi_2(t)\,,\pi_1(t)\,\rangle : B \times A}}{\vdash \lambda\,(t : A \times B)\,.\,\langle\,\pi_2(t)\,,\pi_1(t)\,\rangle : A \times B \to B \times A}$$

4. No, it is not possible to give a type to $\lambda\,x\,.\,x\;x$. To see this, assume that we can find a type A and a derivation tree which establishes $\vdash \lambda\,x\,.\,x\;x : A$.

The only typing rule whose conclusion contains λ is the rule Arr-Intro, so the derivation tree must end with Arr-Intro. This tells us that the type A is of the form $B \to C$ for some types B and C, and that the judgment $x : B \vdash x\ x : C$ is derivable.

Now, the only typing rule whose conclusion contains a function application is the rule Arr-Elim. Thus, the second-to-last rule is necessarily Arr-Elim, which means that B is of the form $D \to C$ for some type D, and that the judgment $x : D \to C \vdash x : C$ is derivable.

Lastly, the only way to give a type to a plain variable such as x is when the typing comes from the context. But the context only contains $x : D \to C$, which tells us that $D \to C$ must be equal to C.

However, we defined the set of simple types as the smallest set generated by a handful of constructors. An easy proof by induction on that set gives us that no spurious equalities such as $(D \to C) = C$ may happen. We have reached a contradiction.

Exercise 2 – The empty type

We extend our simple type theory with an *empty type* $\bot$, which has no inhabitants. For intuition, you might want to think of the empty set in math, or a type of exceptions in functional programming. The empty type has a single typing rule:

$$\frac{\Gamma \vdash t : \bot}{\Gamma \vdash \text{raise}(t) : A}\ \textsc{Empty-Elim}$$

and does not need any computation rule.

1. Find a term t which has type $A \times (A \to \bot) \to B$ for any two types A, B.
2. Find a term t which has type $A \to ((A \to \bot) \to \bot)$ for any type A.
3. ($\star\star$) Is it possible to find a type t which has type $((A \to \bot) \to \bot) \to A$?

Solution

1. The term $\lambda t\,.\,\text{raise}\ ((\pi_2\ t)\ (\pi_1\ t))$ works

$$\dfrac{\dfrac{\dfrac{}{t : A \times (A \to \bot) \vdash t : A \times (A \to \bot)}}{t : A \times (A \to \bot) \vdash (\pi_2\ t) : A \to \bot} \quad \dfrac{\dfrac{}{t : A \times (A \to \bot) \vdash t : A \times (A \to \bot)}}{t : A \times (A \to \bot) \vdash (\pi_1\ t) : A}}{\dfrac{\dfrac{t : A \times (A \to \bot) \vdash (\pi_2\ t)\ (\pi_1\ t) : \bot}{t : A \times (A \to \bot) \vdash \text{raise}\ ((\pi_2\ t)\ (\pi_1\ t)) : B}}{\vdash \lambda t\,.\,\text{raise}\ ((\pi_2\ t)\ (\pi_1\ t)) : A \times (A \to \bot) \to B}}$$

2. The term $\lambda t\,.\,\lambda k\,.\,k\ t$ works

$$\dfrac{\dfrac{\dfrac{}{t : A, k : A \to \bot \vdash k : A \to \bot} \quad \dfrac{}{t : A, k : A \to \bot \vdash t : A}}{t : A, k : A \to \bot \vdash k\ t : \bot}}{\dfrac{t : A \vdash \lambda k\,.\,k\ t : (A \to \bot) \to \bot}{\vdash \lambda t\,.\,\lambda k\,.\,k\ t : A \to ((A \to \bot) \to \bot)}}$$

3. No, it is not possible to find a term of type $((A \to \bot) \to \bot) \to A$. To get an intuition why, recall the Curry-Howard correspondence: giving a term of type $((A \to \bot) \to \bot) \to A$ is equivalent to giving a proof of $(\neg\neg A) \Rightarrow A$ in constructive logic. However, the double negation of a proposition $\neg\neg A$ is not constructively equivalent to the proposition A.

   It is possible to prove this more formally, but that would take us too far for now.

Exercise 3 – Sum types

We extend our simple type theory with *sum types*. The sum type $A + B$ contains a copy of all the elements of A, and a copy of all the elements of B. For intuition, you might want to think about disjoint unions in mathematics, or algebraic datatypes in functional programming.

The rules for sum types are given in the document on simple type theory (*stt.pdf*).

