Homotopy Type Theory

Formal type theory

A.1 The first presentation

Convertibility $t \downarrow t'$ between terms t and t' is the equivalence relation generated by the defining equations for constants, the computation rule

A.1.1 Type universes

We postulate a hierarchy of **universes** denoted by primitive constants

$$\mathcal{U}_0, \quad \mathcal{U}_1, \quad \mathcal{U}_2, \quad \dots$$

The first two rules for universes say that they form a cumulative hierarchy of types:

- $\mathcal{U}_m : \mathcal{U}_n$ for $m < n$,
- if $A : \mathcal{U}_m$ and $m \leq n$, then $A : \mathcal{U}_n$,

and the third expresses the idea that an object of a universe can serve as a type and stand to the right of a colon in judgments:

- if $\Gamma \vdash A : \mathcal{U}_n$, and x is a new variable,[1] then $\vdash (\Gamma, x : A)\ \mathsf{ctx}$.

In the body of the book, an equality judgment $A \equiv B : \mathcal{U}_n$ between types A and B is usually abbreviated to $A \equiv B$. This is an instance of typical ambiguity, as we can always switch to a larger universe, which however does not affect the validity of the judgment.
The following conversion rule allows us to replace a type by one equal to it in a typing judgment:

- if $a : A$ and $A \equiv B$ then $a : B$.

A.1.2 Dependent function types (Π-types)

We introduce a primitive constant c_Π, but write $c_\Pi(A, \lambda x.\, B)$ as $\prod_{(x:A)} B$. Judgments concerning such expressions and expressions of the form $\lambda x.\, b$ are introduced by the following rules:

- if $\Gamma \vdash A : \mathcal{U}_n$ and $\Gamma, x : A \vdash B : \mathcal{U}_n$, then $\Gamma \vdash \prod_{(x:A)} B : \mathcal{U}_n$
- if $\Gamma, x : A \vdash b : B$ then $\Gamma \vdash (\lambda x.\, b) : (\prod_{(x:A)} B)$
- if $\Gamma \vdash g : \prod_{(x:A)} B$ and $\Gamma \vdash t : A$ then $\Gamma \vdash g(t) : B[t/x]$

If x does not occur freely in B, we abbreviate $\prod_{(x:A)} B$ as the non-dependent function type $A \to B$ and derive the following rule:

- if $\Gamma \vdash g : A \to B$ and $\Gamma \vdash t : A$ then $\Gamma \vdash g(t) : B$

Using non-dependent function types and leaving implicit the context Γ, the rules above can be written in the following alternative style that we use in the rest of this section of the appendix:

- if $A : \mathcal{U}_n$ and $B : A \to \mathcal{U}_n$, then $\prod_{(x:A)} B(x) : \mathcal{U}_n$
- if $x : A \vdash b : B$ then $\lambda x.\, b : \prod_{(x:A)} B(x)$
- if $g : \prod_{(x:A)} B(x)$ and $t : A$ then $g(t) : B(t)$

[1] By "new" we mean that it does not appear in Γ or A.

A.1.3 Dependent pair types (Σ-types)

We introduce primitive constants c_Σ and c_{pair}. An expression of the form $c_\Sigma(A, \lambda a.\, B)$ is written as $\sum_{(a:A)} B$, and an expression of the form $c_{\mathsf{pair}}(a, b)$ is written as (a, b). We write $A \times B$ instead of $\sum_{(x:A)} B$ if x is not free in B. Judgments concerning such expressions are introduced by the following rules:

- if $A : \mathcal{U}_n$ and $B : A \to \mathcal{U}_n$, then $\sum_{(x:A)} B(x) : \mathcal{U}_n$
- if, in addition, $a : A$ and $b : B(a)$, then $(a, b) : \sum_{(x:A)} B(x)$

