CUBICAL METHODS IN HOTT/UF

ANDERS MÖRTBERG

Abstract. These are notes for a course given at the 2019 Homotopy Type Theory Summer School at Carnegie Mellon University in Pittsburgh, USA. The course covers the basics of cubical type theory and its semantics in cubical sets. These new type theories provide computational justifications to homotopy type theory and univalent foundations, in particular, Voevodsky's univalence axiom is provable and hence has computational content. This can therefore be seen as a constructive foundation for homotopy type theory and univalent foundations, suitable for computer implementation.

These notes are very much work in progress. Suggestion for improvements and corrections are very welcome and can be sent to anders.mortberg@math.su.se.

Contents

Acknowledgments. I'm very grateful to Carlo Angiuli for proofreading earlier versions of these notes. Multiple explanations and diagrams are also borrowed or directly inspired by his Ph.D. thesis [Ang19].

1. Introduction and background

The original motivation behind the recent developments in HoTT/UF using cubical methods was to give computational/constructive meaning to these new theories. HoTT/UF, as formulated by Voevodsky or in the HoTT book, are axiomatic extensions of type theories of the kind formulated by the Swedish logician Per Martin-Löf. There are multiple different type theories of this kind, see e.g. [Mar75; Mar82; Mar84; Mar98], and we will here use the term "Martin-Löf type theory" (MLTT) for type theories specified using the four hypothetical judgments:

$$\Gamma \vdash A \qquad \Gamma \vdash A = B \qquad \Gamma \vdash a : A \qquad \Gamma \vdash a = b : A$$

We write $\Gamma \vdash \mathcal{J}$ for an arbitrary judgment. We also assume that the type theory supports Π- and Σ-types with judgmental η rules.[1] We also assume an infinite hierarchy

Date: August 6, 2019.

[1]We say that an equality holds "judgmentally" if it is a judgmental equality specified by the type theory, while an equality holds "strictly" if it holds on the nose in the model.

of universes U_n, but the universe level n will be left implicit and we will be a bit sloppy about the exact rules that these satisfy in order not to be bogged down by formal technicalities. Basic datatypes like empty, unit, boolean, and natural number types are also assumed and inhabit the lowest universe. We will write functions on these using pattern-matching equations, however all examples can easily be manually translated to eliminators. We follow the Agda convention (introduced by Nuprl) of writing dependent functions as $(x : A) \to B$ and dependent products as $(x : A) \times B$. The non-dependent variants of these are written $A \to B$ and $A \times B$. We write $p.1$ and $p.2$ for the first and second projections of a pair $p : (x : A) \times B$.

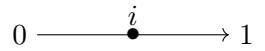
This is essentially the basic type theoretic setup in the various cubical systems that has been implemented, including cubical [cubical], cubicaltt [cubicaltt], yacctt [yacctt], RedPRL [Ang+18; Red16], redtt [Red18] and Cubical Agda [VMA19]. All of these systems build on different cubical type theories and have different standard cubical models [AFH18; Ang+17; BCH14; BCH18; CH19; CHM18; Coh+18], however the underlying ideas behind these are very similar and the goal of this course is to give sufficient background to understand and work with the various systems. The two cubical systems that are most actively developed at the moment are redtt and Cubical Agda. These course notes will mainly present the theory underlying Cubical Agda, but many of the ideas and constructions translate directly to the other systems.

The cubical type theories that underlie these systems are often justified using categorical semantics in various cubical set categories. These are typically formulated using one of the many frameworks for semantics of dependent type theory, like the categories with families (CwF's) of Dybjer [Dyb96], but we will be informal in these notes and not commit to a specific framework. However, a crucial result that we will rely on is the fact that any presheaf category forms a CwF with Π-, Σ-types, and supports basic datatypes like natural numbers, together with universes closed under these type formers. The standard references for these results are Hofmann [Hof97] and Hofmann and Streicher [HS].

A very convenient and elegant way for describing the semantics is to use the internal language of the presheaf topos of cubical sets, following Orton and Pitts [OP18]. As any presheaf category is locally cartesian closed this language supports Π- and Σ-types which means that the semantics can be presented using type theoretic notations. In order to avoid confusion we write $\Pi(x : A).B$ and $\Sigma(x : A).B$ for the semantic notions. This lets us mimic many of the syntactic constructions from cubical type theory also in the semantics, leading to very short and elegant proofs. However, it is sometimes useful or even necessary to express things *externally*, i.e. present the semantics using standard categorical language. We will make it clear when the semantics is described internally or externally.

