Lab 10: Propositions as types

Functional Programming (ITI0212)

This week we are learning about the *propositions as types* paradigm. It tells us that propositions (the mathematical statements) are in fact the same things as types (the functional programming objects). This is known as the Curry-Howard correspondence. When we have `t : T`, then it can be read both as *t is a term of type T* and *t is a proof of the proposition T*. We see here that terms becomes proofs, and constructing a term of a type (= proposition) is the exact same thing as constructing a proof of this type (= proposition).

Facts about less or equal

As a general hint for this section, do not forget that you can use previous result to prove new ones. Recall the type ⩽′ from the lecture:

Task 1
Prove that ⩽′ is transitive, which means you need to give a function of the following type: `leTrans : {m n p : Nat} → (m ⩽′ n) → (n ⩽′ p) → (m ⩽′ p)`

Task 2
State and prove that any integer is less or equal than its successor. This will be the function of the following type : `succLarger : {n : Nat} → n ⩽′ (n + 1)`

Task 3
Prove the following : `leWeakRight : {m n : Nat} → (m ⩽′ n) → m ⩽′ (n + 1)` and `leWeakLeft : {m n : Nat} → ((m + 1) ⩽′ n) → (m ⩽′ n)`

Task 4
Prove the following: `zeroPlusLeft : {m n : Nat} → (0 + n) ⩽′ m + n`

On the length of lists

Task 5
State and prove that if two lists `xs, ys` are such that `length xs ⩽′ length ys`, then `length (x :: xs) ⩽′ length (y :: ys)`.