If we have A and B as above, $C : (\sum_{(x:A)} B(x)) \to \mathcal{U}_m$, and

$$d : \prod_{(x:A)} \prod_{(y:B(x))} C((x, y))$$

we can introduce a defined constant

$$f : \prod_{(p:\sum_{(x:A)} B(x))} C(p)$$

with the defining equation

$$f((x, y)) :\equiv d(x, y).$$

Note that C, d, x, and y may contain extra implicit parameters $x_1, \dots, x_n$ if they were obtained in some non-empty context; therefore, the fully explicit recursion schema is

$$f(x_1, \dots, x_n, (x(x_1, \dots, x_n), y(x_1, \dots, x_n))) :\equiv d(x_1, \dots, x_n, (x(x_1, \dots, x_n), y(x_1, \dots, x_n))).$$

A.1.4 Coproduct types

We introduce primitive constants c_+, c_{inl}, and c_{inr}. We write $A + B$ instead of $c_+(A, B)$, $\mathsf{inl}(a)$ instead of $c_{\mathsf{inl}}(a)$, and $\mathsf{inr}(a)$ instead of $c_{\mathsf{inr}}(a)$:

- if $A, B : \mathcal{U}_n$ then $A + B : \mathcal{U}_n$
- moreover, $\mathsf{inl} : A \to A + B$ and $\mathsf{inr} : B \to A + B$

If we have A and B as above, $C : A + B \to \mathcal{U}_m$, $d : \prod_{(x:A)} C(\mathsf{inl}(x))$, and $e : \prod_{(y:B)} C(\mathsf{inr}(y))$, then we can introduce a defined constant $f : \prod_{(z:A+B)} C(z)$ with the defining equations

$$f(\mathsf{inl}(x)) :\equiv d(x) \qquad \text{and} \qquad f(\mathsf{inr}(y)) :\equiv e(y).$$

A.1.5 The finite types

We introduce primitive constants $\star$, $\mathbf{0}$, $\mathbf{1}$, satisfying the following rules:

- $\mathbf{0} : \mathcal{U}_0$, $\mathbf{1} : \mathcal{U}_0$
- $\star : \mathbf{1}$

Given $C : \mathbf{0} \to \mathcal{U}_n$ we can introduce a defined constant $f : \prod_{(x:\mathbf{0})} C(x)$, with no defining equations.
Given $C : \mathbf{1} \to \mathcal{U}_n$ and $d : C(\star)$ we can introduce a defined constant $f : \prod_{(x:\mathbf{1})} C(x)$, with defining equation $f(\star) :\equiv d$.

A.1.6 Natural numbers

The type of natural numbers is obtained by introducing primitive constants $\mathbb{N}$, 0, and succ with the following rules:

- $\mathbb{N} : \mathcal{U}_0$,
- $0 : \mathbb{N}$,
- $\mathsf{succ} : \mathbb{N} \to \mathbb{N}$.

Furthermore, we can define functions by primitive recursion. If we have $C : \mathbb{N} \to \mathcal{U}_k$ we can introduce a defined constant $f : \prod_{(x:\mathbb{N})} C(x)$ whenever we have

$$d : C(0)$$
$$e : \prod_{(x:\mathbb{N})} (C(x) \to C(\mathsf{succ}(x)))$$

with the defining equations

$$f(0) :\equiv d \qquad \text{and} \qquad f(\mathsf{succ}(x)) :\equiv e(x, f(x)).$$

A.1.7 W-types

For W-types we introduce primitive constants c_{W} and c_{sup}. An expression of the form $c_{\mathsf{W}}(A, \lambda x.\, B)$ is written as $\mathsf{W}_{(x:A)} B$, and an expression of the form $c_{\mathsf{sup}}(x, u)$ is written as $\mathsf{sup}(x, u)$:

- if $A : \mathcal{U}_n$ and $B : A \to \mathcal{U}_n$, then $\mathsf{W}_{(x:A)} B(x) : \mathcal{U}_n$
- if moreover, $a : A$ and $u : B(a) \to \mathsf{W}_{(x:A)} B(x)$ then $\mathsf{sup}(a, u) : \mathsf{W}_{(x:A)} B(x)$.

