Exercises from the *HoTT Book*

John Dougherty

July 6, 2014

Introduction

The following are solutions to exercises from *Homotopy Type Theory: Univalent Foundations of Mathematics*. The Coq code given alongside the by-hand solutions requires the HoTT version of Coq, available at the HoTT github repository. It will be assumed throughout that it has been imported by

```
Require Import HoTT.
```

Each part of each exercise has its own `Section` in the Coq file, so `Context` declarations don't extend beyond the exercise, and sometimes they're even more restricted than that.

1 Type Theory

Exercise 1.1 (p. 56) 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 Define $g \circ f :\equiv \lambda(x : A).\, g(f(x))$. Then if $h : C \to D$, 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).
Goal ∀ (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 $\mathsf{rec}_{A\times B}$ 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 $f : A \times B \to C$ by giving its value on pairs. Suppose that we have projection functions $\mathsf{pr}_1 : A \times B \to A$ and $\mathsf{pr}_2 : A \times B \to B$. Then we can define a function of type

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

in terms of these projections as follows

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

or, in Coq,

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

We must then show that

$$\mathsf{rec}'_{A\times B}(C, g, (a,b)) \equiv g(\mathsf{pr}_1(a,b))(\mathsf{pr}_2(a,b)) \equiv g(a)(b)$$

which in Coq is also trivial:

```
Goal ∀ C g a b, recprod C g (a, b) = g a b. trivial. Qed.
```

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

$$\mathsf{pr}_1 : \left(\sum_{x:A} B(x)\right) \to A$$

and another

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

Define a function of type

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

by

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

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

We then verify that

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

which is again trivial in Coq:

```
Goal ∀ C g a b, recsm C g (a; b) = g a b. trivial. Qed.
```

Exercise 1.3 (p. 56) Derive the induction principle for products $\mathsf{ind}_{A\times B}$ 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

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

For a first pass, we can define

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

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

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

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

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

By the transport principle, there is a function

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

so

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

has the right type. In Coq we first define uppt, then use it with transport to give our $\mathsf{ind}_{A\times B}$.

```
Definition uppt (x : A × B) : (fst x, snd x) = x. destruct x; reflexivity. Defined.
Definition indprd (C : A × B → Type) (g : ∀ (x:A) (y:B), C (x, y)) (z : A × B) :=
  (uppt z) # (g (fst z) (snd z)).
```

We now have to show that

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

Unfolding the left gives

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

which was to be proved. In Coq, it's as trivial as always:

```
Goal ∀ C g a b, indprd C g (a, b) = g a b. trivial. Qed.
```

For Σ-types, we define

$$\mathsf{ind}_{\sum_{(x:A)} B(x)} : \prod_{C:(\sum_{(x:A)} B(x))\to\mathcal{U}} \left(\prod_{(a:A)} \prod_{(b:B(a))} C((a,b)) \right) \to \prod_{p:\sum_{(x:A)} B(x)} C(p)$$

at first pass by

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

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

$$\mathsf{upst} : \prod_{p:\sum_{(x:A)} B(x)} \left((\mathsf{pr}_1 p, \mathsf{pr}_2 p) =_{\sum_{(x:A)} B(x)} p \right)$$

As for product types, we can define

$$\mathsf{upst}((a,b)) :\equiv \mathsf{refl}_{(a,b)}$$

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

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

and in Coq,

```
Definition upst (p : {x:A & B x}) : (projT1 p; projT2 p) = p. destruct p; reflexivity. Defined.
Definition indsm (C : {x:A & B x} → Type) (g : ∀ (a:A) (b:B a), C (a; b)) (p : {x:A & B x}) :=
  (upst p) # (g (projT1 p) (projT2 p)).
```

Now we must verify that

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

We have

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

which Coq finds trivial:

```
Goal ∀ C g a b, indsm C g (a; b) = g a b. trivial. Qed.
```

Exercise 1.4 (p. 56) 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 Fix some $C : \mathcal{U}$, $c_0 : C$, and $c_s : \mathbb{N} \to C \to C$. $\mathsf{iter}(C)$ allows for the n-fold application of a single function to a single input from C, whereas $\mathsf{rec}_{\mathbb{N}}$ 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

$$\mathsf{iter}_{\mathbb{N}\times C} : \mathbb{N} \times C \to (\mathbb{N} \times C \to \mathbb{N} \times C) \to \mathbb{N} \to \mathbb{N} \times C$$

to derive a function

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

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

The first argument of $\mathsf{iter}_{\mathbb{N}\times C}$ 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 first argument, representing the number of steps taken. This gives the function

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

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

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

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

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

which has the same type as $\mathsf{rec}_{\mathbb{N}}$. In Coq we first define the iterator and then our alternative recursor:

```
Fixpoint iter (C : Type) (c0 : C) (cs : C → C) (n : nat) : C :=
  match n with
    | O ⇒ c0
    | S n′ ⇒ cs(iter C c0 cs n′)
  end.
Definition Phi (C : Type) (c0 : C) (cs: nat → C → C) (n : nat) :=
  snd (iter (nat × C)
              (O, c0)
              (fun x ⇒ (S (fst x), cs (fst x) (snd x)))
              n).
```

Now to show that the defining equations hold propositionally for Φ. For clarity of notation, define

$$\Phi'(n) = \mathsf{iter}_{\mathbb{N}\times C}\big((0, c_0), \lambda x.\,(\mathsf{succ}(\mathsf{pr}_1 x), c_s(\mathsf{pr}_1 x, \mathsf{pr}_2 x)), n\big)$$

```
Definition Phi′ (n : nat) := iter (nat × C) (O, c0) (fun x ⇒ (S (fst x), cs (fst x) (snd x))) n.
```

So the propositional equalities can be written

$$\begin{aligned} \mathsf{pr}_2\Phi'(0) &=_C c_0 \\ \prod_{n:\mathbb{N}} \mathsf{pr}_2\Phi'(\mathsf{succ}(n)) &=_C c_s(n, \mathsf{pr}_2\Phi'(n)). \end{aligned}$$

The first is straightforward:

$$\mathsf{pr}_2\Phi'(0) \equiv \mathsf{pr}_2\mathsf{iter}_{\mathbb{N}\times C}\big((0, c_0), \lambda x.\,(\mathsf{succ}(\mathsf{pr}_1 x), c_s(\mathsf{pr}_1 x, \mathsf{pr}_2 x)), 0\big) \equiv \mathsf{pr}_2(0, c_0) \equiv c_0$$

so $\mathsf{refl}_{c_0} : \mathsf{pr}_2\Phi'(0) =_C c_0$. To establish the second, we use induction on a strengthened hypothesis involving Φ'. We will establish that for all $n : \mathbb{N}$,

$$P(n) :\equiv \Phi'(\mathsf{succ}(n)) =_C (\mathsf{succ}(n), c_s(n, \mathsf{pr}_2\Phi'(n)))$$

is inhabited. For the base case, we have

$$\begin{aligned} \Phi'(\mathsf{succ}(0)) &\equiv \mathsf{iter}_{\mathbb{N}\times C}\big((0, c_0), \lambda x.\,(\mathsf{succ}(\mathsf{pr}_1 x), c_s(\mathsf{pr}_1 x, \mathsf{pr}_2 x)), \mathsf{succ}(0)\big) \\ &\equiv \Big(\lambda x.\,(\mathsf{succ}(\mathsf{pr}_1 x), c_s(\mathsf{pr}_1 x, \mathsf{pr}_2 x))\Big)\mathsf{iter}_{\mathbb{N}\times C}\big((0, c_0), \lambda x.\,(\mathsf{succ}(\mathsf{pr}_1 x), c_s(\mathsf{pr}_1 x, \mathsf{pr}_2 x)), 0\big) \\ &\equiv \Big(\lambda x.\,(\mathsf{succ}(\mathsf{pr}_1 x), c_s(\mathsf{pr}_1 x, \mathsf{pr}_2 x))\Big)(0, c_0) \\ &\equiv (\mathsf{succ}(0), c_s(0, c_0)) \\ &\equiv (\mathsf{succ}(0), c_s(0, \mathsf{pr}_2\Phi'(0))) \end{aligned}$$

using the derivation of the first propositional equality. So $P(0)$ is inhabited, or $p_0 : P(0)$. For the induction hypothesis, suppose that $n : \mathbb{N}$ and that $p_n : P(n)$. A little massaging gives

$$\begin{aligned} \Phi'(\mathsf{succ}(\mathsf{succ}(n))) &\equiv \mathsf{iter}_{\mathbb{N}\times C}\big((0, c_0), \lambda x.\,(\mathsf{succ}(\mathsf{pr}_1 x), c_s(\mathsf{pr}_1 x, \mathsf{pr}_2 x)), \mathsf{succ}(\mathsf{succ}(n))\big) \\ &\equiv \Big(\lambda x.\,(\mathsf{succ}(\mathsf{pr}_1 x), c_s(\mathsf{pr}_1 x, \mathsf{pr}_2 x))\Big)\Phi'(\mathsf{succ}(n)) \\ &\equiv (\mathsf{succ}(\mathsf{pr}_1\Phi'(\mathsf{succ}(n))), c_s(\mathsf{pr}_1\Phi'(\mathsf{succ}(n)), \mathsf{pr}_2\Phi'(\mathsf{succ}(n)))) \end{aligned}$$

