

Exercises from the *HoTT Book*

John Dougherty

June 26, 2014

1 Type Theory

Exercise 1.1 (p. 56) Given functions $f : A \rightarrow B$ and $g : B \rightarrow C$, define their **composite** $g \circ f : A \rightarrow C$. Show that we have $h \circ (g \circ f) \equiv (h \circ g) \circ f$.

Solution Define $g \circ f \equiv \lambda(x : A). g(f(x))$. Then if $h : C \rightarrow D$, we have

$$h \circ (g \circ f) \equiv \lambda(x : A). h((g \circ f)x) \equiv \lambda(x : A). h((\lambda(y : A). g(fy))x) \equiv \lambda(x : A). h(g(fx))$$

and

$$(h \circ g) \circ f \equiv \lambda(x : A). (h \circ g)(fx) \equiv \lambda(x : A). (\lambda(y : A). h(gy))(fx) \equiv \lambda(x : A). h(g(fx))$$

So $h \circ (g \circ f) \equiv (h \circ g) \circ f$. In Coq, we have

Definition `compose {A B C : Type} (g : B → C) (f : A → B) :=
 fun x => g (f x).`

Theorem `compose_assoc : forall (A B C D : Type) (f : A → B) (g : B → C) (h : C → D),
 compose h (compose g f) = compose (compose h g) f.`

Proof.

`trivial.`

Qed.

Exercise 1.2 (p. 56) Derive the recursion principle for products $\text{rec}_{A \times B}$ using only the projections, and verify that the definitional equalities are valid. Do the same for Σ -types.

Solution The recursion principle states that we can define a function $f : A \times B \rightarrow C$ by giving its value on pairs. Suppose that we have projection functions $\text{pr}_1 : A \times B \rightarrow A$ and $\text{pr}_2 : A \times B \rightarrow B$. Then we can define a function of type

$$\text{rec}_{A \times B} : \prod_{C : \mathcal{U}} (A \rightarrow B \rightarrow C) \rightarrow A \times B \rightarrow C$$

in terms of these projections as follows

$$\text{rec}_{A \times B}(C, g, p) \equiv g(\text{pr}_1 p)(\text{pr}_2 p)$$

or, in Coq,

Variable `A B : Type.`

Definition `recprod (C : Type) (g : A → B → C) (p : A * B) :=
 g (fst p) (snd p).`

We must then show that $\text{pr}_1 \equiv \text{rec}_{A \times B}(A, \lambda a. \lambda b. a)$ and likewise for pr_2 . We have

$$\begin{aligned} \text{rec}_{A \times B}(A, \lambda a. \lambda b. a) &\equiv \lambda(p : A \times B). (\lambda a. \lambda b. a)(\text{pr}_1 p)(\text{pr}_2 p) \\ &\equiv \lambda(p : A \times B). (\lambda b. \text{pr}_1 p)(\text{pr}_2 p) \\ &\equiv \lambda(p : A \times B). \text{pr}_1 p \\ &\equiv \text{pr}_1 \end{aligned}$$

and

$$\begin{aligned} \text{rec}_{A \times B}(B, \lambda a. \lambda b. b) &\equiv \lambda(p : A \times B). (\lambda a. \lambda b. b)(\text{pr}_1 p)(\text{pr}_2 p) \\ &\equiv \lambda(p : A \times B). (\lambda b. b)(\text{pr}_2 p) \\ &\equiv \lambda(p : A \times B). \text{pr}_2 p \\ &\equiv \text{pr}_2 \end{aligned}$$

Which are direct in Coq:

Goal `recprod A (fun a => fun b => a) = fst. trivial. Qed.`

Goal `recprod B (fun a => fun b => b) = snd. trivial. Qed.`

Now for the Σ -types. Here we have a projection

$$\text{pr}_1 : \left(\sum_{x:A} B(x) \right) \rightarrow A$$

and another

$$\text{pr}_2 : \prod_{p : \sum_{x:A} B(x)} B(\text{pr}_1(p))$$

Define a function of type

$$\text{rec}_{\sum_{x:A} B(x)} : \prod_{C:\mathcal{U}} \left(\prod_{x:A} B(x) \rightarrow C \right) \rightarrow \left(\sum_{x:A} B(x) \right) \rightarrow C$$

by

$$\text{rec}_{\sum_{x:A} B(x)}(C, g, p) \equiv g(\text{pr}_1 p)(\text{pr}_2 p)$$

Variable `A : Type.`

Variable `B : A → Type.`

Definition `recsm (C : Type) (g : forall (x : A), B x → C) (p : exists (x : A), B x) :=
g (projT1 p) (projT2 p).`