Here also we can define functions by total recursion. If we have A and B as above and $C : (\mathsf{W}_{(x:A)} B(x)) \to \mathcal{U}_m$, then we can introduce a defined constant $f : \prod_{(z:\mathsf{W}_{(x:A)} B(x))} C(z)$ whenever we have

$$d : \prod_{(a:A)} \prod_{(u:B(a) \to \mathsf{W}_{(x:A)} B(x))} ((\prod_{(y:B(a))} C(u(y))) \to C(\mathsf{sup}(a, u)))$$

with the defining equation

$$f(\mathsf{sup}(a, u)) :\equiv d(a, u, f \circ u).$$

A.1.8 Identity types

We introduce primitive constants $c_=$ and c_{refl}. We write $a =_A b$ for $c_=(A, a, b)$ and refl_a for $c_{\mathsf{refl}}(A, a)$, when $a : A$ is understood:

- If $A : \mathcal{U}_n$, $a : A$, and $b : A$ then $a =_A b : \mathcal{U}_n$.
- If $a : A$ then $\mathsf{refl}_a : a =_A a$.

Given $a : A$, if $y : A, z : a =_A y \vdash C : \mathcal{U}_m$ and $\vdash d : C[a, \mathsf{refl}_a / y, z]$ then we can introduce a defined constant

$$f : \prod_{(y:A)} \prod_{(z:a=_A y)} C$$

with defining equation

$$f(a, \mathsf{refl}_a) :\equiv d.$$

A.2 The second presentation

In this section, there are three kinds of judgments

$$\Gamma\ \mathsf{ctx} \qquad \Gamma \vdash a : A \qquad \Gamma \vdash a \equiv a' : A$$

which we specify by providing inference rules for deriving them. A typical **inference rule** has the form

$$\frac{\mathcal{J}_1 \quad \cdots \quad \mathcal{J}_k}{\mathcal{J}}\ \textsc{Name}$$

It says that we may derive the **conclusion** $\mathcal{J}$, provided that we have already derived the **hypotheses** $\mathcal{J}_1, \dots, \mathcal{J}_k$.
A **derivation** of a judgment is a tree constructed from such inference rules, with the judgment at the root of the tree.

A.2.1 Contexts

A context is a list

$$x_1{:}A_1, x_2{:}A_2, \ldots, x_n{:}A_n$$

The judgment $\Gamma\ \mathsf{ctx}$ formally expresses the fact that Γ is a well-formed context, and is governed by the rules of inference

$$\frac{}{\cdot\ \mathsf{ctx}}\ \mathsf{ctx\text{-}EMP} \qquad \frac{x_1{:}A_1, \ldots, x_{n-1}{:}A_{n-1} \vdash A_n : \mathcal{U}_i}{(x_1{:}A_1, \ldots, x_n{:}A_n)\ \mathsf{ctx}}\ \mathsf{ctx\text{-}EXT}$$

A.2.2 Structural rules

$$\frac{(x_1{:}A_1, \ldots, x_n{:}A_n)\ \mathsf{ctx}}{x_1{:}A_1, \ldots, x_n{:}A_n \vdash x_i : A_i}\ \mathsf{Vble}$$

The following important principles, called **substitution** and **weakening**, need not be explicitly assumed. For the typing judgments these principles are manifested as

$$\frac{\Gamma \vdash a : A \qquad \Gamma, x{:}A, \Delta \vdash b : B}{\Gamma, \Delta[a/x] \vdash b[a/x] : B[a/x]}\ \mathsf{Subst}_1$$

$$\frac{\Gamma \vdash A : \mathcal{U}_i \qquad \Gamma, \Delta \vdash b : B}{\Gamma, x{:}A, \Delta \vdash b : B}\ \mathsf{Wkg}_1$$

and for judgmental equalities they become

$$\frac{\Gamma \vdash a : A \qquad \Gamma, x{:}A, \Delta \vdash b \equiv c : B}{\Gamma, \Delta[a/x] \vdash b[a/x] \equiv c[a/x] : B[a/x]}\ \mathsf{Subst}_2$$

$$\frac{\Gamma \vdash A : \mathcal{U}_i \qquad \Gamma, \Delta \vdash b \equiv c : B}{\Gamma, x{:}A, \Delta \vdash b \equiv c : B}\ \mathsf{Wkg}_2$$