We now apply based path induction using p_n. Consider the family

$$D : \prod_{z:\mathbb{N}\times C} (\Phi'(\mathsf{succ}(n)) = x) \to \mathcal{U}$$

given by

$$D(z) :\equiv \Big(\mathsf{succ}(\mathsf{pr}_1\Phi'(\mathsf{succ}(n))), c_s(\mathsf{pr}_1\Phi'(\mathsf{succ}(n)), \mathsf{pr}_2\Phi'(\mathsf{succ}(n)))\Big) = (\mathsf{succ}(\mathsf{pr}_1 z), c_s(\mathsf{pr}_1 z, \mathsf{pr}_2\Phi'(\mathsf{succ}(n))))$$

(i.e., D is proof-irrelevant). Clearly, we have

$$\mathsf{refl}_{\Phi'(\mathsf{succ}(\mathsf{succ}(n)))} : D(\Phi'(\mathsf{succ}(n)), \mathsf{refl}_{\Phi'(\mathsf{succ}(n))})$$

so by based path induction, there is an element

$$\begin{aligned} f((\mathsf{succ}(n), c_s(n, \mathsf{pr}_2\Phi'(n))), p_n) : \Big(&\mathsf{succ}(\mathsf{pr}_1\Phi'(\mathsf{succ}(n))), c_s(\mathsf{pr}_1\Phi'(\mathsf{succ}(n)), \mathsf{pr}_2\Phi'(\mathsf{succ}(n)))\Big) \\ = (&\mathsf{succ}(\mathsf{pr}_1(\mathsf{succ}(n), c_s(n, \mathsf{pr}_2\Phi'(n)))), \\ &c_s(\mathsf{pr}_1(\mathsf{succ}(n), c_s(n, \mathsf{pr}_2\Phi'(n))), \mathsf{pr}_2\Phi'(\mathsf{succ}(n)))) \end{aligned}$$

Let $p_{n+1} :\equiv f((\mathsf{succ}(n), c_s(n, \mathsf{pr}_2\Phi'(n))))$. 0ur first bit of massaging allows us to replace the left hand side of this by $\Phi'(\mathsf{succ}(\mathsf{succ}(n))$. As for the right, applying the projections gives

$$p_{n+1} : \Phi'(\mathsf{succ}(\mathsf{succ}(n))) = (\mathsf{succ}(\mathsf{succ}(n)), c_s(\mathsf{succ}(n), \mathsf{pr}_2\Phi'(\mathsf{succ}(n)))) \equiv P(\mathsf{succ}(n))$$

Plugging all this into our induction principle for $\mathbb{N}$, we can discharge the assumption that $p_n : P(n)$ to obtain

$$q :\equiv \mathsf{ind}_{\mathbb{N}}(P, p_0, \lambda n. \lambda p_n. p_{n+1}, n) : P(n)$$

The propositional equality we're after is a consequence of this, which we again obtain by based path induction. Consider the family

$$E : \prod_{z:\mathbb{N}\times C} (\Phi'(n) = z) \to \mathcal{U}$$

given by

$$E(z, p) :\equiv \mathsf{pr}_2\Phi'(\mathsf{succ}(n)) = \mathsf{pr}_2 z$$

Again, it's clear that

$$\mathsf{refl}_{\mathsf{pr}_2\Phi'(\mathsf{succ}(n))} : E(\Phi'(\mathsf{succ}(n)), \mathsf{refl}_{\Phi'(\mathsf{succ}(n))}$$

So based path induction gives us a function

$$g((\mathsf{succ}(n), c_s(n, \mathsf{pr}_2\Phi'(n))), q) : \mathsf{pr}_2\Phi'(\mathsf{succ}(n)) = \mathsf{pr}_2(\mathsf{succ}(n), c_s(n, \mathsf{pr}_2\Phi'(n)))$$

and by applying the projection function on the right and discharging the assumption of n, we have shown that

$$\prod_{n:\mathbb{N}} \mathsf{pr}_2\Phi'(\mathsf{succ}(n)) = c_s(n, \mathsf{pr}_2\Phi'(n))$$

is inhabited. Next chapter we'll prove that functions are functors, and we won't have to do this based path induction every single time. It'll be great. Repeating it all in Coq, we have

```
Goal snd (Phi' 0) = c0. auto. Qed.
Goal ∀ n, Phi'(S n) = (S n, cs n (snd (Phi' n))). Admitted.
```

Exercise 1.5 (p. 56) 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 §1.7 hold.

Solution Define $A + B$ as stated. We need to define a function of type

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

which means that we also need to define $\mathsf{inl}' : A \to A + B$ and $\mathsf{inr}' B \to A + B$; these are

$$\mathsf{inl}'(a) :\equiv (0_{\mathbf{2}}, a) \qquad \mathsf{inr}'(b) :\equiv (1_{\mathbf{2}}, b)$$

In Coq, we can use *sigT* to define coprd as a Σ-type:

```
Definition coprd := {x:Bool & if x then B else A}.
Definition myinl (a : A) := existT (fun x:Bool ⇒ if x then B else A) false a.
Definition myinr (b : B) := existT (fun x:Bool ⇒ if x then B else A) true b.
```

Suppose that $C : A + B \to \mathcal{U}$, $g_0 : \prod_{(a:A)} C(\mathsf{inl}'(a))$, $g_1 : \prod_{(b:B)} C(\mathsf{inr}'(b))$, and $x : A + B$; we're looking to define

$$\mathsf{ind}'_{A+B}(C, g_0, g_1, x)$$

We will use $\mathsf{ind}_{\sum_{(x:\mathbf{2})} \mathsf{rec}_{\mathbf{2}}(\mathcal{U},A,B,x)}$, and for notational convenience will write $\Phi :\equiv \sum_{(x:\mathbf{2})} \mathsf{rec}_{\mathbf{2}}(\mathcal{U}, A, B, x)$. ind_{Φ} has signature

$$\mathsf{ind}_{\Phi} : \prod_{C:(\Phi)\to\mathcal{U}} \left(\prod_{(x:\mathbf{2})}\prod_{(y:\mathsf{rec}_{\mathbf{2}}(\mathcal{U},A,B,x))} C((x, y))\right) \to \prod_{(p:\Phi)} C(p)$$

So

$$\mathsf{ind}_{\Phi}(C) : \left(\prod_{(x:\mathbf{2})}\prod_{(y:\mathsf{rec}_{\mathbf{2}}(\mathcal{U},A,B,x))} C((x, y))\right) \to \prod_{(p:\Phi)} C(p)$$

To obtain something of type $\prod_{(x:\mathbf{2})} \prod_{(y:\mathsf{rec}_{\mathbf{2}}(\mathcal{U},A,B,x))} C((x, y))$ we'll have to use $\mathsf{ind}_{\mathbf{2}}$. In particular, for $B(x) :\equiv \prod_{(y:\mathsf{rec}_{\mathbf{2}}(\mathcal{U},A,B,x))} C((x, y))$ we have

$$\mathsf{ind}_{\mathbf{2}}(B) : B(0_{\mathbf{2}}) \to B(1_{\mathbf{2}}) \to \prod_{x:\mathbf{2}} B(x)$$

along with

$$g_0 : \prod_{a:A} C(\mathsf{inl}'(a)) \equiv \prod_{a:\mathsf{rec}_{\mathbf{2}}(\mathcal{U},A,B,0_{\mathbf{2}})} C((0_{\mathbf{2}}, a)) \equiv B(0_{\mathbf{2}})$$

and similarly for g_1. So

$$\mathsf{ind}_{\mathbf{2}}(B, g_0, g_1) : \prod_{(x:\mathbf{2})} \prod_{(y:\mathsf{rec}_{\mathbf{2}}(\mathcal{U},A,B,x))} C((x, y))$$

which is just what we needed for ind_{Φ}. So we define

$$\mathsf{ind}'_{A+B}(C, g_0, g_1, x) :\equiv \mathsf{ind}_{\sum_{(x:\mathbf{2})} \mathsf{rec}_{\mathbf{2}}(\mathcal{U},A,B,x)}\left(C, \mathsf{ind}_{\mathbf{2}}\left(\prod_{y:\mathsf{rec}_{\mathbf{2}}(\mathcal{U},A,B,x)} C((x, y)), g_0, g_1\right), x\right)$$

and, in Coq, we use sigT_rect, which is the built-in $\mathsf{ind}_{\sum_{(x:A)} B(x)}$:

```
Definition indcoprd (C : coprd → Type) (g0 : ∀ a : A, C (myinl a)) (g1 : ∀ b : B, C (myinr b)) (x : coprd) :=
```

```
sigT_rect C
          (Bool_rect (fun x:Bool ⇒ ∀ (y : if x then B else A), C (x; y))
                     g1
                     g0)
          x.
```

