Chapter 1

Type theory

Exercises

Exercise 1.1. Given functions $f : A \to B$ and $g : B \to C$, define their **composite** $g \circ f : A \to C$. Show that we have $h \circ (g \circ f) \equiv (h \circ g) \circ f$.

Solution (Alan)**.**

Solution (Alex)**.** Define $g \circ f :\equiv \lambda(x : A).g(fx)$. Then if $x : A$ then $fx : B$ and $g(fx) : C$, so $\lambda(x : A).g(fx)$ has the desired type $A \to C$.

Suppose $f : A \to B$, $g : B \to C$, and $h : C \to D$. We have

$$\begin{aligned} & h \circ (g \circ f) \\ & = \lambda x.(h \circ (g \circ f))x \\ & = \lambda x.h((g \circ f)x) \\ & = \lambda x.h(g(fx)) \\ & = \lambda x.(h \circ g)(fx) \\ & = \lambda x.((h \circ g) \circ f)x \\ & = (h \circ g) \circ f. \end{aligned}$$

Exercise 1.2. Derive the recursion principle for products $\mathsf{rec}_{A\times B}$ using only the projections, and verify that the definitional equalities are valid. Do the same for Σ-types.

Solution (Daniel)**.** The *recursion principle* is the rule that states that $\mathsf{rec}_{A\times B}(f) : A \times B \to C$ is well defined, and it is taken as a primitive notion. We are asked to instead take the well-definedness of pr_1 and pr_2 as primitive instead and to derive the recursion principle.

Definitions of $\mathsf{rec}_{A\times B}$, pr_1, and pr_2 are given below for convenience.

$$\mathsf{rec}_{A\times B} : (A \to B \to C) \to A \times B \to C$$
$$\mathsf{rec}_{A\times B}(f)((a,b)) := f(a)(b)$$

$$\mathsf{pr}_1 : A \times B \to A$$
$$\mathsf{pr}_1 := \mathsf{rec}_{A\times B}(\lambda\, a\, b \mapsto a)$$

$$\mathsf{pr}_2 : A \times B \to B$$
$$\mathsf{pr}_2 := \mathsf{rec}_{A\times B}(\lambda\, a\, b \mapsto b)$$

(In agreement with the notation used in the text, $\lambda\, a\, b \mapsto \Phi$ is merely shorthand for $\lambda\, a \mapsto (\lambda\, b \mapsto \Phi)$. We can think of such an expression as a "two-variable" curried function [mmmm, curry].)

We need primitive definitions of pr_1 and pr_2.

$$\mathsf{pr}_1 : A \times B \to A$$
$$\mathsf{pr}_1(a,b) := a$$

$$\mathsf{pr}_2 : A \times B \to B$$
$$\mathsf{pr}_2(a,b) := b$$

We define

$$\mathsf{rec}'_{A\times B} : (A \to B \to C) \to A \times B \to C$$
$$\mathsf{rec}'_{A\times B}(f)((a,b)) := f(\mathsf{pr}_1(a,b))(\mathsf{pr}_2(a,b))$$

We have

$$\begin{aligned}\mathsf{rec}_{A\times B}(f)(a,b) &= f(a)(b)\\ &= f(\mathsf{pr}_1(a,b))(\mathsf{pr}_2(a,b))\\ &= \mathsf{rec}'_{A\times B}(f)(a,b)\end{aligned}$$

So $\mathsf{rec}_{A\times B} = \mathsf{rec}'_{A\times B}$.

Solution (Jake)**.**

Solution (James)**.**

Exercise 1.3. Derive the induction principle for products $\mathsf{ind}_{A\times B}$, using only the projections and the propositional uniqueness principle $\mathsf{uniq}_{A\times B}$. Verify that the definitional equalities are valid. Generalize $\mathsf{uniq}_{A\times B}$ to Σ-types, and do the same for Σ-types. *(This requires concepts from* **??***.)*

Solution (Steven)**.** I actually have no idea (yet), this is just a test. $foo = b^{a^r}$

Solution (Zack)**.**

Exercise 1.4. Assuming as given only the *iterator* for natural numbers

$$\mathsf{iter} : \prod_{C:\mathcal{U}} C \to (C \to C) \to \mathbb{N} \to C$$

with the defining equations

$$\begin{aligned} \mathsf{iter}(C, c_0, c_s, 0) &:\equiv c_0, \\ \mathsf{iter}(C, c_0, c_s, \mathsf{succ}(n)) &:\equiv c_s(\mathsf{iter}(C, c_0, c_s, n)), \end{aligned}$$

derive a function having the type of the recursor $\mathsf{rec}_{\mathbb{N}}$. Show that the defining equations of the recursor hold propositionally for this function, using the induction principle for $\mathbb{N}$.

