Assignment 4

Deadline for solutions: 6.07.2022

Exercise 1 Formal Type Theory (7 Points)

This is a pen-and-paper exercise on understanding the formal development of the Martin-Löf type theory. We refer to the rules from the lecture for constructing derivations.

a) Spell out the rules for non-dependent functions spaces $A \to B$, that is, you need to inspect the rules for $\prod_{x:\,A} B$, and eliminate all ineffective dependencies on x.

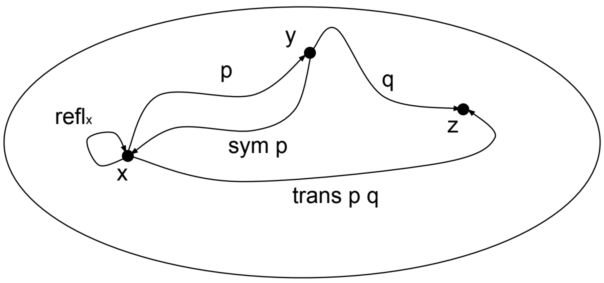
b) Extend the type theory from the lecture with a type of Booleans. Take inspiration from the existing rules for coproducts $A + B$ and note that Booleans corresponds to the case of both A and B being unit types.

c) Analogously to the introduction rule for reflexivity of propositional equality, introduce the corresponding rules for symmetry and transitivity. Prove that they are derivable.

Hint: The case of transitivity is a little tricky: from $a \equiv b$ you should first derive $b \equiv c \to a \equiv c$ and then use the elimination rule for function spaces to derive $a \equiv c$ using the other assumption, i.e. that $b \equiv c$.

Exercise 2 Without K (15 Points)

Implement solutions to the following problems in Agda with the pragma `{-# OPTIONS --without-K #-}` activated*. This corresponds to the general version of Martin-Löf type theory with the elimination principle for the identity types, as explained at the lecture. As a result, proofs of equations themselves become subject to (nontrivial) proofs.



The following intuition is helpful when working with such proofs. You can think of $p : x \equiv y$ as a *path* from x to y on a surface. Then *refl* $: x \equiv x$ is a one point path, symmetry produces a reversed path $(sym\ p) : y \equiv x$, and transitivity concatenates two paths. For example, you can show that $trans\ p\ (sym\ p) \equiv refl$ (do it!). This is called the *groupoid interpretation* of type theory.

The following variant of the identity type eliminator

```
J' : ∀ {A : Set ℓ} {x : A} (P : (z : A) → x ≡ z → Set ℓ)
  → P x refl → (y : A) (x≡y : x ≡ y) → P y x≡y
J' P p ._ refl = p
```

*Consult the relevant page of Agda documentation for more deltails can thus be regarded as *(based) path induction*: to show a property $P\ y\ x{\equiv}y$ of a path $x{\equiv}y$, we show $P\ x\ \mathit{refl}$ (induction base) and that all paths $P\ z\ x{\equiv}z$ can be formed (so, we can continuously move from $z := x$ to $z := y$).

A type is *contractible* if it provably has exactly one inhabitant; a type is a *mere proposition* if all its inhabitants are equal; a type is a *set* if there is at most one proof of equality of any two its inhabitants. This is formalized in Agda as follows:

```
isContr : Set ℓ → Set ℓ
isContr A = Σ A (λ x → ∀ y → x ≡ y)

isProp : Set ℓ → Set ℓ
isProp A = (x y : A) → x ≡ y

isSet : Set ℓ → Set ℓ
isSet A = (x y : A) → isProp (x ≡ y)
```

a) Show that every contractible type is a mere proposition and every mere proposition is a set.

Hint: Second property is non-tivial and requires some exploration of the space of identity proofs $p :\ x \equiv x$. The idea is to prove that every proof $x{\equiv}y : x \equiv y$ is equal to the canonical proof witnessing `isProp A`. As an intermediate step, show the following, using (based) path induction:

```
prop-refl-prop : ∀ {A : Set ℓ}{x : A} (p : isProp A)
  → trans (p x x) (sym (p x x)) ≡ (p x x)
```

b) Show that a type `A` is a mere proposition iff every type $x \equiv y$ with $x\ y :$ `A` is contractible.

c) Show that a type `A` is a set iff it satisfies the *K eliminator*, iff it satisfies *uniqueness of identity proofs*:

```
K :   ∀ (A : Set ℓ) (x : A) (P : x ≡ x → Set)
  → P refl → (x≡x : x ≡ x) → P x≡x

UIP : ∀ (A : Set ℓ) → Set ℓ
```

Hence, removal of the `{-# OPTIONS --without-K #-}` is precisely equivalent to stating that every type is a set. This explains the historical choice of the name `Set` for types in Agda. As many other things, we cannot prove in MLTT that there are types that are not sets, but MLTT can be consistently extended in both directions: by ensuring that every type is a set (set-theoretic interpretation), or by ensuring that non-set types indeed exist (homotopy interpretation).

d) Show that $\mathbb{B}$ and $\mathbb{N}$ are sets.

Hint: The second property is non-trivial and can be proven by induction over natural numbers, for which you will need to prove the following auxiliary property by path induction

```
suc-pre-of-eq : ∀ {x y : ℕ} (sx≡sy : suc x ≡ suc y)
  → cong sx≡sy (λ z → suc (pre z)) ≡ sx≡sy
```

where

```
pre : ℕ → ℕ
pre zero = zero
pre (suc n) = n
```

(you will need to copy `cong` from `eq.agda` and possibly other functions about equalities.)

Exercise 3 Non-negative Rational Numbers (8 Points)

a) Implement non-negative rational numbers $\mathbb{Q}$ as a *setoid*, whose carrier is formed by pairs n, m, representing fractions n/m with natural n and positive natural m and the equivalence relation, which identifies those fractions, which are equal as numbers. Use the following module as a formalization of the notion of equivalence

```
module Equivalence {a ℓ} {A : Set a} (R : A → A → Set ℓ) where

  record IsEquivalence : Set (a ⊔ ℓ) where
    field
      refl  : ∀ {x} → R x x
      sym   : ∀ {x}{y} → R x y → R y x
      trans : ∀ {x}{y}{z} → R x y → R y z → R x z
```

b) Define addition and multiplication of rational numbers and prove that multiplication distributes over addition.

c) Prove that equality of rational numbers (i.e. the above defined setoid equivalence) is a decidable relation.