Computational Higher Type Theory (CHTT)

Robert Harper

Lecture Notes of Week 12 by Ryan Kavanagh, Jon Sterling and Jonathan Laurent

1 Last Time

Last time, we introduced "higher" inductive definitions, simultaneously defining the types A, $\mathsf{Id}_A(-,-)$, $\mathsf{Id}_{\mathsf{Id}_A(-,-)}(-,-)$, etc. by their generators.

We also introduced the abstract interval $\mathbb{I}$ with endpoints $0, 1 : \mathbb{I}$ and path constructor $\mathsf{seg} : \mathsf{Id}_{\mathbb{I}}(0,1)$. To construct a map out of the interval into a type A is to "draw a line" in the type A: you specify two points, and a path in A that connects them. In fact, a map $\mathbb{I}^n \to A$ can be thought of as a defining an n-cube in A.

2 Elimination for the interval

The elimination rule (induction principle) for the interval is written as follows, writing $M_0 =^{i.C}_{\mathsf{seg}} M_1$ for the homogeneous identification $\mathsf{tr}[i.C](\mathsf{seg}; M_0) =_{[1/i]C} M_1$:

$$\frac{\Gamma, u : \mathbb{I} \vdash C\ \mathsf{type} \qquad \Gamma \vdash M : \mathbb{I} \qquad \Gamma \vdash M_0 : [0/i]C \qquad \Gamma \vdash M_1 : [1/i]C \qquad \Gamma \vdash M_{\mathsf{seg}} : M_0 =^{i.C}_{\mathsf{seg}} M_1}{\Gamma \vdash \mathbb{I}\text{-}\mathsf{ind}[i.C](M_0; M_1; M_{\mathsf{seg}})(M) : [M/i]C}$$

For well-typed instances of the elimination form, we impose the following beta rules:

$$\mathbb{I}\text{-}\mathsf{ind}[i.C](M_0; M_1; M_{\mathsf{seg}})(0) \equiv M_0 : [0/i]C \qquad \mathbb{I}\text{-}\mathsf{ind}[i.C](M_0; M_1; M_{\mathsf{seg}})(1) \equiv M_1 : [1/i]C$$

We also have a "propositional beta rule" for the seg constructor; in this particular formal system, we are not free to impose a definitional equality, which is a bit strange:

$$\mathsf{apd}(\mathbb{I}\text{-}\mathsf{ind}[i.C](M_0; M_1; M_{\mathsf{seg}})(-))(\mathsf{seg}) =_{(M_0 =^{i.C}_{\mathsf{seg}} M_1)} M_{\mathsf{seg}}$$

3 The Circle $\mathbb{C}$ aka $\mathbb{S}^1$

The circle $\mathbb{C}$ is given by the following generators, induction principle, and equations:

$$\Gamma \vdash \mathsf{base} : \mathbb{C} \qquad\qquad \Gamma \vdash \mathsf{loop} : \mathsf{base} =_{\mathbb{C}} \mathsf{base}$$

$$\frac{\Gamma, c : \mathbb{C} \vdash C\ \mathsf{type} \qquad \Gamma \vdash M_{\mathsf{base}} : [\mathsf{base}/c]C \qquad \Gamma \vdash M_{\mathsf{loop}} : M_{\mathsf{base}} =^{c.C}_{\mathsf{loop}} M_{\mathsf{base}}}{\Gamma \vdash \mathbb{C}\text{-}\mathsf{ind}[c.C](M_{\mathsf{base}}; M_{\mathsf{loop}})(M) : [M/c]C}$$

$$\mathbb{C}\text{-}\mathsf{ind}[c](C; M_{\mathsf{base}})(M_{\mathsf{loop}})\mathsf{base} \equiv M_{\mathsf{base}} : [\mathsf{base}/c]C$$

$$\mathsf{apd}(\mathbb{C}\text{-}\mathsf{ind}[c.C](M_{\mathsf{base}}; M_{\mathsf{loop}})(-))(\mathsf{loop}) =_{(M_{\mathsf{base}} =^{c.C}_{\mathsf{loop}} M_{\mathsf{base}})} M_{\mathsf{loop}}$$

One thing worth pointing out is that it would have been possible (but not correct) to replace the third premise of the elimination rule with $\Gamma \vdash M_{\mathsf{loop}} : M_{\mathsf{base}} =_{[\mathsf{base}/c]C} M_{\mathsf{base}}$. This would have been type-correct, but it does not capture the right case in the induction principle, which should be to exhibit a loop in C which lies over the generating loop in $\mathbb{C}$.