Solution (Alan)**.**

Solution (Daniel)**.**

Exercise 1.5. Show that if we define $A+B :\equiv \sum_{(x:\mathbf{2})} \mathsf{rec}_{\mathbf{2}}(\mathcal{U}, A, B, x)$, then we can give a definition of ind_{A+B} for which the definitional equalities stated in **??** hold.

Solution (Steven)**.**

Solution (Alex)**.** Recall that the type of ind_{A+B} is

$$\prod_{C:(A+B)\to\mathcal{U}} \left(\prod_{a:A} C(\mathsf{inl}(a))\right) \to \left(\prod_{b:B} C(\mathsf{inr}(b))\right) \to \prod_{x:A+B} C(x).$$

If $x : A + B$ then either $x = (0_{\mathbf{2}}, a)$ with $a : A$ or $x = (1_{\mathbf{2}}, b)$ with $b : B$. Thus we may define ind_{A+B} by the following case analysis.

$$\begin{aligned} \mathsf{ind}_{A+B}(C, g_0, g_1, (0_{\mathbf{2}}, a)) &:\equiv g_0(a) \\ \mathsf{ind}_{A+B}(C, g_0, g_1, (1_{\mathbf{2}}, b)) &:\equiv g_1(b) \end{aligned}$$

Since $\mathsf{inl}(a) = (0_{\mathbf{2}}, a)$ and $\mathsf{inr}(b) = (1_{\mathbf{2}}, b)$ the, types of $g_0(a) : C(\mathsf{inl}(a))$ and $g_1(b) : C(\mathsf{inr}(b))$ are judgementally equal to $C(0_{\mathbf{2}}, a)$ and $C(1_{\mathbf{2}}, b)$ respectively. In either case $\mathsf{ind}_{A+B}(C, g_0, g_1, x) : C(x)$. Additionaly, by substitution we have

$$\begin{aligned} \mathsf{ind}_{A+B}(C, g_0, g_1, \mathsf{inl}(a)) &\equiv g_0(a), \\ \mathsf{ind}_{A+B}(C, g_0, g_1, \mathsf{inr}(b)) &\equiv g_1(b), \end{aligned}$$

as desired.

Solution (James)**.**

Exercise 1.6. Show that if we define $A \times B :\equiv \prod_{(x:\mathbf{2})} \mathsf{rec}_{\mathbf{2}}(\mathcal{U}, A, B, x)$, then we can give a definition of $\mathsf{ind}_{A\times B}$ for which the definitional equalities stated in **??** hold propositionally (i.e. using equality types). *(This requires the function extensionality axiom, which is introduced in* **??***.)*

Solution (Jake)**.**

Solution (Zack)**.**

Exercise 1.7. Give an alternative derivation of $\mathsf{ind}'_{=_A}$ from $\mathsf{ind}_{=_A}$ which avoids the use of universes. *(This is easiest using concepts from later chapters.)*

Solution (Alan)**.**

Solution (Jake)**.**

Exercise 1.8. Define multiplication and exponentiation using $\mathsf{rec}_{\mathbb{N}}$. Verify that $(\mathbb{N}, +, 0, \times, 1)$ is a semiring using only $\mathsf{ind}_{\mathbb{N}}$. You will probably also need to use symmetry and transitivity of equality, **????**.

Solution (Steven)**.**

Solution (Alex)**.** Recall that $\mathsf{rec}_{\mathbb{N}}$ has type

$$\mathsf{rec}_{\mathbb{N}} : \prod_{C:\mathcal{U}} C \to (\mathbb{N} \to C \to C) \to C,$$

and defining equations

$$\begin{aligned} \mathsf{rec}_{\mathbb{N}}(C, c_0, c_s, 0) &:\equiv c_0, \\ \mathsf{rec}_{\mathbb{N}}(C, c_0, c_s, \mathsf{succ}(n)) &:\equiv c_s(n, \mathsf{rec}_{\mathbb{N}}(C, c_0, c_s, n)). \end{aligned}$$

Define

$$\begin{aligned} \mu_0 &:\equiv \lambda n.\, 0, \\ \mu_s &:\equiv \lambda n.\, \lambda g.\, \lambda m.\, g(m) + m, \\ \mathsf{mult} &:\equiv \mathsf{rec}_{\mathbb{N}}(\mathbb{N} \to \mathbb{N}, \mu_0, \mu_s). \end{aligned}$$