We then verify that

$$\begin{aligned} \text{rec}_{\sum_{x:A} B(x)}(A, \lambda a. \lambda b. a) &\equiv \lambda(p : \sum_{x:A} B(x)). (\lambda a. \lambda b. a)(\text{pr}_1 p)(\text{pr}_2 p) \\ &\equiv \lambda(p : \sum_{x:A} B(x)). (\lambda b. \text{pr}_1 p)(\text{pr}_2 p) \\ &\equiv \lambda(p : \sum_{x:A} B(x)). \text{pr}_1 p \\ &\equiv \text{pr}_1 \end{aligned}$$

and

$$\begin{aligned} \text{rec}_{\sum_{x:A} B(x)}(B, \lambda a. \lambda b. b) &\equiv \lambda(p : \sum_{x:A} B(x)). (\lambda a. \lambda b. b)(\text{pr}_1 p)(\text{pr}_2 p) \\ &\equiv \lambda(p : \sum_{x:A} B(x)). (\lambda b. b)(\text{pr}_2 p) \\ &\equiv \lambda(p : \sum_{x:A} B(x)). \text{pr}_2 p \\ &\equiv \text{pr}_2 \end{aligned}$$

Exercise 1.3 (p. 56) Derive the induction principle for products $\text{ind}_{A \times B}$ using only the projections and the propositional uniqueness principle uppt . Verify that the definitional equalities are valid. Generalize uppt to Σ -types, and do the same for Σ -types.

Solution The induction principle has type

$$\text{ind}_{A \times B} : \prod_{C : A \times B \rightarrow \mathcal{U}} \left(\prod_{(x:A)} \prod_{(y:B)} C((x, y)) \right) \rightarrow \prod_{z : A \times B} C(z)$$

For a first pass, we can define

$$\text{ind}_{A \times B}(C, g, z) := g(\text{pr}_1 z)(\text{pr}_2 z)$$

However, we have $g(\text{pr}_1 x)(\text{pr}_2 x) : C((\text{pr}_1 x, \text{pr}_2 x))$, so the type of this $\text{ind}_{A \times B}$ is

$$\text{ind}_{A \times B} : \prod_{C : A \times B \rightarrow \mathcal{U}} \left(\prod_{(x:A)} \prod_{(y:B)} C((x, y)) \right) \rightarrow \prod_{z : A \times B} C((\text{pr}_1 z, \text{pr}_2 z))$$

To define $\text{ind}_{A \times B}$ with the correct type, we need the transport operation from the next chapter. The uniqueness principle for $A \times B$ is

$$\text{uppt} : \prod_{x : A \times B} ((\text{pr}_1 x, \text{pr}_2 x) =_{A \times B} x)$$

By the transport principle, there is a function

$$(\text{uppt } x)_* : C((\text{pr}_1 x, \text{pr}_2 x)) \rightarrow C(x)$$

so

$$\text{ind}_{A \times B}(C, g, z) := (\text{uppt } z)_*(g(\text{pr}_1 z)(\text{pr}_2 z))$$

We now have to show that

$$\text{ind}_{A \times B}(C, g, (a, b)) \equiv g(a)(b)$$

Unfolding the left gives

$$\begin{aligned} \text{ind}_{A \times B}(C, g, (a, b)) &\equiv (\text{uppt } (a, b))_*(g(\text{pr}_1(a, b))(\text{pr}_2(a, b))) \\ &\equiv \text{ind}_{=_{A \times B}}(D, d, (a, b), (a, b), \text{uppt}((a, b)))(g(a)(b)) \\ &\equiv \text{ind}_{=_{A \times B}}(D, d, (a, b), (a, b), \text{refl}_{(a, b)})(g(a)(b)) \\ &\equiv \text{ind}_{=_{A \times B}}(D, d, (a, b), (a, b), \text{refl}_{(a, b)})(g(a)(b)) \\ &\equiv \text{id}_{C((a, b))}(g(a)(b)) \\ &\equiv g(a)(b) \end{aligned}$$

which was to be proved.

For Σ -types, we define

$$\text{ind}_{\Sigma_{(x:A)} B(x)} : \prod_{C : (\Sigma_{(x:A)} B(x)) \rightarrow \mathcal{U}} \left(\prod_{(a:A)} \prod_{(b:B(a))} C((a, b)) \right) \rightarrow \prod_{p : \Sigma_{(x:A)} B(x)} C(p)$$

at first pass by

$$\text{ind}_{\Sigma_{(x:A)} B(x)}(C, g, p) \equiv g(\text{pr}_1 p)(\text{pr}_2 p)$$

We encounter a similar problem as before. We need a uniqueness principle for Σ -types, which would be a function

$$\text{uppt} : \prod_{p : \sum_{(x:A)} B(x)} ((\text{pr}_1 p, \text{pr}_2 p) =_{\sum_{(x:A)} B(x)} p)$$

As for product types, we can define

$$\text{uppt}((a, b)) \equiv \text{refl}_{(a,b)}$$

which is well-typed, since $\text{pr}_1(a, b) \equiv a$ and $\text{pr}_2(a, b) \equiv b$. Thus, we can write

$$\text{ind}_{\sum_{(x:A)} B(x)}(C, g, p) \equiv (\text{uppt } p)_*(g(\text{pr}_1 p)(\text{pr}_2 p)).$$

