Towards a cubical type theory without an interval

Thorsten Altenkirch[1] **and Ambrus Kaposi**[2]

1 **University of Nottingham, Nottingham, United Kingdom**
`txa@cs.nott.ac.uk`
2 **Eötvös Loránd University, Budapest, Hungary**
`akaposi@inf.elte.hu`

Abstract

Following the cubical set model of type theory which validates the univalence axiom, cubical type theories have been developed that interpret the identity type using an interval pretype. These theories start from a geometric view of equality. A proof of equality is encoded as a term in a context extended by the interval pretype. Our goal is to develop a cubical theory where the identity type is defined recursively over the type structure, and the geometry arises from these definitions. In this theory, cubes are present explicitly, e.g. a line is a telescope with 3 elements: two endpoints and the connecting equality. This is in line with Bernardy and Moulin's earlier work on internal parametricity. In this paper we present a naive syntax for internal parametricity and by replacing the parametric interpretation of the universe, we extend it to univalence. However, we don't know how to compute in this theory. As a second step, we present a version of the theory for parametricity with named dimensions which has an operational semantics. Extending this syntax to univalence is left as further work.

1998 ACM Subject Classification F.4.1 Mathematical Logic

Keywords and phrases homotopy type theory, parametricity, univalence

Digital Object Identifier 10.4230/LIPIcs.TYPES.2015.<article-no>

1 Introduction

Homotopy Type Theory [17] introduces the *univalence axiom*, which basically identifies propositional equality with equivalence of types[1]. We can understand the *univalence axiom* as a powerful extensionality principle which allows us to replace one type by another which has the same behaviour. It also entails functional extensionality.

The usual formulation of Homotopy Type Theory (given in the appendix of [17]) lacks a computational understanding of univalence. With some simplification it adds the following axiom to Martin-Löf Type Theory:

$$\mathsf{univ} : A \simeq B \to A =_{\mathsf{U}} B$$

saying that an equality in the universe is provided given an equivalence between the types A and B. Here, equality is defined as an inductive family with the constructor refl and eliminator J. The axiom univ adds another constructor without providing an elimination rule for this constructor resulting in stuck terms. For example, we can define the equivalence $(\mathsf{not}, \dots)$ between Bool and Bool and thus $\mathsf{univ}\,(\mathsf{not}, \dots) : \mathsf{Bool} =_{\mathsf{U}} \mathsf{Bool}$, and then using it,

[1] Equivalence of types is a proof-relevant refinement of isomorphism. Naively it is sufficient to think of isomorphism since this notion is logically equivalent (but not isomorphic as a type).

define a boolean b:

$\mathsf{coe} : (A =_{\mathsf{U}} B) \to A \to B$

$\mathsf{coe}\,(\mathsf{refl}\,A)\,a :\equiv a$

$b : \mathsf{Bool}$

$b :\equiv \mathsf{coe}\,(\mathsf{univ}\,(\mathsf{not}, \dots))\,\mathsf{true}$

We know that b should be false as we coerce along not, but coe was only defined[2] for the case refl so the term b does not compute further. One may think that it would be enought to add the equation $\mathsf{coe}\,(\mathsf{univ}\,(f, \dots)) \equiv f$ but this is not sufficient as shown by the following example. Here we use the ap (apply path) function for a $P : \mathsf{U} \to \mathsf{U}$:

$\mathsf{ap}_P : A =_{\mathsf{U}} B \to P\,A =_{\mathsf{U}} P\,B$

$\mathsf{ap}_P\,(\mathsf{refl}\,A) \equiv \mathsf{refl}\,(P\,A)$

$x : P\,\mathsf{Bool} =_{\mathsf{U}} P\,\mathsf{Bool}$

$x :\equiv \mathsf{ap}_P\,(\mathsf{univ}\,(\mathsf{not}, \dots))$

To reduce this term we need to know how to transport an equivalence along an arbitrary type.

We observe that the problem is caused by viewing equality as an inductive type with only one constructor. Alternatively, we may want to exploit the characterisation of equalities for each type (as described in chapter 2 of [17]): equality of pairs is a pair of equalities, equality of functions is given by a function and equality of types is given by equivalence. That is we want to define the equality relation inductively on the type structure. The theory of logical relations, as known from parametricity ([18, 12, 5]) describes such relations: a logical relation for pairs is a pair of relations, a logical relation for functions is a function which maps related inputs to related outputes. When we view equality as a logical relation, the fundamental lemma of logical relations which says that every term preserves the relation becomes congruence for equality (usually denoted ap or cong). Our work is closely related to the earlier work by Bernardy and Moulin on internal parametricity [7, 14, 8]. Their later [6, 15] work uses as interval type just as [10]. The work on Observational Type Theory [3] is related, however it uses a different heterogeneous equality with built-in proof-irrelevance. Another work very similar to ours is [16], however it only targets functional extensionality.

Our approach is an alternative to introducing an interval pretype as in [10, 4]. Clearly, their work is more complete in that it provides a computational understanding of univalence. While the introduction of the interval is very elegant it is interesting to consider an alternative approach where equality is defined recursively. However, in the moment it is not clear how to interpret univalence in this context. The basic idea is to say that equality in the universe is given by a symmetric notion of equivalence but the problem is that in this definition we refer back to equality for any type in the universe. Hence we don't succeed in justifying the usual rules for equality, in particular the eliminator J. However, justifying univalence is also technically complicated in [10].

Our approach is also closely related to the basic cubical model of type theory as described in [9, 13] (without connections). That is the structure of the presheaf model emerges in our syntax. Hence, we conjecture that there is a natural interpretation of our syntax in this presheaf category. The main question is how a univalent universe of fibrant types can be defined in the syntax.

---

[2] coe was defined by pattern matching, which can be seen as a usage of the eliminator J in this case.

1.1 Structure of the paper

We develop a naive approach to cubical type theory by defining equality in section 2. Our main tool is to introduce heterogenous logical relations similar to Bernardy and Moulin [7]. However, they focus on logical predicates while we are using logical relations and our approach is based on a calculus with explicit subsitutions. We also show how to interpret univalence and derive the eliminator for equality in this calculus (section 2.8). One issue with this calculus as already observed by Bernardy and Moulin is the computational interpretation of the swap operation which swaps dimensions. To address this we introduce a new system with named dimensions in section 3. This is similar to Type Theory in Colour [8] but with explicit substitutions and without the need to annotate all judgements with a list of colours or dimensions. We present an operational semantics for this calculus (appendix A) but don't prove completeness. Also the question how to interpret univalence is left open. We conclude in section 4.

2 Naive cubical type theory

In this section we introduce a naive syntax for cubical type theory. After presenting a core substitution calculus (section 2.1) and the rules for dependent function space (section 2.2) we show how to extend the theory by rules for external parametricity (section 2.3), internal parametricity (section 2.4) and finally we replace relations by equality relations and admit univalence (2.8). In each section, we introduce new derivation rules which extend the theory given in the previous sections. We discuss the metatheoretic properties of the theory in section 2.9.

2.1 Core substitution calculus

We start with an explicit substitution calculus with variable names and universes à la Russell. Weakening is implicit. While we present the system using named variables for readability we assume that terms are identified up to α-conversion and that name capture is always avoided by appropriate renaming. We use $\mathsf{U} : \mathsf{U}$ for presentation purposes, but mean $\mathsf{U}_i : \mathsf{U}_j$ only if $i < j$ officially. We only define well-typed terms (no preterms) and we write equality judgements without context and type information for brevity. For a formal account on such an approach, see [2].

Notations:

Γ, Δ, Θ	contexts
x, y, z, X	variables
t, u, v, f	terms
A, B, C	types
ρ, σ, ν	substitutions

Judgment kinds:

$\Gamma \vdash$	Γ is a valid context
$\Gamma \vdash t : A$	t is a term of type A in context Γ
$\rho : \Delta \Rightarrow \Gamma$	a substitution from Δ to Γ
$\Gamma \equiv \Gamma'$	definitional equality of contexts
$\Gamma \vdash t \equiv t' : A$	definitional equality of terms
$\Gamma \vdash \rho \equiv \rho' : \Delta \Rightarrow \Gamma$	definitional equality of substitutions

Rules:

$$\frac{}{\cdot \vdash} \qquad \frac{\Gamma \vdash \quad \Gamma \vdash A : \mathsf{U}}{\Gamma.x : A \vdash}$$

$$\frac{\Gamma \vdash}{\epsilon : \Gamma \Rightarrow \cdot} \qquad \frac{\Gamma \vdash A : \mathsf{U} \quad \rho : \Delta \Rightarrow \Gamma \quad \Delta \vdash t : A[\rho]}{(\rho, x \mapsto t) : \Delta \Rightarrow \Gamma.x : A} \qquad \frac{\Gamma \vdash}{\mathsf{id}_\Gamma : \Gamma \Rightarrow \Gamma}$$

$$\frac{\rho : \Delta \Rightarrow \Gamma \quad \sigma : \Theta \Rightarrow \Delta}{\rho\sigma : \Theta \Rightarrow \Gamma} \qquad \frac{\Delta \vdash A : \mathsf{U} \quad \rho : \Delta \Rightarrow \Gamma}{\rho : \Delta.x : A \Rightarrow \Gamma}$$

$$\mathsf{id}\rho \equiv \rho \equiv \rho\mathsf{id} \qquad \nu(\rho\sigma) \equiv (\nu\rho)\sigma \qquad (\rho, x \mapsto t)\sigma \equiv (\rho\sigma, x \mapsto t[\sigma])$$

$$(\underbrace{\mathsf{id}_\Gamma}_{:\Gamma.x:A \Rightarrow \Gamma}, x \mapsto x) \equiv \mathsf{id}_{\Gamma.x:A} \qquad \frac{\sigma : \Gamma \Rightarrow \cdot}{\sigma \equiv \epsilon}$$

$$\frac{\Gamma \vdash A : \mathsf{U} \quad \Gamma \vdash t : B}{\Gamma.x : A \vdash t : B} \qquad \frac{\Gamma \vdash t : A \quad \rho : \Delta \Rightarrow \Gamma}{\Delta \vdash t[\rho] : A[\rho]} \qquad t[\mathsf{id}] \equiv t \qquad t[\rho][\sigma] \equiv t[\rho\sigma]$$

$$\frac{\Gamma \vdash A : \mathsf{U}}{\Gamma.x : A \vdash x : A} \qquad \frac{\Gamma \vdash}{\Gamma \vdash \mathsf{U} : \mathsf{U}} \qquad \mathsf{U}[\rho] \equiv \mathsf{U}$$

We assume all coercion rules, congruence rules and reflexivity, symmetry, transitivity of $\equiv$. We usually omit the starting $\cdot$ from contexts and ϵ from substitutions.

2.2 Function space

We list the rules for dependent function space here.

$$\frac{\Gamma.x : A \vdash B : \mathsf{U}}{\Gamma \vdash \Pi(x : A).B : \mathsf{U}} \qquad \frac{\Gamma.x : A \vdash t : B}{\Gamma \vdash \lambda x.t : \Pi(x : A).B} \qquad \frac{\Gamma \vdash f : \Pi(x : A).B \quad \Gamma \vdash u : A}{\Gamma \vdash f\, u : B[(\mathsf{id}, x \mapsto u)]}$$

$$(\lambda x.t)\, u \equiv t[x \mapsto u] \qquad f \equiv (\lambda x.f\, x)$$

$$(\Pi(x : A).B)[\rho] \equiv \Pi(x : A[\rho]).B[(\rho, x \mapsto x)]$$

$$(\lambda x.t)[\rho] \equiv \lambda x.t[(\rho, x \mapsto x)] \qquad (f\, u)[\rho] \equiv f[\rho]\, u[\rho]$$

$A \to B$ is an abbreviation for $\Pi(x : A).B$ where B does not depend on x. $\Pi(x : A, y : B).C$ is an abbreviation for $\Pi(x : A).\Pi(y : B).C$.

2.3 External parametricity

Binary parametricity says that logically related interpretations of a term are logically related. We can express this syntactically in the following way: given a term $\Gamma \vdash t : A$, and two substitutions into Γ denoted ρ_0 and ρ_1 which are pointwise related at each type in Γ, then $t[\rho_0]$ and $t[\rho_1]$ will be related at A. We formalise this idea by adding the following rules to our theory:

$$\frac{\Gamma \vdash}{\Gamma^= \vdash} \qquad \mathsf{0}_\Gamma, \mathsf{1}_\Gamma : \Gamma^= \Rightarrow \Gamma \qquad \frac{\Gamma \vdash t : A}{\Gamma^= \vdash t^* : A^*\, t[\mathsf{0}_\Gamma]\, t[\mathsf{1}_\Gamma]} \qquad \frac{\rho : \Gamma \Rightarrow \Delta}{\rho^= : \Gamma^= \Rightarrow \Delta^=}$$

$\Gamma^=$ contains two copies of Γ (ρ_0 and ρ_1 in the above example) and a logical relation between them. A^* is the logical relation at A. It relates the two interpretations of the term t, in the two different copies of Γ, which are projected out by the substitutions $\mathsf{0}_\Gamma$, $\mathsf{1}_\Gamma$. t^* is the witness of this relation. The parametricity rule for a term t says that the two versions of the term are related at A^* and this is witnessed by t^*. Also, substitutions can be lifted to the $-^=$-d contexts. We use the notation $-^=$ because (from section 2.8) we will view the relation A^* as the heterogeneous equality relation on A.

The following rule for relations is admissible as it follows from the rule for terms and the equality rule for U^* below.

$$\frac{\Gamma \vdash A : \mathsf{U}}{\Gamma^= \vdash A^* : A[\mathsf{0}_\Gamma] \to A[\mathsf{1}_\Gamma] \to \mathsf{U}}$$

The operation $-^=$ duplicates a context and adds witnesses that the two new subcontexts are pointwise related. We add lower indices to the variable names to get new variable names.

$$\cdot^= \equiv \cdot$$
$$(\Gamma.x : A)^= \equiv \Gamma^=.x_0 : A[\mathsf{0}_\Gamma].x_1 : A[\mathsf{1}_\Gamma].x_2 : A^*\, x_0\, x_1$$

The substitutions $\mathsf{0}$ and $\mathsf{1}$ project out the corresponding components ($b = \mathsf{0}, \mathsf{1}$).

$$b. \equiv \epsilon : \cdot \Rightarrow \cdot$$
$$b_{(\Gamma.x:A)} \equiv (b_\Gamma, x \mapsto x_b) : (\Gamma.x : A)^= \Rightarrow \Gamma.x : A$$

The relation A^* is generated by induction on A: for function types, it says that related inputs are mapped to related outputs, for the universe, it is relation space[3]. As types are just terms, we define t^* for every term t uniformly, including the cases when it is a type:

$$\begin{aligned}
(\Pi(x : A).B)^* &\equiv \lambda f_0\, f_1.\Pi(x_0 : A[\mathsf{0}], x_1 : A[\mathsf{1}], x_2 : A^*\, x_0\, x_1).B^*\, (f_0\, x_0)\, (f_1\, x_1)\\
\mathsf{U}^* &\equiv \lambda X_0\, X_1.X_0 \to X_1 \to \mathsf{U}\\
x^* &\equiv x_2\\
(t[\rho])^* &\equiv t^*[\rho^=]\\
(\lambda x.t)^* &\equiv \lambda x_0\, x_1\, x_2.t^*\\
(f\, u)^* &\equiv f^*\, u[\mathsf{0}]\, u[\mathsf{1}]\, u^*
\end{aligned}$$

Note that U^* validates the rule $\mathsf{U}^* : \mathsf{U}^*\, \mathsf{U}\, \mathsf{U}$ by reducing its type. The cases for variables, substituted terms, abstraction and application can be seen as the proof of the fundamental theorem of the logical relation.