Then we have

$$\begin{aligned} \mathsf{mult}(0, m) &\equiv \mathsf{rec}_{\mathbb{N}}(\mathbb{N} \to \mathbb{N}, \mu_0, \mu_s, 0)(m), \\ &\equiv \mu_0(m), \\ &\equiv 0, \end{aligned}$$

and

$$\begin{aligned} \mathsf{mult}(\mathsf{succ}(n), m) &\equiv \mathsf{rec}_{\mathbb{N}}(\mathbb{N} \to \mathbb{N}, \mu_0, \mu_s, \mathsf{succ}(n))(m), \\ &\equiv \mu_s(n, \mathsf{rec}_{\mathbb{N}}(\mathbb{N} \to \mathbb{N}, \mu_0, \mu_s, n))(m), \\ &\equiv \mu_s(n, \mathsf{mult}(n))(m), \\ &\equiv \mathsf{mult}(n, m) + m, \end{aligned}$$

which behaves like we would expect of multiplication. Similarly, we define

$$\begin{aligned} e_0 &:\equiv 1, \\ e_s &:\equiv \lambda n.\, \lambda g.\, \lambda m.\, \mathsf{mult}(g(m), m), \\ \mathsf{exp} &:\equiv \mathsf{rec}_{\mathbb{N}}(\mathbb{N} \to \mathbb{N}, e_0, e_s), \end{aligned}$$

and everything works out similarly. (Note that here $\mathsf{exp}(n, m)$ means m^n.)

Next, we will show that $(\mathbb{N}, 0, +, \times, 1)$ forms a semiring. We will be needing the following functions throughout:

$\mathsf{ap}_{\mathsf{succ}} : \prod_{m,n:\mathbb{N}} (m =_{\mathbb{N}} n) \to (\mathsf{succ}(m) =_{\mathbb{N}} \mathsf{succ}(n))$	§1.9
$\mathsf{tr}_{\mathbb{N}} : \prod_{a,b,c:A} a =_A b \to b =_A c \to a =_A c$	§2.1
$\mathsf{sym}_{\mathbb{N}} : \prod_{a,b:A} a =_A b \to b =_A a$	§2.1
$\mathsf{ipl} : \prod_{a,b:\mathbb{N}} \mathsf{succ}(a) + b =_{\mathbb{N}} \mathsf{succ}(a + b)$	From judgemental equality.
$\mathsf{ipr} : \prod_{a,b:\mathbb{N}} a + \mathsf{succ}(b) =_{\mathbb{N}} \mathsf{succ}(a + b)$	From judgemental equality.

For convenience most dependent arguments are omitted when clear from context and $\mathsf{tr}_{\mathbb{N}}$ is written infix as $\star$. I may be doing something wrong here, but most of these follow from judgemental equality with $\mathsf{ind}_{\mathbb{N}}$ just there to pack things into the dependent type. That's why there are so many refls.

Left identity: $P_0 :\equiv \prod_{n:\mathbb{N}} 0 + n = n$.

$$p_0 :\equiv \mathsf{ind}_{\mathbb{N}}(P_0, \mathsf{refl}_0, \lambda n.\, \lambda p.\, \mathsf{refl}_n)$$

Right identity: $P_1 :\equiv \prod_{n:\mathbb{N}} n + 0 = n$.

$$\begin{aligned} r_1 &:\equiv \lambda n.\, \lambda p.\, \mathsf{ipl} \star \mathsf{ap}_{\mathsf{succ}}(p) \\ p_1 &:\equiv \mathsf{ind}_{\mathbb{N}}(P_1, \mathsf{refl}_0, r_1) \end{aligned}$$

Commutativity: $P_2 :\equiv \prod_{a,b:\mathbb{N}} a + b = b + a$.

$$\begin{aligned} r_2 &:\equiv \lambda n.\, \lambda p.\, \mathsf{ipl} \star \mathsf{ap}_{\mathsf{succ}}(p) \star \mathsf{ipr} \\ p_2 &:\equiv \mathsf{ind}_{\mathbb{N}}(P_2, p_0, r_2) \end{aligned}$$

Zero annihilation (left): $P_3 :\equiv \prod_{a:\mathbb{N}} 0 \times a = 0$.

$$p_3 :\equiv \mathsf{ind}_{\mathbb{N}}(P_3, \mathsf{refl}_0, \lambda n.\, \lambda p.\, \mathsf{refl}_0)$$

Zero annihilation (right): $P_4 :\equiv \prod_{a:\mathbb{N}} a \times 0 = 0$.

$$p_4 :\equiv \mathsf{ind}_{\mathbb{N}}(P_4, \mathsf{refl}_0, \lambda n.\, \lambda p.\, \mathsf{refl}_0)$$