Now we must show that the definitional equalities

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

hold. For the first, we have

$$\begin{aligned}
\mathsf{ind}'_{A+B}(C, g_0, g_1, \mathsf{inl}'(a)) &\equiv \mathsf{ind}'_{A+B}(C, g_0, g_1, (0_{\mathbf{2}}, a)) \\
&\equiv \mathsf{ind}_{\sum_{(x:\mathbf{2})} \mathsf{rec}_{\mathbf{2}}(\mathcal{U}, A, B, x)} \left(C, \mathsf{ind}_{\mathbf{2}} \left(\prod_{y:\mathsf{rec}_{\mathbf{2}}(\mathcal{U}, A, B, x)} C((x, y)), g_0, g_1 \right), (0_{\mathbf{2}}, a) \right) \\
&\equiv \mathsf{ind}_{\mathbf{2}} \left(\prod_{y:\mathsf{rec}_{\mathbf{2}}(\mathcal{U}, A, B, x)} C((x, y)), g_0, g_1, 0_{\mathbf{2}} \right) (a) \\
&\equiv g_0(a)
\end{aligned}$$

and for the second,

$$\begin{aligned}
\mathsf{ind}'_{A+B}(C, g_0, g_1, \mathsf{inr}'(b)) &\equiv \mathsf{ind}'_{A+B}(C, g_0, g_1, (1_{\mathbf{2}}, b)) \\
&\equiv \mathsf{ind}_{\sum_{(x:\mathbf{2})} \mathsf{rec}_{\mathbf{2}}(\mathcal{U}, A, B, x)} \left(C, \mathsf{ind}_{\mathbf{2}} \left(\prod_{y:\mathsf{rec}_{\mathbf{2}}(\mathcal{U}, A, B, x)} C((x, y)), g_0, g_1 \right), (1_{\mathbf{2}}, b) \right) \\
&\equiv \mathsf{ind}_{\mathbf{2}} \left(\prod_{y:\mathsf{rec}_{\mathbf{2}}(\mathcal{U}, A, B, x)} C((x, y)), g_0, g_1, 1_{\mathbf{2}} \right) (b) \\
&\equiv g_1(b)
\end{aligned}$$

Trivial calculations, as Coq can attest:

```
Goal ∀ C g0 g1 a, indcoprd C g0 g1 (myinl a) = g0 a. trivial. Qed.
Goal ∀ C g0 g1 b, indcoprd C g0 g1 (myinr b) = g1 b. trivial. Qed.
```

Exercise 1.6 (p. 56) 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 §1.5 hold propositionally (i.e. using equality types).

Solution Define

$$A \times B :\equiv \prod_{x:\mathbf{2}} \mathsf{rec}_{\mathbf{2}}(\mathcal{U}, A, B, x)$$

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

$$(a, b) :\equiv \mathsf{ind}_{\mathbf{2}}(\mathsf{rec}_{\mathbf{2}}(\mathcal{U}, A, B), a, b)$$

Defining this type and constructor in Coq, we have

```
Definition prd := ∀ x : Bool, if x then B else A.
Definition mypair (a : A) (b : B) := Bool_rect (fun x : Bool ⇒ if x then B else A) b a.
```

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

$$C : \left(\prod_{x:\mathbf{2}} \mathsf{rec}_{\mathbf{2}}(\mathcal{U}, A, B, x)\right) \to \mathcal{U}$$
$$g : \prod_{(x:A)} \prod_{(y:B)} C(\mathsf{ind}_{\mathbf{2}}(\mathsf{rec}_{\mathbf{2}}(\mathcal{U}, A, B), x, y))$$

We can define projections by

$$\mathsf{pr}_1 p :\equiv p(0_{\mathbf{2}}) \qquad \mathsf{pr}_2 p :\equiv p(1_{\mathbf{2}})$$

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

$$p(0_{\mathbf{2}}) : \mathsf{rec}_{\mathbf{2}}(\mathcal{U}, A, B, 0_{\mathbf{2}}) \equiv A$$
$$p(1_{\mathbf{2}}) : \mathsf{rec}_{\mathbf{2}}(\mathcal{U}, A, B, 1_{\mathbf{2}}) \equiv B$$

```
Definition myfst (p : prd) := p false.
Definition mysnd (p : prd) := p true.
```

Then we have

$$g(\mathsf{pr}_1 p)(\mathsf{pr}_2 p) : C(\mathsf{ind}_{\mathbf{2}}(\mathsf{rec}_{\mathbf{2}}(\mathcal{U}, A, B), (\mathsf{pr}_1 p), (\mathsf{pr}_2 p))) \equiv C((p(0_{\mathbf{2}}), p(1_{\mathbf{2}})))$$

So we have defined a function

$$f' : \prod_{p:A\times B} C((p(0_{\mathbf{2}}), p(1_{\mathbf{2}})))$$

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

$$\mathsf{funext} : \left(\prod_{x:\mathbf{2}} ((\mathsf{pr}_1 p, \mathsf{pr}_2 p)(x) =_{\mathsf{rec}_{\mathbf{2}}(\mathcal{U},A,B,x)} p(x))\right) \to ((\mathsf{pr}_1 p, \mathsf{pr}_2 p) =_{A\times B} p)$$

We just need to show that the antecedent is inhabited, which we can do with $\mathsf{ind}_{\mathbf{2}}$. So consider the family

$$E :\equiv \lambda(x:\mathbf{2}).\, ((p(0_{\mathbf{2}}), p(1_{\mathbf{2}}))(x) =_{\mathsf{rec}_{\mathbf{2}}(\mathcal{U},A,B,x)} p(x)))$$
$$\equiv \lambda(x:\mathbf{2}).\, (\mathsf{ind}_{\mathbf{2}}(\mathsf{rec}_{\mathbf{2}}(\mathcal{U}, A, B), p(0_{\mathbf{2}}), p(1_{\mathbf{2}}), x) =_{\mathsf{rec}_{\mathbf{2}}(\mathcal{U},A,B,x)} p(x))$$

We have

$$E(0_{\mathbf{2}}) \equiv (\mathsf{ind}_{\mathbf{2}}(\mathsf{rec}_{\mathbf{2}}(\mathcal{U}, A, B), p(0_{\mathbf{2}}), p(1_{\mathbf{2}}), 0_{\mathbf{2}}) =_{\mathsf{rec}_{\mathbf{2}}(\mathcal{U},A,B,0_{\mathbf{2}})} p(0_{\mathbf{2}}))$$
$$\equiv (p(0_{\mathbf{2}}) =_{\mathsf{rec}_{\mathbf{2}}(\mathcal{U},A,B,0_{\mathbf{2}})} p(0_{\mathbf{2}}))$$

Thus $\mathsf{refl}_{p(0_\mathbf{2})} : E(0_\mathbf{2})$. The same argument goes through to show that $\mathsf{refl}_{p(1_\mathbf{2})} : E(1_\mathbf{2})$. This means that

$$h :\equiv \mathsf{ind}_\mathbf{2}(E, \mathsf{refl}_{p(0_\mathbf{2})}, \mathsf{refl}_{p(1_\mathbf{2})}) : \prod_{x:\mathbf{2}} ((\mathsf{pr}_1 p, \mathsf{pr}_2 p)(x) =_{\mathsf{rec}_\mathbf{2}(\mathcal{U},A,B,x)} p(x))$$

and thus

$$\mathsf{funext}(h) : (p(0_\mathbf{2}), p(1_\mathbf{2})) =_{A\times B} p$$

This allows us to define the uniqueness principle for products:

$$\mathsf{uppt} :\equiv \lambda p.\, \mathsf{funext}(h) : \prod_{p:A\times B} (\mathsf{pr}_1 p, \mathsf{pr}_2 p) =_{A\times B} p$$

where funext implicitly depends on p in the way we've been assuming. Now we can define $\mathsf{ind}_{A\times B}$ as

$$\mathsf{ind}_{A\times B}(C, g, p) :\equiv (\mathsf{uppt}\ p)_*(g(\mathsf{pr}_1 p)(\mathsf{pr}_2 p))$$

In Coq we can repeat this construction using Funext.

```
Context `{Funext}.
Definition myuppt (p : prd) : mypair (myfst p) (mysnd p) = p.
  apply path_forall.
  unfold pointwise_paths; apply Bool_rect; reflexivity.
Defined.
Definition indprd' (C : prd → Type) (g : ∀ (x:A) (y:B), C (mypair x y)) (z : prd) :=
  (myuppt z) # (g (myfst z) (mysnd z)).
```