[3] In section 2.8 we will replace relation space by equivalence

$-^=$ on substitutions is defined componentwise.

$$\begin{aligned}
\epsilon^= &\equiv \epsilon \\
\mathsf{id}^= &\equiv \mathsf{id} \\
(\sigma\rho)^= &\equiv \sigma^=\rho^= \\
(\rho, x \mapsto t)^= &\equiv (\rho^=, x_0 \mapsto t[\mathsf{0}], x_1 \mapsto t[\mathsf{1}], x_2 \mapsto t^*)
\end{aligned}$$

Finally, we add naturality of b for $b = \mathsf{0}, \mathsf{1}$.

$$\frac{\rho : \Gamma \Rightarrow \Delta}{\rho b_\Gamma \equiv b_\Delta \rho^=}$$

The above rules add external parametricity to the theory. For example, using the new rules, we can derive that a particular inhabitant of the type $\Pi(A : \mathsf{U}).A \to A$ is the identity function. Using the parametricity rule for the term f and expanding its type using the computation rules we get

$$\frac{f : \Pi(A : \mathsf{U}).A \to A}{f^* : \Pi \left(\begin{array}{l} A_0 : \mathsf{U}, A_1 : \mathsf{U}, A_2 : A_0 \to A_1 \to \mathsf{U}, \\ x_0 : A_0, x_1 : A_1, x_2 : A_2\, x_0\, x_1 \end{array} \right) .A_2\, (f\, A_0\, x_0)\, (f\, A_1\, x_1)} .$$

Now we define a function that for any type A and element $y : A$ shows that $f\, A\, y$ is equal to y. For this example we need an identity type Id_A and its constructor refl to be part of our theory. The binary relation on A says that the first component is equal to y. Obviously, this y witnesses this relation by $\mathsf{refl}\, y$.

$$\lambda A : \mathsf{U}\, y : A.f^*\, A\, A\, (\lambda x_0\, x_1.\mathsf{Id}_A\, x_0\, y)\, y\, y\, (\mathsf{refl}\, y) : \Pi(A : \mathsf{U}, y : A).\mathsf{Id}_A\, (f\, A\, y)\, y$$

Another example of using parametricity is given for the following term.

$$A : \mathsf{U}.z : A.s : A \to A \vdash t : A$$

We can interpret the context by the following two substitutions. For this example, we need natural numbers and booleans in our type theory.

$$\begin{aligned}
\rho_0 &= (A := \mathbb{N}, \quad z := \mathsf{zero}, s := \mathsf{suc}) \\
\rho_1 &= (A := \mathsf{Bool}, z := \mathsf{true}, s := \mathsf{not})
\end{aligned}$$

That is, in the first case we use natural numbers with the Peano constructors for z and s and in the second case we have booleans with true for z and negation for successor. We would like to prove that whatever t is, it is not possible to have $t[\rho_0] = \mathsf{suc}\, (\mathsf{suc}\, \mathsf{zero})$ and $t[\rho_1] = \mathsf{false}$. Binary parametricity says in this case that if we have a relation between $\mathbb{N}$ and Bool such that if $z[\rho_0]$ is related to $z[\rho_1]$ and $s[\rho_0]$ is related to $s[\rho_1]$ then $t[\rho_0]$ is related to $t[\rho_1]$. We can define the relation $A_2 : \mathbb{N} \to \mathsf{Bool} \to \mathsf{U}$ to be $A_2\, x\, b := \mathsf{if}\, b\, \mathsf{then}\, \mathsf{Even}\, x\, \mathsf{else}\, \mathsf{Odd}\, x$. We can show that zero and true are related as zero is even and that the successor of an even number is odd and the successor of an odd number is even. Parametricity tells us that $t[\rho_0]$ and $t[\rho_1]$ need to be related by A_2, hence it can't happen that $t[\rho_0]$ is 2 and $t[\rho_1]$ is false. Collecting together ρ_0, ρ_1 and the proof of their relatedness into ρ, we can express this in our theory as follows.

$$\frac{\dfrac{A : \mathsf{U}.z : A.s : A \to A \vdash t : A}{(A : \mathsf{U}.z : A.s : A \to A)^= \vdash t^* : A_2\, (t[\mathsf{0}])\, (t[\mathsf{1}])}}{\cdot \vdash t^*[\rho] : A_2[\rho]\, (t[\rho_0])\, (t[\rho_1])}$$

We call this theory *external* because even though we can show for every instance of $f : \Pi(A : \mathsf{U}).A \to A$ that it behaves like the identity but we cannot show this *internally* for an assumption of the form $f : \Pi(A : \mathsf{U}).A \to A$.

2.4 Internal parametricity

To internally derive that every function of type $\Pi(A : \mathsf{U}).A \to A$ is the identity, we would need a term t expressing parametricity for f assuming it in the context containing f:

$$f : \Pi(A : \mathsf{U}).A \to A \vdash t : \Pi(A_0, A_1 : \mathsf{U}, A_2 : A_0 \to A_1 \to \mathsf{U})\,. \\ \Pi(x_0 : A_0, x_1 : A_1, x_2 : A_2\,x_0\,x_1)\,.\,A_2\,(f\,A_0\,x_0)\,(f\,A_1\,x_1).$$

We can try to choose t to be f^*, but that lives in the context $(f : \Pi(A : \mathsf{U}).A \to A)^=$.

This motivates the definition of a substitution that goes from a context Γ into $\Gamma^=$. Using such a substitution R_Γ, we can choose the above t to be $f^*[\mathsf{R}_{f:\Pi(A:\mathsf{U}).A\to A}]$.

We add the substitution R by the following rule.

$$\frac{\Gamma \vdash}{\mathsf{R}_\Gamma : \Gamma \Rightarrow \Gamma^=}$$

We also add the following computation rules.

$$\mathsf{R}_\cdot \equiv \epsilon \qquad \mathsf{R}_{\Gamma.x:A} \equiv (\mathsf{R}_\Gamma, x_0 \mapsto x, x_1 \mapsto x, x_2 \mapsto x^*[\mathsf{R}_{\Gamma.x:A}]) \qquad \frac{\rho : \Delta \Rightarrow \Gamma}{\mathsf{R}_\Gamma \rho \equiv \rho^= \mathsf{R}_\Delta}$$

The first two rules explain how R is duplicating the elements in the context, while the last one is a naturality rule making it possible to commute substitutions with R.

Note that $x^*[\mathsf{R}_{\Gamma.x:A}]$ is equal to $x_2[\mathsf{R}_{\Gamma.x:A}]$ however we don't have any rules to compute such an expression any further. Hence, this becomes a new normal form. We can use the naturality rule to substitute into such a normal form.

2.5 Geometry arising from the syntax

A context $(x : A)$ can be viewed as a context of points of type A. The context $(x : A)^=$ is a context of lines where x_2 is a line between the edges x_0 and x_1 (3 elements), A^* is a relation (the type of lines). The $(x : A)^{==}$ is a type of squares with 9 components: 4 points, 4 lines and a filler of. A^{**} is a two-dimensional relation defining the type of squares: it has 8 arguments, the 4 points and the 4 lines of the square.

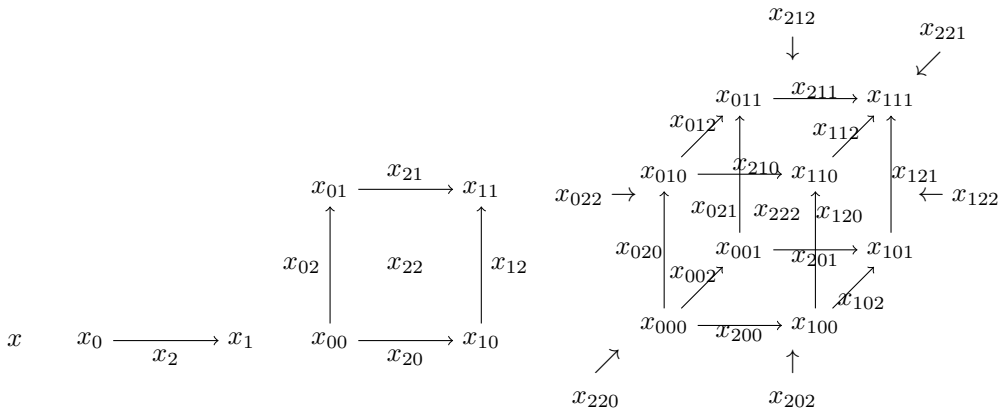
When iterating $-^=$ for a context with more than one type, we get a series of cubes. The later cubes can depend on the previous ones, hence the lines are heterogeneous (the endpoints don't necessarily have the same types). As an example, we expand the first two iterations.

$$(\Gamma.x : A)^= \equiv \Gamma^=.x_0 : A[\mathsf{0}_\Gamma].x_1 : A[\mathsf{1}_\Gamma].x_2 : A^*\,x_0\,x_1$$
$$\begin{aligned}(\Gamma.x : A)^{==} \equiv \Gamma^{==}.&x_{00} : A[\mathsf{0}_\Gamma \mathsf{0}_{\Gamma^=}] \quad .x_{01} : A[\mathsf{0}_\Gamma \mathsf{1}_{\Gamma^=}] \quad .x_{02} : A[\mathsf{0}_\Gamma]^*\,x_{00}\,x_{01}\\ .&x_{10} : A[\mathsf{1}_\Gamma \mathsf{0}_{\Gamma^=}] \quad .x_{11} : A[\mathsf{1}_\Gamma \mathsf{1}_{\Gamma^=}] \quad .x_{12} : A[\mathsf{1}_\Gamma]^*\,x_{10}\,x_{11}\\ .&x_{20} : A^*[\mathsf{0}_{\Gamma^=}]\,x_{00}\,x_{10}.x_{21} : A^*[\mathsf{1}_{\Gamma^=}]\,x_{01}\,x_{11}\,.x_{22} : (A^*\,x_0\,x_1)^*\,x_{20}\,x_{21}\end{aligned}$$

Note that e.g. the type of x_{20} was computed from $(A^*\,x_0\,x_1)[\mathsf{0}_{\Gamma^=.x_0:A[\mathsf{0}].x_1:A[\mathsf{1}]}] \equiv A^*[\mathsf{0}_{\Gamma^=}]\,x_{00}\,x_{10}$. Also, the type of x_{22} is equal to $A^{**}\,x_{00}\,x_{01}\,x_{02}\,x_{10}\,x_{11}\,x_{12}\,x_{20}\,x_{21}$ by the computation rule of $-^*$ for applications.

More generally, the context $(x : A)^n$ ($-^=$ iterated n times) has 3^n components which make an n-dimensional cube. We can depict the first three iterations as follows. The first

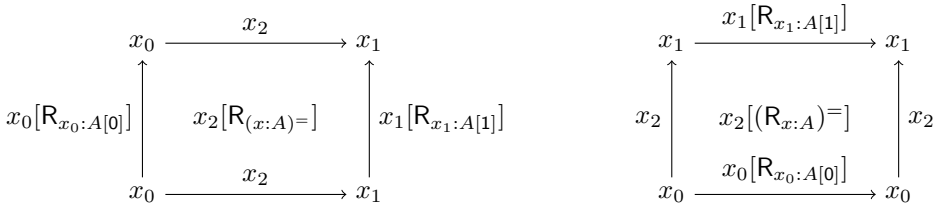
dimension is drawn horizontal, the second vertical and the third is perpendicular to the plain of the paper.



As an example we describe how to view a variable in a 3-dimensional context. If we name the dimensions d_0, d_1, d_2, the variable x_{ijk} represents a face of the 3-dimensional cube which has coordinate i in dimension d_0, coordinate j in dimension d_1, and coordinate k in dimension d_2. By face we mean any 0,1,2 or 3-dimensional face, a 3-dimensional cube only has one 3-dimensional face, the filler. A coordinate can be 0 or 1, in this case we can interpret it as a usual cartesian coordinate. If the coordinate is 2, it means that this is a face spanning through that dimension. Eg. x_{011} is the $d_0 = 0, d_1 = 1, d_2 = 1$ point of a 3-dimensional cube (left upper back if the dimensions are oriented in the standard way); x_{021} is a line in the vertical dimension d_1, it connects the points x_{001} and x_{011} on the left and in the back of the cube; x_{221} is the back face of the cube; x_{222} is the filler of the cube.

For a two-dimensional cube $(x : A)^{==}$, we have four ways to project out the lines: $\mathsf{0}_{(x:A)^=}, (\mathsf{0}_{x:A})^=, \mathsf{1}_{(x:A)^=}$ and $(\mathsf{1}_{x:A})^=$. If we expand $\mathsf{0}_{(x:A)^=}$, we get $x_0 \mapsto x_{00}, x_1 \mapsto x_{10}, x_2 \mapsto x_{20}$, the bottom line of the square. However expanding $(\mathsf{0}_{x:A})^=$ gives $x_0 \mapsto x_{00}, x_1 \mapsto x_{01}, x_2 \mapsto x_{02}$, the left line. In general, $(b_{\Gamma^i})^{n-i}$ projects dimension i to b while decreasing the dimension, i.e. it consists of $x_{j_0 \dots j_{n-1}} \mapsto x_{j_0 \dots j_{i-1} b j_i \dots j_{n-1}}$ maps.

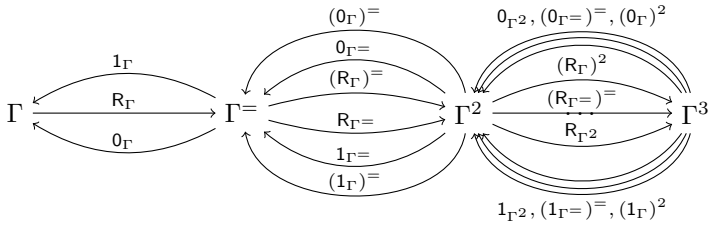
Similarly, from a one-dimensional cube $(x : A)^=$ there are two ways to go to the two-dimensional $(x : A)^{==}$: $\mathsf{R}_{(x:A)^=}$ and $(\mathsf{R}_{x:A})^=$, depicted as the left and right square, respectively.