Unit (left): $P_5 :\equiv \prod_{a:\mathbb{N}} 1 \times a = a$

$$p_5 :\equiv \mathsf{ind}_{\mathbb{N}}(P_5, \mathsf{refl}_0, \lambda n.\, \lambda p.\, \mathsf{refl}_n)$$

Unit (right): $P_6 :\equiv \prod_{a:\mathbb{N}} a \times 1 = a$

$$\begin{aligned} h_6 &:\equiv \mathsf{ind}_{\mathbb{N}}(\textstyle\prod_{n:\mathbb{N}} \mathsf{succ}(n) \times 1 = \mathsf{succ}(n \times 1), \mathsf{refl}_0, \lambda n.\, \lambda p.\, \mathsf{refl}_n) \\ r_6 &:\equiv \lambda n.\, \lambda p.\, h_6 \star \mathsf{ap}_{\mathsf{succ}}(p) \\ p_6 &:\equiv \mathsf{ind}_{\mathbb{N}}(P_5, \mathsf{refl}_0, r_6) \end{aligned}$$

To be continued maybe. This exercise is pretty long, and I'm pretty sure the hairy part is still ahead.

Exercise 1.9. Define the type family $\mathsf{Fin} : \mathbb{N} \to \mathcal{U}$ mentioned at the end of **??**, and the dependent function $\mathsf{fmax} : \prod_{(n:\mathbb{N})} \mathsf{Fin}(n+1)$ mentioned in **??**.

Solution (Daniel)**.**

Solution (Zack)**.**

Solution (James)**.**

Exercise 1.10. Show that the Ackermann function $\mathsf{ack} : \mathbb{N} \to \mathbb{N} \to \mathbb{N}$ is definable using only $\mathsf{rec}_{\mathbb{N}}$ satisfying the following equations:

$$\begin{aligned} \mathsf{ack}(0, n) &\equiv \mathsf{succ}(n), \\ \mathsf{ack}(\mathsf{succ}(m), 0) &\equiv \mathsf{ack}(m, 1), \\ \mathsf{ack}(\mathsf{succ}(m), \mathsf{succ}(n)) &\equiv \mathsf{ack}(m, \mathsf{ack}(\mathsf{succ}(m), n)). \end{aligned}$$

Solution (Alan)**.**

Solution (Steven)**.**

Exercise 1.11. Show that for any type A, we have $\neg\neg\neg A \to \neg A$.

Solution (Jake)**.**

Solution (Daniel)**.**

Solution (James)**.**

Exercise 1.12. Using the propositions as types interpretation, derive the following tautologies.

(i) If A, then (if B then A).
(ii) If A, then not (not A).
(iii) If (not A or not B), then not (A and B).

Solution (Alex)**.** Translating them into types, we get

(i) $\prod_{A,B:\mathcal{U}} A \to B \to A$,
(ii) $\prod_{A:\mathcal{U}} A \to (A \to 0) \to 0$,

(iii) $\prod_{A,B:\mathcal{U}}((A \to 0) + (B \to 0)) \to (A \times B) \to 0$.

For values of each type we have

(i) $\lambda a.\, \lambda b.\, a$,
(ii) $\lambda a.\, \lambda f.\, f(a)$,
(iii) $\mathsf{rec}_{(A\to 0)+(B\to 0)}((A \times B) \to 0, \lambda f.\, f \circ \mathsf{pr}_1, \lambda f.\, f \circ \mathsf{pr}_2)$.

The third value can be written without the recursor as

$$g(\mathsf{inl}(f), (a, b)) :\equiv f(a),$$
$$g(\mathsf{inr}(f), (a, b)) :\equiv f(b).$$

Solution (Zack)**.**

Exercise 1.13. Using propositions-as-types, derive the double negation of the principle of excluded middle, i.e. prove *not (not (P or not P))*.

Solution (Alan)**.**

Solution (Zack)**.**

Exercise 1.14. Why do the induction principles for identity types not allow us to construct a function $f : \prod_{(x:A)} \prod_{(p:x=x)} (p = \mathsf{refl}_x)$ with the defining equation

$$f(x, \mathsf{refl}_x) :\equiv \mathsf{refl}_{\mathsf{refl}_x} \quad ?$$

Solution (Daniel)**.**

Solution (Alex)**.**

Solution (James)**.**

Exercise 1.15. Show that indiscernability of identicals follows from path induction.

Solution (Steven)**.**

Solution (Jake)**.**

Exercise 1.16. Show that addition of natural numbers is commutative: $\prod_{(i,j:\mathbb{N})} (i + j = j + i)$.