In addition to the judgmental equality rules given for each type former, we also assume that judgmental equality is an equivalence relation respected by typing.

$$\frac{\Gamma \vdash a : A}{\Gamma \vdash a \equiv a : A} \qquad \frac{\Gamma \vdash a \equiv b : A}{\Gamma \vdash b \equiv a : A} \qquad \frac{\Gamma \vdash a \equiv b : A \qquad \Gamma \vdash b \equiv c : A}{\Gamma \vdash a \equiv c : A}$$

$$\frac{\Gamma \vdash a : A \qquad \Gamma \vdash A \equiv B : \mathcal{U}_i}{\Gamma \vdash a : B} \qquad \frac{\Gamma \vdash a \equiv b : A \qquad \Gamma \vdash A \equiv B : \mathcal{U}_i}{\Gamma \vdash a \equiv b : B}$$

Additionally, for all the type formers below, we assume rules stating that each constructor preserves definitional equality in each of its arguments; for instance, along with the Π-INTRO rule, we assume the rule

$$\frac{\Gamma \vdash A : \mathcal{U}_i \qquad \Gamma, x{:}A \vdash B : \mathcal{U}_i \qquad \Gamma, x{:}A \vdash b \equiv b' : B}{\Gamma \vdash \lambda x.\, b \equiv \lambda x.\, b' : \prod_{(x:A)} B}\ \Pi\text{-INTRO-EQ}$$

However, we omit these rules for brevity.

A.2.3 Type universes

We postulate an infinite hierarchy of type universes

$$\mathcal{U}_0, \quad \mathcal{U}_1, \quad \mathcal{U}_2, \quad \ldots$$

Each universe is contained in the next, and any type in $\mathcal{U}_i$ is also in $\mathcal{U}_{i+1}$:

$$\frac{\Gamma\ \mathsf{ctx}}{\Gamma \vdash \mathcal{U}_i : \mathcal{U}_{i+1}}\ \mathcal{U}\text{-INTRO} \qquad \frac{\Gamma \vdash A : \mathcal{U}_i}{\Gamma \vdash A : \mathcal{U}_{i+1}}\ \mathcal{U}\text{-CUMUL}$$

A.2.4 Dependent function types (Π-types)

$$x{:}A \vdash B : \mathcal{U}_i.$$

- a **formation rule**, stating when the type former can be applied;
- some **introduction rules**, stating how to inhabit the type;
- **elimination rules**, or an induction principle, stating how to use an element of the type;
- **computation rules**, which are judgmental equalities explaining what happens when elimination rules are applied to results of introduction rules;
- optional **uniqueness principles**, which are judgmental equalities explaining how every element of the type is uniquely determined by the results of elimination rules applied to it.

For the dependent function type these rules are:

$$\frac{\Gamma \vdash A : \mathcal{U}_i \qquad \Gamma, x{:}A \vdash B : \mathcal{U}_i}{\Gamma \vdash \prod_{(x:A)} B : \mathcal{U}_i}\ \Pi\text{-FORM}$$

$$\frac{\Gamma, x{:}A \vdash b : B}{\Gamma \vdash \lambda(x:A).\, b : \prod_{(x:A)} B}\ \Pi\text{-INTRO}$$

$$\frac{\Gamma \vdash f : \prod_{(x:A)} B \qquad \Gamma \vdash a : A}{\Gamma \vdash f(a) : B[a/x]}\ \Pi\text{-ELIM}$$

$$\frac{\Gamma, x{:}A \vdash b : B \qquad \Gamma \vdash a : A}{\Gamma \vdash (\lambda(x:A).\, b)(a) \equiv b[a/x] : B[a/x]}\ \Pi\text{-COMP}$$

$$\frac{\Gamma \vdash f : \prod_{(x:A)} B}{\Gamma \vdash f \equiv (\lambda x.\, f(x)) : \prod_{(x:A)} B}\ \Pi\text{-UNIQ}$$