In general, $(\mathsf{R}_{\Gamma^i})^{n-i}$ adds a degenerate dimension at index i, i.e. $x_{j_0 \dots j_{i-1} b j_i \dots j_{n-1}} \mapsto x_{j_0 \dots j_{n-1}}$ for $b = 0, 1$ and $x_{j_0 \dots j_{i-1} 2 j_i \dots j_{n-1}} \mapsto x_{j_0 \dots j_{n-1}}[(\mathsf{R}_{\Gamma^i})^{n-i}]$. This can be shown as follows: $(\mathsf{R}_{(x:A)^i})^{n-i}$ for arbitrary indices $j_0 \dots j_{i-1}$ includes $(x_{j_0 \dots j_{i-1} 0} \mapsto x_{j_0 \dots j_{i-1}}, x_{j_0 \dots j_{i-1} 1} \mapsto x_{j_0 \dots j_{i-1}}, x_{j_0 \dots j_{i-1} 2} \mapsto x_{j_0 \dots j_{i-1}}[R_{(x:A)^i}])^{n-i}$. We need to apply $-^=$ $(n - i)$ times to the part in parentheses. Applying $-^=$ once means substituting each component with $\mathsf{0}$ and $\mathsf{1}$ and applying $-^*$ while adding indices $0, 1$ and 2, respectively. For the first two components in the substitution we simply get $x_{j_0 \dots j_{i-1} b k} \mapsto x_{j_0 \dots j_{i-1} k}$ this way for $k = 0, 1, 2$. If the third component is substituted by b, we can use the naturality rule for b: $x_{j_0 \dots j_{i-1}}[\mathsf{R}_{(x:A)^i} b_{(x:A)^i}] \equiv x_{j_0 \dots j_{i-1}}[b_{(x:A)^{i+1}} (\mathsf{R}_{(x:A)^i})^=] \equiv x_{j_0 \dots j_{i-1} b}[(\mathsf{R}_{(x:A)^i})^=]$. And using the rule $(t[\rho])^* \equiv t^*[\rho^=]$ we

get $(x_{j_0 \dots j_{i-1}}[\mathsf{R}_{(x:A)^i} b_{(x:A)^i}])^* \equiv x_{j_0 \dots j_{i-1} 2}[(\mathsf{R}_{(x:A)^i})^=]$. By induction on $n - i$, we obtain the above rule.

The substitutions coming from iterating $-^=$ on $\mathsf{0}_\Gamma$, $\mathsf{1}_\Gamma$ and R_Γ can be depicted as follows.



2.6 A swapping substitution

We add another substitution to explain the relationship between $\mathsf{R}_{\Gamma=}$ and $(\mathsf{R}_\Gamma)^=$. This is a two-dimensional substitution which swaps the two dimensions of the square. We specify it with the following rule: it does not change the dimension of the context.

$$\frac{\Gamma \vdash}{\mathsf{S}_\Gamma : \Gamma^{==} \Rightarrow \Gamma^{==}}$$

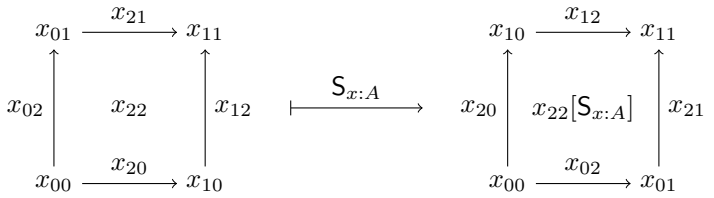
The first two computation rules explain how it acts on the empty and extended contexts. For the extended context, it swaps the dimensions i.e. maps x_{ij} to x_{ji} while the swapped x_{22} needs to be substituted with S because it has a different type now: the arguments of A^{**} have been swapped.

$$\begin{aligned}
\mathsf{S}_\cdot &\equiv \epsilon && : \cdot \Rightarrow \cdot \\
\mathsf{S}_{\Gamma.x:A} &\equiv (\mathsf{S}_\Gamma, x_{00} \mapsto x_{00}, x_{01} \mapsto x_{10}, x_{02} \mapsto x_{20}, \\
&\qquad x_{10} \mapsto x_{01}, x_{11} \mapsto x_{11}, x_{10} \mapsto x_{01}, \\
&\qquad x_{20} \mapsto x_{02}, x_{21} \mapsto x_{12}, x_{22} \mapsto x_{22}[\mathsf{S}_{\Gamma.x:A}]) : (\Gamma.x:A)^2 \Rightarrow (\Gamma.x:A)^2
\end{aligned}$$

The following three computation rules express that S is an isomorphism (applied twice it gives the identity), it has a naturality rule and it can be used to express $(\mathsf{R}_\Gamma)^=$: we swap the results of $\mathsf{R}_{\Gamma=}$.

$$\mathsf{S}_\Gamma \mathsf{S}_\Gamma \equiv \mathsf{id}_{\Gamma==} \qquad \frac{\rho : \Gamma \Rightarrow \Delta}{\mathsf{S}_\Delta \rho^{==} \equiv \rho^{==} \mathsf{S}_\Gamma} \qquad \mathsf{S}_\Gamma \mathsf{R}_{\Gamma=} \equiv (\mathsf{R}_\Gamma)^=$$

We can depict the effect of $\mathsf{S}_{x:A}$ as follows.



Just as $x_2[\mathsf{R}_{\Gamma.x:A}]$ introduced a new normal form, $x_{22}[\mathsf{S}_{\Gamma.x:A}]$ is a new normal form, however unfortunately we only know how to commute $-^{==}$-d substitutions with S using the naturality rule. For arbitrary substitutions, we are stuck. We come back to this problem in section 2.9.

By iterating $-^=$ on S we observe that at dimension n there are $n - 1$ different swap substitutions. $(\mathsf{S}_{\Gamma^i})^{n-2-i}$ swaps dimensions i and $i + 1$ of an n-dimensional cube ($n \geq 2$). With the same line of thought as for $(\mathsf{R}_{\Gamma^i})^{n-i}$, we can show that $(\mathsf{S}_{\Gamma^i})^{n-2-i}$ constitutes of

maps $x_{j_0 \dots j_{i-1} k l j_{i+2} \dots j_{n-1}} \mapsto x_{j_0 \dots j_{i-1} l k j_{i+2} \dots j_{n-1}}$ for $k, l = 0, 1, 2$ when k and l are not both 2 and has $x_{j_0 \dots j_{i-1} 22 j_{i+2} \dots j_{n-1}} \mapsto x_{j_0 \dots j_{i-1} 22 j_{i+2} \dots j_{n-1}}[(\mathsf{S}_{\Gamma^i})^{n-2-i}]$ for the other case.

Note that as we don't know how to commute arbitrary substitutions and swap, we also don't have all the rules for computing with compositions of swap substitutions. For example, we would like to have the equality $\mathsf{S}_\Gamma{=}\mathsf{S}_\Gamma{}^{=}\mathsf{S}_{\Gamma=} \equiv \mathsf{S}_\Gamma{}^{=}\mathsf{S}_{\Gamma=}\mathsf{S}_\Gamma{}^{=}$ (having 3 dimensions, swapping dimensions 1–2, then 0–1, then 1–2 gives the same result as swapping 0–1, then 1–2, then 0–1), however this is not yet justified by our rules.

2.7 Sigma types

We will need Σ types to express the constructs in the next section, we add them using the following rules in the usual way. We have both β and η computation rules.

$$\frac{\Gamma.x : A \vdash B : \mathsf{U}}{\Gamma \vdash \Sigma(x : A).B : \mathsf{U}} \qquad \big(\Sigma(x : A).B\big)[\rho] \equiv \Sigma(x : A[\rho]).B[\rho, x \mapsto x]$$

$$\frac{\Gamma \vdash u : A \qquad \Gamma \vdash v : B[x \mapsto u]}{\Gamma \vdash (u, v) : \Sigma(x : A).B} \qquad \frac{\Gamma \vdash w : \Sigma(x : A).B}{\Gamma \vdash \mathsf{proj}_1\, w : A} \qquad \frac{\Gamma \vdash w : \Sigma(x : A).B}{\Gamma \vdash \mathsf{proj}_2\, w : B[x \mapsto \mathsf{proj}_1\, w]}$$

$$(u, v)[\rho] \equiv (u[\rho], v[\rho]) \qquad (\mathsf{proj}_1\, w)[\rho] \equiv \mathsf{proj}_1\, (w[\rho]) \qquad (\mathsf{proj}_2\, w)[\rho] \equiv \mathsf{proj}_2\, (w[\rho])$$

$$\mathsf{proj}_1\, (u, v) \equiv u \qquad \mathsf{proj}_2\, (u, v) \equiv v \qquad (\mathsf{proj}_1\, w, \mathsf{proj}_2\, w) \equiv w$$

$-^*$ for Σ encodes that pairs are related if they are componentwise related. The relation for the second components depends on the relatedness of the first components (B^* only makes sense in context $(\Gamma.x : A)^=$), this is why we need to substitute it.

$$\big(\Sigma(x : A).B\big)^* \equiv \lambda w_0\, w_1.\Sigma\big(x_2 : A^*\, (\mathsf{proj}_1\, w_0)\, (\mathsf{proj}_1\, w_1)\big) \\ .B^*[x_0 \mapsto \mathsf{proj}_1\, w_0, x_1 \mapsto \mathsf{proj}_1\, w_1, x_2 \mapsto x_2]\, (\mathsf{proj}_2\, w_0)\, (\mathsf{proj}_2\, w_1)$$

$-^*$ on the constructor is componentwise and it commutes with the projections.

$$(u, v)^* \equiv (u^*, v^*) \qquad (\mathsf{proj}_1\, w)^* \equiv \mathsf{proj}_1\, (w^*) \qquad (\mathsf{proj}_2\, w)^* \equiv \mathsf{proj}_2\, (w^*)$$

2.8 Univalence

For a type A, $A^*[\mathsf{R}] : A \to A \to \mathsf{U}$ can be viewed as the identity type of A. We have reflexivity of this identity by defining

$$\mathsf{refl}\, a :\equiv a^*[\mathsf{R}],$$

however we cannot show transitivity and symmetry, let alone the J rule. To address this, as a first step we replace the definition of U^*. Instead of relation space, we would like to encode what equality should really be for the universe: equivalence. We use a notion of equivalence which includes a relation. Such a notion is given in exercise 4.2 in [17] where an equivalence of types A and B is given by the following Σ type where $\mathsf{isContr}\, X$ says that the type X is contractible (this is equivalent to the other notions of equivalence in [17]).

$$\Sigma(\sim : A \to B \to \mathsf{U}).\Pi(x : A).\mathsf{isContr}\, \big(\Sigma(y : B).x \sim y\big) \times \Pi(y : B).\mathsf{isContr}\, \big(\Sigma(x : A).x \sim y\big)$$

We replace the definition of U^* by an expansion of this definition. For a type A, A^* will now not only contain the relation, but also the $\mathsf{isContr}$ proofs, hence we need to adjust the

parametricity rule to use the relation part. We can call this rule congruence rule with the view of $\sim_{A^*}$ as the heterogeneous equality for A. $\sim$ projects out the relation part from A^*.

$$\frac{\Gamma \vdash t : A}{\Gamma^{=} \vdash t^* : t[\mathsf{0}] \sim_{A^*} t[\mathsf{1}]}$$

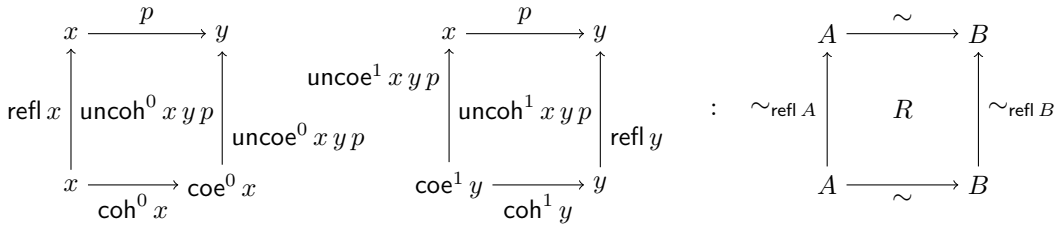
Similarly, we need to adjust the computation rule for $-^{=}$ on context extensions to add the projection $\sim$.

$$(\Gamma.x : A)^{=} \equiv \Gamma^{=}.x_0 : A[\mathsf{0}_\Gamma].x_1 : A[\mathsf{1}_\Gamma].x_2 : x_0 \sim_{A^*} x_1$$

We define equality in the universe by the following iterated Σ-type. The first component is an infix relation and we use $(x \sim)$ to denote $\lambda y.x \sim y$ and $(\sim y)$ to denote $\lambda x.x \sim y$.

$$\begin{aligned}
\sim_{\mathsf{U}^*} \equiv \lambda\, A\, B\, .\, \Sigma & - \sim - && : A \to B \to \mathsf{U} \\
& \mathsf{coe}^0 && : A \to B \\
& \mathsf{coh}^0 && : \Pi(x : A).x \sim \mathsf{coe}^0\, x \\
& \mathsf{uncoe}^0 && : \Pi(x : A, y : B, p : x \sim y).\mathsf{coe}^0\, x \sim_{\mathsf{refl}\, B}\, y \\
& \mathsf{uncoh}^0 && : \Pi(x : A, y : B, p : x \sim y).\mathsf{coh}^0\, x \sim_{\mathsf{refl}\, (x\sim)\, (\mathsf{coe}^0\, x)\, y\, (\mathsf{uncoe}^0\, x\, y\, p)}\, p \\
& \mathsf{coe}^1 && : B \to A \\
& \mathsf{coh}^1 && : \Pi(y : B).\mathsf{coe}^1\, y \sim y \\
& \mathsf{uncoe}^1 && : \Pi(x : A, y : B, p : x \sim y).\mathsf{coe}^1\, y \sim_{\mathsf{refl}\, A\, x} \\
& \mathsf{uncoh}^1 && : \Pi(x : A, y : B, p : x \sim y).\mathsf{coh}^1\, y \sim_{\mathsf{refl}\, (\sim y)\, (\mathsf{coe}^1\, y)\, x\, (\mathsf{uncoe}^1\, x\, y\, p)}\, p \\
: \mathsf{U} \to \mathsf{U} & \to \mathsf{U}
\end{aligned}$$

An equality between two types $A \sim_{\mathsf{U}^*} B$ is a relation between them, coercion functions (coe) from A to B and backwards, together with coherence proofs (coh) that the coercions respect the relations and uniqueness proofs (uncoe, uncoh) stating that any equality is equal to a coercion. We can depict these conditions as follows. The components with 0 indices give us the left square, the components with 1 indices give the middle square, while the right square shows the types of the other two squares.



$$R\, x_{00}\, x_{01}\, x_{02}\, x_{10}\, x_{11}\, x_{12}\, x_{20}\, x_{21} \equiv x_{20} \sim_{\mathsf{refl} \sim x_{00}\, x_{01}\, x_{02}\, x_{10}\, x_{11}\, x_{12}}\, x_{21}$$

Note that the type of the squares is degenerate in the vertical dimension. uncoe says that any y which is equal to x (by p) is equal to the coercion of x, while uncoh gives the equality of the coherence and the equality proof p. Using the definition of refl, we can unfold the types of the uncoh components as follows.

$$\begin{aligned}
&\mathsf{uncoh}^0\, x\, y\, p : \mathsf{coh}^0\, x \sim_{(x\sim y)^*[\mathsf{R}_{A:\mathsf{U}.B:\mathsf{U}.\sim:A\to B\to \mathsf{U}.x:A}, y_0 \mapsto \mathsf{coe}^0\, x, y_1 \mapsto y, y_2 \mapsto \mathsf{uncoe}^0\, x\, y\, p]}\, p \\
&\mathsf{uncoh}^1\, x\, y\, p : \mathsf{coh}^1\, y \sim_{(x\sim y)^*[\mathsf{R}_{A:\mathsf{U}.B:\mathsf{U}.\sim:A\to B\to \mathsf{U}.y:B}, x_0 \mapsto \mathsf{coe}^1\, y, x_1 \mapsto x, x_2 \mapsto \mathsf{uncoe}^1\, x\, y\, p]}\, p
\end{aligned}$$

As we replaced the relation space for U^* by the above definition, we need to define the new components $\sim$, (un)coe and (un)coh for the type formers U, Π and Σ. The $\sim$ component

for U^* is defined above, and for Π and Σ we just use the relations given in the previous sections, i.e.

$$\sim_{(\Pi(x:A).B)^*} \equiv \lambda f_0\, f_1.\Pi(x_0 : A[\mathsf{0}], x_1 : A[\mathsf{1}], x_2 : x_0 \sim_{A^*} x_1).f_0\, x_0 \sim_{B^*} f_1\, x_1$$
$$\sim_{(\Sigma(x:A).B)^*} \equiv \lambda w_0\, w_1.\Sigma(x_2 : \mathsf{proj}_1\, w_0 \sim_{A^*} \mathsf{proj}_1\, w_1)$$
$$.\mathsf{proj}_2\, w_0 \sim_{B^*[x_0 \mapsto \mathsf{proj}_1\, w_0, x_1 \mapsto \mathsf{proj}_2\, w_1, x_2 \mapsto x_2]} \mathsf{proj}_2\, w_1.$$

First we show how to define coe and coh for U and Σ (section 2.8.1), then we show how to derive coercion and coherence for squares (2-dimensional version of the 1-dimensional coe and coh) and show how to use this to derive uncoe and uncoh (section 2.8.2). Then we derive coe and coh for Π (section 2.8.3). Finally, we show how to derive the usual elimination rule for homogeneous equality (section 2.8.4).

