# 1 Dependent Type Theory

## 1.1 Equalities

**Definition 1.1.** Definitional equality is a syntactic notion of equality decided by the normalization or reduction rules of a type theory. If two terms are definitionally equal, they can be freely substituted for eachother in any context without changing the meaning of the program.

**Definition 1.2.** Propositional equality is a semantic notion of equality that is a proposition or type in the type theory. Two terms are propositionally equal if there exists a proof or a witness inhabiting a type expressing propositional equality between the two terms.

*Remark* 1.3. Definitional equality is automated in Agda through the normalization of terms, while proving propositional equality is a task for the user. Definitional equality is a stronger notion than propositional equality and is usually decidable.The main distinction is that definitional equality is a syntactic notion built into the type theory, while propositional equality is a semantic notion expressed within the type theory itself.

# 2 Homotopy Type Theory

## 2.1 Path Induction

## 2.2 Contractible Types

**Definition 2.1.** The type A is *contractible* if it comes equipped with an element of the type

$$is - Contr(A) =: \Sigma_{(c:A)}\Pi_{(x:A)}a = x$$

Given $(c, C) : is - Contr(A)$, $c : A$ is the *center of contraction* for $A$ and $C : \Pi_{(x:A)}$ is the *contraction of* $A$.

*Remark* 2.2. A type is contractible if there exists $a : A$ such that for every term $x : A$, there is a path $p : a = x$. We can say that a type is contractible if it is equivalent to the unit type, $\mathbf{1}$. A contractible type can be thought of as a type that contains only one unique element up to homotopy [?].