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

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

is inhabited. Unfolding the left gives

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

where $D : \prod_{(x,y:A\times B)}(x = y) \to \mathcal{U}$ is given by $D(x, y, p) :\equiv C(x) \to C(y)$ and

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

Now,

$$\mathsf{uppt}\ (a, b) \equiv \mathsf{funext}(h) : (a, b) =_{A\times B} (a, b)$$

and, in particular, we have $h : x \mapsto \mathsf{refl}_{(a,b)(x)}$, so $\mathsf{funext}(h) = \mathsf{refl}_{(a,b)}$. Plugging this into $\mathsf{ind}_{=_{C((a,b))}}$ and applying its defining equality gives

$$\begin{aligned}\mathsf{ind}_{A\times B}(C, g, (a, b)) &= \mathsf{ind}_{=_{C((a,b))}}(D, d, (a, b), (a, b), \mathsf{refl}_{(a,b)})(g(a)(b)) \\ &= d((a, b))(g(a)(b)) \\ &= \mathsf{id}_{C((a,b))}(g(a)(b)) \\ &= g(a)(b)\end{aligned}$$

Verifying that the definitional equality holds propositionally. The reason we can only get propositional equality, not judgemental equality, is that $\mathsf{funext}(h) = \mathsf{refl}_{(a,b)}$ is just a propositional equality. Understanding this better requires stuff from next chapter.

```
Goal ∀ C g a b, indprd' C g (mypair a b) = g a b. Admitted.
```

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

Solution To avoid universes, we follow the plan from p. 53 of the text: show that $\mathsf{ind}_{=_A}$ entails Lemmas 2.3.1 and 3.11.8, and that these two principles imply $\mathsf{ind}'_{=_A}$ directly.

First we have Lemma 2.3.1, which states that for any type family P over A and $p : x =_A y$, there is a function $p_* : P(x) \to P(y)$. The proof for this can be taken directly from the text. Consider the type family

$$D : \prod_{x,y:A} (x = y) \to \mathcal{U}, \qquad\qquad D(x,y,p) :\equiv P(x) \to P(y)$$

which exists, since $P(x) : \mathcal{U}$ for all $x : A$ and these can be used to form function types. We also have

$$d :\equiv \lambda x.\, \mathsf{id}_{P(x)} : \prod_{x:A} D(x,x,\mathsf{refl}_x) \equiv \prod_{x:A} P(x) \to P(x)$$

We now apply $\mathsf{ind}_{=_A}$ to obtain

$$p_* :\equiv \mathsf{ind}_{=_A}(D,d,x,y,p) : P(x) \to P(y)$$

establishing the Lemma.

Next we have Lemma 3.11.8, which states that for any A and any $a : A$, the type $\sum_{(x:A)}(a = x)$ is contractible; that is, there is some $w : \sum_{(x:A)}(a = x)$ such that $w = w'$ for all $w' : \sum_{(x:A)}(a = x)$. Consider the point $(a, \mathsf{refl}_a) : \sum_{(a:A)}(a = x)$ and the family $C : \prod_{(x,y:A)}(x = y) \to \mathcal{U}$ given by

$$C(x,y,p) :\equiv ((x, \mathsf{refl}_x) =_{\sum_{(z:A)}(x=z)} (y,p))$$

Take also the function

$$\mathsf{refl}_{(x,\mathsf{refl}_x)} : \prod_{x:A} ((x, \mathsf{refl}_x) =_{\sum_{(x:A)}(x=z)} (x, \mathsf{refl}_x))$$

By path induction, then, we have a function

$$g : \prod_{(x,y:A)} \prod_{(p:x=_A y)} ((x, \mathsf{refl}_x) =_{\sum_{(z:A)}(x=z)} (y,p))$$

such that $g(x,x,\mathsf{refl}_x) :\equiv \mathsf{refl}_{(x,\mathsf{refl}_x)}$. This allows us to construct

$$\lambda p.\, g(a, \mathsf{pr}_1 p, \mathsf{pr}_2 p) : \prod_{p:\sum_{(x:A)}(a=x)} (a, \mathsf{refl}_a) =_{\sum_{(z:A)}(a=z)} (\mathsf{pr}_1 p, \mathsf{pr}_2 p)$$

And upst lets us transport this, using the first lemma, to the statement that $\sum_{(x:A)}(a = x)$ is contractible:

$$\mathsf{contr} :\equiv \lambda p.\, \Big((\mathsf{upst}\, p)_* g(a, \mathsf{pr}_1 p, \mathsf{pr}_2 p)\Big) : \prod_{p:\sum_{(x:A)}(a=x)} (a, \mathsf{refl}_a) =_{\sum_{(z:A)}(a=z)} p$$

With these two lemmas we can derive based path induction. Fix some $a : A$ and suppose we have a family

$$C : \prod_{x:A} (a = x) \to \mathcal{U}$$

and an element

$$c : C(a, \mathsf{refl}_a).$$

Suppose we have $x : A$ and $p : a = x$. Then we have $(x, p) : \sum_{(x:A)}(a = x)$, and because this type is contractible, an element $\mathsf{contr}_{(x,p)} : (a, \mathsf{refl}_a) = (x, p)$. So for any type family P over $\sum_{(x:A)}(a = x)$, we have the function $(\mathsf{contr}_{(x,p)})_* : P((a, \mathsf{refl}_a)) \to P((x, p))$. In particular, we have the type family

$$\tilde{C} :\equiv \lambda p.\, C(\mathsf{pr}_1 p, \mathsf{pr}_2 p)$$

so

$$(\mathsf{contr}_{(x,p)})_* : \tilde{C}((a, \mathsf{refl}_a)) \to \tilde{C}((x, p)) \equiv C(a, \mathsf{refl}_a) \to C(x, p).$$

thus

$$(\mathsf{contr}_{(x,p)})(c) : C(x, p)$$

or, abstracting out the x and p,

$$f :\equiv \lambda x.\, \lambda p.\, (\mathsf{contr}_{(x,p)})_*(c) : \prod_{(x:A)} \prod_{(p:x=y)} C(x, p).$$

We also have

$$\begin{aligned}
f(a, \mathsf{refl}_a) &\equiv (\mathsf{contr}_{(a,\mathsf{refl}_a)})_*(c) \\
&\equiv ((\mathsf{upst}\,(a, \mathsf{refl}_a))_* g(a, a, \mathsf{refl}_a))_*(c) \\
&\equiv ((\mathsf{upst}\,(a, \mathsf{refl}_a))_* \mathsf{refl}_{(a,\mathsf{refl}_a)})_*(c) \\
&\equiv (\mathsf{ind}_=(\lambda x.\, ((a, \mathsf{refl}_a) = x), \lambda x.\, \mathsf{id}_{(a,\mathsf{refl}_a)=x}, (a, \mathsf{refl}_a), (a, \mathsf{refl}_a), \mathsf{refl}_{(a,\mathsf{refl}_a)})\mathsf{refl}_{(a,\mathsf{refl}_a)})_*(c) \\
&\equiv (\mathsf{id}_{(a,\mathsf{refl}_a)=(a,\mathsf{refl}_a)}\mathsf{refl}_{(a,\mathsf{refl}_a)})_*(c) \\
&\equiv (\mathsf{refl}_{(a,\mathsf{refl}_a)})_*(c) \\
&\equiv \mathsf{ind}_=(\tilde{C}, \lambda x.\, \mathsf{id}_{\tilde{C}(x)}, (a, \mathsf{refl}_a), (a, \mathsf{refl}_a), \mathsf{refl}_{(a,\mathsf{refl}_a)})(c) \\
&\equiv \mathsf{id}_{\tilde{C}((a,\mathsf{refl}_a))}(c) \\
&\equiv \mathsf{id}_{C(a,\mathsf{refl}_a)}(c) \\
&\equiv c
\end{aligned}$$

So we have derived based path induction.