2.8.1 coe and coh for U and Σ

We use the following abbreviation to concisely express $\sim$ in both directions.

$$x \overset{0}{\sim}_e y \equiv x \sim_e y \qquad y \overset{1}{\sim}_e x \equiv x \sim_e y$$

For U we simply use the identity function as coercion and reflexivity for coherence.

$$\Gamma^= \vdash \mathsf{coe}^b_{\mathsf{U}^*} \equiv \mathsf{id} : \mathsf{U} \to \mathsf{U}$$
$$\Gamma^= \vdash \mathsf{coh}^b_{\mathsf{U}^*} \equiv \mathsf{refl} : \Pi(A : \mathsf{U})\,.\, A \overset{b}{\sim}_{\mathsf{U}^*} A$$

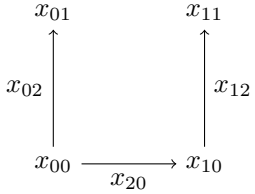
For Σ types, coercion and coherence are pointwise.

$$\Gamma^= \vdash \mathsf{coe}^b_{(\Sigma(x:A).B)^*} \equiv \lambda(u, v).(\mathsf{coe}^b_{A^*}\, u, \mathsf{coe}^b_{B^*}[x_2 \mapsto \mathsf{coh}^b_{A^*}\, u]\, v)$$
$$: (\Sigma(x : A).B)[b] \to (\Sigma(x : A).B)[1 - b]$$
$$\Gamma^= \vdash \mathsf{coh}^b_{(\Sigma(x:A).B)^*} \equiv \lambda(u, v).(\mathsf{coh}^b_{A^*}\, u, \mathsf{coh}^b_{B^*}[x_2 \mapsto \mathsf{coh}^b_{A^*}\, u]\, v)$$
$$: \Pi(w : (\Sigma(x : A).B)[b])\,.\, w \overset{b}{\sim}_{(\Sigma(x:A).B)^*} \mathsf{coe}^b_{(\Sigma(x:A).B)^*}\, w$$

We used pattern matching lambdas for ease of notation and in the substitution $[x_2 \mapsto \mathsf{coh}^b_{A^*}\, u]$ we left the x_0, x_1 components implicit. We need the $[x_2 \mapsto \mathsf{coh}^b_{A^*}\, u]$ substitution when we using coe_{B^*} and coh_{B^*} because B is in the context $\Gamma, x : A$, hence B^* is in the context $(\Gamma, x : A)^=$ which includes the components $x_0 : A[\mathsf{0}], x_1 : A[\mathsf{1}], x_2 : x_0 \sim_{A^*} x_1$ and we need to provide these components.

2.8.2 Deriving coe, coh for squares from coe, coh for lines

A square with a missing top line and a filler can be depicted as follows.



This corresponds to the last 7 components of the context $(\Gamma^=.x_0 : A[\mathsf{0}].x_1 : A[\mathsf{1}])^=.x_{20} : x_{00} \sim_{A^*[\mathsf{0}]} x_{10}$. Coercion for the type $x_0 \sim_{A^*} x_1$ in this context has the following type.

$$(\Gamma^=.x_0 : A[\mathsf{0}].x_1 : A[\mathsf{1}])^= \vdash \mathsf{coe}^{\mathsf{0}}_{(x_0 \sim_{A^*} x_1)^*} : x_{00} \sim_{A^*[\mathsf{0}]} x_{10} \to x_{01} \sim_{A^*[\mathsf{1}]} x_{11}$$

We can use this coercion to get the top of the square from x_{20} and similarly, we can use coherence to obtain the filler. Putting them together we get a substitution which fills in the last two components of the square.

$$\begin{aligned}\mathsf{K}^{0,0}_{\Gamma.x:A} :\equiv \big(&\mathsf{id}_{(\Gamma^{=}.x_0:A[0].x_1:A[1])^{=}},\\ &x_{20}\mapsto x_{20}, x_{21}\mapsto \mathsf{coe}^0_{(x_0\sim_{A^*}x_1)^*}\, x_{20}, x_{22}\mapsto \mathsf{coh}^0_{(x_0\sim_{A^*}x_1)^*}\, x_{20}\big)\\ &: (\Gamma^{=}.x_0 : A[0].x_1 : A[1])^{=}.x_{20} : x_{00}\sim_{A^*[0]} x_{10} \Rightarrow (\Gamma.x:A)^{==}\end{aligned}$$

If the right side of a square is missing, we can use the swap substitution to obtain a filler. First we swap the components we have, then apply the above $\mathsf{K}^{0,0}_{\Gamma.x:A}$, then swap the components back so that everything stays in its original place. We denote this substitution $\mathsf{K}^{1,0}_{\Gamma.x:A}$.

$$\begin{array}{c}(\Gamma^{=}.x:A[0])^{=}.x_{10}:A[\mathsf{10}].x_{11}:A[\mathsf{11}].x_{20}:x_{00}\sim_{A^*[0]}x_{10}.x_{21}:x_{01}\sim_{A^*[1]}x_{11}\\ \big(\mathsf{S}_\Gamma, x_{00}\mapsto x_{00}, x_{01}\mapsto x_{10}, x_{02}\mapsto x_{20}, x_{10}\mapsto x_{01}, x_{11}\mapsto x_{11}, x_{12}\mapsto x_{21}, x_{20}\mapsto x_{02}\big)\downarrow\\ (\Gamma^{=}.x_0:A[0].x_1:A[1])^{=}.x_{20}:x_{00}\sim_{A^*[0]}x_{10}\\ \downarrow \mathsf{K}^{0,0}_{\Gamma.x:A}\\ (\Gamma.x:A)^{==}\\ \downarrow \mathsf{S}_{\Gamma.x:A}\\ (\Gamma.x:A)^{==}\end{array}$$

With the help of $\mathsf{K}^{1,0}$ we can define uncoe^0 and uncoh^0 for a given type $\Gamma\vdash A:\mathsf{U}$. We use a substitution ρ to define the common parts.

$$\begin{aligned}\rho :\equiv\ &\big((\mathsf{R}_\Gamma)^{=}, x_{00}\mapsto x, x_{01}\mapsto x, x_{02}\mapsto \mathsf{refl}\, x,\\ &x_{10}\mapsto \mathsf{coe}_{A^*}\, x, x_{11}\mapsto y, x_{20}\mapsto \mathsf{coh}_{A^*}\, x, x_{21}\mapsto p\big)\\ &: (\Gamma^{=}.x:A[0].y:A[1].p:x\sim_{A^*} y)\Rightarrow(\Gamma.x:A)^2\\ \Gamma^{=}\vdash\ &\mathsf{uncoe}_{A^*}\equiv \lambda\, x\, y\, p.x_{12}[\mathsf{K}^{1,0}_{\Gamma.x:A}\rho] : \Pi(x:A, y:B, p:x\sim y).\mathsf{coe}^0\, x\sim_{\mathsf{refl}\, B}\, y\\ \Gamma^{=}\vdash\ &\mathsf{uncoh}_{A^*}\equiv \lambda\, x\, y\, p.x_{22}[\mathsf{K}^{1,0}_{\Gamma.x:A}\rho] : \Pi(x:A, y:B, p:x\sim y)\\ &\qquad\qquad .\mathsf{coh}^0\, x\sim_{\mathsf{refl}\,(x\sim)\,(\mathsf{coe}^0\, x)\, y\,(\mathsf{uncoe}^0\, x\, y\, p)}\, p\end{aligned}$$

We can analogously define $\mathsf{K}^{0,1}$ and $\mathsf{K}^{1,1}$ which go in the opposite directions (and $\mathsf{K}^{1,1}$ can be used to define uncoe^1 and uncoh^1). As a summary, we list here the directions of filling for a square that we get using these substitutions.

$\mathsf{K}^{0,0}$	bottom to top	$\uparrow$
$\mathsf{K}^{0,1}$	top to bottom	$\downarrow$
$\mathsf{K}^{1,0}$	left to right	$\rightarrow$
$\mathsf{K}^{1,1}$	right to left	$\leftarrow$

2.8.3 coe and coh for Π

The coerce operation for Π types takes a function of type $(\Pi(x:A).B)[0]$ and needs to return a function of type $(\Pi(x:A).B)[1]$. We define the latter function by first coercing backwards from $A[1]$ to $A[0]$, then applying the original function, then coercing forward the result from

$B[0]$ to $B[1]$. We define it for both directions as follows.

$$\begin{aligned}\Gamma^{=} \vdash \mathsf{coe}^{b}_{(\Pi(x:A).B)^*} &\equiv \lambda f.\lambda x.\mathsf{coe}^{b}_{B^*}[x_2 \mapsto \mathsf{coh}^{1-b}_{A^*}\, x]\,(f\,(\mathsf{coe}^{1-b}_{A^*}\, x)) \\ &: (\Pi(x:A).B)[b] \to (\Pi(x:A).B)[1-b]\end{aligned}$$

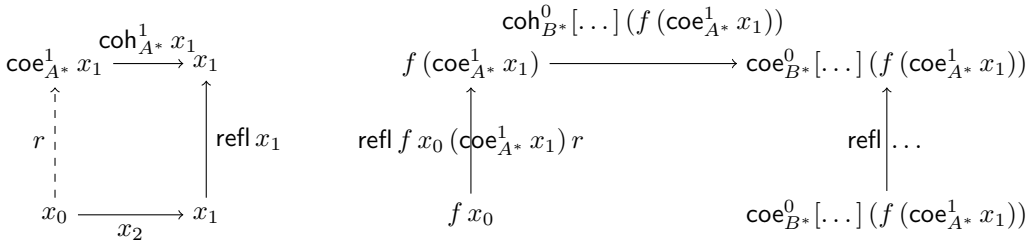
We left the other two components of the substitution $[x_2 \mapsto \mathsf{coh}^{1-b}_{A^*}\, x]$ implicit.

The coherence operation for Π needs to have the following type.

$$\begin{aligned}\Gamma^{=} \vdash \mathsf{coh}^{b}_{(\Pi(x:A).B)^*} :&\Pi\big(f : (\Pi(x:A).B)[b], x_0 : A[0], x_1 : A[1], x_2 : x_0 \sim_{A^*} x_1\big) \\ &.\, f\, x_b \overset{b}{\sim}_{B^*} \mathsf{coe}^{b}_{B^*}[x_2 \mapsto \mathsf{coh}^{1-b}_{A^*}\, x_{1-b}]\,(f\,(\mathsf{coe}^{1-b}_{A^*}\, x_{1-b}))\end{aligned}$$

We explain in detail how to define $\mathsf{coh}^{0}_{(\Pi(x:A)B)^*}\, f\, x_0\, x_1\, x_2$, the other direction is symmetric.

We start in the context $\Gamma^{=}.f : (\Pi(x:A).B)[0].x_0 : A[0].x_1 : A[1].x_2 : x_0 \sim_{A^*} x_1$. First we will fill the left incomplete square thereby obtaining the dashed line r and using this r we construct the right incomplete square and filling it gives us the line in the bottom which is exactly what we need. Note that the left square has telescope type $(x:A)^2[(\mathsf{R}_\Gamma)^{=}]$ while the right square has telescope type $(y:B)^2[(\mathsf{R}_\Gamma)^{=}]$ and depends on the first square (by $(x:A)^2$ we mean the telescope $x_{00} : A[00], x_{01} : A[01], \ldots, x_{22} : x_{20} \sim_{(x_0 \sim_{A^*} x_1)^*} x_{21}$ which contains the last 9 elements of the context $(\Gamma, x:A)^{==}$).