2. Cubical type theories and their models

The key thing that makes a type theory "*cubical*" is the idea to add a primitive interval $\mathbb{I}$ and allow the judgmental structure to also incorporate contexts with interval variables. Intuitively one may think of $\mathbb{I}$ as a formal analogue of the real interval $[0, 1]$. This interval has two distinct endpoints $0 : \mathbb{I}$ and $1 : \mathbb{I}$. A variable $i : \mathbb{I}$ should be thought of as a point that is varying continuously between these endpoints:



By extending the judgmental structure of type theory with these variables we get "cubical" judgments of the form

$$i_1 : \mathbb{I}, ..., i_n : \mathbb{I} \vdash \mathcal{J}$$

Given a judgment $\mathcal{J}$ depending on an interval variable $i : \mathbb{I}$ we write $\mathcal{J}(r/i)$ for $\mathcal{J}$ with r substituted for i (so $A(0/i)$ is a type where 0 has been substituted for i). Types and terms in a context with n dimension variables correspond to n-dimensional cubes:

$\vdash A : \mathsf{U}$	$\bullet A$
$i : \mathbb{I} \vdash A : \mathsf{U}$	$A(0/i) \xrightarrow{A} A(1/i)$
$i : \mathbb{I}, j : \mathbb{I} \vdash A : \mathsf{U}$	$\begin{array}{ccc} A(0/i)(1/j) & \xrightarrow{A(1/j)} & A(1/i)(1/j) \\ {\scriptstyle A(0/i)}\uparrow & & \uparrow{\scriptstyle A(1/i)} \\ A(0/i)(0/j) & \xrightarrow[A(0/j)]{} & A(1/i)(0/j) \end{array}$
$\vdots$	$\vdots$

By the standard rules for substitutions we have $A(0/i)(0/j) = A(0/j)(0/i)$, etc. These equations correspond to the lines in the square matching up, namely the source of the left-most line is the same as the source of the bottom one.

Semantically all cubical set models are based on presheaves on some cube category $\mathcal{C}$, i.e. functor categories $\widehat{\mathcal{C}} := [\mathcal{C}^{\mathrm{op}}, \mathbf{Set}]$. The contexts are modelled by cubical sets and substitutions (or context maps) are modelled by natural transformations between these cubical sets. A very important cube category is the *cartesian* one.

Definition 1. The *cartesian* cube category $\square$ has as objects finite sets, and as morphisms $\mathrm{Hom}_{\square}(I, J)$ functions $J \to I + \{0, 1\}$ with the evident Kleisli composition.

Remark 2. This is equivalent to the free finite product category on an interval object [Awo18; Par14].

We write $I, J, K, ...$ for objects of $\square$, and say that a finite set $\{i_1, \ldots, i_n\}$ is an n-cube of "dimensions". Notable morphisms in $\square$ include:

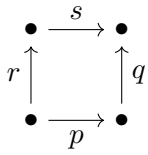
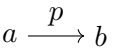
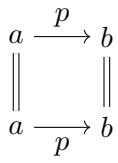
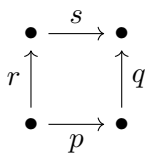
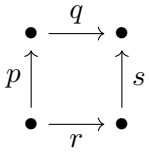
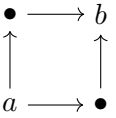
- Given $\varepsilon \in \{0, 1\}$, a dimension i and a finite set I there are *face maps*
$$d^i_\varepsilon \in \mathrm{Hom}_{\square}(I, I + \{i\})$$
$$d^i_\varepsilon(j) = \begin{cases} \varepsilon & \text{if } i = j \\ j & \text{otherwise} \end{cases}$$
- Given a dimension i and a finite set I there are *degeneracy maps*
$$s^i \in \mathrm{Hom}_{\square}(I + \{i\}, I)$$
$$s^i(j) = j$$
- Given dimensions i, j and a finite set I there are *symmetry maps*
$$t^{i,j} \in \mathrm{Hom}_{\square}(I + \{j, i\}, I + \{i, j\})$$
$$t^{i,j}(k) = \begin{cases} j & \text{if } k = i \\ i & \text{if } k = j \\ k & \text{otherwise} \end{cases}$$

- Given dimensions i, j and a finite set I there are *diagonal maps*

$$c^{i,j} \in \mathrm{Hom}_{\square}(I + \{i\}, I + \{i, j\})$$

$$c^{i,j}(k) = \begin{cases} i & \text{if } k = i \text{ or } k = j \\ k & \text{otherwise} \end{cases}$$