A.2.5 Dependent pair types (Σ-types)

$$\frac{\Gamma \vdash A : \mathcal{U}_i \qquad \Gamma, x{:}A \vdash B : \mathcal{U}_i}{\Gamma \vdash \sum_{(x:A)} B : \mathcal{U}_i}\ \Sigma\text{-FORM}$$

$$\frac{\Gamma, x{:}A \vdash B : \mathcal{U}_i \qquad \Gamma \vdash a : A \qquad \Gamma \vdash b : B[a/x]}{\Gamma \vdash (a, b) : \sum_{(x:A)} B}\ \Sigma\text{-INTRO}$$

$$\frac{\begin{array}{c}\Gamma, z{:}\sum_{(x:A)} B \vdash C : \mathcal{U}_i \\ \Gamma, x{:}A, y{:}B \vdash g : C[(x,y)/z] \qquad \Gamma \vdash p : \sum_{(x:A)} B\end{array}}{\Gamma \vdash \mathsf{ind}_{\sum_{(x:A)} B}(z.C, x.y.g, p) : C[p/z]}\ \Sigma\text{-ELIM}$$

$$\frac{\begin{array}{c}\Gamma, z{:}\sum_{(x:A)} B \vdash C : \mathcal{U}_i \qquad \Gamma, x{:}A, y{:}B \vdash g : C[(x,y)/z] \\ \Gamma \vdash a : A \qquad \Gamma \vdash b : B[a/x]\end{array}}{\Gamma \vdash \mathsf{ind}_{\sum_{(x:A)} B}(z.C, x.y.g, (a,b)) \equiv g[a,b/x,y] : C[(a,b)/z]}\ \Sigma\text{-COMP}$$

A.2.6 Coproduct types

$$\frac{\Gamma \vdash A : \mathcal{U}_i \qquad \Gamma \vdash B : \mathcal{U}_i}{\Gamma \vdash A + B : \mathcal{U}_i}\ +\text{-FORM}$$

$$\frac{\Gamma \vdash A : \mathcal{U}_i \qquad \Gamma \vdash B : \mathcal{U}_i \qquad \Gamma \vdash a : A}{\Gamma \vdash \mathsf{inl}(a) : A + B}\ +\text{-INTRO}_1$$

$$\frac{\Gamma \vdash A : \mathcal{U}_i \qquad \Gamma \vdash B : \mathcal{U}_i \qquad \Gamma \vdash b : B}{\Gamma \vdash \mathsf{inr}(b) : A + B}\ +\text{-INTRO}_2$$

$$\frac{\begin{array}{c}\Gamma, z{:}(A+B) \vdash C : \mathcal{U}_i \\ \Gamma, x{:}A \vdash c : C[\mathsf{inl}(x)/z] \qquad \Gamma, y{:}B \vdash d : C[\mathsf{inr}(y)/z] \\ \Gamma \vdash e : A + B\end{array}}{\Gamma \vdash \mathsf{ind}_{A+B}(z.C, x.c, y.d, e) : C[e/z]}\ +\text{-ELIM}$$

$$\frac{\begin{array}{c}\Gamma, z{:}(A+B) \vdash C : \mathcal{U}_i \\ \Gamma, x{:}A \vdash c : C[\mathsf{inl}(x)/z] \qquad \Gamma, y{:}B \vdash d : C[\mathsf{inr}(y)/z] \\ \Gamma \vdash a : A\end{array}}{\Gamma \vdash \mathsf{ind}_{A+B}(z.C, x.c, y.d, \mathsf{inl}(a)) \equiv c[a/x] : C[\mathsf{inl}(a)/z]}\ +\text{-COMP}_1$$