First we define the incomplete first square as a substitution ρ.

$$\begin{aligned}\rho \equiv&\big((\mathsf{R}_\Gamma)^{=}, x_{00} \mapsto x_0, x_{01} \mapsto \mathsf{coe}^1_{A^*}\, x_1, x_{10} \mapsto x_1, x_{11} \mapsto x_1, x_{12} \mapsto \mathsf{refl}\, x_1, \\ &x_{20} \mapsto x_2, x_{21} \mapsto \mathsf{coh}^1_{A^*}\, x_1\big) \\ &: \big(\Gamma^{=}.f : (\Pi(x:A).B)[0].x_0 : A[0].x_1 : A[1].x_2 : x_0 \sim_{A^*} x_1\big) \\ &\Rightarrow \Gamma^2.x_{00} : A[00].x_{01} : A[01].x_{10} : A[10].x_{11} : A[11].x_{12} : x_{10} \sim_{(A[1])^*} x_{11} \\ &\quad .x_{20} : x_{00} \sim_{A^*[0]} x_{10}.x_{21} : x_{01} \sim_{A^*[1]} x_{11}\end{aligned}$$

The following σ substitution defines the (complete) first square and the incomplete second square.

$$\begin{aligned}\sigma \equiv \big((\mathsf{R}_\Gamma)^{=},\ & x_{00} \mapsto x_0, x_{01} \mapsto \mathsf{coe}^1_{A^*}\, x_1, x_{02} \mapsto x_{02}[\mathsf{K}^{1,1}][\rho], \\ & x_{10} \mapsto x_1, x_{11} \mapsto x_1, x_{12} \mapsto \mathsf{refl}\, x_1 \\ & x_{20} \mapsto x_2, x_{21} \mapsto \mathsf{coh}^1_{A^*}\, x_1, x_{22} \mapsto x_{22}[\mathsf{K}^{1,1}][\rho], \\ & y_{00} \mapsto f\, x_0, y_{01} \mapsto f\,(\mathsf{coe}^1_A\, x_1), y_{02} \mapsto f^*[\mathsf{R}_{\Gamma^=}]\, x_0\, (\mathsf{coe}^1_A\, x_1)\, (x_{02}[\mathsf{K}^{1,1}][\rho]) \\ & y_{10}, y_{11} \mapsto \mathsf{coe}^0_{B^*}[x_2 \mapsto \mathsf{coh}^1_{A^*}\, x]\,(f\,(\mathsf{coe}^1_{A^*}\, x)), \\ & y_{12} \mapsto \mathsf{refl}\,(\mathsf{coe}^0_{B^*}[x_2 \mapsto \mathsf{coh}^1_{A^*}\, x]\,(f\,(\mathsf{coe}^1_{A^*}\, x)))\big) \\ : \big(&\Gamma^{=}.f : (\Pi(x:A).B)[0].x_0 : A[0].x_1 : A[1].x_2 : x_0 \sim_{A^*} x_1\big) \\ \Rightarrow\ & ((\Gamma.x:A)^{=}.y_0 : B[0].y_1 : B[1])^{=}.y_{21} : y_{01} \sim_{B^*[1]} y_{11}\end{aligned}$$

Using σ we can define the coherence operation for Π as follows.

$$\Gamma^{=} \vdash \mathsf{coh}^0_{(\Pi(x:A).B)^*} \equiv \lambda f\, x_0\, x_1.\, x_2.\mathsf{coe}^1_{(y_0 \sim_{B^*} y_1)^*[\sigma]} \left(\mathsf{coh}^0_{B^*}[x_2 \mapsto \mathsf{coh}^1_{A^*}\, x]\, (f\, (\mathsf{coe}^1_{A^*}\, x))\right)$$
$$: \Pi\big(f : (\Pi(x : A).B)[\mathsf{0}], x_0 : A[\mathsf{0}], x_1 : A[\mathsf{1}], x_2 : x_0 \sim_{A^*} x_1\big).f\, x_0 \sim_{B^*} \mathsf{coe}_{(\Pi(x:A).B)^*}\, f\, x_1$$

2.8.4 The elimination rule for equality

We define the homogeneous equality $\Gamma \vdash a =_A b : \mathsf{U}$ as $\Gamma \vdash a \sim_{A^*[\mathsf{R}_\Gamma]} b : \mathsf{U}$.

We express the eliminator for equality as the transport function and the fact that singletons are contractible. From these two principles the eliminator J can be derived.

$$\frac{\Gamma \vdash P : A \to \mathsf{U} \quad \Gamma \vdash r : a =_A b \quad \Gamma \vdash u : P\, a}{\Gamma \vdash \mathsf{transport}_P\, r\, u : P\, b} \qquad \frac{\Gamma \vdash a, b : A \quad \Gamma \vdash r : a =_A b}{\Gamma \vdash\ (s, t) : (a, \mathsf{refl}\, a) =_{\Sigma(x:A).a=_A x} (b, r)}$$

We validate transport using coe^0 and $-^*$ on the predicate P.

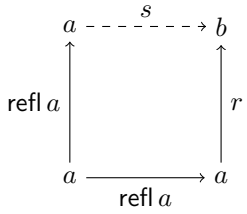
$$\Gamma \vdash \mathsf{transport}_P\, r\, u \equiv \mathsf{coe}^0_{P^*[\mathsf{R}_\Gamma]\, a\, b\, r}\, u : P\, b$$

The computation rule for transport is validated by the corresponding coherence law (up to equality, but not definitional equality). $\mathsf{coh}^0_{P^*[\mathsf{R}_\Gamma]\, a\, a\, (\mathsf{refl}\, a)}\, u : u =_{P\, a} \mathsf{transport}_P\, (\mathsf{refl}\, a)\, u\, u$.

The type of (s, t) in the second principle is equal to

$$\Sigma(s : a \sim_{A^*[\mathsf{R}_\Gamma]} b).\mathsf{refl}\, a \sim_{(a \sim_{A^*[\mathsf{R}_\Gamma]} x)^*[\mathsf{R}_\Gamma, a, b, s]} r.$$

We construct s by filling the following incomplete square from bottom to top.



We spell it out explicitly below.

$$\rho :\equiv (\mathsf{R}_{\Gamma^=} \mathsf{R}_\Gamma, x_{00} \mapsto a, x_{01} \mapsto a, x_{02} \mapsto \mathsf{refl}\, x_{10} \mapsto a, x_{11} \mapsto b, x_{12} \mapsto r)$$
$$: \Gamma \Rightarrow (\Gamma^{=}.x_0 : A[\mathsf{0}].x_1 : A[\mathsf{1}])^{=}$$
$$\Gamma \vdash s :\equiv \mathsf{coe}^0_{(x_0 \sim_{A^*} x_1)^*}[\rho](\mathsf{refl}\, a) : a \sim_{A^*[\mathsf{R}_\Gamma]} b$$

The coherence gives us the filler, however we need to swap it to get the type that we need. Without swapping we get the following type.

$$\Gamma \vdash \mathsf{coh}^0_{(x_0 \sim_{A^*} x_1)^*}[\rho](\mathsf{refl}\, a) : \mathsf{refl}\, a \sim_{(x_0 \sim_{A^*} x_1)^*[\rho]} s$$

We define a substitution σ and compose it with $\mathsf{S}_{\Gamma.x:A}$ and we define t as the last term in the resulting substitution.

$$\Gamma \xrightarrow{\sigma} (\Gamma.x : A)^{==} \xrightarrow{\mathsf{S}_{\Gamma.x:A}} (\Gamma.x : A)^{==}$$

$$\sigma :\equiv \big(\mathsf{R}_{\Gamma^=} \mathsf{R}_\Gamma, x_{00} \mapsto a, x_{01} \mapsto a, x_{02} \mapsto \mathsf{refl}\, a,$$
$$x_{10} \mapsto a, x_{11} \mapsto b, x_{12} \mapsto r,$$
$$x_{20} \mapsto \mathsf{refl}\, a, x_{21} \mapsto \mathsf{coe}^0_{(x_0 \sim_{A^*} x_1)^*}[\rho](\mathsf{refl}\, a), x_{22} \mapsto \mathsf{coh}^0_{(x_0 \sim_{A^*} x_1)^*}[\rho](\mathsf{refl}\, a)\big)$$
$$\Gamma \vdash t :\equiv x_{22}[\mathsf{S}_{\Gamma.x:A}\sigma] : \mathsf{refl}\, a \sim_{(a \sim_{A^*[\mathsf{R}_\Gamma]} x)^*[\mathsf{R}_\Gamma, a, b, s]} r$$

2.9 Metatheoretic properties

The operational semantics for the theory of external parametricity (section 2.3) is clear, we can always eliminate the new constructs $-^{=}, -^{*}$, $\mathsf{0}$, $\mathsf{1}$. We formalised the unary version of this theory as a syntactic translation, see [2].

After adding reflexivities (section 2.4), we lose the property that we can translate everything back to the original theory as we have new normal forms such as $x_2[\mathsf{R}_{\Gamma.x:A}]$. And even worse, we don't know how to substitute arbitrary substitutions into $x_{22}[(\mathsf{R}_{\Gamma.x:A})^{=}]$, i.e. $x_{22}[(\mathsf{R}_{\Gamma.x:A})^{=}\rho]$ is stuck. We can defer the problem by introducing S and expressing $(\mathsf{R}_\Gamma)^{=}$ as $\mathsf{S}_\Gamma\mathsf{R}_{\Gamma^{=}}$, but then we need to commute S and arbitrary substitutions which we don't know how to do. This is a problem indeed in practice, e.g. we don't know how to compute with $\mathsf{K}^{1,0}$ from section 2.8.2 which is defined by postcomposing S with another substitution. We describe a possible solution in section 3, but this is notationally quite heavy.

We believe however that the syntax up to section 2.6 has a presheaf model with base category the category of renamings from [13]. This category does not include connections [11], only face maps (corresponding to our $\mathsf{0}$ and $\mathsf{1}$), degeneracies (corresponding to R) and renamings (corresponding to S). The idea of the construction is as follows. We denote the set with n elements (an object of the base category) $\overline{n}$. An interpretation of a context Γ is a covariant presheaf $[\![\Gamma]\!] : \mathcal{C} \to \mathbf{Set}$, and for lifted contexts we define $[\![\Gamma^{=}]\!]\,\overline{n} := [\![\Gamma]\!]\,\overline{n+1}$. The interpretations of $\mathsf{0}$ and $\mathsf{1}$ are natural transformations and we set $[\![\mathsf{0}_\Gamma]\!]_{\overline{n}} := [\![\Gamma]\!]\,(d_n = 0)$ i.e. we use the action on morphisms for $[\![\Gamma]\!]$ at the morphism which sends dimension n to 0. A lifted substitution is the unlifted substitution at a higher dimensional object, $[\![\rho^{=}]\!]_{\overline{n}} := [\![\rho]\!]_{\overline{n+1}}$. A type $\Gamma \vdash A$ is interpreted as a family of presheaves over $[\![\Gamma]\!]$. A lifted type $A^{*}\,x_0\,x_1$ (which depends on the last two elements in the context being $x_0 : A[\mathsf{0}], x_1 : A[\mathsf{1}]$) is interpreted as the element of the higher type where the projections give the two faces in the context.

$$[\![A^{*}\,x_0\,x_1]\!]_{\overline{n}}\,(\gamma, a_0, a_1) := \left(a : [\![A]\!]_{\overline{n+1}}\,\gamma\right) \times [\![A]\!]\,(d_n = 0)\,a = a_0 \times [\![A]\!]\,(d_n = 1)\,a = a_1$$

Terms are interpreted as sections, lifted terms are interpreted as triples corresponding to the above sum types: $[\![t^{*}]\!]_{\overline{n}}\,\gamma := ([\![t]\!]_{\overline{n+1}}\,\gamma, \mathsf{refl}, \mathsf{refl})$. Some equalities however are only validated up to isomorphism in this model. E.g. $-^{=}$ on context extension is only validated up to an isomorphism which follows from the fact that singletons are contractible.

$$\begin{aligned}
&[\![(\Gamma.x : A)^{=}]\!]_{\overline{n}}\\
=\ &(\gamma : [\![\Gamma]\!]_{\overline{n+1}}) \times [\![A]\!]_{\overline{n+1}}\,\gamma\\
\simeq\ &(\gamma : [\![\Gamma]\!]_{\overline{n+1}}) \times \left(a_0 : [\![A]\!]\,([\![\Gamma]\!]\,(d_n = 0)\,\gamma)\right) \times \left(a_1 : [\![A]\!]\,([\![\Gamma]\!]\,(d_n = 1)\,\gamma)\right)\\
&\times \left(a : [\![A]\!]_{\overline{n+1}}\,\gamma\right) \times [\![A]\!]\,(d_n = 0)\,a = a_0 \times [\![A]\!]\,(d_n = 1)\,a = a_1\\
=\ &[\![\Gamma^{=}.x_0 : A[\mathsf{0}].x_1 : A[\mathsf{1}].x_2 : A^{*}\,x_0\,x_1]\!]_{\overline{n}}
\end{aligned}$$

This can be remedied using a univalent metatheory (where isomorphic sets are equal) or by a refinement of the presheaf model as in [6] where presheaves return sets with additional structure (called I-sets for an object I (set of dimension names) in the base category; an I-set is a set of I-indexed tuples).

We also believe that the presheaf model of the syntax with the Kan operations (section 2.8) has Kan operations with the uniformity condition as in [13].

3 Cubical type theory with named dimensions

In this section, we remedy the problem of the previous approach which was that we didn't know how to compute with the S substitutions. We define a new syntax where each usage of

the $-^=$ and $-^*$ operators is decorated with a dimension name. A two-dimensional context $\Gamma^{==}$ will be denoted Γ^{ij} where i and j are dimension names. They are assumed to be fresh in every rule. With named dimensions at our hand, we will equate the contexts Γ^{ij} and Γ^{ji} as they contain the same information in different order. A context $(x : A)^{ij}$ can be viewed as a collection of types indexed by $i = 0, 1, 2$ and $j = 0, 1, 2$ regardless of the order; i and j are the dimension names, and given a coordinate for each dimension, we get a type. The information about which dimension has which index will be stored in the variable names, e.g. x_{i0j1}, x_{i1j2}. This eliminates the need of the swapping substitution S and the problem with its computation rules. However the theory becomes more involved as we need Π types to remember their dimensions as well. As higher dimensional contexts are order-agnostic, lifted Π types need to be order-agnostic too, hence they need to carry information about their dimensions. Instead of defining the lifted relation $(\Pi(x : A).B)^i$ as an iterated Π type $\Pi(x_{i0} : A[0]).\Pi(x_{i1} : A[1].\Pi(x_{i2} : x_{i0} \sim_{A^i} x_{i1}) \dots$, we will use $\Pi(x : A)^i \dots$ which shows that this type has arguments of dimension i.

We start with the core substitution calculus given in section 2.1. First we add telescopes to the calculus in section 3.1. Then we define the function space with dimensional information in section 3.2. Sections 3.3 and 3.4 show how to define the parametricity operation (called $-^=$ before) for this calculus. Section 3.5 adds a definitional quotient thereby e.g. equating Γ^{ij} and Γ^{ji}. We need this quotient to internalise parametricity in section 3.6. This version corresponds to section 2.4 of the naive calculus. We stop here and don't interpret univalence in the nominal calculus. In appendix A we define an operational semantics.

3.1 Telescopes

We add new judgment types to the core substitution calculus defined in section 2.1.