A cartesian cubical set is a functor $\Gamma : \widehat{\square}$. Geometrically we may think of such a Γ as a space and $\Gamma(\{i_1, \ldots, i_n\})$ as the set of continuous functions $[0,1]^n \to \Gamma$. The face maps $\Gamma(d^i_\varepsilon) : \Gamma(I + \{i\}) \to \Gamma(I)$ restrict $(n+1)$-cubes to n-cubes by setting the i coordinate to ε. The degeneracy maps $\Gamma(s^i) : \Gamma(I) \to \Gamma(I + \{i\})$ lets us regard n-cubes as $(n+1)$-cubes. The symmetry maps rotate cubes by permuting the axes. Finally, the diagonal maps extract the various diagonal n-cubes of $(n+1)$-cubes. We illustrate these maps informally below.

Faces		$\mapsto$	
Degeneracies		$\mapsto$	
Symmetries		$\mapsto$	
Diagonals		$\mapsto$	

These satisfy various evident cubical identities, e.g. degenerating and taking the corresponding face does nothing. We will now see that there is a close relationship between these identities and the rules satisfied by the cubical judgments.

Let $\mathbf{y} : \square \to \widehat{\square}$ be the Yoneda embedding. The interval of cubical type theory is modelled by $\mathbf{y}(\{i\})$ where i is an arbitrary dimension. We write $\mathbb{I}$ for this representable 1-cube as well. An important property of cubical sets is that the product of representable cubical sets is again representable.[2] Combined with the fact that $\mathbf{y}$ preserves products we get that the representable n-cube is an n-fold product of $\mathbb{I}$, i.e. $\mathbf{y}(\{i_1, \ldots, i_n\}) \simeq \mathbb{I}^n$. The Yoneda lemma hence gives a bijection between the n-cubes of a cubical set Γ and natural transformations $\mathbb{I}^n \to \Gamma$. This means that the structure of a cubical set is determined by maps out of products of cubical sets, justifying the geometric intuition above. Furthermore, a type $i_1 : \mathbb{I}, \ldots, i_n : \mathbb{I} \vdash A : \mathsf{U}$ corresponds to a morphism $A : \mathbb{I}^n \to \mathsf{U}$, justifying the cubical judgments from above. This is one reason why cubical sets are so well-suited as a basis for higher dimensional type theory.

[2]This is not true for simplicial sets where the product of representables has to be subdivided in order to form a simplicial set again.

Given an n-cube $I = \{i_1, \ldots, i_n\}$ and context $\Gamma = i_1 : \mathbb{I}, \ldots, i_n : \mathbb{I}$ we can now clarify the relationship between the proof theory of cubical type theory and its semantics.

Syntax/proof theory	Semantics
$\dfrac{\Gamma, i : \mathbb{I} \vdash \mathcal{J}}{\Gamma \vdash \mathcal{J}(\varepsilon/i)}$ face	$\mathbf{y}(I) \xrightarrow{\mathbf{y}(d^i_\varepsilon)} \mathbf{y}(I + \{i\})$
$\dfrac{\Gamma \vdash \mathcal{J}}{\Gamma, i : \mathbb{I} \vdash \mathcal{J}}$ weakening	$\mathbf{y}(I + \{i\}) \xrightarrow{\mathbf{y}(s^i)} \mathbf{y}(I)$
$\dfrac{\Gamma, i : \mathbb{I}, j : \mathbb{I} \vdash \mathcal{J}}{\Gamma, j : \mathbb{I}, i : \mathbb{I} \vdash \mathcal{J}}$ exchange	$\mathbf{y}(I + \{j, i\}) \xrightarrow{\mathbf{y}(t^{i,j})} \mathbf{y}(I + \{i, j\})$
$\dfrac{\Gamma, i : \mathbb{I}, j : \mathbb{I} \vdash \mathcal{J}}{\Gamma, i : \mathbb{I} \vdash \mathcal{J}(i/j)}$ contraction	$\mathbf{y}(I + \{i, j\}) \xrightarrow{\mathbf{y}(c^{i,j})} \mathbf{y}(I + \{i\})$

This shows that there is a close correspondence between the maps in the cubical set categories and the structure of the cubical judgments in cubical type theory.

Remark 3. If we omit diagonals we obtain the *substructural* cubical sets that underlie the first cubical set model of Bezem et al. [BCH14]. However, because of the substructural nature of this cube category nobody has so far developed a cubical type theory based on this model.