3.1 Loop space

There is the (based) loop space $\Omega(A; a_0) \triangleq a_0 =_A a_0$ for any point $a : A$. Here, the groupoid laws become *group* laws, because the endpoints are the same. To ensure that this is actually a group, one can take its "zero-truncation" $\| \Omega(A; a_0) \|_0$, in which all the higher structure has been squashed down to reflexivities. This truncation of the loop space is called the "fundamental group" of a type.

Using the univalence axiom, it can be shown that the loop space of the circle $\mathbb{C}$ with base point base can be identified with the group of integers under addition. Because the integers are already 0-truncated, we have also shown that the fundamental group of the circle is the integers [Licata and Shulman, 2013].

4 Judgmental methodology

To study the degree to which HoTT has computational meaning (i.e. develop a way to phrase homotopy type theory which does have computational meaning), we return to Martin-Löf's version of the notion of a *judgment*, which forms the epistemic foundation of modern type theory [Martin-Löf, 1987a, 1994, 1996].

The guiding idea is that judgments come first: structure which appears in the connectives and type constructors is developed using analogous structure at the level of judgments.

Gentzen's key contribution was his stress on *entailment* (hypothetical judgment) as being prior to implication; compare Hilbert systems which have *only* implication. The contradiction between entailment and implication has a parallel in the old battle between lambda calculus and combinators.

Entailment $J_0 \vdash J_1$ expresses that the fact J_1 can be concluded once the fact J_0 is assumed, but it does not yet express *generality*. The *generic judgment* is the structure from which the universal quantification connective is built, in the same way as the hypothetical judgment lies underneath the implication connective; using the notation of Martin-Löf [1987b]:

$$\frac{\vert_x \ (A\ \mathsf{true})}{\forall x.A\ \mathsf{true}}$$

Constructivism (which we take here as the principle that the evidence that makes a proposition true is a mathematical object) suggests a consolidation of the hypothetical and the generic judgments into a single form, named "hypothetico-general judgment" by Martin-Löf:

$$\boxed{x_1 : A_1, \ldots, x_n : A_n \vdash M : A}$$

This notation emphasizes the centrality of variables in type theory. We must reiterate that type theory does not fix a particular interpretation of variables, but is instead compatible with multiple explanations of variables, each of which can be deployed in different settings with different aims. We will call out two possible interpretations of variables in particular:

formal	**semantic**
indeterminates / generic values	placeholders
"derivability"	"admissibility" (careful!)
$\Gamma \vdash M : A$	$\Gamma \gg M \in A$

Both forms of judgment above share the *structural properties* and present a consequence relation. However, the variables on the formal side range over open elements (elements that may themselves have free variables), whereas the variables on the semantic side range over only closed elements. The semantic consequence relation and account of variables as placeholders is essentially the mathematical notion of a variable, whereas the formal consequence relation and account of variables as indeterminates is proof-theoretic in nature, making terms analogous to polynomials.

The defeasible structural principles which define the structure of contexts (weakening/projections, contraction/diagonals, exchange/symmetries) are designed to be admissible each consequence relation, as are the non-defeasible ones (identity and composition). The negotiation of the structural maps will appear again later in the course in the context of cubical type theory.

Remark 1. *In the first version of Church's lambda calculus, you were required to use every variable that is introduced. Later this idea of working without weakening was codified into what some call "relevant logic" and others call "strict logic".*

5 Judgmental structure of identifications

To give a type-theoretic account of paths/identifications, we must conceive them as an intrinsic concept in type theory, emanating from the judgmental base. We are not free to simply "throw in" identifications without explaining them at the judgmental level, since this would disrupt many important properties of type theory, including its computational content.