Now we must verify that

$$\text{ind}_{\sum_{(x:A)} B(x)}(C, g, (a, b)) \equiv g(a)(b)$$

We have

$$\begin{aligned} \text{ind}_{\sum_{(x:A)} B(x)}(C, g, (a, b)) &\equiv (\text{uppt } (a, b))_*(g(\text{pr}_1(a, b))(\text{pr}_2(a, b))) \\ &\equiv \text{ind}_{=\sum_{(x:A)} B(x)}(D, d, (a, b), (a, b), \text{uppt } (a, b))(g(a)(b)) \\ &\equiv \text{ind}_{=\sum_{(x:A)} B(x)}(D, d, (a, b), (a, b), \text{refl}_{(a,b)})(g(a)(b)) \\ &\equiv \text{id}_{C((a,b))}(g(a)(b)) \\ &\equiv g(a)(b) \end{aligned}$$

Exercise 1.4 (p. 56) Assuming as given only the *iterator* for natural numbers

$$\text{iter} : \prod_{C:\mathcal{U}} C \rightarrow (C \rightarrow C) \rightarrow \mathbb{N} \rightarrow C$$

with the defining equations

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

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

Solution Fix some $C : \mathcal{U}$, $c_0 : C$, and $c_s : \mathbb{N} \rightarrow C \rightarrow C$. $\text{iter}(C)$ allows for the n -fold application of a single function to a single input from C , whereas $\text{rec}_{\mathbb{N}}$ allows each application to depend on n , as well. Since n just tracks how many applications we've done, we can construct n on the fly, iterating over elements of $\mathbb{N} \times C$. So we will use the iterator

$$\text{iter}_{\mathbb{N} \times C} : \mathbb{N} \times C \rightarrow (\mathbb{N} \times C \rightarrow \mathbb{N} \times C) \rightarrow \mathbb{N} \rightarrow \mathbb{N} \times C$$

to derive a function

$$\Phi : \prod_{C:\mathcal{U}} C \rightarrow (\mathbb{N} \rightarrow C \rightarrow C) \rightarrow \mathbb{N} \rightarrow C$$

which has the same type as $\text{rec}_{\mathbb{N}}$. The first argument of $\text{iter}_{\mathbb{N} \times C}$ is the starting point, which we'll make $(0, c_0)$. The second input takes an element of $\mathbb{N} \times C$ as an argument and uses c_s to construct a new element

of $\mathbb{N} \times C$. We can use the first and second elements of the pair as arguments for c_s , and we'll use succ to advance the second argument, representing the number of steps taken. This gives the function

$$\lambda x. (\text{succ}(\text{pr}_1 x), c_s(\text{pr}_1 x, \text{pr}_2 x)) : \mathbb{N} \times C \rightarrow \mathbb{N} \times C$$

for the second input to $\text{iter}_{\mathbb{N} \times C}$. The third input is just n , which we can pass through. Plugging these in gives

$$\text{iter}_{\mathbb{N} \times C}((0, c_0), \lambda x. (\text{succ}(\text{pr}_1 x), c_s(\text{pr}_1 x, \text{pr}_2 x)), n) : \mathbb{N} \times C$$

from which we need to extract an element of C . This is easily done with the projection operator, so we have

$$\Phi_C(c_0, c_s, n) := \text{pr}_2 \left(\text{iter}_{\mathbb{N} \times C}((0, c_0), \lambda x. (\text{succ}(\text{pr}_1 x), c_s(\text{pr}_1 x, \text{pr}_2 x)), n) \right)$$

which has the same type as $\text{rec}_{\mathbb{N}}$.

Now to show that the defining equations hold propositionally for Φ . First we need a small calculation about how the first element of the pair advances through iteration. That is, we will be interested in the function

$$\Theta(n) := \text{pr}_1 \left(\text{iter}_{\mathbb{N} \times C}((0, c_0), \lambda x. (\text{succ}(\text{pr}_1 x), c_s(\text{pr}_1 x, \text{pr}_2 x)), n) \right)$$

We show that $\Theta(n) =_{\mathbb{N}} n$ is inhabited for all n , by induction. For the family

$$E(n) := (\Theta(n) =_{\mathbb{N}} n)$$

The induction principle for \mathbb{N} gives us a function

$$\text{ind}_{\mathbb{N}}(E) : E(0) \rightarrow \left(\prod_{(n:\mathbb{N})} E(n) \rightarrow E(\text{succ}(n)) \right) \rightarrow \prod_{(n:\mathbb{N})} E(n)$$

which is just a functional version of usual notion of induction on \mathbb{N} . So for the base case, we show that

$$\Theta(0) \equiv \text{pr}_1 \left(\text{iter}_{\mathbb{N} \times C}((0, c_0), \lambda x. (\text{succ}(\text{pr}_1 x), c_s(\text{pr}_1 x, \text{pr}_2 x)), 0) \right) \equiv \text{pr}_1(0, c_0) \equiv 0$$