$$\frac{\begin{array}{c}\Gamma, z{:}(A+B) \vdash C : \mathcal{U}_i \\ \Gamma, x{:}A \vdash c : C[\mathsf{inl}(x)/z] \qquad \Gamma, y{:}B \vdash d : C[\mathsf{inr}(y)/z] \\ \Gamma \vdash b : B\end{array}}{\Gamma \vdash \mathsf{ind}_{A+B}(z.C, x.c, y.d, \mathsf{inr}(b)) \equiv d[b/y] : C[\mathsf{inr}(b)/z]}\ +\text{-COMP}_2$$

A.2.7 The empty type 0

$$\frac{\Gamma\ \mathsf{ctx}}{\Gamma \vdash \mathbf{0} : \mathcal{U}_i}\ \mathbf{0}\text{-FORM} \qquad \frac{\Gamma, x{:}\mathbf{0} \vdash C : \mathcal{U}_i \qquad \Gamma \vdash a : \mathbf{0}}{\Gamma \vdash \mathsf{ind}_{\mathbf{0}}(x.C, a) : C[a/x]}\ \mathbf{0}\text{-ELIM}$$

A.2.8 The unit type 1

$$\frac{\Gamma\ \mathsf{ctx}}{\Gamma \vdash \mathbf{1} : \mathcal{U}_i}\ \mathbf{1}\text{-FORM} \qquad \frac{\Gamma\ \mathsf{ctx}}{\Gamma \vdash \star : \mathbf{1}}\ \mathbf{1}\text{-INTRO}$$

$$\frac{\Gamma, x{:}\mathbf{1} \vdash C : \mathcal{U}_i \qquad \Gamma \vdash c : C[\star/x] \qquad \Gamma \vdash a : \mathbf{1}}{\Gamma \vdash \mathsf{ind}_{\mathbf{1}}(x.C, c, a) : C[a/x]}\ \mathbf{1}\text{-ELIM}$$

$$\frac{\Gamma, x{:}\mathbf{1} \vdash C : \mathcal{U}_i \qquad \Gamma \vdash c : C[\star/x]}{\Gamma \vdash \mathsf{ind}_{\mathbf{1}}(x.C, c, \star) \equiv c : C[\star/x]}\ \mathbf{1}\text{-COMP}$$

A.2.9 The natural number type

$$\frac{\Gamma\ \mathsf{ctx}}{\Gamma \vdash \mathbb{N} : \mathcal{U}_i}\ \mathbb{N}\text{-FORM} \qquad \frac{\Gamma\ \mathsf{ctx}}{\Gamma \vdash 0 : \mathbb{N}}\ \mathbb{N}\text{-INTRO}_1$$

$$\frac{\Gamma \vdash n : \mathbb{N}}{\Gamma \vdash \mathsf{succ}(n) : \mathbb{N}}\ \mathbb{N}\text{-INTRO}_2$$

$$\frac{\begin{array}{c}\Gamma, x{:}\mathbb{N} \vdash C : \mathcal{U}_i \qquad \Gamma \vdash c_0 : C[0/x] \\ \Gamma, x{:}\mathbb{N}, y{:}C \vdash c_s : C[\mathsf{succ}(x)/x] \qquad \Gamma \vdash n : \mathbb{N}\end{array}}{\Gamma \vdash \mathsf{ind}_{\mathbb{N}}(x.C, c_0, x.y.c_s, n) : C[n/x]}\ \mathbb{N}\text{-ELIM}$$