There are multiple possible ways to give a judgmental account of higher-dimensional structure, each of which is based on a different kind of shape; the type-theoretic community has mostly settled on a *cubical* account of higher dimensional structure.[1]

In HoTT (which can be thought of as a "globular" account of higher structure), higher structure is captured by iterating the identification type. Instead, we want to give these dimension levels names:

1. points
2. lines
3. squares
4. cubes

... (and beyond)

Accordingly, we extend the judgments of type theory to indicate whether they are constructing a point in a type, or a line, or a cube, or a higher cube, etc. For now, we might write $\Gamma \gg M \doteq M' \in A \mathbin{@} n$ to mean that M and M' are equal "cubes" of type A of dimension n. When $n = 0$, this is the familiar member equality judgment, but when $n \geq 1$, this form of judgment asserts the unfamiliar equation between higher-dimensional elements of a higher-dimensional type.

The way that we make sense of this is to say that an "n-dimensional type" is a *family* of types that is indexed by the generic n-cube, i.e. the nth power of the interval. To make this precise, we replace this n parameter with a *context* Ψ (of length n) of dimension variables, which are allowed to occur in Γ, M, M', A.

[1] There are several different possible flavors of cubical structure; it is a matter of current scientific inquiry to determine the benefits and drawbacks of each version of cubical structure.

When we assert the judgment $M \in A\ [x]$, we are saying that A is a *line of types* in the x direction, and that M is a *line of elements* of A. To get the left endpoint of the type A we write $M\langle 0/x\rangle$, and we have $M\langle 0/x\rangle \in A\langle 0/x\rangle\ [\cdot]$. One of the big difficulties of higher dimensional type theory is to develop a theory of coercions and compositions which let us move an element from the 0 fiber to the 1 fiber, or even the 0 fiber to the x fiber! We will study this further in future lectures.

It is important to note that the variables $\Psi = x, y, \ldots$ range over an abstract notion of interval, rather than a metric one. In particular, we do not have any notion of distance; you are either at the endpoints $A\langle 0/x\rangle$, $A\langle 1/x\rangle$ or even in "middle" $A\langle y/x\rangle$, but there is no sense in which you can be "two-thirds of the way", etc.

6 Dimensions and dimension substitutions

Formally, we define a *dimension context* Ψ as a finite sequence of *dimension variables*:

$$\Psi = x_1, \cdots, x_n \quad (n \geq 0).$$

These variables can be thought as cartesian coordinates in an n-space. A *dimension term* is either 0, 1 or a dimension variable. We write $\Psi \vdash r\ \mathsf{dim}$ when r is a valid dimension term in dimension context Ψ:

$$\Psi \vdash 0\ \mathsf{dim} \qquad \Psi \vdash 1\ \mathsf{dim} \qquad \Psi, x, \Psi' \vdash x\ \mathsf{dim}.$$

A λ-calculus term M whose free dimension variables belong to Ψ is called a Ψ-cube. In this case, we write $M\ \mathsf{tm}\ [\Psi]$. The dimension variables in M can be subsituted for dimension terms via a *dimension substitution*. A dimension substitution $\psi : \Psi' \to \Psi$ maps variables of $\Psi = x_1, \cdots, x_n$ to dimension terms that are valid in Ψ':

$$\Psi' \vdash \psi(x_i)\ \mathsf{dim} \qquad (\text{for } 1 \leq i \leq n).$$

Thus, a dimension substitution $\psi : \Psi' \to \Psi$ takes a Ψ-cube M into a Ψ'-cube $M\psi$. Formally,

$$\text{if } M\ \mathsf{tm}\ [\Psi] \text{ and } \psi : \Psi' \to \Psi \text{ then } M\psi\ \mathsf{tm}\ [\Psi'].$$

Substitutions define contravariant *functorial actions* on λ-terms, in the sense that they preserve identity and composition:

$$M \cdot \mathsf{id} = M \qquad M \cdot (\psi_1\psi_2) = (M \cdot \psi_1) \cdot \psi_2$$

where id refers to the identity substitution and $\psi_1\psi_2$ refers to the composition of substitutions ψ_1 and ψ_2.

Dimension contexts may be thought of as "possible worlds" in the sense of Kripke, or more generally pre-sheaf, interpretations in which worlds specify a collection of "indeterminates". Worlds are related by substitutions, which interpret one collection of indeterminates in terms of another by either specifying an end point (0 or 1) for a variable or replacing a variable by another variable, with possibly several variables being replaced by one. Dimension substitutions induce a contravariant, functorial action on terms, by performing the given substitutions on them.