2.1. **Path types.** In order to be able to talk internally about paths in cubical type theory we now extend the system with path types. These types are a type theoretic rendering of the idea that a path is just a function out of the interval.

$$\frac{\Gamma, i : \mathbb{I} \vdash A \qquad \Gamma \vdash a : A(0/i) \qquad \Gamma \vdash b : A(1/i)}{\Gamma \vdash \mathsf{Path}^i\ A\ a\ b}$$

$$\frac{\Gamma, i : \mathbb{I} \vdash A \qquad \Gamma, i : \mathbb{I} \vdash a : A}{\Gamma \vdash \lambda(i : \mathbb{I}).\ a : \mathsf{Path}^i\ A\ a(0/i)\ a(1/i)} \qquad \frac{\Gamma \vdash p : \mathsf{Path}^i\ A\ a\ b \qquad \Gamma \vdash r : \mathbb{I}}{\Gamma \vdash p\ r : A(r/i)}$$

$$\frac{\Gamma, i : \mathbb{I} \vdash A \qquad \Gamma, i : \mathbb{I} \vdash a : A \qquad \Gamma \vdash r : \mathbb{I}}{\Gamma \vdash (\lambda(i : \mathbb{I}).\ a)\ r = a(r/i) : A(r/i)}\beta \qquad \frac{\Gamma \vdash p : \mathsf{Path}^i\ A\ a\ b}{\Gamma \vdash (\lambda(j : \mathbb{I}).\ p\ j) = p : \mathsf{Path}^i\ A\ a\ b}\eta$$

$$\frac{\Gamma \vdash p : \mathsf{Path}^i\ A\ a\ b}{\Gamma \vdash p\ 0 = a : A(0/i)} \qquad \frac{\Gamma \vdash p : \mathsf{Path}^i\ A\ a\ b}{\Gamma \vdash p\ 1 = b : A(1/i)}$$

Note that these rules are very similar to those of Π-types, except that special care has to be taken when applying a path to one of the endpoints of the interval. If A doesn't depend on i we write simply $\mathsf{Path}\ A\ u_0\ u_1$. This non-dependent version is supposed to correspond to regular identity types of HoTT/UF while Path^i is a cubical version of the "path-over" types of HoTT, i.e. paths living over a line of types. Having built-in path-over types is very useful, especially when working with higher inductive types, as we can directly represent heterogeneous equalities (i.e. equalities where the endpoints are in different types). Furthermore, representing equalities using path types allows

direct definitions of many standard operations on identity types that are usually proved by identity elimination.

For example, given $A : \mathsf{U}$ we may define a proof of reflexivity as a constant path

$$\mathsf{refl} : (a : A) \to \mathsf{Path}\ A\ a\ a$$

$$\mathsf{refl}\ a \triangleq \lambda(i : \mathbb{I}).\ a$$

Given $A, B : \mathsf{U}$ we can also prove that the images of two path-equal elements are path-equal

$$\mathsf{ap} : (a\ b : A)\ (f : A \to B)\ (p : \mathsf{Path}\ A\ a\ B) \to \mathsf{Path}\ B\ (f\ a)\ (f\ b)$$

$$\mathsf{ap}\ a\ b\ f\ p \triangleq \lambda(i : \mathbb{I}).\ f\ (p\ i)$$

This operation satisfies some judgmental equalities that do not hold judgmentally when ap is defined using identity elimination, for example:

$$\mathsf{ap}\ \mathrm{id}\ p = p$$
$$\mathsf{ap}\ (g \circ f)\ p = \mathsf{ap}\ f\ (\mathsf{ap}\ g\ p)$$

We can also define new operations that doesn't always hold for identity types, for instance, function extensionality for path types can be proved as:

$$\mathsf{funext} : (f\ g : (x : A) \to B)\ (p : (x : A) \to \mathsf{Path}\ B\ (f\ x)\ (g\ x)) \to \mathsf{Path}\ ((x : A) \to B)\ f\ g$$

$$\mathsf{funext}\ f\ g\ p \triangleq \lambda(i : \mathbb{I}).\ \lambda(x : A).\ p\ x\ i$$

To see that this is correct we check that the term has the correct faces, for instance:

$$(\lambda(i : \mathbb{I}).\ \lambda(x : A).\ p\ x\ i)\ 0 = \lambda(x : A).\ p\ x\ 0 = \lambda(x : A).\ f\ x = f$$

Note that the last equality holds by the η rule for Π-types.