Thus $\text{refl}_0 : (\Theta(0) =_{\mathbb{N}} 0) \equiv E(0)$. For the induction step, we suppose that $n : \mathbb{N}$ and that $\text{refl}_n : E(n)$. Unravelling the definition a bit, we get

$$\begin{aligned} & \Theta(\text{succ}(n)) \\ & \equiv \text{pr}_1 \left(\text{iter}_{\mathbb{N} \times C}((0, c_0), \lambda x. (\text{succ}(\text{pr}_1 x), c_s(\text{pr}_1 x, \text{pr}_2 x)), \text{succ}(n)) \right) \\ & \equiv \text{pr}_1(\text{succ}(\Theta(n)), c_s(\Theta(n), \Phi_C(c_0, c_s, n))) \\ & \equiv \text{succ}(\Theta(n)) \end{aligned}$$

From which we conclude that $\text{refl}_{\Theta(\text{succ}(n))} : \Theta(\text{succ}(n)) =_{\mathbb{N}} \text{succ}(\Theta(n))$. To replace the $\Theta(n)$ on the right hand side of this equality, we use refl_n and the indiscernability of identicals. We have the family

$$F(m) := (\Theta(\text{succ}(n)) =_{\mathbb{N}} \text{succ}(m))$$

and the indiscernability of identicals gives us a function

$$f : \prod_{(x,y:\mathbb{N})} \prod_{(p:x=\mathbb{N}y)} F(x) \rightarrow F(y)$$

on the basis of this. Plugging in the appropriate arguments, we obtain

$$f \left(\Theta(n), n, \text{refl}_n, \text{refl}_{\Theta(\text{succ}(n))} \right) : F(n) \equiv (\Theta(\text{succ}(n)) =_{\mathbb{N}} \text{succ}(n)) \equiv E(\text{succ}(n))$$

So, discharging the assumption of the induction step,

$$e \equiv \lambda n. \lambda \text{refl}_n. f \left(\Theta(n), n, \text{refl}_n, \text{refl}_{\Theta(\text{succ}(n))} \right) : \prod_{n:\mathbb{N}} E(n) \rightarrow E(\text{succ}(n))$$

thus

$$\text{ind}_{\mathbb{N}}(E, \text{refl}_0, e) : \prod_{n:\mathbb{N}} E(n) \equiv \prod_{n:\mathbb{N}} (\Theta(n) =_{\mathbb{N}} n)$$

We're now prepared to show that the definitional equalities hold propositionally for Φ . To do this, we must show that

$$\begin{aligned} & \Phi_C(c_0, c_s, 0) =_C c_0 \\ & \prod_{n:\mathbb{N}} \left(\Phi_C(c_0, c_s, \text{succ}(n)) =_C c_s(n, \Phi_C(c_0, c_s, n)) \right) \end{aligned}$$

are inhabited. Since C , c_0 , and c_s are fixed, define

$$\Psi(n) \equiv \Phi_C(c_0, c_s, n)$$

for brevity. The first equality is straightforward:

$$\Psi(0) \equiv \text{pr}_2 \left(\text{iter}_{\mathbb{N} \times C}((0, c_0), \lambda x. (\text{succ}(\text{pr}_1 x), c_s(\text{pr}_1 x, \text{pr}_2 x)), 0) \right) \equiv \text{pr}_2(0, c_0) \equiv c_0$$

So $\text{refl}_{\Psi(0)} : \Psi(0) =_C c_0$. This establishes the first equality.

Given the family

$$G(n) \equiv \Psi(\text{succ}(n)) =_C c_s(n, \Psi(n)),$$

the induction principle for \mathbb{N} gives us a function

$$\text{ind}_{\mathbb{N}}(G) : G(0) \rightarrow \left(\prod_{(n:\mathbb{N})} G(n) \rightarrow G(\text{succ}(n)) \right) \rightarrow \prod_{(n:\mathbb{N})} G(n)$$

In the base case,

$$\begin{aligned} & \Psi(1) \\ & \equiv \text{pr}_2 \left(\text{iter}_{\mathbb{N} \times C}((0, c_0), \lambda x. (\text{succ}(\text{pr}_1 x), c_s(\text{pr}_1 x, \text{pr}_2 x)), 1) \right) \\ & \equiv \text{pr}_2 \left(\text{succ}(\text{pr}_1(0, c_0)), c_s(\text{pr}_1(0, c_0), \text{pr}_2(0, c_0)) \right) \\ & \equiv c_s(0, c_0) \end{aligned}$$