Example: getting the endpoints of a line via substitution If M_x is a λ-term with a single free dimension variable x, it is called an (x)-cube. Because it is one dimensional, we also say that M_x is a line. We can get its endpoints by substituting 0 or 1 into x (for the "left" and "right" endpoints respectively):

$$M\langle 0/x\rangle \xrightarrow{\qquad M_x \qquad} M\langle 1/x\rangle$$

Dimension contexts are structural Dimension contexts are structural: they admit *exchange*, *weakening* and *contraction* in the following sense:

- **Exchange:** the order of variables in a dimension context does not matter and variables can be permuted silently:

$$\Psi, x, y, \Psi' \xrightarrow{\text{silent}} \Psi, y, x, \Psi'.$$

  For example, an (x, y)-cube is a (y, x)-cube and vice versa.

- **Weakening:** a Ψ-cube is also a (degenerate) (Ψ, x)-cube ($x \notin \Psi$):

$$\Psi, x, y \xrightarrow{\text{silent}} \Psi.$$

  For example, the x-line in the previous example is also a degenerate (x, y)-square.

- **Contraction:** the substitution $\langle z, z \,/\, x, y\rangle$ turns a (Ψ, x, y)-cube into a (Ψ, z)-cube:

$$\Psi, z \xrightarrow{\langle z, z \,/\, x, y\rangle} \Psi, x, y.$$

We can now investigate the definition of a simple cubical programming language.

7 A cubical programming language

We define an operational semantics for a cubical programming language using two judgments

$$M \ \mathsf{value} \qquad M \mapsto M'$$

for when M is a value and when M transitions to M' respectively. We write $M \Downarrow M'$ when $M \mapsto^* M'$ and M value, in which case we say that M *evaluates* to value M'. For clarity, we sometimes annotate these judgments with the dimension context of M:

$$M \ \mathsf{value}_\Psi \qquad M \mapsto_\Psi M' \qquad M \Downarrow^\Psi M'.$$

Such a notation makes sense because a Ψ-cube can only transition to a (possibly degenerate[2]) Ψ-cube. Indeed, a transition cannot make new free dimension variables appear in a term.

Note that these two judgments do not commute with dimension substitution. Typically,

- we can have $M \ \mathsf{value}_{(x)}$ but not $M\langle 0/x\rangle \ \mathsf{value}_\emptyset$
- we can have $M \mapsto_{(x)} M'$ but not $M\langle 0/x\rangle \mapsto_\emptyset M'\langle 0/x\rangle$.

We can observe this phenomenon on the example of the circle $\mathbb{C}$.

7.1 Operational semantics of $\mathbb{C}$

In order to augment our language with the circle type $\mathbb{C}$, we need to add the following λ-terms in our grammar:

$$\text{Terms } M \ := \ \cdots \mid \mathbb{C} \mid \mathsf{base} \mid \mathsf{loop}_r \mid \mathbb{C}\text{-}\mathsf{ind}[c.C](M_{\mathsf{base}}; M_{\mathsf{loop}})(M)$$

where r refers to a dimension term: it can be either 0, 1 or a dimension variable x. These terms are subject to the following operational semantics:

[2] A Ψ-cube M is said to be *degenerate* if there are dimension variables in Ψ that do not appear free in M.

$$\mathsf{base}\ \mathsf{value} \qquad \mathsf{loop}_x\ \mathsf{value} \qquad \mathsf{loop}_0 \mapsto \mathsf{base} \qquad \mathsf{loop}_1 \mapsto \mathsf{base}$$

Moreover, in order to evaluate $\mathbb{C}\text{-}\mathsf{ind}[c.C](M_{\mathsf{base}}; x.M_{\mathsf{loop}})(M)$, we first evaluate M:

$$\frac{M \mapsto M'}{\mathbb{C}\text{-}\mathsf{ind}[c.C](M_{\mathsf{base}}; x.M_{\mathsf{loop}})(M) \mapsto \mathbb{C}\text{-}\mathsf{ind}[c.C](M_{\mathsf{base}}; x.M_{\mathsf{loop}})(M')}$$