$$\frac{\begin{array}{c}\Gamma, x{:}\mathbb{N} \vdash C : \mathcal{U}_i \\ \Gamma \vdash c_0 : C[0/x] \qquad \Gamma, x{:}\mathbb{N}, y{:}C \vdash c_s : C[\mathsf{succ}(x)/x]\end{array}}{\Gamma \vdash \mathsf{ind}_{\mathbb{N}}(x.C, c_0, x.y.c_s, 0) \equiv c_0 : C[0/x]}\ \mathbb{N}\text{-COMP}_1$$

$$\frac{\begin{array}{c}\Gamma, x{:}\mathbb{N} \vdash C : \mathcal{U}_i \qquad \Gamma \vdash c_0 : C[0/x] \\ \Gamma, x{:}\mathbb{N}, y{:}C \vdash c_s : C[\mathsf{succ}(x)/x] \qquad \Gamma \vdash n : \mathbb{N}\end{array}}{\begin{array}{l}\Gamma \vdash \mathsf{ind}_{\mathbb{N}}(x.C, c_0, x.y.c_s, \mathsf{succ}(n)) \\ \quad \equiv c_s[n, \mathsf{ind}_{\mathbb{N}}(x.C, c_0, x.y.c_s, n)/x, y] : C[\mathsf{succ}(n)/x]\end{array}}\ \mathbb{N}\text{-COMP}_2$$

A.2.10 Identity types

$$\frac{\Gamma \vdash A : \mathcal{U}_i \qquad \Gamma \vdash a : A \qquad \Gamma \vdash b : A}{\Gamma \vdash a =_A b : \mathcal{U}_i}\ =\text{-FORM}$$

$$\frac{\Gamma \vdash A : \mathcal{U}_i \qquad \Gamma \vdash a : A}{\Gamma \vdash \mathsf{refl}_a : a =_A a}\ =\text{-INTRO}$$

$$\frac{\begin{array}{c}\Gamma, x{:}A, y{:}A, p{:}x =_A y \vdash C : \mathcal{U}_i \qquad \Gamma, z{:}A \vdash c : C[z, z, \mathsf{refl}_z/x, y, p] \\ \Gamma \vdash a : A \qquad \Gamma \vdash b : A \qquad \Gamma \vdash p' : a =_A b\end{array}}{\Gamma \vdash \mathsf{ind}_{=_A}(x.y.p.C, z.c, a, b, p') : C[a, b, p'/x, y, p]}\ =\text{-ELIM}$$

$$\frac{\begin{array}{c}\Gamma, x{:}A, y{:}A, p{:}x =_A y \vdash C : \mathcal{U}_i \\ \Gamma, z{:}A \vdash c : C[z, z, \mathsf{refl}_z/x, y, p] \qquad \Gamma \vdash a : A\end{array}}{\Gamma \vdash \mathsf{ind}_{=_A}(x.y.p.C, z.c, a, a, \mathsf{refl}_a) \equiv c[a/z] : C[a, a, \mathsf{refl}_a/x, y, p]}\ =\text{-COMP}$$

A.3 Homotopy type theory

A.3.1 Function extensionality and univalence

?? is formalized by introduction of a constant funext which asserts that happly is an equivalence:

$$\frac{\Gamma \vdash f : \prod_{(x:A)} B \qquad \Gamma \vdash g : \prod_{(x:A)} B}{\Gamma \vdash \mathsf{funext}(f, g) : \mathsf{isequiv}(\mathsf{happly}_{f,g})}\ \Pi\text{-EXT}$$

The definitions of happly and isequiv can be found in (**??**) and **??**, respectively.
?? is formalized in a similar fashion, too:

$$\frac{\Gamma \vdash A : \mathcal{U}_i \qquad \Gamma \vdash B : \mathcal{U}_i}{\Gamma \vdash \mathsf{univalence}(A, B) : \mathsf{isequiv}(\mathsf{idtoeqv}_{A,B})}\ \mathcal{U}_i\text{-UNIV}$$

The definition of idtoeqv can be found in (**??**).