$\Gamma \vdash \Omega$	Ω is a telescope context in context Γ
$\Gamma \vdash \omega : \Omega$	ω is a telescope substitution of type Ω in context Γ
$\Gamma \vdash \Omega \equiv \Omega'$	definitional equality of telescope contexts
$\Gamma \vdash \omega \equiv \omega' : \Omega$	definitional equality of telescope substitutions

Explanation:

$$\begin{array}{rlll} \text{Just as} & \Gamma \vdash t : A & \text{can be viewed as} & (\mathsf{id}_\Gamma, x \mapsto t) : \Gamma \Rightarrow \Gamma.x : A, \\ & \Gamma \vdash \Omega & \text{can be viewed as} & \Gamma +\!\!+ \Omega \vdash \\ \text{and} & \Gamma \vdash \omega : \Omega & \text{can be viewed as} & (\mathsf{id}_\Gamma +\!\!+ \omega) : \Gamma \Rightarrow \Gamma +\!\!+ \Omega. \end{array}$$

Telescope contexts are generalisations of types into lists of types which might depend on each other. They can also be thought of as named iterated Σ-types.

$$\frac{\Gamma \vdash}{\Gamma \vdash \cdot} \qquad \frac{\Gamma \vdash \quad \Gamma \vdash \Omega \quad \Gamma +\!\!+ \Omega \vdash A : \mathsf{U}}{\Gamma \vdash \Omega.x : A}$$

We explain how to extend contexts with telescope contexts:

$$\frac{\Gamma \vdash \Omega}{\Gamma +\!\!+ \Omega \vdash} \qquad \Gamma +\!\!+ \cdot \equiv \Gamma \qquad \Gamma +\!\!+ (\Omega.x : A) \equiv (\Gamma +\!\!+ \Omega).x : A$$

Substitution of telescope contexts is pointwise:

$$\frac{\Gamma \vdash \Omega \quad \rho : \Delta \Rightarrow \Gamma}{\Delta \vdash \Omega[\rho]} \qquad \cdot[\rho] \equiv \cdot \qquad (\Omega.x : A)[\rho] \equiv \Omega[\rho].x : A[\rho]$$

Telescope substitutions are generalisations of terms into lists of terms.

$$\frac{\Gamma \vdash}{\Gamma \vdash \epsilon : \cdot} \qquad \frac{\Gamma \vdash \omega : \Omega \quad \Gamma +\!\!+ \Omega \vdash A : \mathsf{U} \quad \Gamma \vdash t : A[(\mathsf{id}_\Gamma +\!\!+ \omega)]}{\Gamma \vdash (\omega, x \mapsto t) : \Omega.x : A}$$

We explain how to extend normal substitutions with telescope substitutions:

$$\frac{\rho : \Delta \Rightarrow \Gamma \quad \Delta \vdash \omega : \Omega[\rho]}{\rho +\!\!+ \omega : \Delta \Rightarrow \Gamma +\!\!+ \Omega} \qquad \rho +\!\!+ \epsilon \equiv \rho \qquad \rho +\!\!+ (\omega, x \mapsto t) \equiv (\rho +\!\!+ \omega, x \mapsto t)$$

Substitution of telescope substitutions is pointwise:

$$\frac{\Gamma \vdash \omega : \Omega \quad \rho : \Delta \Rightarrow \Gamma}{\Delta \vdash \omega[\rho] : \Omega[\rho]} \qquad \epsilon[\rho] \equiv \epsilon \qquad (\omega, x \mapsto t)[\rho] \equiv (\omega[\rho], x \mapsto t[\rho])$$

Extending telescope contexts with telescope contexts and telescope substitutions with telescope substitutions is done in the obvious way. Specification:

$$\frac{\Gamma \vdash \Omega \quad \Gamma +\!\!+ \Omega \vdash \Omega'}{\Gamma \vdash \Omega +\!\!+ \Omega'} \qquad \Omega +\!\!+ \cdot \equiv \Omega \qquad \Omega +\!\!+ (\Omega'.x : A) \equiv \Omega +\!\!+ \Omega'.x : A$$

$$\frac{\Gamma \vdash \omega : \Omega \quad \Gamma \vdash \omega' : \Omega'[\omega]}{\Gamma \vdash \omega +\!\!+ \omega' : \Omega +\!\!+ \Omega'} \qquad \omega +\!\!+ \epsilon \equiv \omega \qquad \omega +\!\!+ (\omega', x \mapsto t) \equiv (\omega +\!\!+ \omega', x \mapsto t)$$

The pr telescope substitution generalises the projection $\Gamma.x : A \vdash x : A$.

$$\frac{\Gamma \vdash \Omega}{\Gamma +\!\!+ \Omega \vdash \mathsf{pr}_\Omega : \Omega} \qquad \Gamma +\!\!+ \cdot \vdash \mathsf{pr}_\cdot \equiv \epsilon : \cdot \qquad \Gamma +\!\!+ (\Omega.x : A) \vdash \mathsf{pr}_{\Omega.x:A} \equiv (\mathsf{pr}_\Omega, x \mapsto x) : (\Omega.x : A)$$

Note that we haven't added new types in the universe, telescopes live outside the world of types.

3.2 Function space

We would like to have $\Gamma^{ij} \equiv \Gamma^{ji}$. This forces us to provide a new type former for $\Pi(x : A)^{ij}.B$ instead of using the iterated $\Pi(x_{i0j0} : A[\mathsf{0}_i\mathsf{0}_j]).\Pi(x_{i0j1} : A[\mathsf{0}_i\mathsf{1}_j]).\ldots.B$. When applying arguments to such a function, we need to supply all of them at the same time, because the order of arguments won't matter, only their dimension indices.

For this reason, we add functions where the domain can be a telescope context. However, we only allow special telescope contexts, ones which arise from applying the lifting operator zero or more times. Hence, a function will be able to have only 3^k number of arguments where k is the number of dimensions. We will denote such a telescope context $(x : A)^I$, where I is a set of dimensions which might be empty.

Also, to express the type of the relation $A^i : A[\mathsf{0}_i] \to A[\mathsf{1}_i] \to \mathsf{U}$, we need functions with incomplete cube arguments, $\{x : A\}_I$ will denote $(x : A)^I$ without the last element, so $A^i : \Pi\{x : A\}_i.\mathsf{U}$. Hence, we will also have functions with $3^k - 1$ number of arguments.

We will add rules to compute $(x : A)^I$ and $\{x : A\}_I$ in the next section.

Our function space also needs to be stable under substitution. For example, consider the type $(X : \mathsf{U})^i \vdash \Pi(x : X)^i.B : \mathsf{U}$. Here all the 3 types in the domain of the function are variables, hence they can be given arbitrary values by a substitution $\rho : \cdot \Rightarrow (X : \mathsf{U})^i$. Performing the substitution $(\Pi(x : X)^i.B)[\rho]$ results in $\Pi(x_{i0} : X_{i0}[\rho], x_{i1} : X_{i1}[\rho], x_{i2} : X_{i2}[\rho]\, x_{i0}\, x_{i1}).B[\rho]$ which does not have the form $\Pi(x : A)^i.B$ for some A anymore. This is

we decorate the domain of the function with a telescope substitution providing the types and A will be fixed to be a variable X. The formation rules:

$$\frac{\Gamma \vdash \xi : (X : \mathsf{U})^I \quad \Gamma +\!\!+ (x : X)^I[\xi] \vdash B : \mathsf{U}}{\Gamma \vdash \Pi(x : X)^I[\xi].B : \mathsf{U}} \qquad \frac{\Gamma \vdash \xi : \{X : \mathsf{U}\}_{Ii}}{\Gamma \vdash \Pi\{x : X\}_{Ii}[\xi].\mathsf{U} : \mathsf{U}}$$

These are just the usual rules for telescope functions, restricted on the domain (and for relations, in the codomain too). Note that the domain for relation space can't be zero-dimensional. Also, we haven't defined substituting a telescope context with a telescope substitution, so we mean $(x : X)^I[\mathsf{id} +\!\!+ \xi]$ when we write $(x : X)^I[\xi]$.

Most rules have the same shape for functions and relations, for these, we introduce the notation $(\!|x : A|\!)_I$ which can mean $(x : A)^I$ or $\{x : A\}_I$. We use the notation $\mathsf{app}^I(f, \omega)$ for function application, $\mathsf{app}_I(R, \omega)$ for relation application and $\mathsf{app}_I(f, \omega)$ can be either.

$$\frac{\Gamma +\!\!+ (\!|x : X|\!)_I[\xi] \vdash t : B}{\Gamma \vdash \lambda(\!|x : X|\!)_I[\xi].t : \Pi(\!|x : X|\!)_I[\xi].B} \qquad \frac{\Gamma \vdash f : \Pi(\!|x : X|\!)_I[\xi].B \quad \Gamma \vdash \omega : (\!|y : X|\!)_I[\xi]}{\Gamma \vdash \mathsf{app}_I(f, \omega) : B[\mathsf{id} +\!\!+ \omega]}$$

The computation rules include η:

$$\begin{aligned}
\mathsf{app}_I(\lambda(\!|x|\!)_I.t, \omega) &\equiv t[\mathsf{id} +\!\!+ \omega] \\
f &\equiv \lambda(\!|x : X|\!)_I[\xi].\mathsf{app}_I(f, \mathsf{pr}_{\{x:X\}I[\xi]}) \\
(\Pi(\!|x : X|\!)_I[\xi].B)[\sigma] &\equiv \Pi(\!|x : X|\!)_I[\xi[\sigma]].B[\sigma +\!\!+ \mathsf{pr}_{\{x:X\}I[\xi[\sigma]]}] \\
(\lambda(\!|x : X|\!)_I[\xi].t)[\sigma] &\equiv \lambda(\!|x : X|\!)_I[\xi[\sigma]].t[\sigma +\!\!+ \mathsf{pr}_{\{x:X\}I[\xi[\sigma]]}] \\
\mathsf{app}_I(f, \omega)[\sigma] &\equiv \mathsf{app}_I(f[\sigma], \omega[\sigma])
\end{aligned}$$

If the telescope substitution ξ is the projection pr, we omit it, eg. we write $\Pi(x : X)^I.B$ for $\Pi(x : X)^I[\mathsf{pr}].B$. Also, we write $\Pi(x : A)^I.B$ for $\Pi(x : X)^I[(X \mapsto A)^I].B$ and $\Pi\{x : A\}_I.B$ for $\Pi\{x : X\}_I[\{X \mapsto A\}_I].B$.

Note that in the case when I is empty, $\Pi(x : A)^I.B$ becomes the usual function space, we denote it as $\Pi(x : A).B$.

3.3 The operations $(-)^i$ and $\{-\}_i$ on contexts and substitutions

We define $(-)^i$ and $\{-\}_i$ on contexts, telescope contexts, substitutions, telescope substitutions. The definitions for contexts are the same as in the naive version.

Specification:

$$\frac{\Gamma \vdash}{(\Gamma)^i \vdash} \qquad \frac{\rho : \Delta \Rightarrow \Gamma}{(\rho)^i : \Delta^i \Rightarrow \Gamma^i} \qquad \frac{\Gamma \vdash \Omega}{(\Gamma)^i \vdash \Omega^i} \qquad \frac{\Gamma \vdash \omega : \Omega}{(\Gamma)^i \vdash \omega^i : \Omega^i}$$

$$\frac{\Gamma.x : A \vdash}{\{\Gamma.x : A\}_i \vdash} \qquad \frac{\rho : \Delta \Rightarrow \Gamma.x : A}{\{\rho\}_i : \Delta^i \Rightarrow \{\Gamma.x : A\}_i} \qquad \frac{\Gamma \vdash \Omega.x : A}{\Gamma^i \vdash \{\Omega.x : A\}_i} \qquad \frac{\Gamma \vdash \omega : \Omega.x : A}{\Gamma^i \vdash \{\omega\}_i : \{\Omega.x : A\}_i}$$

Implementation:

$$
\begin{array}{lll}
(\cdot)^i & \equiv \cdot & \\
(\Gamma.x : A)^i & \equiv \Gamma^i +\!\!+ (x : A)^i & \\
\epsilon^i & \equiv \epsilon & : \Gamma^i \Rightarrow \cdot^i \\
(\rho, x \mapsto t)^i & \equiv (\rho)^i +\!\!+ (x \mapsto t)^i & : \Delta^i \Rightarrow (\Gamma.x : A)^i \\
\Gamma^i \vdash (\cdot)^i & \equiv \cdot & \\
\Gamma^i \vdash (\Omega.x : A)^i & \equiv \Omega^i.x_{i0} : A[\mathsf{0}_{i\Gamma}].x_{i1} : A[\mathsf{1}_{i\Gamma}].x_{i2} : \mathsf{app}_i(A^i, (x)_i) & \\
\Gamma^i \vdash \epsilon^i & \equiv \epsilon & : \cdot^i \\
\Gamma^i \vdash (\omega, x \mapsto t)^i & \equiv (\omega^i, x_{i0} \mapsto t[\mathsf{0}_{i\Gamma}], x_{i1} \mapsto t[\mathsf{1}_{i\Gamma}], x_{i2} \mapsto t^i) & : (\Omega.x : A)^i \\
\{\Gamma.x : A\}_i & \equiv \Gamma^i +\!\!+ \{x : A\}_i & \\
\{\rho, x \mapsto t\}_i & \equiv \rho^i +\!\!+ \{x \mapsto t\}_i & : \Delta^i \Rightarrow \{\Omega.x : A\}_i \\
\Gamma^i \vdash \{\Omega.x : A\}_i & \equiv \Omega^i.x_{i0} : A[\mathsf{0}_{i\Gamma}].x_{i1} : A[\mathsf{1}_{i\Gamma}] & \\
\Gamma^i \vdash \{\omega, x \mapsto t\}_i & \equiv (\omega^i, x_{i0} \mapsto t[\mathsf{0}_{i\Gamma}], x_{i1} \mapsto t[\mathsf{1}_{i\Gamma}]) & : \{\Omega.x : A\}_i
\end{array}
$$

The operation $\{\text{–}\}_i$ does the same as $(\text{–})^i$ but omits the very last element (because of this, it does not work on empty contexts or substitutions).

The substitutions $\mathsf{0}$ and $\mathsf{1}$ project out the corresponding components ($b = \mathsf{0}, \mathsf{1}$) while losing the dimension i.

$$
\begin{array}{ll}
b_{i\cdot} & \equiv \epsilon : \cdot \Rightarrow \cdot \\
b_{i(\Gamma.x:A)} & \equiv (b_{i\Gamma}, x \mapsto x_{ib}) : (\Gamma.x : A)^i \Rightarrow \Gamma.x : A
\end{array}
$$

We also add how $(\text{–})^i$ operates on special substitutions:

$$(\rho\sigma)^i \equiv \rho^i\sigma^i \qquad (\mathsf{id}_\Gamma)^i \equiv \mathsf{id}_{\Gamma^i}$$

$(\text{–})^I$ is an iterated version of $(\text{–})^i$. $\{\text{–}\}_{Ii}$ is just $(\text{–})^I$ composed by $\{\text{–}\}_i$. For contexts we express this by the following rules, for the substitutions etc. we have analogous rules.

$$\Gamma^\emptyset \equiv \Gamma \qquad \Gamma^{Ij} \equiv (\Gamma^I)^j \qquad \{\Gamma\}_{Ij} \equiv \{\Gamma^I\}_j$$

An element of a (telescope) context produced by multiple applications of $(\text{–})^i$ can be viewed as an $|I|$-dimensional cube.

3.4 $(\text{–})^i$ on terms

The operation which maps a term to the witness of parametricity is the same as in the naive version, but with the additional information added for the function space. We need to handle functions and relations separately because performing $(\text{–})^i$ on a full cube gives a full cube, but on incomplete cubes it does not even give an incomplete cube. We need to fill in the omitted two elements in the latter case. Note that $(x : A)^I$ has $3^{|I|}$ elements and $\{x : A\}_I$ has $3^{|I|} - 1$ elements.

Specification:

$$\frac{\Gamma \vdash t : A}{\Gamma^i \vdash t^i : \mathsf{app}_i(A^i, \{x \mapsto t\}_i)} \tag{1}$$

Implementation:

$$
\begin{aligned}
(t[\rho])^i &\equiv t^i[\rho^i] \\
x^i &\equiv x_{i2} \\
\mathsf{U}^i &\equiv \lambda\{X\}_i.\Pi\{x : X\}_i.\mathsf{U} \\
(\Pi(x : X)^I[\xi].B)^i &\equiv \lambda\{f\}_i.\Pi(x : X)^{Ii}[\xi^i].\mathsf{app}_i(B^i, \{y \mapsto \mathsf{app}^I(f, (x)^I)\}_i) \\
(\lambda(x)^I.t)^i &\equiv \lambda(x)^{Ii}.t^i \\
(\mathsf{app}^I(f, \omega))^i &\equiv \mathsf{app}^{Ii}(t^i, \omega^i) \\
(\Pi\{x : X\}_I[\xi].\mathsf{U})^i &\equiv \lambda\{y\}_i.\Pi\{x : X\}_{Ii}[\xi^i +\!\!+ \{X_{I2} \mapsto y\}_i].\mathsf{U} \\
(\lambda\{x\}_I.B)^i &\equiv \lambda\{x\}_{Ii}.\mathsf{app}_i(B^i, \{y \mapsto x_{I2}\}_i) \\
(\mathsf{app}_I(R, \omega))^i &\equiv \lambda\{y\}_i.\mathsf{app}_{Ij}(R^i, \omega^i +\!\!+ \{x_{I2} \mapsto y\}_i)
\end{aligned}
$$

x_{I2} is a notation for the variable x where all the dimensions have index 2, eg. x_{ijk2} is x_{i2j2k2}.