Then, if Ψ is a dimension context, $x \notin \Psi$, $M_{\mathsf{base}}\ \mathsf{tm}\ [\Psi]$ and $M_{\mathsf{loop}}\ \mathsf{tm}\ [\Psi, x]$, we have:

$$\mathbb{C}\text{-}\mathsf{ind}[c.C](M_{\mathsf{base}}; x.M_{\mathsf{loop}})(\mathsf{base}) \mapsto_{\Psi} M_{\mathsf{base}}$$

$$\mathbb{C}\text{-}\mathsf{ind}[c.C](M_{\mathsf{base}}; x.M_{\mathsf{loop}})(\mathsf{loop}_x) \mapsto_{\Psi, x} M_{\mathsf{loop}}$$

As we can see, substituting into a loop changes a value into a non-value: $\mathsf{loop}_x\ \mathsf{value}$ but $\mathsf{loop}_0 \mapsto \mathsf{base}$. Besides, reduction does not commute with substitution. Indeed,

$$\begin{aligned}
&(\mathbb{C}\text{-}\mathsf{ind}[c.C](M_{\mathsf{base}}; x.M_{\mathsf{loop}})(\mathsf{loop}_x))\ \langle 0/x \rangle \\
&= \mathbb{C}\text{-}\mathsf{ind}[c.C](M_{\mathsf{base}}; x.M_{\mathsf{loop}})(\mathsf{loop}_0) \\
&\mapsto \mathbb{C}\text{-}\mathsf{ind}[c.C](M_{\mathsf{base}}; x.M_{\mathsf{loop}})(\mathsf{base}) \\
&\mapsto M_{\mathsf{base}}
\end{aligned}$$

but

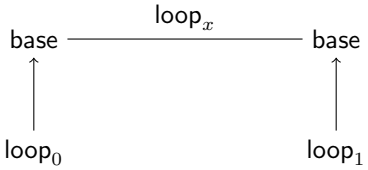
$$\mathbb{C}\text{-}\mathsf{ind}[c.C](M_{\mathsf{base}}; x.M_{\mathsf{loop}})(\mathsf{loop}_x) \mapsto M_{\mathsf{loop}}$$

and there is no reason to expect $M_{\mathsf{base}} \equiv M_{\mathsf{loop}}\langle x/0 \rangle$ in general. Although there is no syntactic confluence in cubical dependent type theory, we use types to enforce semantic confluence. Coherence between different aspects of a cube is not obtained via the operational semantics but via the type system, as detailed in section 7.2.

7.2 Giving a meaning to the judgments $M \in A\ [\Psi]$

In standard (0-dimensional) computational type theory, the judgment $M \in A$ means that $A \Downarrow A_0$, $M \Downarrow M_0$, and that M_0 is a (canonical) value of type A_0. Everything is about running programs and using types to specify their values. Similarly, in a higher dimensional setting, we say that $M \in A\ [\Psi]$ if $A \Downarrow^{\Psi} A_0$, $M \Downarrow^{\Psi} M_0$ and M_0 is a Ψ-value of type A_0. It remains to explain what Ψ-values are and what type membership is. We must be careful in defining these though. For example, the meaning of membership must account for substitutions, as we argue below.

Indeed, if a cube belongs to a type, we expect all its faces to belong to the same type. For example, because $\mathsf{loop}_x \in \mathbb{C}\ [x]$ and $\mathsf{loop}_x\langle 0/x \rangle \equiv \mathsf{loop}_0$, we expect $\mathsf{loop}_0 \in \mathbb{C}\ [\cdot]$. Because $\mathsf{loop}_0 \Downarrow \mathsf{base}$, this is equivalent to $\mathsf{base} \in \mathbb{C}\ [\cdot]$.

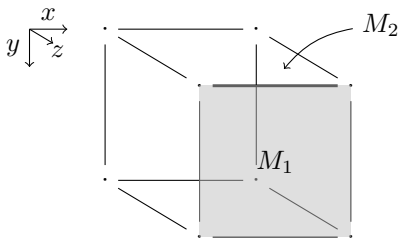


More generally, we want the following to hold.

Desideratum 1 (first attempt) If $M \in A\,[\Psi]$, then for any substitution $\psi : \Psi' \to \Psi$ we have $M\psi \in A\,[\Psi']$ (i.e. $A \Downarrow A_0$, $M\psi \Downarrow M_0$ and M_0 is a Ψ'-value of A_0).

