Definition 4: The **substitution** of a preterm t in an expression $\phi \in \mathrm{Pretype} \cup \mathrm{Preterm}$ for a variable x is defined by recursion again in habitually straight-forward fashion, i.e.

- $x[t/x] \equiv t$

- $s[t/x] \equiv s$ for all $s \in \mathrm{Var} - \{x\}$, s constant

- $\left(\prod_{x:A} B\right)[t/x] \equiv \prod_{x:A[t/x]} B$ (formal non-sense, as x won't occur in A if A is well-formed)

- $\left(\prod_{y:A} B\right)[t/x] \equiv \prod_{z:A[t/x]} B[z/y][t/x]$ for $z \in \mathrm{Var} - (FV(A) \cup FV(B) \cup FV(t)), z \neq x$.

- Same for $\sum_{y:A} B$ and $\lambda(y:A).t$ in both cases $x \equiv y$, $x \not\equiv y$.

- All other type and term formers commute with substitution in obvious fashion.

The **simultaneous substitution** of preterms t_i in ϕ for variables x_i, $i \leq n$, can be defined by

$$\phi[\vec{t}/\vec{x}] := \phi[j x_0/x_0] \dots [j x_n/x_n][t_0/j x_0] \dots [t_n/j x_n]$$

for any injection $j : \{x_1, \dots, x_n\} \to \mathrm{Var}$ with

$$\mathrm{im} j \cap \left(\{x_1, \dots, x_n\} \cup \bigcup_{i \leq n} FV(t_i)\right) = \emptyset.$$

Exercise: $\phi[\vec{t}/\vec{x}]$ does not depend on j or on the order of the pairs $\{(t_i, x_i) \mid i \leq n\}$.

Whenever $\Gamma \equiv \{x_1 : A_1, \dots, x_n : A_n\}$ is a precontext, we define

$$\phi[\vec{t}/\Gamma] := \phi(\vec{t}/\vec{x}], \quad \phi[\Gamma/\vec{\zeta}] := \phi[\vec{x}/\vec{\zeta}].$$

Remark 5: As before, one may impose α-congruence on the sets of preterms and pretypes meta-theoretically, using that substitution is invariant under the change of bound variables, or impose it later on as a congruence rule. Let's just implicitly work on quotients of α-congruent terms and types, so it's out of the way.

Notation	Reading
$\vdash \Gamma$ ctx	Γ is a valid context
$\vdash \Gamma \equiv \Delta$ ctx	Γ, Δ are judgementally equal contexts
$\Gamma \vdash A$ type	A is a type in context Γ
$\Gamma \vdash A \equiv B$ type	A, B are judgementally equal types in ctx Γ
$\Gamma \vdash a : A$	a is a term of type A in context Γ
$\Gamma \vdash a_1 \equiv a_2 : A$	a_1, a_2 are judgementally equal terms of type A in ctx Γ.

Blueprint: A precontext $\Gamma = (x_1 : A_1, \dots, x_n : A_n)$ is a (valid) context if for all $i \le n$: $(x_1 : A_1, \dots, x_{i-1} : A_{i-1}) \vdash A_i$ type.

$\Gamma \vdash -$ type is a unary relation on Pretype. The process of validating whether a given expression is a type in context Γ is called typability checking.

The process of validating whether a given preterm is of a given type in context Γ is called **type checking**.

Assuming $\Gamma \vdash A$ type, one says that A is **inhabited** in context Γ if there is some preterm a such that $\Gamma \vdash a : A$.

The construction of contexts occurs recursively along the length of precontexts, applying at each step the structural and the type formation rules from the logical rules below.

We thus introduce the following rules of **context formation** and **context congruence**:

$$\frac{}{\phi \ \ ctx} \quad , \quad \frac{\Gamma \vdash A \ type}{\vdash \Gamma, x:A \ \ ctx} \quad \text{for } x \text{ not subject in } \Gamma \ ,$$

$$\frac{\vdash \Gamma \equiv \Delta \ \ ctx \qquad \Gamma \vdash A \equiv B \ type}{\vdash \Gamma, x:A \equiv \Delta, y:B \ \ ctx} \quad \text{f.e. } x \text{ not subject in } \Gamma, \ y \text{ not subject in } \Delta \ .$$