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

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

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

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

$$\mathsf{mult} :\equiv \mathsf{rec}_{\mathbb{N}}(\mathbb{N} \to \mathbb{N}, \lambda n.\, 0, \lambda n.\, \lambda g.\, \lambda m.\, \mathsf{add}(m, g(m)))$$

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

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

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

$$\mathsf{exp} :\equiv \mathsf{rec}_{\mathbb{N}}(\mathbb{N} \to \mathbb{N}, \lambda n. 1, \lambda n. \lambda g. \lambda m. \mathsf{mult}(m, g(m)))$$

In Coq, we can define these by

```
Fixpoint mult (n m : nat) :=
  match n with
    | 0 ⇒ 0
    | S n' ⇒ m + (mult n' m)
  end.
Notation "x * y" := (mult x y) : nat_scope.
Fixpoint myexp (e b : nat) :=
  match e with
    | 0 ⇒ S 0
    | S e' ⇒ mult b (myexp e' b)
  end.
```

To verify that $(\mathbb{N}, +, 0, \times, 1)$ is a semiring, we need stuff from Chapter 2. In particular, we need the following properties of the identity. First, for all types A and $x, y : A$, we have the inversion mapping, with type

$$p \mapsto p^{-1} : (x = y) \to (y = x)$$

and such that $\mathsf{refl}_x^{-1} \equiv \mathsf{refl}_x$ for each $x : A$. Second, for $x, y, z : A$ we have concatenation:

$$p \mapsto q \mapsto p \cdot q : (x = y) \to (y = z) \to (x = z)$$

such that $\mathsf{refl}_x \cdot \mathsf{refl}_x \equiv \mathsf{refl}_x$ for any $x : A$. To show that $(\mathbb{N}, +, 0, \times, 1)$ is a semiring, we need to verify that for all $n, m, k : \mathbb{N}$,

(i) $\prod_{(n:\mathbb{N})} 0 + n = n = n + 0$
(ii) $\prod_{(n:\mathbb{N})} 0 \times n = 0 = n \times 0.$
(iii) $\prod_{(n:\mathbb{N})} 1 \times n = n = n \times 1$
(iv) $\prod_{(n,m:\mathbb{N})} n + m = m + n$
(v) $\prod_{(n,m,k:\mathbb{N})} (n + m) + k = n + (m + k)$
(vi) $\prod_{(n,m,k:\mathbb{N})} (n \times m) \times k = n \times (m \times k)$
(vii) $\prod_{(n,m,k:\mathbb{N})} n \times (m + k) = (n \times m) + (n \times k)$
(viii) $\prod_{(n,m,k:\mathbb{N})} (n + m) \times k = (n \times k) + (m \times k)$

For (i)–(iii), we show each equality separately and then use concatenation to show the implicit third equality. We dream of next chapter, where we obtain the function ap.

(i) For all $n : \mathbb{N}$, we have

$$0 + n \equiv \mathsf{add}(0, n) \equiv n$$

so $\mathsf{refl} : \prod_{(n:\mathbb{N})} 0 + n = n$. For the other equality we'll need induction on n. For the base case, we have

$$0 + 0 \equiv \mathsf{add}(0, 0) \equiv 0.$$

so $\mathsf{refl}_0 : 0 = 0 + 0$. Fix n and suppose for the induction step that $p_n : n = n + 0$. Then we have

$$\mathsf{succ}(n) + 0 \equiv \mathsf{add}(\mathsf{succ}(n), 0) \equiv \mathsf{succ}(\mathsf{add}(n, 0))$$

so we turn again to based path induction, with the family

$$C : \prod_{m:\mathbb{N}} (n = m) \to \mathcal{U} \qquad\qquad C(m, p) :\equiv (\mathsf{succ}(n) = \mathsf{succ}(m))$$

and the element $\mathsf{refl}_{\mathsf{succ}(n)} : C(n, \mathsf{refl}_n)$. So we have

$$\mathsf{ind}'_{=}(n, C, \mathsf{refl}_{\mathsf{succ}(n)}, \mathsf{refl}_n, \mathsf{add}(n, 0), p_n) : \mathsf{succ}(n) = \mathsf{succ}(\mathsf{add}(n, 0))$$

and discharging our induction step gives

$$q :\equiv \mathsf{ind}_{\mathbb{N}}(\lambda n.\, (n = n + 0), \mathsf{refl}_0, \lambda n.\, \mathsf{ind}'_{=}(n, C, \mathsf{refl}_{\mathsf{succ}(n)}, \mathsf{refl}_n, \mathsf{add}(n, 0))) : \prod_{n:\mathbb{N}} (n = n + 0)$$

For the final equality, we use concatenation. From $\mathsf{refl}_n : 0 + n = n$ and $q_n : n = n + 0$, we have $\mathsf{refl}_n \cdot q_n : 0 + n = n + 0$.

(ii) For all $n : \mathbb{N}$,

$$0 \times n \equiv \mathsf{mult}(0, n) \equiv 0$$

so $\lambda n.\, \mathsf{refl}_0 : \prod_{(n:\mathbb{N})} 0 \times n = 0$. For the other direction, induction on n. The base case is

$$0 \times 0 = \mathsf{mult}(0, 0) = 0$$

so $\mathsf{refl}_0 : 0 = 0 \times 0$. Fixing n and supposing for the induction step that $p_n : 0 = n \times 0$, we have

$$\mathsf{mult}(\mathsf{succ}(n), 0) \equiv 0 + \mathsf{mult}(n, 0) \equiv \mathsf{add}(0, \mathsf{mult}(n, 0)) \equiv \mathsf{mult}(n, 0)$$

so $p_n : 0 = \mathsf{succ}(n) \times 0$. Thus

$$q :\equiv \mathsf{ind}_{\mathbb{N}}(\lambda n.\, (0 = n \times 0), \mathsf{refl}_0, \lambda n.\, \mathsf{id}_{n=n\times 0}) : \prod_{n:\mathbb{N}} (n = n \times 0).$$

And again, $\mathsf{refl}_0 \cdot q_n : 0 \times n = n \times 0$ gives us the last equality.

(iii) For all $n : \mathbb{N}$,

$$1 \times n \equiv \mathsf{succ}(0) \times n \equiv n + (0 \times n) \equiv n + 0$$

so, recalling q_n from (i), we have $\mathsf{refl}_{1\times n} \cdot q_n^{-1} : 1 \times n = n$. For the other direction, we proceed by induction on n. For the base case we have

$$0 \times 1 \equiv \mathsf{mult}(0, 1) \equiv 0$$

so $\mathsf{refl}_0 : 0 = 0 \times 1$. Fixing n and supposing for induction that $p_n : n = n \times 1$, we have

$$\mathsf{mult}(\mathsf{succ}(n), 1) \equiv 1 + \mathsf{mult}(n, 1) \equiv \mathsf{succ}(0) + \mathsf{mult}(n, 1) \equiv \mathsf{succ}(n \times 1)$$

So we turn to based path induction again. Let $C(m) = \mathsf{succ}(n) = \mathsf{succ}(m)$; then

$$\mathsf{ind}'_{=}(n, C, \mathsf{refl}_{\mathsf{succ}(n)}, n \times 1, p_n) : \mathsf{succ}(n) = \mathsf{succ}(n \times 1)$$

and

$$r :\equiv \mathsf{ind}_{\mathbb{N}}(\lambda n.\, (n = n \times 1), \mathsf{refl}_0, \lambda n.\, \mathsf{ind}'_{=}(n, C, \mathsf{refl}_{\mathsf{succ}(n)}, n \times 1)) : \prod_{n:\mathbb{N}} (n = n \times 1)$$

For the third equality, finally, $\mathsf{refl}_{1\times n} \cdot q_n^{-1} \cdot r_n : 1 \times n = n \times 1$.

(iv) We first prove an auxiliary lemma by induction: $\prod_{(n,m:\mathbb{N})} \mathsf{succ}(n+m) = n + \mathsf{succ}(m)$. For the base case, we have $\mathsf{succ}(0+m) \equiv \mathsf{succ}(m) \equiv 0 + \mathsf{succ}(m)$, so $\mathsf{refl}_{\mathsf{succ}(m)} : \mathsf{succ}(0+m) = 0 + \mathsf{succ}(m)$. Fix $n : \mathbb{N}$, and suppose for induction that $p_n : \mathsf{succ}(n+m) = n + \mathsf{succ}(m)$. Then

$$\mathsf{succ}(\mathsf{succ}(n) + m) \equiv \mathsf{succ}(\mathsf{succ}(n+m))$$

and based path induction on $C(m) :\equiv \mathsf{succ}(\mathsf{succ}(n+m)) = \mathsf{succ}(m)$ gives

$$\mathsf{ind}'_{=}(\mathsf{succ}(n+m), C, \mathsf{refl}_{\mathsf{succ}(\mathsf{succ}(n+m))}, n + \mathsf{succ}(m), p_n) : \mathsf{succ}(\mathsf{succ}(n+m)) = \mathsf{succ}(n + \mathsf{succ}(m))$$

so letting $D(n) :\equiv \prod_{(m:\mathbb{N})}(\mathsf{succ}(n+m) = n + \mathsf{succ}(m))$,

$$r :\equiv \mathsf{ind}_{\mathbb{N}}(D, \mathsf{refl}_{\mathsf{succ}(m)}, \lambda n.\, \mathsf{ind}'_{=}(\mathsf{succ}(n+m), C, \mathsf{refl}_{\mathsf{succ}(\mathsf{succ}(n+m))}, n + \mathsf{succ}(m))) : \prod_{n:\mathbb{N}} D(n)$$

We now proceed by induction on n to show (iv). For the base case, recalling q_n from (i), we have $\mathsf{refl}_m \cdot q_m : 0 + m = m + 0$. Fixing n and supposing for induction that $p_n : n + m = m + n$, we have

$$\mathsf{succ}(n) + m \equiv \mathsf{succ}(n+m)$$

We then apply based path induction on $E(k) :\equiv \mathsf{succ}(n+m) = \mathsf{succ}(k)$ to obtain

$$\begin{aligned} \mathsf{ind}'_{=}(n+m, E, \mathsf{refl}_{\mathsf{succ}(n+m)}, m+n, p_n) &: \mathsf{succ}(n) + m = \mathsf{succ}(m+n) \\ \mathsf{ind}'_{=}(n+m, E, \mathsf{refl}_{\mathsf{succ}(n+m)}, m+n, p_n) \cdot r_{m,n} &: \mathsf{succ}(n) + m = m + \mathsf{succ}(n) \end{aligned}$$

and, finally, for the family $F(n) = n + m = m + n$,

$$\mathsf{ind}_{\mathbb{N}}(F, \mathsf{refl}_m \cdot q_m, \lambda n.\, \lambda p.\, (\mathsf{ind}'_{=}(n+m, E, \mathsf{refl}_{\mathsf{succ}(n+m)}, m+n, p) \cdot r_{m,n})) : \prod_{n:\mathbb{M}} n + m = m + m$$

Abstracting out the m gives us (iv).