Equality in Σ-types is notoriously complicated to work with in traditional type theory, but with dependent path types things are easier. Given $A : \mathsf{U}$ and a family $B : A \to \mathsf{U}$ we define

$$\Sigma_{\mathsf{eq}} : (s\ t : (x : A) \times B)\ (p : \mathsf{Path}\ A\ s.1\ t.1)\ (q : \mathsf{Path}^i\ (B\ (p\ i))\ s.2\ t.2) \to$$
$$\mathsf{Path}\ ((x : A) \times B)\ s\ t$$

$$\Sigma_{\mathsf{eq}}\ s\ t\ p\ q \triangleq \lambda(j : \mathbb{I}).\ (p\ j, q\ j)$$

Working with path-overs like this is very convenient compared to HoTT/UF as no transports are necessary.

Semantically we can easily justify path types using the internal language of $\widehat{\square}$. Given a family of types $A : \mathbb{I} \to \mathsf{U}$ we define:

$$\mathsf{Path}(A) \triangleq \Pi(i : \mathbb{I}).A(i)$$

We then define the type of paths between $a : A(0)$ and $b : A(1)$ as

$$\mathsf{Path}_A(a, b) \triangleq \Sigma(p : \mathsf{Path}(A)).(p\ 0 = a \wedge p\ 1 = b)$$

It is easy to verify that this satisfies the rules of path types as they are constructed using semantic Π and Σ's. Furthermore, the exact same operations as above are easily definable semantically.

2.2. **Connections and reversals.** It is very useful to assume more structure in the underlying cube category, both when constructing models and for making proofs simpler in the cubical type theory based on the model. This is the strategy in both cubicaltt and Cubical Agda which are based on the cube category of the CCHM cubical set model [Coh+18]. Before defining this category we have to introduce the notion of a De Morgan algebra:

Definition 4. A bounded distributive lattice $(A, 0, 1, \wedge, \vee)$ is a De Morgan algebra if it has an involution $\neg : A \to A$ satisfying the De Morgan identities:

$$\neg(r \vee s) = \neg r \wedge \neg s \qquad\qquad \neg(r \wedge s) = \neg r \vee \neg s$$

We write DM for the monad on the category of sets associating to each set A the free De Morgan algebra on A.

Definition 5. The *De Morgan* cube category $\square_{\mathsf{DM}}$ has as objects finite sets, and as morphisms $\mathrm{Hom}_{\square_{\mathsf{DM}}}(I, J)$ functions $J \to \mathsf{DM}(I)$. Identity and composition inherited from the Kleisli category of DM.

This category has all of the morphisms of $\square$, but there are very many more ones (see exercise 7). Notable new morphisms in this category are:

- Given a dimension i and a finite set I there are *connection maps*

$$c^i_\wedge \in \mathrm{Hom}_{\square_{\mathsf{DM}}}(I + \{i\}, I) \qquad\qquad c^i_\vee : \mathrm{Hom}_{\square_{\mathsf{DM}}}(I + \{i\}, I)$$
$$c^i_\wedge(j) = i \wedge j \qquad\qquad c^i_\vee(j) = i \vee j$$

- Given a dimension i and a finite set I there are *reversal* maps:

$$r^i \in \mathrm{Hom}_{\square_{\mathsf{DM}}}(I + \{i\}, I)$$
$$r^i(j) = \begin{cases} \neg i & \text{if } i = j \\ j & \text{otherwise} \end{cases}$$

A CCHM cubical set is a functor $\Gamma : \widehat{\square_{\mathsf{DM}}}$. The connections can be thought of as new kinds of degeneracies while the reversals lets us invert lines.

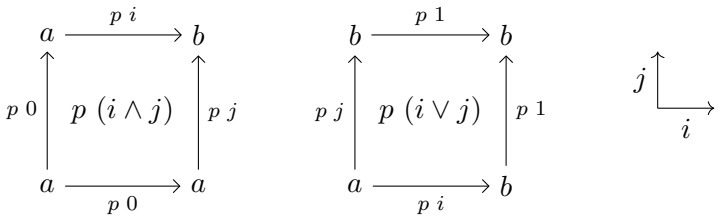
Connections $\quad a \xrightarrow{p} b \quad \mapsto \quad \begin{array}{ccc} b & = & b \\ {\scriptstyle p}\uparrow & \vee & \| \\ a & \xrightarrow[p]{} & b \end{array} \qquad \begin{array}{ccc} a & \xrightarrow{p} & b \\ \| & \wedge & \uparrow{\scriptstyle p} \\ a & = & a \end{array}$

Reversals $\quad a \longrightarrow b \quad \mapsto \quad b \longrightarrow a$

We may see these new morphisms as operations in $\widehat{\square_{\mathsf{DM}}}$ of type $\wedge, \vee : \mathbb{I} \to \mathbb{I} \to \mathbb{I}$ and $\neg : \mathbb{I} \to \mathbb{I}$ satisfying the axioms of a De Morgan algebra. The topological intuition behind these operations is that $r \wedge s$ corresponds to $\min(r, s)$, $r \vee s$ to $\max(r, s)$ and $\neg r$ to $1 - r$ for $r, s \in [0, 1]$.