3.5 A definitional quotient

Now we can add rules equating $(\text{–})^{ij}$ and $(\text{–})^{ji}$.

$$\frac{\Gamma \vdash}{(\Gamma^i)^j \equiv (\Gamma^j)^i} \qquad \frac{\Gamma \vdash \Omega}{\Gamma^{ij} \vdash (\Omega^i)^j \equiv (\Omega^j)^i} \qquad \frac{\Gamma \vdash \omega : \Omega}{\Gamma^{ij} \vdash (\omega^i)^j \equiv (\omega^j)^i : \Omega^{ij}}$$

$$\frac{\rho : \Delta \Rightarrow \Gamma}{(\rho^i)^j \equiv (\rho^j)^i : \Delta^{ij} \Rightarrow \Gamma^{ij}} \qquad \frac{\Gamma \vdash t : A}{\Gamma^{ij} \vdash (t^i)^j \equiv (t^j)^i : \mathsf{app}_{ij}(A^{ij}, \{x \mapsto t\}_{ij})}$$

$$\frac{\Gamma \vdash}{\{\Gamma^i\}_j \equiv \{\Gamma^j\}_i} \qquad \frac{\Gamma \vdash \Omega}{\Gamma^{ij} \vdash \{\Omega^i\}_j \equiv \{\Omega^j\}_i} \qquad \frac{\Gamma \vdash \omega : \Omega}{\Gamma^{ij} \vdash \{\omega^i\}_j \equiv \{\omega^j\}_i : \{\Omega\}_{ij}}$$

$$\frac{\rho : \Delta \Rightarrow \Gamma}{\{\rho^i\}_j \equiv \{\rho^j\}_i : \Delta^{ij} \Rightarrow \{\Gamma\}_{ij}}$$

So we can treat the dimensions of a context, substitution etc. as a set and write them without parentheses.

Now we have $(b_{i\Gamma})^j \equiv b_{i\Gamma^j}$ for $b = \mathsf{0}, \mathsf{1}$.

3.6 Internalisation of parametricity

We add a substitution R that adds a dimension to a context.

$$\frac{\Gamma \vdash}{\mathsf{R}_{i\Gamma} : \Gamma \Rightarrow \Gamma^i} \qquad \frac{\Gamma \vdash t : A}{\Gamma \vdash t^{ⓘ} : \mathsf{app}_i(A^i[\mathsf{R}_{i\Gamma}], (x_{i0} \mapsto t, x_{i1} \mapsto t))}$$

$$
\begin{aligned}
&\mathsf{R}_{i\cdot} \equiv \epsilon : \cdot \Rightarrow \cdot \\
&\mathsf{R}_{i(\Gamma.x:A)} \equiv (\mathsf{R}_{i\Gamma}, x_{i0} \mapsto x, x_{i1} \mapsto x, x_{i2} \mapsto x^{ⓘ}) : (\Gamma.x : A) \Rightarrow (\Gamma.x : A)^i \\
&t^{ⓘ} \equiv t^i[\mathsf{R}_{i\Gamma}] \qquad \frac{\rho : \Delta \to \Gamma}{\mathsf{R}_{i\Gamma}\rho \equiv \rho^i \mathsf{R}_{i\Delta}}
\end{aligned}
$$

Note that we can derive $a^{ⓘ}[\rho] \equiv (a[\rho])^{ⓘ}$ from the above rules. Also, the computation rule doesn't give us anything new if a is a variable:

$$\Gamma.x : A \vdash x^{ⓘ} \equiv x^i[\mathsf{R}_{i(\Gamma.x:A)}] \equiv x_{i2}[\mathsf{R}_{i(\Gamma.x:A)}] \equiv x^{ⓘ}.$$

We finally add the rule describing how $(-)^i$ works on the substitution R and how it interacts with the projections $b = \mathsf{0}, \mathsf{1}$:

$$(\mathsf{R}_{j\Gamma})^i \equiv \mathsf{R}_{j\Gamma^i} \qquad b_{i\Gamma}\mathsf{R}_{i\Gamma} \equiv \mathsf{id}_\Gamma$$

The first rule needs the definitional quotient to typecheck.

With this rule, we defined a type theory with internal parametricity. We describe an operational semantics in appendix A.

4 Conclusions and further work

We defined a naive version of cubical type theory in which univalence is admissible however we don't yet know how to compute in this theory. An advantage of this theory is that it is close to the usual syntax of Martin-Löf's type theory, hence it might be a step towards a translation from cubical type theory into intensional type theory. We conjecture that it has a presheaf model in a univalent metatheory (as some equations are only validated up to isomorphism – note that this problem does not appear for Cohen et al's cubical type theory [10], however it does appear in [6]).

We also defined a nominal version of the theory for parametricity together with a big-step normalisation function. We still have to establish that this normalisation function is sound and complete following [1]. However, as far as we know the computational behaviour of the operational semantics suggested in [10] hasn't been fully investigated yet, either.

In our approach the cubical structure arises naturally by iterating the $-^=$ and $-^i$ operators and it is not a consequence of extending the theory with a pretype for the interval as in [10, 4]. One interesting difference is that we directly define a family of equality types while the approach based on the interval type defines the family by fixing the endpoints of functions on the interval. On the other hand the notational overhead in the interval type based approach seems to be less than in our approach. Introducing an univalent universe is hard and is the most difficult aspect in [10], hence it is maybe unsurprising that we haven't achieved this yet in this more low level approach.

While in the moment the interval based approach presented in [10] seems to offer many advantages it may be a good idea to consider alternative approaches as the ones suggested here. We may also wonder whether we should need to explain an interval type when we want to introduce type theory?

Acknowledgements. We would like to thank the anonymous reviewers for their helpful comments and suggestions.

References

1 Thorsten Altenkirch and James Chapman. Big-step normalisation. *Journal of Functional Programming*, 19(3-4):311–333, 2009. doi:10.1017/S0956796809007278.

2 Thorsten Altenkirch and Ambrus Kaposi. Type theory in type theory using quotient inductive types. In *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL 2016, pages 18–29, New York, NY, USA, 2016. ACM. doi:10.1145/2837614.2837638.

3 Thorsten Altenkirch, Conor Mcbride, and Wouter Swierstra. Observational equality, now! In *PLPV '07: Proceedings of the 2007 workshop on Programming languages meets program verification*, pages 57–58. ACM, 2007. doi:10.1145/1292597.1292608.

4 Carlo Angiuli, Robert Harper, and Todd Wilson. Computational higher-dimensional type theory. In *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages*, POPL 2017, pages 680–693, New York, NY, USA, 2017. ACM. doi:10.1145/3009837.3009861.

5 Robert Atkey, Neil Ghani, and Patricia Johann. A relationally parametric model of dependent type theory. In *Proceedings of the 41st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL 2014)*, 2014. doi:10.1145/2535838.2535852.

6 Jean-Philippe Bernardy, Thierry Coquand, and Guilhem Moulin. A presheaf model of parametric type theory. *Electronic Notes in Theoretical Computer Science*, 319:67 – 82, 2015. doi:http://dx.doi.org/10.1016/j.entcs.2015.12.006.

7 Jean-Philippe Bernardy and Guilhem Moulin. A computational interpretation of parametricity. In *Proceedings of the 2012 27th Annual IEEE/ACM Symposium on Logic in Computer Science*, LICS '12, pages 135–144, Washington, DC, USA, 2012. IEEE Computer Society. doi:10.1109/LICS.2012.25.

8 Jean-Philippe Bernardy and Guilhem Moulin. Type-theory in color. In *Proceedings of the 18th ACM SIGPLAN International Conference on Functional Programming*, ICFP '13, pages 61–72, New York, NY, USA, 2013. ACM. doi:10.1145/2500365.2500577.

9 Marc Bezem, Thierry Coquand, and Simon Huber. A Model of Type Theory in Cubical Sets. In Ralph Matthes and Aleksy Schubert, editors, *19th International Conference on Types for Proofs and Programs (TYPES 2013)*, volume 26 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 107–128, Dagstuhl, Germany, 2014. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:http://dx.doi.org/10.4230/LIPIcs.TYPES.2013.107.

10 Cyril Cohen, Thierry Coquand, Simon Huber, and Anders Mörtberg. Cubical type theory: a constructive interpretation of the univalence axiom. In *To appear in: Proceedings of the 21st International Conference on Types for Proofs and Programs (TYPES 2015), 18-21 May 2015, Tallinn, Estonia*, December 2015.

11 Thierry Coquand. Variation on cubical sets. Note available at the website of the author, November 2014.

12 Claudio Hermida, Uday S. Reddy, and Edmund P. Robinson. Logical relations and parametricity - a reynolds programme for category theory and programming languages. *Electron. Notes Theor. Comput. Sci.*, 303:149–180, March 2014. doi:10.1016/j.entcs.2014.02.008.

13 Simon Huber. *A model of type theory in cubical sets*. Chalmers University of Technology, 2015. Licentiate thesis.

14 Guilhem Moulin. *Pure Type Systems with an Internalized Parametricity Theorem*. Chalmers University of Technology, 2013. Licentiate thesis.

15 Guilhem Moulin. *Internalizing Parametricity*. Chalmers University of Technology, 2016. PhD thesis.

16 Andrew Polonsky. Extensionality of lambda-*. In Hugo Herbelin, Pierre Letouzey, and Matthieu Sozeau, editors, *20th International Conference on Types for Proofs and Programs, TYPES 2014, May 12-15, 2014, Paris, France*, volume 39 of *LIPIcs*, pages 221–250. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2014. doi:10.4230/LIPIcs.TYPES.2014.221.

17 The Univalent Foundations Program. Homotopy type theory: Univalent foundations of mathematics. Technical report, Institute for Advanced Study, 2013.

18 J. C. Reynolds. Types, abstraction and parametric polymorphism. In R. E. A. Mason, editor, *Information Processing 83, Proceedings of the IFIP 9th World Computer Congress, Paris, September 19-23, 1983*, pages 513–523. Elsevier Science Publishers B. V. (North-Holland), Amsterdam, 1983. doi:10.1145/6041.6042.

A An operational semantics for the nominal calculus

A.1 Evaluation

We describe a call by name operational semantics for the theory defined in section 3. We conjecture that the semantics defined below is terminating, and when viewed as a syntactic model construction, sound and complete (we refer to [1] for the meaning of these properties); decidability of definitional would equality.

We define values (weak-head normal forms), and show how to evaluate terms into values. Next we define normal forms, and show that by recursively applying evaluation, we can normalise terms.

In what follows, $(\!|-|\!)_I$ can mean both $(-)^I$ and $\{-\}_I$. Note that in the latter case I is nonempty. We define the following inductive sets by listing their constructors.

t, A	$::= t[\rho] \mid x \mid \mathsf{U} \mid \Pi(x : X)^I[\omega].A \mid \Pi\{x : X\}_I[\omega].\mathsf{U}$	
	$\mid \lambda(\!|x|\!)_I.t \mid \mathsf{app}_I(t, \omega) \mid t^i \mid t^{\textcircled{i}}$	terms
ρ	$::= \mathsf{id} \mid \epsilon \mid (\rho, x \mapsto t) \mid \rho\rho' \mid \rho +\!\!+ \omega \mid \mathsf{0}_i \mid \mathsf{1}_i \mid \mathsf{R}_i \mid \rho^i \mid \{\rho\}_i$	substitutions
ω	$::= \epsilon \mid (\omega, x \mapsto t) \mid \omega[\rho] \mid \omega +\!\!+ \omega' \mid \mathsf{pr}_\Omega \mid \omega^i \mid \{\omega\}_i$	telescope substitutions
x, f, X	$::= \ldots$	variables
i	$::= \ldots$	dimension names
I	$::= \emptyset \mid Ii$	sets of dimension names
v	$::= \mathsf{U} \mid (\Pi(\!|x : X|\!)_I[\omega].A)[\nu] \mid (\lambda(\!|x|\!)_I.t)[\nu] \mid n$	values (whnfs)
ν	$::= \epsilon \mid (\nu, x \mapsto g) \mid (\nu, x \mapsto t[\nu])$	environments
n	$::= g \mid \mathsf{app}_I(n, \psi)$	neutral values
g	$::= x \mid g^{\textcircled{i}}$	generic values
ψ	$::= \epsilon \mid (\psi, x \mapsto g) \mid (\psi, x \mapsto t[\nu])$	telescope environments
Ω	$::= \cdot \mid \Omega.x : A \mid \Omega[\rho] \mid \Omega +\!\!+ \Omega' \mid \Omega^i \mid \{\Omega\}_i$	telescope contexts
Ξ	$::= \cdot \mid \Xi.x : A$	linear telescope contexts

We have the following judgment types for evaluation.

$t[\nu] \Downarrow v$	evaluate a closure to a value
$\rho\nu \Downarrow \nu'$	evaluate a substitution closure to an environment
$\omega[\nu] \Downarrow \psi$	evaluate a telescope substitution closure into a telescope environment
$\Omega \Downarrow \Xi$	evaluate a telescope context into a linear telescope context
$\nu(x) \rightsquigarrow v$	look up the value of a variable in an environment
$t^i \rightsquigarrow t'$	one step in calculating the meaning of a lifted term
$\rho^i \rightsquigarrow \rho'$	one step in calculating the meaning of a lifted substitution
$\omega^i \rightsquigarrow \omega'$	one step in calculating the meaning of a lifted telescope substitution
$\nu +\!\!+ \psi \rightsquigarrow \nu'$	composition of environments and telescope environments
$\psi +\!\!+ \psi' \rightsquigarrow \psi''$	composition of telescope environments
$\mathsf{pr}_\Xi[\nu] \rightsquigarrow \psi$	evaluating a telescope projection

We give a case for each judgment type for each constructor. Evaluation of terms is standard, the interesting cases are for t^i and $t^{\textcircled{i}}$: for the former we use the one-step evaluation relation $t^i \rightsquigarrow t'$, for the latter we use the computation rule for $-^{\textcircled{i}}$.