So $\text{refl}_{\Psi(1)} : G(0)$. Now suppose that $n : \mathbb{N}$ and $\text{refl}_{\Psi(\text{succ}(n))} : G(n)$. We have

$$\begin{aligned} & \Psi(\text{succ}(\text{succ}(n))) \\ & \equiv \text{pr}_2 \left(\text{iter}_{\mathbb{N} \times C}((0, c_0), \lambda x. (\text{succ}(\text{pr}_1 x), c_s(\text{pr}_1 x, \text{pr}_2 x)), \text{succ}(\text{succ}(n))) \right) \\ & \equiv \text{pr}_2 \left(\text{succ}(\Theta(\text{succ}(n))), c_s(\Theta(\text{succ}(n)), \Psi(\text{succ}(n))) \right) \\ & \equiv c_s(\Theta(\text{succ}(n)), \Psi(\text{succ}(n))) \end{aligned}$$

We can again use the indiscernability of identicals here. Define the family

$$H(m) \equiv \Psi(\text{succ}(\text{succ}(n))) =_C c_s(m, \Psi(\text{succ}(n)))$$

Then the indiscernability of identicals gives us a function

$$h : \prod_{(m,i:\mathbb{N})} \prod_{(p:m=\mathbb{N}i)} H(m) \rightarrow H(i)$$

and

$$h(\Theta(\text{succ}(n)), \text{succ}(n), \text{ind}_{\mathbb{N}}(E, \text{refl}_0, g, \text{succ}(n))) : \Psi(\text{succ}(\text{succ}(n))) =_C c_s(\text{succ}(n), \Psi(\text{succ}(n)))$$

Abstracting out the context, we obtain an object

$$j : \prod_{n:\mathbb{N}} G(n) \rightarrow G(\text{succ}(n))$$

So

$$\text{ind}_{\mathbb{N}}(G, \text{refl}_{\Psi(1)}, j) : \prod_{n:\mathbb{N}} G(n) \equiv \prod_{n:\mathbb{N}} \left(\Phi_C(c_0, c_s, \text{succ}(n)) =_C c_s(n, \Phi_C(c_0, c_s, n)) \right)$$

Thus proving the second equality propositionally.

Exercise 1.5 (p. 56) Show that if we define $A + B := \sum_{(x:2)} \text{rec}_2(\mathcal{U}, A, B, x)$, then we can give a definition of ind_{A+B} for which the definitional equalities stated in §1.7 hold.

Solution Define $A + B$ as stated, with

$$\text{inl}(a) := (0_2, a)$$

$$\text{inr}(b) := (1_2, b)$$

This means that $A + B$ is a Σ -type, so we can use the definition of $\text{ind}_{\sum_{(x:A)} B(x)}$

$$\text{ind}_{\sum_{(x:2)} \text{rec}_2(\mathcal{U}, A, B, x)}(C, g, (a, b)) := g(a)(b)$$

where $C : (\sum_{(x:2)} \text{rec}_2(\mathcal{U}, A, B, x)) \rightarrow \mathcal{U}$, $g : \prod_{(a:2)} \prod_{(b:\text{rec}_2(\mathcal{U}, A, B, a))} C((a, b))$, $a : 2$, and $b : \text{rec}_2(\mathcal{U}, A, B, a)$. The definitional equalities are given in terms of $g_0 = g(0_2)$ and $g_1 = g(1_2)$. To show them true, we compute

$$f(\text{inl}(a)) \equiv \text{ind}_{\sum_{(x:2)} \text{rec}_2(\mathcal{U}, A, B, x)}(C, g, \text{inl}(a)) \equiv \text{ind}_{\sum_{(x:2)} \text{rec}_2(\mathcal{U}, A, B, x)}(C, g, (0_2, a)) \equiv g(0_2)(a) \equiv g_0(a)$$

$$f(\text{inr}(b)) \equiv \text{ind}_{\sum_{(x:2)} \text{rec}_2(\mathcal{U}, A, B, x)}(C, g, \text{inr}(b)) \equiv \text{ind}_{\sum_{(x:2)} \text{rec}_2(\mathcal{U}, A, B, x)}(C, g, (1_2, b)) \equiv g(1_2)(b) \equiv g_1(b)$$

Exercise 1.6 (p. 56) Show that if we define $A \times B := \prod_{(x:2)} \text{rec}_2(\mathcal{U}, A, B, x)$, then we can give a definition of $\text{ind}_{A \times B}$ for which the definitional equalities stated in §1.5 hold propositionally (i.e. using equality types).