(v) Fix m and k. We proceed by induction on n. For the base case,

$$(0+m) + k \equiv m + k \equiv 0 + (m+k)$$

By the definition of add. Fix n, and suppose that $p_n : (n+m) + k = n + (m+k)$. We have

$$(\mathsf{succ}(n) + m) + k \equiv \mathsf{succ}(n+m) + k \equiv \mathsf{succ}((n+m)+k)$$

So based path induction on $C(\ell) = \mathsf{succ}((n+m)+k) = \mathsf{succ}(\ell)$ gives

$$\mathsf{ind}'_{=}((n+m)+k, C, \mathsf{refl}_{\mathsf{succ}((n+m)+k)}, n + (m+k), p_n) : \mathsf{succ}((n+m)+k) = \mathsf{succ}(n + (m+k))$$

which is equivalently the type $(\mathsf{succ}(n) + m) + k = \mathsf{succ}(n) + (m+k)$. So induction over $D(n) = (n+m) + k = n + (m+k)$ gives

$$\mathsf{ind}_{\mathbb{N}}(D, \mathsf{refl}_{(0+m)+k}, \lambda n.\, \lambda p.\, \mathsf{ind}'_{=}((n+m)+k, C, \mathsf{refl}_{\mathsf{succ}((n+m)+k)}, n + (m+k), p)) : \prod_{n:\mathbb{N}} D(n)$$

and abstracting out the m and k gives us (v).

(vi) Fix m and k. First an auxiliary lemma; we show that $(n+m) \times k = (n \times k) + (m \times k)$ by induction on n. For the base case,

$$(0+m) \times k \equiv m \times k \equiv 0 + (m \times k) \equiv (0 \times k) + (m \times k)$$

Now fix n and suppose that $p_n : (n+m) \times k = n \times k + m \times k$.

$$(\mathsf{succ}(n) + m) \times k \equiv \mathsf{succ}(n+m) \times k \equiv k + (n+m) \times k$$

and

$$\mathsf{succ}(n) \times k + m \times k \equiv (k + n \times k) + m \times k$$

Using based path induction over $C(\ell) :\equiv k + (n+m) \times k = k + \ell$, we get

$$\mathsf{ind}'_{=}((n+m) \times k, C, \mathsf{refl}_{k+(n+m)\times k}, n \times k + m \times k, p_n) : k + (n+m) \times k = k + (n \times k + m \times k)$$

We established in (v) that addition is associative, so we have some

$$r^{-1}_{k,n\times k,m\times k} : k + (n \times k + m \times k) = (k + n \times k) + m \times k$$

and concatenating this with the result of the based path induction gives something of type

$$k + (n+m) \times k = (k + n \times k) + m \times k$$

0ur two strings of judgemental equalities mean that this is the same as the type

$$(\mathsf{succ}(n) + m) \times k = \mathsf{succ}(n) \times k + m \times k.$$

So we can now perform the induction over $D(\ell) = (n+m) \times k = n \times k + m \times k$ to obtain

$$\mathsf{ind}_{\mathbb{N}}(D, \mathsf{refl}_{(0+m)\times k}, \lambda n.\, \lambda p.\, (\mathsf{ind}'_{=}((n+m) \times k, C, \mathsf{refl}_{k+(n+m)\times k}, n \times k + m \times k, p_n) \cdot r^{-1}_{k,n\times k,m\times k}))$$

which is of type

$$\prod_{n:\mathbb{N}} (n+m) \times k = n \times k + m \times k$$

abstracting out the m and k give the final result (i.e., that multiplication on the right distributes over addition).

Now, for (vi). As always, it's induction on n. For the base case

$$(0 \times m) \times k \equiv 0 \times k \equiv 0 \equiv 0 \times (m \times k)$$

Now fix n and assume that $p_n : (n \times m) \times k = n \times (m \times k)$. We have

$$(\mathsf{succ}(n) \times m) \times k \equiv (m + n \times m) \times k$$

and

$$\mathsf{succ}(n) \times (m \times k) \equiv m \times k + n \times (m \times k)$$

From our lemma, then, there is a function

$$q : \prod_{n:\mathbb{N}} (\mathsf{succ}(n) \times m) \times k = m \times k + (n \times m) \times k$$

we use based path induction over $E(\ell) :\equiv m \times k + \ell$ to obtain

$$\mathsf{ind}'_{=}((n \times m) \times k, E, \mathsf{refl}_{m\times k+(n\times m)\times k}, n \times (m \times k), p_n) : m \times k + (n \times m) \times k = m \times k + n \times (m \times k)$$

which, concatenated with q_n and altered by the second judgemental equality, gives something of type

$$(\mathsf{succ}(n) \times m) \times k = \mathsf{succ}(n) \times (m \times k)$$

So our induction principle over $F(\ell) :\equiv (n \times m) \times k = n \times (m \times k)$ gives

$$\mathsf{ind}_{\mathbb{N}}(F, \mathsf{refl}_{(0\times m)\times k}, \lambda n.\, \lambda p.\, (q_n \cdot \mathsf{ind}'_{=}((n \times m) \times k, E, \mathsf{refl}_{m\times k+(n\times m)\times k}, n \times (m \times k), p_n)))$$

of type

$$\prod_{n:\mathbb{N}} (n \times m) \times k = n \times (m \times k)$$

and abstracting out the m and k gives (vi).

(vii) Fix m and k. We proceed by induction on n. For the base case we have

$$0 \times (m+k) \equiv 0 \equiv 0 + 0 \equiv (0 \times m) + (0 \times k)$$

So fix $n : \mathbb{N}$ and suppose that $p_n : n \times (m+k) = (n \times m) + (n \times k)$. We have

$$\mathsf{succ}(n) \times (m+k) \equiv (m+k) + n \times (m+k)$$

and

$$(\mathsf{succ}(n) \times m) + (\mathsf{succ}(n) \times k) \equiv (m + n \times m) + (k + n \times k)$$

Now by (iv) and (v) we have the following two functions

$$q : \prod_{n,m:\mathbb{N}} n + m = m + n \qquad\qquad r : \prod_{n,m,k:\mathbb{N}} (n+m) + k = n + (m+k)$$

A long chain of based path inductions allows us to construct an object of type

$$(\mathsf{succ}(n) \times m) + (\mathsf{succ}(n) \times k) = (m+k) + (n \times m + n \times k)$$

In the interest of masochism, I'll do them explicitly. We start with

$$r_1 :\equiv r_{m, n\times m, k + n \times k} : (m + n \times m) + (k + n \times k) = m + (n \times m + (k + n \times k))$$

Based path induction over $C_1(\ell) :\equiv m + (n \times m + (k + n \times k)) = m + \ell$ and using

$$r_2 :\equiv r_{n \times m, k, n \times k} : n \times m + (k + n \times k) = (n \times m + k) + n \times k$$

gives

$$\langle r_2 \rangle :\equiv \mathsf{ind}'_{=}(n \times m + (k + n \times k), C_1, \mathsf{refl}_{m + (n \times m + (k + n \times k))}, (n \times m + k) + n \times k, r_2)$$

which results in

$$r_1 \cdot \langle r_2 \rangle : (m + n \times m) + (k + n \times k) = m + ((n \times m + k) + n \times k)$$

Next consider

$$q_1 :\equiv q_{n \times m, k} : n \times m + k = k + n \times m$$

which is passed through a based path induction on $C_2(\ell) :\equiv m + ((n \times m + k) + n \times k) = m + (\ell + n \times k)$ to get

$$\langle q_1 \rangle :\equiv \mathsf{ind}'_{=}(n \times m + k, C_2, \mathsf{refl}_{m + ((n \times m + k) + n \times k)}, k + n \times m, q_1)$$

which adds to our chain, giving

$$r_1 \cdot \langle r_2 \rangle \cdot \langle q_1 \rangle : (m + n \times m) + (k + n \times k) = m + ((k + n \times m) + n \times k)$$