$\boxed{t[\nu] \Downarrow v}$

$$\frac{\rho\nu \Downarrow \nu' \quad t[\nu'] \Downarrow v}{t[\rho][\nu] \Downarrow v} \qquad \frac{\nu(x) \rightsquigarrow v}{x[\nu] \Downarrow v} \qquad \mathsf{U}[\nu] \Downarrow \mathsf{U} \qquad (\Pi\{x : X\}_I[\omega].A)[\nu] \Downarrow (\Pi\{x : X\}_I[\omega].A)[\nu]$$

$$(\lambda\{x\}_I.t)[\nu] \Downarrow (\lambda\{x\}_I.t)[\nu] \qquad \frac{t[\nu] \Downarrow (\lambda(x)_I.t')[\nu'] \quad \omega[\nu] \Downarrow \psi \quad \nu' +\!\!+ \psi \rightsquigarrow \nu'' \quad t'[\nu''] \Downarrow v}{\mathsf{app}_I(t,\omega)[\nu] \Downarrow v}$$

$$\frac{t[\nu] \Downarrow n \quad \omega[\nu] \Downarrow \psi}{\mathsf{app}_I(t,\omega)[\nu] \Downarrow \mathsf{app}_I(n,\psi)} \qquad \frac{t^i \rightsquigarrow t' \quad t'[\nu] \Downarrow v}{t^i[\nu] \Downarrow v} \qquad \frac{t^i[\mathsf{R}_i][\nu] \Downarrow v}{t^{\textcircled{i}}[\nu] \Downarrow v}$$

For substitutions, the interesting cases are again for our special constructions: the substitutions $\mathsf{0}_i, \mathsf{1}_i$ (denoted b_i) project out the corresponding components, while the substitution R_i duplicates the content of the environment. Its substitution rule ensures that we can move the $-^{\textcircled{i}}$ through the environment ν'.

$\boxed{\rho\nu \Downarrow \nu'}$

$$\mathsf{id}\nu \Downarrow \nu \qquad \epsilon\nu \Downarrow \epsilon \qquad \frac{\rho\nu \Downarrow \nu'}{(\rho, x \mapsto t)\nu \Downarrow (\nu', x \mapsto t[\nu])} \qquad \frac{\rho'\nu \Downarrow \nu' \quad \rho\nu' \Downarrow \nu''}{(\rho\rho')\nu \Downarrow \nu''}$$

$$\frac{\rho\nu \Downarrow \nu' \qquad \omega[\nu] \Downarrow \psi \quad \nu' +\!\!+ \psi \rightsquigarrow \nu''}{(\rho +\!\!+ \omega)\nu \Downarrow \nu''}$$

$$b_i\epsilon \Downarrow \epsilon \qquad \frac{b_i\nu \Downarrow \nu'}{b_i(\nu, x_{ib} \mapsto t) \Downarrow (\nu', x \mapsto t)} \qquad \frac{b_i\nu \Downarrow \nu'}{b_i(\nu, x_{ic} \mapsto t) \Downarrow \nu'}\ {\scriptstyle c\mathsf{not}=b}$$

$$\mathsf{R}_i\epsilon \Downarrow \epsilon \qquad \frac{\mathsf{R}_i\nu \Downarrow \nu'}{\mathsf{R}_i(\nu, x \mapsto g) \Downarrow (\nu', x_{i0} \mapsto g, x_{i1} \mapsto g, x_{i2} \mapsto g^{\textcircled{i}})}$$

$$\frac{\mathsf{R}_i\nu \Downarrow \nu''}{\mathsf{R}_i(\nu, x \mapsto t[\nu']) \Downarrow (\nu'', x_{i0} \mapsto t[\nu'], x_{i1} \mapsto t[\nu'], x_{i2} \mapsto t^{\textcircled{i}}[\nu'])}$$

$$\frac{\rho^i \rightsquigarrow \rho' \quad \rho'\nu \Downarrow \nu'}{(\rho)^i\nu \Downarrow \nu'} \qquad \frac{\rho^i \rightsquigarrow \rho' \quad \rho'\nu \Downarrow (\nu', x \mapsto t)}{\{\rho\}_i\nu \Downarrow \nu'}$$

$\boxed{\omega[\nu] \Downarrow \psi}$

$$\epsilon[\nu] \Downarrow \epsilon \qquad \frac{\omega[\nu] \Downarrow \psi}{(\omega, x \mapsto t)[\nu] \Downarrow (\psi, x \mapsto t[\nu])} \qquad \frac{\rho\nu \Downarrow \nu' \quad \omega[\nu'] \Downarrow \psi}{\omega[\rho][\nu] \Downarrow \psi}$$

$$\frac{\omega[\nu] \Downarrow \psi \quad \omega'[\nu] \Downarrow \psi \quad \psi +\!\!+ \psi' \rightsquigarrow \psi''}{(\omega +\!\!+ \omega')[\nu] \Downarrow \psi''} \qquad \frac{\Omega \Downarrow \Xi \quad \mathsf{pr}_\Xi \rightsquigarrow \psi}{\mathsf{pr}_\Omega[\nu] \Downarrow \psi} \qquad \frac{\omega^i \rightsquigarrow \omega' \quad \omega'[\nu] \Downarrow \psi}{\omega^i[\nu] \Downarrow \psi}$$

$$\frac{\omega^i \rightsquigarrow \omega' \quad \omega'[\nu] \Downarrow (\psi, x \mapsto t)}{\{\omega\}_i[\nu] \Downarrow \psi}$$

$\boxed{\Omega \Downarrow \Xi}$

$$\cdot \Downarrow \cdot \qquad \frac{\Omega \Downarrow \Xi}{\Omega.x : A \Downarrow \Xi.x : A} \qquad \frac{\Omega \Downarrow \cdot}{\Omega[\rho] \Downarrow \cdot} \qquad \frac{\Omega \Downarrow \Xi.x : A \quad \Xi[\rho] \Downarrow \Xi'}{\Omega[\rho] \Downarrow \Xi'.x : A[\rho]}$$

$$\frac{\Omega \Downarrow \cdot \qquad \Omega' \Downarrow \Xi}{\Omega +\!\!+ \Omega' \Downarrow \Xi} \qquad \frac{\Omega' \Downarrow (\Xi.x : A) \qquad \Omega +\!\!+ \Xi \Downarrow \Xi'}{\Omega +\!\!+ \Omega' \Downarrow \Xi'.x : A} \qquad \frac{\Omega \Downarrow \cdot}{\Omega^i \Downarrow \cdot}$$

$$\frac{\Omega \Downarrow \Xi.x : A \qquad \Xi^i \Downarrow \Xi'}{\Omega^i \Downarrow \Xi'.x_{i0} : A[0_i].x_{i1} : A[1_i].x_{i2} : \mathsf{app}_i(A^i, (x \mapsto x)_i)} \qquad \frac{\Omega \Downarrow \Xi.x : A \qquad \Xi^i \Downarrow \Xi'}{\{\Omega\}_i \Downarrow \Xi'.x_{i0} : A[0_i].x_{i1} : A[1_i]}$$

$\boxed{\nu(x) \rightsquigarrow v}$

$$\frac{\nu(x) = g}{\nu(x) \rightsquigarrow g} \qquad \frac{\nu(x) = t[\nu'] \quad t[\nu'] \Downarrow v}{\nu(x) \rightsquigarrow v}$$

The one-step part of the evaluation is given by following the rules in section 3.4.

$\boxed{t^i \rightsquigarrow t'}$

$$\begin{aligned}
(t[\rho])^i &\rightsquigarrow t^i[\rho^i] \\
x^i &\rightsquigarrow x_{i2} \\
\mathsf{U}^i &\rightsquigarrow \lambda\{X\}_i.\Pi(x : X)_i[\{X \mapsto X\}_i].\mathsf{U} \\
(\Pi(x : X)^I[\omega].A)^i &\rightsquigarrow \lambda\{f\}_i.\Pi(x : X)^{Ii}[\omega^i].\mathsf{app}_i(A^i, \{x' \mapsto \mathsf{app}^I(f, (x'' \mapsto x)^I)\}_i) \\
(\lambda(x)^I.t)^i &\rightsquigarrow \lambda(x)^{Ii}.t^i \\
\mathsf{app}^I(t, \omega)^i &\rightsquigarrow \mathsf{app}^{Ii}(t^i, \omega^i) \\
(\Pi\{x : X\}_I[\omega].\mathsf{U})^i &\rightsquigarrow \lambda(X')_i.\Pi\{x : X\}_{Ii}[(\omega^i, \{X_{I2} \mapsto X'\}_i)].\mathsf{U} \\
(\lambda\{x\}_I.A)^i &\rightsquigarrow \lambda\{x\}_{Ii}.\mathsf{app}_i(A^i, \{x' \mapsto x_{I2}\}_i) \\
\mathsf{app}_I(t, \omega)^i &\rightsquigarrow \lambda\{x'\}_i.\mathsf{app}_{Ii}(t^i, \{\omega^i, (x_{I2} \mapsto x')_i\})
\end{aligned}$$

$$\frac{t^j \rightsquigarrow t'}{(t^j)^i \rightsquigarrow t'^i} \qquad (t\textcircled{\scriptsize j})^i \rightsquigarrow (t^j[\mathsf{R}_j])^i$$

The definition of $\rho^i \rightsquigarrow \rho'$ for substitutions follows a similar pattern, it uses the functor laws for substitutions given in section 3.3. $\omega^i \rightsquigarrow \omega'$ describes one-step evaluation of telescope substitutions.

$\boxed{\rho^i \rightsquigarrow \rho'}$

$$\begin{aligned}
\mathsf{id}^i &\rightsquigarrow \mathsf{id} \\
\epsilon^i &\rightsquigarrow \epsilon \\
(\rho, x \mapsto t)^i &\rightsquigarrow (\rho^i, x_{i0} \mapsto t[0_i], x_{i1} \mapsto t[1_i], x_{i2} \mapsto t^i) \\
(\rho\rho')^i &\rightsquigarrow \rho^i \rho'^i \\
(\rho +\!\!+ \omega)^i &\rightsquigarrow (\rho^i +\!\!+ \omega^i) \\
(b_j)^i &\rightsquigarrow b_j \\
(\mathsf{R}_j)^i &\rightsquigarrow \mathsf{R}_j
\end{aligned}$$

$$\frac{\rho^j \rightsquigarrow \rho'}{(\!|\rho|\!)_j)^i \rightsquigarrow \rho'^i}$$

$\boxed{\omega^i \rightsquigarrow \omega'}$

$$\begin{aligned}
\epsilon^i &\rightsquigarrow \epsilon \\
(\omega, x \mapsto t)^i &\rightsquigarrow (\omega^i, x_{i0} \mapsto t[0_i], x_{i1} \mapsto t[1_i], x_{i2} \mapsto t^i) \\
(\omega[\rho])^i &\rightsquigarrow \omega^i[\rho^i] \\
(\omega +\!\!+ \omega')^i &\rightsquigarrow (\omega^i +\!\!+ \omega'^i)
\end{aligned}$$

$$\frac{\omega^j \rightsquigarrow \omega'}{(\{\!|\omega|\!\}_j)^i \rightsquigarrow \omega'^i}$$

$\boxed{\nu +\!\!+ \psi \rightsquigarrow \nu'}$

$$\nu +\!\!+ \epsilon \rightsquigarrow \nu \qquad \frac{\nu +\!\!+ \psi \rightsquigarrow \nu'}{\nu +\!\!+ (\psi, x \mapsto t) \rightsquigarrow (\nu', x \mapsto t)}$$

$\boxed{\psi +\!\!+ \psi' \rightsquigarrow \psi''}$

$$\psi +\!\!+ \epsilon \rightsquigarrow \psi \qquad \frac{\psi +\!\!+ \psi' \rightsquigarrow \psi''}{\psi +\!\!+ (\psi', x \mapsto t) \rightsquigarrow (\psi'', x \mapsto t)}$$

$\boxed{\mathsf{pr}_\Xi[\nu] \rightsquigarrow \psi}$

$$\mathsf{pr}_\cdot[\nu] \Downarrow \epsilon \qquad \frac{\mathsf{pr}_\Xi[\nu] \Downarrow \psi}{\mathsf{pr}_{\Xi.x:A}[(\nu, x \mapsto t)] \Downarrow (\psi, x \mapsto t)}$$

A.2 Normalisation

$$\begin{array}{lll} v_n & ::= \mathsf{U} \mid \Pi\{\!|x : X|\!\}_I[\psi_n].v_n \mid \lambda\{\!|x|\!\}_I.v_n \mid n_n & \text{normal forms} \\ n_n & ::= g \mid \mathsf{app}_I(n, \psi_n) & \text{neutral normal forms} \\ \psi_n & ::= \epsilon \mid (\psi_n, x \mapsto v_n) & \text{normal telescopes} \end{array}$$

$\boxed{v \Rrightarrow v_n}$

$$\mathsf{U} \Rrightarrow \mathsf{U}$$

$$\frac{\{\!|x : X|\!\}_I[\mathsf{id} +\!\!+ \omega] \Downarrow \Xi \quad \mathsf{pr}_\Xi \rightsquigarrow \psi \quad \nu +\!\!+ \psi \rightsquigarrow \nu' \quad A[\nu'] \Downarrow v \quad v \Rrightarrow v_n \quad \omega[\nu] \Downarrow \psi' \quad \psi' \Rrightarrow \psi'_n}{(\Pi\{\!|x : X|\!\}_I[\omega].A)[\nu] \Rrightarrow \Pi\{\!|x : X|\!\}_I[\psi'_n].v_n}$$

$$\frac{\{\!|x : X|\!\}_I[\mathsf{id} +\!\!+ \omega] \Downarrow \Xi \quad \mathsf{pr}_\Xi \rightsquigarrow \psi \quad \nu +\!\!+ \psi \rightsquigarrow \nu' \quad t[\nu'] \Downarrow v \quad v \Rrightarrow v_n}{(\lambda\{\!|x|\!\}_I.t)[\nu] \Rrightarrow \lambda\{\!|x|\!\}_I.v_n} \qquad \frac{n \Rrightarrow n_n}{n \Rrightarrow n_n}$$

$\boxed{n \Rrightarrow n_n}$

$$g \Rrightarrow g \qquad \frac{n \Rrightarrow n_n \quad \psi \Rrightarrow \psi_n}{\mathsf{app}_I(n, \psi) \Rrightarrow \mathsf{app}_I(n_n, \psi_n)}$$

$\boxed{\psi \Rrightarrow \psi_n}$

$$\epsilon \Rrightarrow \epsilon \qquad \frac{\psi \Rrightarrow \psi_n}{(\psi, x \mapsto g) \Rrightarrow (\psi_n, x \mapsto g)} \qquad \frac{\psi \Rrightarrow \psi_n \quad t[\nu] \Downarrow v \quad v \Rrightarrow v_n}{(\psi, x \mapsto t[\nu]) \Rrightarrow (\psi_n, x \mapsto v_n)}$$

To generate the identity substitution, we need to linearize contexts, it can be done in the same way as for telescope contexts before ($\Omega \Downarrow \Xi$). We denote this by $\Gamma \Downarrow \Delta$.

$\boxed{\mathsf{id}_\Delta \Downarrow \nu}$

$$\mathsf{id}_\cdot \Downarrow \epsilon \qquad \frac{\mathsf{id}_\Delta \Downarrow \nu}{\mathsf{id}_{\Delta.x:t} \Downarrow (\nu, x \mapsto x)}$$

Normalisation can be performed as follows:

$$\frac{\Gamma \vdash t : A \qquad \Gamma \Downarrow \Delta \qquad \mathsf{id}_\Delta \Downarrow \nu \qquad t[\nu] \Downarrow v \qquad v \Rrightarrow v_n}{t \text{ normalises to } v_n}$$