Remark 6. One might wonder why De Morgan algebras and not Boolean algebras? The reason is that Boolean algebras do not describe the theory of the real interval. Indeed, the equations $r \wedge \neg r = 0$ and $r \vee \neg r = 1$ are not generally true for $r \in [0, 1]$ except for at the endpoints.

Type theoretically we may now substitute interval variables for formulas built using connections and reversals in order to construct more complex cubes out of simple ones. For example, given $p : \mathsf{Path}\ A\ a\ b$ and $i, j : \mathbb{I}$ we can construct the connection squares $p\ (i \wedge j)$ and $p\ (i \vee j)$ as



where, for instance, the right-hand side of the left square is computed as

$$p\ (i \wedge j)(1/i) = p\ (1 \wedge j) = p\ j$$

The ability to directly construct squares with boundaries given by the formulas that can be formed in a De Morgan algebra like this is very convenient. An example of this is the proof that singletons are contractible, that is: any element in $(x : A) \times (\mathsf{Path}\ A\ a\ x)$ is path-equal to (a, refl_a).

$$\mathsf{contrSingl} : (a\ b : A)\ (p : \mathsf{Path}\ A\ a\ b) \to \mathsf{Path}\ ((x : A) \times (\mathsf{Path}\ A\ a\ x))\ (a, \mathsf{refl}_a)\ (b, p)$$
$$\mathsf{contrSingl}\ a\ b\ p \triangleq \lambda(i : \mathbb{I}).\ (p\ i, \lambda(j : \mathbb{I}).\ p\ (i \wedge j))$$

Obviously the first component is a path between a and b. The second component is path from $\lambda(j : \mathbb{I}).\ p\ 0$ to $\lambda(j : \mathbb{I}).\ p\ j$, i.e. from refl_a to p by the computation and η rules for path types.

Given $a, b : A$ we may also define the symmetry of a path as follows:

$$\mathsf{sym} : \mathsf{Path}\ A\ a\ b \to \mathsf{Path}\ A\ b\ a$$
$$\mathsf{sym}\ p \triangleq \lambda(i : \mathbb{I}).\ p\ (\neg i)$$

This satisfies $\mathsf{sym}\ (\mathsf{sym}\ p) = p$ judgmentally. This is very useful when formalizing mathematics, for example we may directly define the opposite of a category so that $\mathcal{C}^{\mathrm{op}^{\mathrm{op}}} = \mathcal{C}$ judgmentally without relying on any tricks.

With this new additional structure the path types almost behave like the identity types of HoTT/UF, however we need to add additional structure that lets us prove the path-elimination principle:

$$\mathsf{J} : (A : \mathsf{U})\ (a : A)\ (C : (x : A) \to \mathsf{Path}\ A\ a\ x \to \mathsf{U})$$
$$(d : C\ a\ \mathsf{refl}_a)\ (x : A)\ (p : \mathsf{Path}\ A\ a\ x) \to C\ x\ p$$

This may in fact be decomposed as contractibility of singletons (which we have already proved) and subst:

$$\mathsf{subst} : (A : \mathsf{U})\ (P : A \to \mathsf{U})\ (a\ b : A)\ (p : \mathsf{Path}\ A\ a\ b)\ (e : P\ a) \to P\ b$$

In fact, in the next section we will reduce subst to an even simpler operation that we call $\mathsf{transport}$. We will then equip the cubical sets with structure corresponding to this operation and explain how it computes in the type theory.

2.3. **Exercises.**

(1) Prove the binary version of ap

$$\mathsf{ap}_2 : (a\ a' : A)\ (b\ b' : B)\ (f : A \to B \to C)$$
$$(p : \mathsf{Path}\ A\ a\ a')\ (q : \mathsf{Path}\ B\ b\ b') \to \mathsf{Path}\ C\ (f\ a\ b)\ (f\ a'\ b')$$

(2) Prove the binary (non-dependent) version of function extensionality

$$\mathsf{funext}_2 : (f\ g : A \to B \to C)\ (p : (x : A)\ (y : B) \to \mathsf{Path}\ B\ (f\ x\ y)\ (g\ x\ y)) \to$$
$$\mathsf{Path}\ (A \to B \to C)\ f\ g$$