Now just two applications of associativity are left. We have

$$r_3 :\equiv r_{k, n \times m, n \times k} : (k + n \times m) + n \times k = k + (n \times m + n \times k)$$

so for $C_3(\ell) :\equiv m + ((k + n \times m) + n \times k) = m + \ell$, we have

$$\langle r_3 \rangle :\equiv \mathsf{ind}'_{=}((k + n \times m) + n \times k, C_3, \mathsf{refl}_{m + ((k + n \times m) + n \times k)}, k + (n \times m + n \times k), r_3)$$

making our chain of type

$$r_1 \cdot \langle r_2 \rangle \cdot \langle q_1 \rangle \cdot \langle r_3 \rangle : (m + n \times m) + (k + n \times k) = m + (k + (n \times m + n \times k))$$

Finally, take

$$r_4 :\equiv r^{-1}_{m,k,n\times m+n\times k} : m + (k + (n \times m + n \times k)) = (m + k) + (n \times m + n \times k)$$

so after applying the last judgemental equality above, we have

$$f :\equiv r_1 \cdot \langle r_2 \rangle \cdot \langle q_1 \rangle \cdot \langle r_3 \rangle \cdot r_4 : (\mathsf{succ}(n) \times m) + (\mathsf{succ}(n) \times k) = (m + k) + (n \times m + n \times k)$$

Now, consider the family $D(\ell) :\equiv (m + k) + n \times (m + k) = (m + k) + \ell$. Based path induction once more gives us

$$\mathsf{ind}'_{=}(n \times (m + k), D, \mathsf{refl}_{(m+k)+n\times(m+k)}, n \times m + n \times k, p_n) \cdot f^{-1}$$

which, after application of our judgemental equalities, is of type

$$\mathsf{succ}(n) \times (m + k) = (\mathsf{succ}(n) \times m) + (\mathsf{succ}(n) \times k)$$

So we can at last apply induction over $\mathbb{N}$, using the family $E(n) : n \times (m + k) = (n \times m) + (n \times k)$, giving

$$\mathsf{ind}_{\mathbb{N}}(E, \mathsf{refl}_{0\times(m+k)}, \lambda n. \lambda p. (\mathsf{ind}'_{=}(n \times (m + k), D, \mathsf{refl}_{(m+k)+n\times(m+k)}, n \times m + n \times k, p) \cdot f^{-1}))$$

which is of type

$$\prod_{n:\mathbb{N}} n \times (m + k) = (n \times m) + (n \times k)$$

and m and k may be abstracted out to give (vii).

(viii) This was shown as a lemma in proving (vi).

In Coq we'll do things a touch out of order, so as to appeal to (viii) in the proof of (vi).

```
Theorem plus_0_r : ∀ (n : nat), n = n + 0.
Proof.
  induction n; [reflexivity | simpl; rewrite ← IHn; reflexivity].
Qed.

Theorem ex1_8_i : ∀ (n : nat),
  (0 + n = n) ∧ (n = n + 0) ∧ (0 + n = n + 0).
Proof.
  split; [reflexivity | split; rewrite ← plus_0_r; reflexivity].
Qed.

Theorem mult_0_r : ∀ (n : nat), 0 = n × 0.
Proof.
  induction n; [reflexivity
               | simpl; rewrite ← IHn; reflexivity].
Qed.

Theorem ex1_8_ii : ∀ (n : nat),
  (0 × n = 0) ∧ (0 = n × 0) ∧ (0 × n = n × 0).
Proof.
  split; [reflexivity | split; rewrite ← mult_0_r; reflexivity].
Qed.
```

```
Theorem mult_1_r : ∀ (n : nat), n = n × 1.
Proof.
  induction n.
  reflexivity.
  simpl. rewrite ← IHn. reflexivity.
Qed.
  Theorem mult_1_l : ∀ (n : nat), 1 × n = n.
Proof.
  simpl.
  intro n.
  rewrite ← plus_0_r.
  reflexivity.
Qed.
Theorem ex1_8_iii : ∀ (n : nat),
  (1 × n = n) ∧ (n = n × 1) ∧ (1 × n = n × 1).
Proof.
  split; [rewrite mult_1_l; reflexivity | split; rewrite ← mult_1_r].
  reflexivity.
  rewrite mult_1_l. reflexivity.
Qed.
Theorem plus_n_Sm : ∀ (n m : nat), S (n + m) = n + (S m).
Proof.
  intros n m.
  induction n.
  reflexivity.
  simpl. rewrite IHn. reflexivity.
Qed.
Theorem ex1_8_iv : ∀ (n m : nat), n + m = m + n.
Proof.
  intros n m.
  induction n.
  rewrite ← plus_0_r. reflexivity.
  simpl. rewrite ← plus_n_Sm. rewrite IHn. reflexivity.
Qed.
Theorem ex1_8_v : ∀ (n m k : nat),
  (n + m) + k = n + (m + k).
Proof.
  intros n m k.
  induction n.
  reflexivity.
  simpl. rewrite IHn. reflexivity.
Qed.
Theorem ex1_8_viii : ∀ (n m k : nat),
  (n + m) × k = (n × k) + (m × k).
Proof.
  intros n m k.
  induction n.
  reflexivity.
  simpl. rewrite IHn. rewrite ex1_8_v. reflexivity.
Qed.
```

```
Theorem ex1_8_vi : ∀ (n m k : nat),
  (n × m) × k = n × (m × k).
Proof.
  intros n m k.
  induction n.
  reflexivity.
  simpl. rewrite ← IHn. rewrite ← ex1_8_viii. reflexivity.
Qed.
Theorem ex1_8_vii : ∀ (n m k : nat),
  n × (m + k) = (n × m) + (n × k).
Proof.
  intros n m k.
  induction n.
  reflexivity.
  simpl. rewrite IHn. rewrite ← ex1_8_v. rewrite ← ex1_8_v.
  cut (m + n × m + k = m + k + n × m). intro H. rewrite H. reflexivity.
  rewrite ex1_8_v.
  cut (n × m + k = k + n × m). intro H. rewrite H. rewrite ← ex1_8_v. reflexivity.
  rewrite ex1_8_iv. reflexivity.
Qed.
```

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

Solution $\mathsf{Fin}(n)$ is a type with exactly n elements. Consider $\mathsf{Fin}(n)$ from the types-as-propositions point of view; if $x : \mathsf{Fin}(n)$, then x must be one of the n elements. So we would like a type that is guaranteed to have n elements. Recalling that $\sum_{(m:\mathbb{N})}(m < n)$ may be regarded as "the type of all elements $m : \mathbb{N}$ such that $(m < n)$", we note that there are n such elements, and define

$$\mathsf{Fin}(n) :\equiv \sum_{m:\mathbb{N}} (m < n) \equiv \sum_{m:\mathbb{N}} \sum_{k:\mathbb{N}} (n + k = m)$$

We should show that $\mathsf{Fin}(n)$ has exactly n elements, which we'll do by induction. We have

To define fmax, note that we are looking for the element of $\mathsf{Fin}(n+1)$ that is greater than all others. That is,

$$\mathsf{fmax}(n) : \sum_{(m:\mathsf{Fin}(n+1))} \prod_{(k:\mathsf{Fin}(n+1))} (k \leq m)$$

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

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

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

Solution ack must be of the form

$$\mathsf{ack} :\equiv \mathsf{rec}_{\mathbb{N}}(\mathbb{N} \to \mathbb{N}, \Phi, \Psi)$$

with

$$\Phi : \mathbb{N} \to \mathbb{N} \qquad \Psi : \mathbb{N} \to (\mathbb{N} \to \mathbb{N}) \to (\mathbb{N} \to \mathbb{N})$$

which we can determine by their intended behaviour. We have

$$\mathsf{ack}(0, n) \equiv \mathsf{rec}_{\mathbb{N}}(\mathbb{N} \to \mathbb{N}, \Phi, \Psi, 0)(n) \equiv \Phi(n)$$

So we must have $\Phi :\equiv \mathsf{succ}$, which is of the correct type. The next equation gives us

$$\begin{aligned}
\mathsf{ack}(\mathsf{succ}(m), 0) &\equiv \mathsf{rec}_{\mathbb{N}}(\mathbb{N} \to \mathbb{N}, \mathsf{succ}, \Psi, \mathsf{succ}(m))(0) \\
&\equiv \Psi(m, \mathsf{rec}_{\mathbb{N}}(\mathbb{N} \to \mathbb{N}, \mathsf{succ}, \Psi, m))(0) \\
&\equiv \Psi(m, \mathsf{ack}(m, -), 0)
\end{aligned}$$