To assure that typability, type checking and type assignment are invariant under all kinds of judgemental equality, we have the following **conversion rules**:

$$\frac{\Gamma \vdash a : A \qquad \vdash \Gamma \equiv \Delta \ \ ctx \qquad \Gamma \vdash A \equiv B \ type}{\Delta \vdash a[\Delta/\Gamma] : B[\Delta/\Gamma]} \quad (\text{Term-conversion})$$

$$\frac{\vdash \Gamma \equiv \Delta \ \ ctx \qquad \Gamma \vdash A \ type}{\Delta \vdash A[\Delta/\Gamma] \ type} \quad (\text{Type-Conversion})$$

$$\frac{\Gamma \vdash a \equiv b : A}{\Gamma \vdash a : A} \qquad \frac{\Gamma \vdash A \equiv B \ \mathrm{type}}{\Gamma \vdash A \ \mathrm{type}}$$

$$\frac{\vdash \Gamma \equiv \Delta \ \mathrm{ctx} \qquad \Gamma \vdash a \equiv b : A \qquad \Gamma \vdash A \equiv B \ \mathrm{type}}{\Delta \vdash a[\Delta / \Gamma] \equiv b[\Delta / \Gamma] : B[\Delta / \Gamma]},$$

$$\frac{\vdash \Gamma \equiv \Delta \ \mathrm{ctx} \qquad \Gamma \vdash A \equiv B \ \mathrm{type}}{\Delta \vdash A[\Delta / \Gamma] \equiv B[\Delta / \Gamma] \ \mathrm{type}}$$

The basic calculus is complete with the following structural rules.

$$\frac{\vdash \Gamma, x : A, \Delta \ \mathrm{ctx}}{\Gamma, x : A, \Delta \vdash x : A} \quad (\text{Var-Introduction / Monotonicity})$$

$$\frac{\Gamma, \Delta \vdash \phi \qquad \Gamma \vdash A \ \mathrm{type}}{\Gamma, x : A, \Delta \vdash \phi} \quad (\text{Weakening})$$

$$\frac{\Gamma, x : A, \Delta \vdash \phi \qquad \Gamma \vdash a : A}{\Gamma, \Delta[a/x] \vdash \phi[a/x]} \quad (\text{Substitution})$$

for all expressions ϕ of the form $b : B$, $B \ \mathrm{type}$, $B \equiv C \ \mathrm{type}$, or $b \equiv c : B$. The variable x in Weakening has to be fresh.

Lastly, one imposes that judgemental equality is an equivalence relation in the obvious way.

While the structural rules so far are an invariant foundation for all dependent type theories, the following set of logical rules depends on which type constructions we wish to include. In any case, the logical rules conventionally come in packages of six (or sometimes seven) patterns:

1. The type formation rules which generate new types from given ones.
2. Type-Equality rules ensuring judgemental equality is defined accordingly by recursion.
3.-6. Term-Introduction, -Elimination, -Computation and -Congruence rules which "animate" types: Term-Intro. rules declare canonical terms of the given types; the Elimination and Computation rules state that every type is freely generated by its canonical terms; the Congruence rules again ensure that judgemental equality is defined accordingly by recursion.

[7. Term-Uniqueness rules which state that every term of a given type is canonical in canonical fashion up to judgemental equality. What this means will become clear from the examples.]

The logical rules of $MLTT^{\Pi,\Sigma,Id,1,0,(W_i)_{i<\omega}}$:

(1.) $$\frac{\Gamma, x:A \vdash B \text{ type}}{\Gamma \vdash \prod_{x:A} B \text{ type}} \quad (\Pi\text{-formation})$$

The type $\prod_{x:A} B$ is the type of dependent functions from A to B.

$$\frac{\Gamma, x:A \vdash b : B}{\Gamma \vdash \lambda(x:A).b : \prod_{x:A} B} \quad (\Pi\text{-Introduction} \;/\; \lambda\text{-Abstraction})$$

$$\frac{\Gamma \vdash a : A \qquad \Gamma \vdash f : \prod_{x:A} B}{\Gamma \vdash App(f,a) : B[a/x]} \quad (\Pi\text{-Elimination} \;/\; \lambda\text{-Application})$$

$$\frac{\Gamma \vdash A \equiv A' \text{ type} \qquad \Gamma, x:A \vdash B \equiv B' \text{ type}}{\Gamma \vdash \prod_{x:A} B \equiv \prod_{x:A'} B' \text{ type}} \quad (\Pi\text{-Equality})$$