Solution Define

$$A \times B := \prod_{x:2} \text{rec}_2(\mathcal{U}, A, B, x)$$

Supposing that $a : A$ and $b : B$, we have an element $(a, b) : A \times B$ given by

$$(a, b) := \text{ind}_2(\text{rec}_2(\mathcal{U}, A, B), a, b)$$

An induction principle for $A \times B$ will, given a family $C : A \times B \rightarrow \mathcal{U}$ and a function

$$g : \prod_{(x:A)} \prod_{(y:B)} C((x, y)),$$

give a function $f : \prod_{(x:A \times B)} C(x)$ defined by

$$f((x, y)) \equiv g(x)(y)$$

So suppose that we have such a C and g . Writing things out in terms of the definitions, we have

$$\begin{aligned} C &: \left(\prod_{x:2} \text{rec}_2(\mathcal{U}, A, B, x) \right) \rightarrow \mathcal{U} \\ g &: \prod_{(x:A)} \prod_{(y:B)} C(\text{ind}_2(\text{rec}_2(\mathcal{U}, A, B), x, y)) \end{aligned}$$

We can define projections by

$$\text{pr}_1 p \equiv p(0_2) \quad \text{pr}_2 p \equiv p(1_2)$$

Since p is an element of a dependent type, we have

$$\begin{aligned} p(0_2) &: \text{rec}_2(\mathcal{U}, A, B, 0_2) \equiv A \\ p(1_2) &: \text{rec}_2(\mathcal{U}, A, B, 1_2) \equiv B \end{aligned}$$

which checks out. Then we have

$$\begin{aligned} g(\text{pr}_1 p)(\text{pr}_2 p) &: C(\text{ind}_2(\text{rec}_2(\mathcal{U}, A, B), (\text{pr}_1 p), (\text{pr}_2 p))) \\ &\equiv C(\text{ind}_2(\text{rec}_2(\mathcal{U}, A, B), (\text{pr}_1 p), (\text{pr}_2 p))) \\ &\equiv C((p(0_2), p(1_2))) \end{aligned}$$

So we have defined a function

$$f' : \prod_{p:A \times B} C((p(0_2), p(1_2)))$$

But we need one of the type

$$f : \prod_{p:A \times B} C(p)$$

To solve this problem, we need to appeal to function extensionality from §2.9. This implies that there is a function

$$\text{funext} : \prod_{f,g:A \times B} \left(\prod_{x:2} (f(x) =_{\text{rec}_2(\mathcal{U}, A, B, x)} g(x)) \right) \rightarrow (f =_{A \times B} g)$$

So, consider

$$\text{funext}(p, (\text{pr}_1 p, \text{pr}_2 p)) : \left(\prod_{x:2} (p(x) =_{\text{rec}_2(\mathcal{U}, A, B, x)} (p(0_2), p(1_2))(x)) \right) \rightarrow (p =_{A \times B} (p(0_2), p(1_2)))$$

We just need to show that the antecedent is inhabited, which we can do with ind_2 . So consider the family

$$\begin{aligned} E &\equiv \lambda(x:2). (p(x) =_{\text{rec}_2(\mathcal{U}, A, B, x)} (p(0_2), p(1_2))(x)) \\ &\equiv \lambda(x:2). (p(x) =_{\text{rec}_2(\mathcal{U}, A, B, x)} \text{ind}_2(\text{rec}_2(\mathcal{U}, A, B), p(0_2), p(1_2), x)) \end{aligned}$$

We have

$$\begin{aligned} E(0_2) &\equiv (p(0_2) =_{\text{rec}_2(\mathcal{U}, A, B, 0_2)} \text{ind}_2(\text{rec}_2(\mathcal{U}, A, B), p(0_2), p(1_2), 0_2)) \\ &\equiv (p(0_2) =_{\text{rec}_2(\mathcal{U}, A, B, 0_2)} p(0_2)) \end{aligned}$$

Thus $\text{refl}_{p(0_2)} : E(0_2)$. The same argument goes through to show that $\text{refl}_{p(1_2)} : E(1_2)$. This means that

$$h \equiv \text{ind}_2(E, \text{refl}_{p(0_2)}, \text{refl}_{p(1_2)}) : \prod_{x:2} (p(x) =_{\text{rec}_2(\mathcal{U}, A, B, x)} (p(0_2), p(1_2)))$$

and thus

$$\text{funext}(p, (\text{pr}_1 p, \text{pr}_2 p), h) : p =_{A \times B} (p(0_2), p(1_2))$$

So, by the transport principle, there is a function

$$(\text{funext}(p, (\text{pr}_1 p, \text{pr}_2 p), h))_* : C((\text{pr}_1 p, \text{pr}_2 p)) \rightarrow C(p)$$

and we may define

$$\text{ind}_{A \times B}(C, g, p) \equiv (\text{funext}(p, (\text{pr}_1 p, \text{pr}_2 p), h))_*(g(\text{pr}_1 p)(\text{pr}_2 p))$$

