Exercises in Type Theory

These exercises should help you get familiar with type theory as a language for writing down mathematical structures, constructions, statements, and proofs. If you are only beginning to use type theory, you will likely have many questions, as even simple notational conventions and basics will be new to you. We encourage you to work together with others, and to ask lots of questions!

You should try to solve the exercises on paper as well as with Coq (see the accompanying file Spartan_exercises.v). The paper is good for understanding and Coq is good for precision.

Exercise 1

For each of the following types, write down an element of that type, if it has one. If it does not have any elements, you should establish that this is the case. Recall the lecture: to show that A has no elements, we construct an element of `A → empty`.

1. `A × (B + C) → A × B + A × C`, given types `A`, `B`, and `C`
2. `(A → A) → (A → A)`, given type `A` (for extra credit, write down *five* elements of this type)
3. `Id_nat (0, succ 0)`
4. `∑ (A : Universe) (A → empty) → empty`
5. `∏ (n : nat), ∑ (m : nat), Id_nat n (2 · m) + Id_nat n (2 · m + 1)`, assuming you have got arithmetic.
6. `(∑ (x : A) B × P x) → B × ∑ (x : A) P x`, given types `A`, `B`, and `P : A → Universe`
7. `B → (B → A) → A`, given types `A` and `B`
8. `B → ∏ (A : Universe) (B → A) → A`, given type `B`
9. `(∏ (A : Universe) (B → A) → A) → B`, given type `B`

Exercise 2

1. Using the basic rules for natural numbers, construct addition on natural numbers.
2. State associativity and commutativity of addition in a type-theoretic way.

Exercise 3

Write down the following types and elements:

1. even natural numbers, and `4` as an element of this type
2. prime numbers, and `7` as an element of this type
3. functions `A → nat` with a given argument at which they are zero (set-theoretically this would be `{ (x, f) ∈ A × (A → nat) | f x = 0 }`).
4. pairs `(m, n)` of natural numbers such that `m ≤ n`, and `(2, 4)` as an element of this type.

Exercise 4

We say that a dependent type `P : A → Universe` is decidable if there is an element of `∏ (x:A), P x + (P x → Empty)`. Which of the following are decidable?

1. `P : bool → Universe`, defined by `P true ≡ unit` and `P false ≡ empty`.
2. `P : nat → Universe`, defined by `P n ≡ Id_nat (n, 42)`.