$$\frac{\Gamma, x:A \vdash b_1 \equiv b_2 : B}{\Gamma \vdash \lambda(x:A).b_1 \equiv \lambda(x:A).b_2 : \prod_{x:A} B} \quad (\Pi\text{-Congruence})$$

$$\frac{\Gamma \vdash a : A \qquad \Gamma, x:A \vdash b : B}{\Gamma \vdash App(\lambda(x:A).b, a) \equiv b[a/x] : B[a/x]} \quad (\Pi\text{-Computation} \;/\; \beta\text{-Congruence})$$

$$\frac{\Gamma \vdash f : \prod_{x:A} B}{\Gamma \vdash \lambda(x:A).App(f,x) \equiv f : \prod_{x:A} B} \quad (\Pi\text{-Uniqueness} \;/\; \eta\text{-Congruence})$$

Recall that η-Congruence is equivalent to meta-theoretical Function Extensionality:

$$\frac{\Gamma \vdash App(f,a) \equiv App(g,a) : B}{\Gamma \vdash f \equiv g : \prod_{x:A} B} \quad \text{f.a. } a \text{ s.th. } \Gamma \vdash a : A.$$

Π-Uniqueness is the only Uniqueness-rule commonly stipulated. We will see that internal versions of some other Uniqueness-rules can be derived in the presence of Martin-Löf's Identity-types.

Notation: If $\Gamma, x:A \vdash B$ type, but $x \notin FV(B)$, the type $\prod_{x:A} B$ is the plain function type from A to B and denoted by $(A \to B)$.

Exercise: Define the composition of two dependent functions and show that composition is associative up to judgemental equality.

2.)
$$\frac{\Gamma, x:A \vdash B \text{ type}}{\Gamma \vdash \sum_{x:A} B \text{ type}} \quad (\Sigma\text{-Formation})$$

The type $\sum_{x:A} B$ is the type of dependent pairs of A wrt. B.

$$\frac{\Gamma, x:A \vdash B \text{ type} \qquad \Gamma \vdash a:A \qquad \Gamma \vdash b:B[a/x]}{\Gamma \vdash (a,b) : \sum_{x:A} B} \quad (\Sigma\text{-Introduction})$$

$$\frac{\Gamma \vdash t : \sum_{x:A} B \qquad \Gamma, z:\sum_{x:A} B \vdash C \text{ type} \qquad \Gamma, x:A, y:B \vdash c : C[(x,y)/z]}{\Gamma \vdash \mathrm{ind}_{\sum_{x:A} B}\left(\prod_{z:\sum_{x:A} B} C, \lambda(x:A).\lambda(y:B).c, t\right) : C[t/z]} \quad (\Sigma\text{-Elim.})$$

$$\frac{\begin{array}{ll}\Gamma \vdash a:A & \Gamma, z:\sum_{x:A} B \vdash C \text{ type} \\ \Gamma \vdash b:B[a/x] & \Gamma, x:A, y:B \vdash c : C[(x,y)/z]\end{array}}{\Gamma \vdash \mathrm{ind}_{\sum_{x:A} B}\left(\prod_{z:\sum_{x:A} B} C, \lambda(x:A).\lambda(y:B).c, (a,b)\right) \equiv c[a/x, b/y] : C[(a,b)/z]} \quad (\Sigma\text{-Comp.})$$

+ the obvious Equality and Congruence rule.

(3.) $\dfrac{\vdash \Gamma \ ctx}{\Gamma \vdash 1 \ type}$ (1-formation) : 1 is the terminal type.

$\dfrac{\vdash \Gamma \ ctx}{\Gamma \vdash * : 1}$ (1-Introduction)

$$\frac{\Gamma \vdash a : 1 \quad \Gamma, x : 1 \vdash C \ type \quad \Gamma \vdash c_* : C[*/x]}{\Gamma \vdash ind_1(\prod_{x:1} C, c_*, a) : C[a/x]} \quad (1\text{-Elimination})$$

$$\frac{\Gamma, x : 1 \vdash C \ type \quad \Gamma \vdash c_* : C[*/x]}{\Gamma \vdash ind_1(\prod_{x:1} C, c_*, *) \equiv c_* : C[*/x]} \quad (1\text{-Computation})$$

+ 1-Equality & 1-Congruence.

(4.) $\dfrac{\vdash \Gamma \ ctx}{\Gamma \vdash 0 \ type}$ (0-Formation) : 0 is the initial type.