(3) Define negation on booleans $\mathsf{not} : \mathsf{bool} \to \mathsf{bool}$ and prove that

$$\mathsf{notK} : \mathsf{Path}\ (\mathsf{bool} \to \mathsf{bool})\ (\mathsf{not} \circ \mathsf{not})\ \mathrm{id}$$

(4) We can define a predicate $\mathsf{isProp} : \mathsf{U} \to \mathsf{U}$ that expresses that a type is an h-proposition as

$$\mathsf{isProp}\ A \triangleq (x\ y : A) \to \mathsf{Path}\ A\ x\ y$$

Prove that a family of propositions is a proposition:

$$\mathsf{propPi} : (A : \mathsf{U})\ (B : A \to \mathsf{U})\ (h : (x : A) \to \mathsf{isProp}\ (B\ x)) \to$$
$$\mathsf{isProp}\ ((x : A) \to B\ x)$$

(5) Given $p : \mathsf{Path}\ A\ a\ b$ and $i, j : \mathbb{I}$, draw the squares corresponding to
   (a) $p\ (\neg i \wedge j)$
   (b) $p\ (i \wedge \neg j)$
   (c) $p\ (\neg i \wedge \neg j)$
   (d) $p\ (\neg i \vee j)$
   (e) $p\ (i \vee \neg j)$
   (f) $p\ (\neg i \vee \neg j)$

(6) Given $p : \mathsf{Path}\ A\ a\ b$ and $i, j, k : \mathbb{I}$, draw the cubes corresponding to
   (a) $p\ (i \wedge j \wedge k)$
   (b) $p\ (i \wedge \neg j \vee k)$
   (c) $p\ (\neg i \vee \neg j \vee \neg k)$

(7) How many elements does $\mathrm{Hom}_{\square}(\{i_1, \ldots, i_n\}, \{i\})$ have? What about $\mathrm{Hom}_{\square_{\mathsf{DM}}}(\{i_1, \ldots, i_n\}, \{i\})$?

(8) Prove the following variation of $\mathsf{contrSingl}$

$$\mathsf{contrSingl}' : (a\ b : A)\ (p : \mathsf{Path}\ A\ a\ b) \to$$
$$\mathsf{Path}\ ((x : A) \times (\mathsf{Path}\ A\ x\ b))\ (b, \mathsf{refl}_b)\ (a, p)$$

(9) Prove J using subst and $\mathsf{contrSingl}$. What does $\mathsf{J}\ A\ a\ C\ d\ a\ \mathsf{refl}_a$ evaluate to?

(10) The circle $\mathbb{S}^1$ has constructors base and $\mathsf{loop} : \mathbb{I} \to \mathbb{S}^1$ such that $\mathsf{loop}\ 0 = \mathsf{loop}\ 1 = \mathsf{base}$. Why is this type non-trivial even though $\lambda(i\ j : \mathbb{I}).\ \mathsf{loop}\ (i \wedge j)$ proves that $\mathsf{refl}_{\mathsf{base}}$ is path-equal to loop? (hint: draw a picture)

(11) Construct a homotopy on the circle that shows that the path going halfway around and back is contractible:

$$\mathsf{hmtpy} : \mathsf{Path}\ (\mathsf{Path}\ \mathbb{S}^1\ \mathsf{base}\ \mathsf{base})\ \mathsf{refl}_{\mathsf{base}}\ (\lambda(i : \mathbb{I}).\ \mathsf{loop}\ (i \wedge \neg i))$$

(12) Prove that it is inconsistent to assume decidable equality of $\mathbb{I}$ internally. (hint: construct a path from the unit type to the empty type)

References

[AFH18] Carlo Angiuli, Kuen-Bang Hou (Favonia), and Robert Harper. "Cartesian Cubical Computational Type Theory: Constructive Reasoning with Paths and Equalities". In: *27th EACSL Annual Conference on Computer Science Logic, CSL 2018, September 4-7, 2018, Birmingham, UK*. 2018, 6:1–6:17.

[Ang+17] Carlo Angiuli, Guillaume Brunerie, Thierry Coquand, Kuen-Bang Hou (Favonia), Robert Harper, and Daniel R. Licata. "Syntax and Models of Cartesian Cubical Type Theory". Draft available at https://github.com/dlicata335/cart-cube/blob/master/cart-cube.pdf. 2017.

[Ang+18] Carlo Angiuli, Evan Cavallo, Kuen-Bang Hou (Favonia), Robert Harper, and Jonathan Sterling. "The RedPRL Proof Assistant (Invited Paper)". In: *13th International Workshop on Logical Frameworks and Meta-Languages: Theory and Practice (LFMTP 2018)*. Ed. by Frédéric Blanqui and Giselle Reis. Vol. 274. Electronic Proceedings in Theoretical Computer Science. Oxford, UK, 2018.