A.3.2 The circle

Here we give an example of a basic higher inductive type; others follow the same general scheme, albeit with elaborations.
Note that the rules below do not precisely follow the pattern of the ordinary inductive types in **??**: the rules refer to the notions of transport and functoriality of maps (**??**), and the second computation rule is a propositional, not judgmental, equality. These differences are discussed in **??**.

$$\frac{\Gamma\ \mathsf{ctx}}{\Gamma \vdash \mathbb{S}^1 : \mathcal{U}_i}\ \mathbb{S}^1\text{-FORM} \qquad \frac{\Gamma\ \mathsf{ctx}}{\Gamma \vdash \mathsf{base} : \mathbb{S}^1}\ \mathbb{S}^1\text{-INTRO}_1$$

$$\frac{\Gamma\ \mathsf{ctx}}{\Gamma \vdash \mathsf{loop} : \mathsf{base} =_{\mathbb{S}^1} \mathsf{base}}\ \mathbb{S}^1\text{-INTRO}_2$$

$$\frac{\begin{array}{c}\Gamma, x{:}\mathbb{S}^1 \vdash C : \mathcal{U}_i \\ \Gamma \vdash b : C[\mathsf{base}/x] \qquad \Gamma \vdash \ell : b =^{C}_{\mathsf{loop}} b \qquad \Gamma \vdash p : \mathbb{S}^1\end{array}}{\Gamma \vdash \mathsf{ind}_{\mathbb{S}^1}(x.C, b, \ell, p) : C[p/x]}\ \mathbb{S}^1\text{-ELIM}$$

$$\frac{\Gamma, x{:}\mathbb{S}^1 \vdash C : \mathcal{U}_i \qquad \Gamma \vdash b : C[\mathsf{base}/x] \qquad \Gamma \vdash \ell : b =^{C}_{\mathsf{loop}} b}{\Gamma \vdash \mathsf{ind}_{\mathbb{S}^1}(x.C, b, \ell, \mathsf{base}) \equiv b : C[\mathsf{base}/x]}\ \mathbb{S}^1\text{-COMP}_1$$

$$\frac{\Gamma, x{:}\mathbb{S}^1 \vdash C : \mathcal{U}_i \qquad \Gamma \vdash b : C[\mathsf{base}/x] \qquad \Gamma \vdash \ell : b =^{C}_{\mathsf{loop}} b}{\Gamma \vdash \mathbb{S}^1\text{-loopcomp} : \mathsf{apd}_{(\lambda y.\, \mathsf{ind}_{\mathbb{S}^1}(x.C,b,\ell,y))}(\mathsf{loop}) = \ell}\ \mathbb{S}^1\text{-COMP}_2$$

In $\mathsf{ind}_{\mathbb{S}^1}$, x is bound in C. The notation $b =^{C}_{\mathsf{loop}} b$ for dependent paths was introduced in **??**.

A.4 Basic metatheory

Theorem A.4.1. *If $A : \mathcal{U}$ and $A \downarrow A'$ then $A' : \mathcal{U}$. If $t : A$ and $t \downarrow t'$ then $t' : A$.*

Theorem A.4.2. *If $A : \mathcal{U}$ then A is strongly normalizable. If $t : A$ then A and t are strongly normalizable.*

Lemma A.4.3. *The terms in normal form can be described by the following syntax:*

$$v ::= k \mid \lambda x.\, v \mid c(\vec{v}) \mid f(\vec{v}),$$
$$k ::= x \mid k(v) \mid f(\vec{v})(k),$$

where $f(\vec{v})$ represents a partial application of the defined function f. In particular, a type in normal form is of the form k or $c(\vec{v})$.

Theorem A.4.4. *If A is in normal form then the judgment $A : \mathcal{U}$ is decidable. If $A : \mathcal{U}$ and t is in normal form then the judgment $t : A$ is decidable.*

Corollary A.4.5. *The system in **??** is logically consistent.*

Corollary A.4.6. *The system in **??** has the canonicity property.*

Corollary A.4.7. *The property of being a proof in the system in **??** is decidable.*