(It has no Introduction rule, and hence no Computation rule either.

$$\frac{\Gamma \vdash a : 0 \qquad \Gamma, x : 0 \vdash C \ type}{\Gamma \vdash ind_0(\prod_{x:0} C, a) : C[a/x]} \quad (0\text{-Elimination})$$

+ 0-Equality.

Notation: A type theory with initial type 0 is inconsistent if the type 0 is inhabited in the empty context. It is consistent otherwise.

5. $\dfrac{\Gamma \vdash a : A \qquad \Gamma \vdash b : A}{\Gamma \vdash a =_A b \ \text{type}}$ (Id-formation)

The type $a =_A b$ is the "intensional" identity-type of $a, b : A$. We say that $a, b : A$ are "propositionally" or "intensionally" equal in context Γ if $a =_A b$ is inhabited in context Γ.

$\dfrac{\Gamma \vdash a : A}{\Gamma \vdash \mathrm{refl}_a : a =_A a}$ (Id-Introduction)

Note already here that in contrast to former type constructors, the identity-types $a =_A b$ are not introduced each individually, but inductively as a type family over A.

$$\frac{\Gamma \vdash a : A \qquad \Gamma \vdash b : A \qquad \Gamma, x : A, y : A, z : x =_A y \vdash C \ \text{type} \qquad \Gamma, x : A \vdash c : C[x/y, \mathrm{refl}_x/z] \qquad \Gamma \vdash p : a =_A b}{\Gamma \vdash \mathrm{ind}_{=_A}\left(\prod_{x:A}\prod_{y:A}\prod_{z:x=_A y} C, \lambda(x:A).c, a, b, p\right) : C[a/x, b/y, p/z]} \ (\text{Id-Elim.})$$

$$\frac{\Gamma \vdash a : A \qquad \Gamma, x : A, y : A, z : x =_A y \vdash C \ \text{type} \qquad \Gamma, x : A \vdash c : C[x/y, \mathrm{refl}_x/z]}{\Gamma \vdash \mathrm{ind}_{=_A}\left(\prod_{x:A}\prod_{y:A}\prod_{z:x=_A y} C, \lambda(x:A).c, a, a, \mathrm{refl}_a\right) \equiv c[a/x] : C[a/x, a/y, \mathrm{refl}_a/z]} \ (\text{Id-Comp.})$$

+ Id-Equality and Id-Congruence.

$\Gamma \vdash a \equiv b : A$ implies $\Gamma \vdash \mathrm{refl}_a : a =_A b$, inc

judgemental equality implies propositional equality.

The strength of propositional equality has been mysterious for decades after their introduction by Martin-Löf. He for instance considered extensions by

$$\frac{\Gamma \vdash p : a =_A b}{\Gamma \vdash a \equiv b : A} \qquad \frac{\Gamma \vdash p : a =_A a}{\Gamma \vdash p = \mathrm{refl}\, a : a =_A a} \qquad \text{(Equality-Reflection)}$$

which identifies judgemental and propositional equality up to derivability. It hence collapses the potentially multi-valued truth value of $a =_A b$ to the binary datum on whether it is inhabited or not.

Remark: MLTT$^{\mathrm{Id}, \vdash}$ is "extensional intensional type theory", extensional MLTT$^{\Pi,\Sigma,\mathrm{Id},\mathbb{1},\mathbb{0}}$ has a convivent semantics in locally cartesian closed categories, see Seely's "Locally cartesian closed categories and type theory" with a caveat on coherence. But it destroys decidability of type checking, see e.g. Hofmann's "Extensional concepts in intensional type theory".

A propositional analogue of Equality-Reflection is given by Id-Uniqueness, or classically referred to as Axiom K or UIP (Uniqueness of Identity Proofs):

$$\frac{\Gamma \vdash a : A \quad \Gamma \vdash p : a =_A a}{\Gamma \vdash \mathrm{K}(A, a, p) : p = (a =_A a)\ \mathrm{refl}\, a} \qquad \text{(Axiom K)}$$

This is equivalent under Id-Elimination and the Π-rules to

$$\frac{\Gamma \vdash a : A \quad \Gamma \vdash b : A \quad \Gamma \vdash p : a =_A b \quad \Gamma \vdash q : a =_A b}{\Gamma \vdash \mathrm{UIP}(A, a, b, p, q) : p =_{a =_A b} q} \qquad \text{(UIP)}$$

(t was unknown whether $MLTT^{Id1-} + \neg UIP$ is consistent until the work of Hofmann and Streicher on the groupoid model of intensional type theory.