Suppose that Ψ is also defined in terms of $\mathsf{rec}_{\mathbb{N}}$. We know its signature, giving the first arg, and this second equation gives its behavior on 0, the second arg. So it must be of the form

$$\Psi = \lambda m.\, \lambda r.\, \mathsf{rec}_{\mathbb{N}}(\mathbb{N}, r(1), \Theta(m, r)) \qquad \Theta : \mathbb{N} \to (\mathbb{N} \to \mathbb{N}) \to \mathbb{N} \to \mathbb{N} \to \mathbb{N}$$

The final equation fixes Θ:

$$\begin{aligned}
&\mathsf{ack}(\mathsf{succ}(m), \mathsf{succ}(n)) \\
&\equiv \mathsf{rec}_{\mathbb{N}}(\mathbb{N} \to \mathbb{N}, \mathsf{succ}, \lambda m.\, \lambda r.\, \mathsf{rec}_{\mathbb{N}}(\mathbb{N}, r(1), \Theta(m, r)), \mathsf{succ}(m))(\mathsf{succ}(n)) \\
&\equiv \mathsf{rec}_{\mathbb{N}}(\mathbb{N}, \mathsf{ack}(m, 1), \Theta(m, \mathsf{ack}(m, -)), \mathsf{succ}(n)) \\
&\equiv \Theta(m, \mathsf{ack}(m, -), n, \mathsf{rec}_{\mathbb{N}}(\mathbb{N}, \mathsf{ack}(m, 1), \Theta(m, \mathsf{ack}(m, -)), n)) \\
&\equiv \Theta(m, \mathsf{ack}(m, -), n, \Psi(m, \mathsf{ack}(m, -), n))
\end{aligned}$$

Looking at the second equation again suggests that the final argument to Θ is really $\mathsf{ack}(\mathsf{succ}(m), n)$. Supposing this is true,

$$\Theta :\equiv \lambda m.\, \lambda r.\, \lambda n.\, \lambda s.\, r(s)$$

should work. Putting it all together, we have

$$\mathsf{ack} :\equiv \mathsf{rec}_{\mathbb{N}}(\mathbb{N} \to \mathbb{N}, \mathsf{succ}, \lambda m.\, \lambda r.\, \mathsf{rec}_{\mathbb{N}}(\mathbb{N}, r(1), \lambda n.\, \lambda s.\, r(s)))$$

In Coq, we define

```
Definition ack : nat → nat → nat :=
    nat_rect (fun _ ⇒ nat → nat)
             S
             (fun m r ⇒ nat_rect (fun _ ⇒ nat)
                                 (r (S 0))
                                 (fun n s ⇒ (r s))).
```

Now, to show that the three equations hold, we just calculate

$$\mathsf{ack}(0, n) \equiv \mathsf{rec}_{\mathbb{N}}(\mathbb{N} \to \mathbb{N}, \mathsf{succ}, \lambda m.\, \lambda r.\, \mathsf{rec}_{\mathbb{N}}(\mathbb{N}, r(1), \lambda n.\, \lambda s.\, r(s)), 0)(n) \equiv \mathsf{succ}(n)$$

for the first,

$$\begin{aligned}
\mathsf{ack}(\mathsf{succ}(m), 0) &\equiv \mathsf{rec}_{\mathbb{N}}(\mathbb{N} \to \mathbb{N}, \mathsf{succ}, \lambda m.\, \lambda r.\, \mathsf{rec}_{\mathbb{N}}(\mathbb{N}, r(1), \lambda n.\, \lambda s.\, r(s)), \mathsf{succ}(m))(0) \\
&\equiv \mathsf{rec}_{\mathbb{N}}(\mathbb{N}, \mathsf{ack}(m, 1), \lambda n.\, \lambda s.\, \mathsf{ack}(m, s), 0) \\
&\equiv \mathsf{ack}(m, 1)
\end{aligned}$$

for the second, and finally

$$\begin{aligned}\mathsf{ack}(\mathsf{succ}(m),\mathsf{succ}(n)) &\equiv \mathsf{rec}_{\mathbb{N}}(\mathbb{N}\to\mathbb{N},\mathsf{succ},\lambda m.\,\lambda r.\,\mathsf{rec}_{\mathbb{N}}(\mathbb{N},r(1),\lambda n.\,\lambda s.\,r(s)),\mathsf{succ}(m))(\mathsf{succ}(n))\\ &\equiv \mathsf{rec}_{\mathbb{N}}(\mathbb{N},\mathsf{ack}(m,1),\lambda n.\,\lambda s.\,\mathsf{ack}(m,s),\mathsf{succ}(n))\\ &\equiv \mathsf{ack}(m,\mathsf{rec}_{\mathbb{N}}(\mathbb{N},\mathsf{ack}(m,1),\lambda n.\,\lambda s.\,\mathsf{ack}(m,s),n))\end{aligned}$$

Focus on the second argument of the outer ack. We have

$$\begin{aligned}\mathsf{ack}(\mathsf{succ}(m),n) &\equiv \mathsf{rec}_{\mathbb{N}}(\mathbb{N}\to\mathbb{N},\mathsf{succ},\lambda m.\,\lambda r.\,\mathsf{rec}_{\mathbb{N}}(\mathbb{N},r(1),\lambda n.\,\lambda s.\,r(s)),\mathsf{succ}(m))(n)\\ &\equiv \mathsf{rec}_{\mathbb{N}}(\mathbb{N},\mathsf{ack}(m,1),\lambda n.\,\lambda s.\,\mathsf{ack}(m,s),n)\end{aligned}$$

and so we may substitute it back in to get

$$\mathsf{ack}(\mathsf{succ}(m),\mathsf{succ}(n)) \equiv \mathsf{ack}(m,\mathsf{ack}(\mathsf{succ}(m),n))$$

which is the third equality. In Coq,

```
Goal ∀ n, ack 0 n = S n. auto. Qed.
Goal ∀ m, ack (S m) 0 = ack m (S 0). auto. Qed.
Goal ∀ m n, ack (S m) (S n) = ack m (ack (S m) n). auto. Qed.
```

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

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 $\neg\neg\neg A \to \neg A$.

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

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

But then we have

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

so discharging the a gives

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

And discharging the first assumption gives

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

This is automatic for Coq, though not trivial:

```
Goal ∀ A, ¬ ¬ ¬ A → ¬A. auto. Qed.
```

We can get a proof out of Coq by printing this `Goal`. It returns

```
fun (A : Type) (X : ¬ ¬ ¬ A) (X0 : A) ⇒ X (fun X1 : A → Empty ⇒ X1 X0)
    : ∀ A : Type, ¬ ¬ ¬ A → ¬ A
```

which is just the function obtained by hand.

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, $A \to B \to A$. That is, we have

$$\lambda(a:A).\,\lambda(b:B).\,a : A \to B \to A$$

and in Coq,

```
Goal A → B → A. trivial. Qed.
```

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

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

```
Goal A → ¬ ¬ A. auto. Qed.
```

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

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

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

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

So

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

```
Goal (¬ A + ¬ B) → ¬ (A × B).
Proof.
  unfold not.
  intros H x.
  apply H.
  destruct x.
  constructor.
  exact a.
Qed.
```

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 $g : P + (P \to \mathbf{0}) \to \mathbf{0}$. Assuming $p : P$, disjunction introduction results in $\mathsf{inl}(p) : P + (P \to \mathbf{0})$. But then $g(\mathsf{inl}(p)) : \mathbf{0}$, so we discharge the assumption of $p : P$ to get

$$\lambda(p:P).\,g(\mathsf{inl}(p)) : P \to \mathbf{0}$$

Applying disjunction introduction again leads to contradiction, as

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

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

$$\lambda(g : P + (P \to \mathbf{0}) \to \mathbf{0}).\, g(\mathsf{inr}(\lambda(p : P).\, g(\mathsf{inl}(p)))) : (P + (P \to \mathbf{0}) \to \mathbf{0}) \to \mathbf{0}$$

Finally, in Coq,

```
Goal ¬ ¬ (P + ¬P).
Proof.
  unfold not.
  intro H.
  apply H.
  right.
  intro p.
  apply H.
  left.
  apply p.
Qed.
```

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

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

Solution

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

Solution Consider some family $C : A \to \mathcal{U}$, and define

$$D : \prod_{x,y:A} (x =_A y) \to \mathcal{U}, \qquad D(x, y, p) :\equiv C(x) \to C(y)$$

Note that we have the function

$$\lambda x.\, \mathsf{id}_{C(x)} : \prod_{x:A} C(x) \to C(x) \equiv \prod_{x:A} D(x, x, \mathsf{refl}_x)$$

So by path induction there is a function

$$f : \prod_{(x,y:A)} \prod_{(p:x=_A y)} D(x, y, p) \equiv \prod_{(x,y:A)} \prod_{(p:x=_A y)} C(x) \to C(y)$$

such that

$$f(x, x, \mathsf{refl}_x) :\equiv \mathsf{id}_{C(x)}$$

But this is just the statement of the indiscernability of identicals: for every such family C, there is such an f.