Now, we must show that the definitional equality holds propositionally. That is, we must show that the type

$$\text{ind}_{A \times B}(C, g, (a, b)) =_{C((a, b))} g(a)(b)$$

is inhabited. Unfolding the left hand side gives

$$\begin{aligned} \text{ind}_{A \times B}(C, g, (a, b)) &\equiv (\text{funext}((a, b), (\text{pr}_1(a, b), \text{pr}_2(a, b)), h))_*(g(\text{pr}_1(a, b))(\text{pr}_2(a, b))) \\ &\equiv (\text{funext}((a, b), (a, b), h))_*(g(a)(b)) \\ &\equiv \text{ind}_{=_{A \times B}}(D, d, (a, b), (a, b), \text{funext}((a, b), (a, b), h))(g(a)(b)) \end{aligned}$$

where $D(x, y, \text{funext}((a, b), (a, b), h)) \equiv C(x) \rightarrow C(y)$ and

$$d \equiv \lambda x. \text{id}_{C(x)} : \prod_{x:A \times B} D(x, x, \text{refl}_x)$$

But the defining equality of

Exercise 1.7 (p. 56) Give an alternative derivation of $\text{ind}'_{=A}$ from $\text{ind}_{=A}$ which avoids the use of universes.

Exercise 1.8 (p. 56) Define multiplication and exponentiation using $\text{rec}_{\mathbb{N}}$. Verify that $(\mathbb{N}, +, 0, \times, 1)$ is a semiring using only $\text{ind}_{\mathbb{N}}$.

Solution For multiplication, we need to construct a function $\text{mult} : \mathbb{N} \rightarrow \mathbb{N} \rightarrow \mathbb{N}$. Defined with pattern-matching, we would have

$$\begin{aligned} \text{mult}(0, m) &\equiv 0 \\ \text{mult}(\text{succ}(n), m) &\equiv m + \text{mult}(n, m) \end{aligned}$$

so in terms of $\text{rec}_{\mathbb{N}}$ we have

$$\text{mult} \equiv \text{rec}_{\mathbb{N}}(\mathbb{N} \rightarrow \mathbb{N}, \lambda n. 0, \lambda n. \lambda r. \lambda m. \text{add}(m, r(n, m)))$$

For exponentiation, we have the function $\text{exp} : \mathbb{N} \rightarrow \mathbb{N} \rightarrow \mathbb{N}$, with the intention that $\text{exp}(e, b) = b^e$. In terms of pattern matching,

$$\begin{aligned} \text{exp}(0, b) &\equiv 1 \\ \text{exp}(\text{succ}(e), b) &\equiv \text{mult}(b, \text{exp}(e, b)) \end{aligned}$$

or, in terms of $\text{rec}_{\mathbb{N}}$,

$$\text{exp} \equiv \text{rec}_{\mathbb{N}}(\mathbb{N} \rightarrow \mathbb{N}, \lambda n. 1, \lambda n. \lambda r. \lambda m. \text{mult}(m, r(n, m)))$$

To verify that $(\mathbb{N}, +, 0, \times, 1)$ is a semiring, we need stuff from Chapter 2.

Exercise 1.9 (p. 56) Define the type family $\text{Fin} : \mathbb{N} \rightarrow \mathcal{U}$ mentioned at the end of §1.3, and the dependent function $\text{fmax} : \prod_{(n:\mathbb{N})} \text{Fin}(n+1)$ mentioned in §1.4.

Solution $\text{Fin}(n)$ is a type with exactly n elements. Essentially, we want to recreate \mathbb{N} using types; so we will replace 0 with $\mathbf{0}$ and succ with a coproduct. So we define Fin recursively:

$$\begin{aligned}\text{Fin}(\mathbf{0}) &\equiv \mathbf{0} \\ \text{Fin}(\text{succ}(n)) &\equiv \text{Fin}(n) + \mathbf{1}\end{aligned}$$

or, equivalently,

$$\text{Fin} \equiv \text{rec}_{\mathbb{N}}(\mathcal{U}, \mathbf{0}, \lambda C. C + \mathbf{1})$$

Exercise 1.10 (p. 56) Show that the Ackermann function $\text{ack} : \mathbb{N} \rightarrow \mathbb{N} \rightarrow \mathbb{N}$, satisfying the following equations

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

is definable using only $\text{rec}_{\mathbb{N}}$.

Exercise 1.11 (p. 56) Show that for any type A , we have $\neg\neg\neg A \rightarrow \neg A$.

Solution Suppose that $\neg\neg\neg A$ and A . Supposing further that $\neg A$, we get a contradiction with the second assumption, so $\neg\neg A$. But this contradicts the first assumption that $\neg\neg\neg A$, so $\neg A$. Discharging the first assumption gives $\neg\neg\neg A \rightarrow \neg A$.