[Ang19] Carlo Angiuli. "Computational Semantics of Cartesian Cubical Type Theory". PhD thesis. Carnegie Mellon University, 2019.

[Awo18] Steve Awodey. "A cubical model of homotopy type theory". In: *Annals of Pure and Applied Logic* 169.12 (2018). Logic Colloquium 2015, pp. 1270–1294.

[BCH14] Marc Bezem, Thierry Coquand, and Simon Huber. "A Model of Type Theory in Cubical Sets". In: *19th International Conference on Types for Proofs and Programs (TYPES 2013)*. Vol. 26. Leibniz International Proceedings in Informatics (LIPIcs). 2014, pp. 107–128.

[BCH18] Marc Bezem, Thierry Coquand, and Simon Huber. "The Univalence Axiom in Cubical Sets". In: *Journal of Automated Reasoning* (June 2018).

[CH19] Evan Cavallo and Robert Harper. "Higher Inductive Types in Cubical Computational Type Theory". In: *Proc. ACM Program. Lang.* 3.POPL (Jan. 2019), 1:1–1:27.

[CHM18] Thierry Coquand, Simon Huber, and Anders Mörtberg. "On Higher Inductive Types in Cubical Type Theory". In: *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*. LICS '18. ACM, 2018, pp. 255–264.

[Coh+18] Cyril Cohen, Thierry Coquand, Simon Huber, and Anders Mörtberg. "Cubical Type Theory: A Constructive Interpretation of the Univalence Axiom". In: *Types for Proofs and Programs (TYPES 2015)*. Vol. 69. LIPIcs. 2018, 5:1–5:34.

[cubical] The cubical Development Team. *cubical*. Available at https://github.com/simhu/cubical/. 2013–.

[cubicaltt] The cubicaltt Development Team. *cubicaltt*. Available at https://github.com/mortberg/cubicaltt/. 2015–.

[Dyb96] Peter Dybjer. "Internal Type Theory". In: *Lecture Notes in Computer Science*. Springer Verlag, Berlin, Heidelberg, New York, 1996, pp. 120–134.

[Hof97] Martin Hofmann. "Syntax and semantics of dependent types". In: *Semantics and logics of computation*. Ed. by A.M. Pitts and P. Dybjer. Vol. 14. Publ. Newton Inst. Papers from the Summer School held at the University of Cambridge, Cambridge, September 1995. Cambridge: Cambridge University Press, 1997, pp. 79–130.

[HS] Martin Hofmann and Thomas Streicher. “Lifting Grothendieck Universes”. Unpublished Note.

[Mar75] Per Martin-Löf. “An Intiutionistic Theory of Types: Predicative Part”. In: *Logic Colloquium '73*. Ed. by H. E. Rose and J. Shepherdson. North–Holland, Amsterdam, 1975, pp. 73–118.

[Mar82] Per Martin-Löf. “Constructive Mathematics and Computer Programming”. In: *Logic, Methodology and Philosophy of Science, VI*. 1982, pp. 153–175.

[Mar84] Per Martin-Löf. *Intuitionistic Type Theory*. Bibliopolis, 1984.

[Mar98] Per Martin-Löf. “An intuitionistic theory of types”. In: *Twenty-five years of constructive type theory (Venice, 1995)*. Vol. 36. Oxford Logic Guides. New York: Oxford Univ. Press, 1998, pp. 127–172.

[OP18] Ian Orton and Andrew M. Pitts. “Axioms for Modelling Cubical Type Theory in a Topos”. In: *Logical Methods in Computer Science* Volume 14, Issue 4 (Dec. 2018).

[Par14] Jason Parker. “Duality between Cubes and Bipointed Sets”. MA thesis. Carnegie Mellon University, 2014.

[Red16] The RedPRL Development Team. *The RedPRL Proof Assistant*. Available at http://www.redprl.org/. 2016–2018.

[Red18] The RedPRL Development Team. *redtt*. Available at https://github.com/RedPRL/redtt. 2018.

[VMA19] Andrea Vezzosi, Anders Mörtberg, and Andreas Abel. “Cubical Agda: A Dependently Typed Programming Language with Univalence and Higher Inductive Types”. Preprint available at http://www.cs.cmu.edu/~amoertbe/papers/cubicalagda.pdf. 2019.

[yacctt] The yacctt Development Team. *yacctt*. Available at https://github.com/mortberg/yacctt/. 2018.