1. Find a term t which has type $A + B \to B + A$ for any two types A, B.
2. Find a term t which has type $(A + B \to C) \to (A \to C) \times (B \to C)$ for any types A, B, C.
3. ($\star\star$) Find a term t which has type $((A + (A \to \bot)) \to \bot) \to \bot$ for any type A.

Solution

1. The following term works:

$$\lambda\,(x : A + B)\,.\ \begin{array}{l}\mathsf{match}\ x\ \mathsf{with}\\ \mid \mathsf{in}_1\ a \mapsto \mathsf{in}_2\ a\\ \mid \mathsf{in}_2\ b \mapsto \mathsf{in}_1\ b\end{array}$$

$$\dfrac{\dfrac{}{x : A + B \vdash x : A + B} \quad \dfrac{\dfrac{}{x : A + B, a : A \vdash a : A}}{x : A + B, a : A \vdash \mathsf{in}_2\ a : B + A} \quad \dfrac{\dfrac{}{x : A + B, b : B \vdash b : B}}{x : A + B, b : B \vdash \mathsf{in}_2\ b : B + A}}{\dfrac{x : A + B \vdash \mathsf{match}\ x\ \mathsf{with} \mid \mathsf{in}_1\ a \mapsto \mathsf{in}_2\ a \mid \mathsf{in}_2\ b \mapsto \mathsf{in}_1\ b : B + A}{\vdash \lambda\,x\,.\,\mathsf{match}\ x\ \mathsf{with} \mid \mathsf{in}_1\ a \mapsto \mathsf{in}_2\ a \mid \mathsf{in}_2\ b \mapsto \mathsf{in}_1\ b : A + B \to B + A}}$$

2. The following term works: $\lambda\,(f : A + B \to C)\,.\,\langle\,\lambda\,(x : A)\,.\,f\ (\mathsf{in}_1\ x)\,,\,\lambda\,(x : B)\,.\,f\ (\mathsf{in}_2\ x)\,\rangle$

$$\dfrac{\dfrac{\dfrac{}{[\ldots] \vdash f : A + B \to C} \quad \dfrac{\dfrac{}{[\ldots] \vdash x : A}}{[\ldots] \vdash \mathsf{in}_1\ x : A + B}}{f : A + B \to C, x : A \vdash f\ (\mathsf{in}_1\ x) : C}}{f : A + B \to C \vdash \lambda\,x\,.\,f\ (\mathsf{in}_1\ x) : A \to C} \quad \dfrac{\dfrac{\dfrac{}{[\ldots] \vdash f : A + B \to C} \quad \dfrac{\dfrac{}{[\ldots] \vdash x : B}}{[\ldots] \vdash \mathsf{in}_2\ x : A + B}}{f : A + B \to C, x : B \vdash f\ (\mathsf{in}_2\ x) : C}}{f : A + B \to C \vdash \lambda\,x\,.\,f\ (\mathsf{in}_2\ x) : B \to C}$$

$$\dfrac{f : A + B \to C \vdash \langle\,\lambda\,x\,.\,f\ (\mathsf{in}_1\ x)\,,\,\lambda\,x\,.\,f\ (\mathsf{in}_2\ x)\,\rangle : (A \to C) \times (B \to C)}{\vdash \lambda\,f\,.\,\langle\,\lambda\,x\,.\,f\ (\mathsf{in}_1\ x)\,,\,\lambda\,x\,.\,f\ (\mathsf{in}_2\ x)\,\rangle : (A + B \to C) \to (A \to C) \times (B \to C)}$$

3. Before giving an explicit term, let us build a bit of intuition. In order to construct a term of type $((A + (A \to \bot)) \to \bot) \to \bot$, we should assume some variable $k : ((A + (A \to \bot)) \to \bot)$ in our context and construct a term of type $\bot$. From the previous question, we know how to get from $(A + (A \to \bot)) \to \bot$ to $(A \to \bot) \times ((A \to \bot) \to \bot)$. And from question 1 of exercise 2, we know how to get from there to any other type B, which we might as well take to be $\bot$.

   If we put all of this together and unfold a few definitional equalities, we end up with a startingly simple term:

$$\lambda\,k\,.\,k\ (\mathsf{in}_2\ (\lambda\,a\,.\,k\ (\mathsf{in}_1\ a)))$$

   From a Curry-Howard perspective, we are proving that the double negation of the law of excluded middle is constructively true. From a functional programming perspective, we are duplicating the continuation k.