In type-theoretic terms, the first assumption is $x : ((A \rightarrow \mathbf{0}) \rightarrow \mathbf{0}) \rightarrow \mathbf{0}$, and the second is $a : A$. If we further assume that $h : A \rightarrow \mathbf{0}$, then $h(a) : \mathbf{0}$, so discharging the h gives

$$\lambda(h : A \rightarrow \mathbf{0}). h(a) : (A \rightarrow \mathbf{0}) \rightarrow \mathbf{0}$$

But then we have

$$x(\lambda(h : A \rightarrow \mathbf{0}). h(a)) : \mathbf{0}$$

so discharging the a gives

$$\lambda(a : A). x(\lambda(h : A \rightarrow \mathbf{0}). h(a)) : A \rightarrow \mathbf{0}$$

And discharging the first assumption gives

$$\lambda(x : ((A \rightarrow \mathbf{0}) \rightarrow \mathbf{0}) \rightarrow \mathbf{0}). \lambda(a : A). x(\lambda(h : A \rightarrow \mathbf{0}). h(a)) : (((A \rightarrow \mathbf{0}) \rightarrow \mathbf{0}) \rightarrow \mathbf{0}) \rightarrow (A \rightarrow \mathbf{0}))$$

Exercise 1.12 (p. 56) 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 (i) Suppose that A and B ; then A . Discharging the assumptions, $A \rightarrow B \rightarrow A$. That is, we have

$$\lambda(a : A). \lambda(b : B). a : A \rightarrow B \rightarrow A$$

(ii) Suppose that A . Supposing further that $\neg A$ gives a contradiction, so $\neg \neg A$. That is,

$$\lambda(a : A). \lambda(f : A \rightarrow \mathbf{0}). f(a) : A \rightarrow (A \rightarrow \mathbf{0}) \rightarrow \mathbf{0}$$

(iii) Finally, suppose $\neg A \vee \neg B$. Supposing further that $A \wedge B$ means that A and that B . There are two cases. If $\neg A$, then we have a contradiction; but also if $\neg B$ we have a contradiction. Thus $\neg(A \wedge B)$.

Type-theoretically, we assume that $x : (A \rightarrow \mathbf{0}) + (B \rightarrow \mathbf{0})$ and $z : A \times B$. Conjunction elimination gives $\text{pr}_1 z : A$ and $\text{pr}_2 z : B$. We can now perform a case analysis. Suppose that $x_A : A \rightarrow \mathbf{0}$; then $x_A(\text{pr}_1 z) : \mathbf{0}$, a contradiction; if instead $x_B : B \rightarrow \mathbf{0}$, then $x_B(\text{pr}_2 z) : \mathbf{0}$. By the recursion principle for the coproduct, then,

$$f(z) \equiv \text{rec}_{(A \rightarrow \mathbf{0}) + (B \rightarrow \mathbf{0})}(\mathbf{0}, \lambda x. x(\text{pr}_1 z), \lambda x. x(\text{pr}_2 z)) : (A \rightarrow \mathbf{0}) + (B \rightarrow \mathbf{0}) \rightarrow \mathbf{0}$$

Discharging the assumption that $A \times B$ is inhabited, we have

$$f : A \times B \rightarrow (A \rightarrow \mathbf{0}) + (B \rightarrow \mathbf{0}) \rightarrow \mathbf{0}$$

So

$$\text{swap}(A \times B, (A \rightarrow \mathbf{0}) + (B \rightarrow \mathbf{0}), \mathbf{0}, f) : (A \rightarrow \mathbf{0}) + (B \rightarrow \mathbf{0}) \rightarrow A \times B \rightarrow \mathbf{0}$$

Exercise 1.13 (p. 57) Using propositions-as-types, derive the double negation of the principle of excluded middle, i.e. prove *not (not (P or not P))*.

Solution Suppose that $\neg(P \vee \neg P)$. Then, assuming P , we have $P \vee \neg P$ by disjunction introduction, a contradiction. Hence $\neg P$. But disjunction introduction on this again gives $P \vee \neg P$, a contradiction. So we must reject the remaining assumption, giving $\neg \neg(P \vee \neg P)$.

In type-theoretic terms, the initial assumption is that $g : P + (P \rightarrow \mathbf{0}) \rightarrow \mathbf{0}$. Assuming $p : P$, disjunction introduction results in $\text{inl}(p) : P + (P \rightarrow \mathbf{0})$. But then $g(\text{inl}(p)) : \mathbf{0}$, so we discharge the assumption of $p : P$ to get

$$\lambda(p : P). g(\text{inl}(p)) : P \rightarrow \mathbf{0}$$

Applying disjunction introduction again leads to contradiction, as

$$g(\text{inr}(\lambda(p : P). g(\text{inl}(p)))) : \mathbf{0}$$

So we must reject the assumption of $\neg(P \vee \neg P)$, giving the result:

$$\lambda(g : P + (P \rightarrow \mathbf{0}) \rightarrow \mathbf{0}). g(\text{inr}(\lambda(p : P). g(\text{inl}(p)))) : (P + (P \rightarrow \mathbf{0}) \rightarrow \mathbf{0}) \rightarrow \mathbf{0}$$

Exercise 1.14 (p. 57) Why do the induction principles for identity types not allow us to construct a function $f : \prod_{(x:A)} \prod_{(p:x=x)} (p = \text{refl}_x)$ with the defining equation

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

Exercise 1.15 (p. 57) Show that indiscernability of identicals follows from path induction.