In other words, the meaning of a cube must account for all its *aspects*, where the aspects of a Ψ-cube M is the set of all $M\psi$ for $\psi : \Psi' \to \Psi$ a substitution (in particular, faces are aspects). This is not enough to get the kind of semantic confluence properties that we discussed in section 7.1 though. Indeed, the aspects of a cube must also be *coherent* in a sense that we make precise below.

Suppose we have the following (x, y, z)-cube. Let's call it M.



We can compute the front face of M: $M\langle 1/z\rangle \Downarrow M_1$. Also, we can compute the top edge of M_1: $M_1\langle 0/y\rangle \Downarrow M_2$. Intuitively, M_2 should also be an aspect of M. Indeed, if $M\langle 1, 0/z, y\rangle \Downarrow M_{1,2}$, then we expect that M_2 and $M_{1,2}$ should coincide somehow. More precisely, if M has type A_0, we expect that $M_2 \doteq M_{1,2} \in A_0\,[x]$.[3] Generalizing, this leads to the following desideratum.

Desideratum 2 (first attempt) If the following conditions hold

- $M \in A\,[\Psi]$
- $A \Downarrow^{\Psi} A_0$ and $M \Downarrow^{\Psi} M_0$
- $\psi_1 : \Psi_1 \to \Psi$ and $\psi_2 : \Psi_2 \to \Psi_1$
- $M_0\psi_1 \Downarrow^{\Psi_1} M_1$ and $M_1\psi_2 \Downarrow^{\Psi_2} M_2$
- $M_0(\psi_1\psi_2) \Downarrow^{\Psi_2} M_{1,2}$,

then we have $M_{1,2} \doteq M_2 \in A_0\,[\Psi_2]$.

We insist that nothing forces this to hold for every term in our language that is syntactically correct. It is always possible to write bad programs violating this, but we simply decide to not give them a type. Types are expected to be specifications of good programs, whose meaning accounts for all their aspects and whose aspects are coherent (Desiderata 1 and 2).

Adding dependency In our first attempt to formulate Desiderata 1 and 2, we neglected dependency. Indeed, because types can depend on higher dimensional λ-terms, then types can also vary in dimensions. For example, if F is an A-indexed family of types and $M \in A\,[x]$, then $F[M]$ is a type which varies in dimension x: $F[M]\langle 0/x\rangle = F[M\langle 0/x\rangle]$.

Taking this into account, we generalize Desiderata 1 and 2 as follows.

[3] Remember that the judgment $M \in A\,[\Psi]$ is only a shorthand for $M \doteq M \in A\,[\Psi]$.

Desideratum 1 *The meaning of a cube must account for all its aspects.* That is, if $M \in A\ [\Psi]$, then for any substitution $\psi : \Psi' \to \Psi$ we have $M\psi \in A\psi\ [\Psi']$ (i.e. $A\psi \Downarrow A_0$, $M\psi \Downarrow M_0$ and M_0 is a Ψ'-value of A_0).

Desideratum 2 *The aspects of a cube must be coherent.* That is, if

- $M \in A\ [\Psi]$
- $A \Downarrow^{\Psi} A_0$ and $M \Downarrow^{\Psi} M_0$
- $\psi_1 : \Psi_1 \to \Psi$ and $\psi_2 : \Psi_2 \to \Psi_1$
- $M_0\psi_1 \Downarrow^{\Psi_1} M_1$ and $M_1\psi_2 \Downarrow^{\Psi_2} M_2$
- $A_0\psi_1 \Downarrow^{\Psi_1} A_1$ and $A_1\psi_2 \Downarrow^{\Psi_2} A_2$
- $M_0(\psi_1\psi_2) \Downarrow^{\Psi_2} M_{1,2}$,

then we have $M_{1,2} \doteq M_2 \in A_2\ [\Psi_2]$.

One should picture a cube M being typed by another cube A: the front face of M must inhabit the front face of A, the top right edge of M must inhabit the top right edge of $A\ldots$

Also, one should note that equality is prior to the notion of path. Indeed, in order to make paths "fit together" to form cubes, we need a notion of equality of endpoints.

7.3 Hypothetico-General judgments

We now investigate the meaning of the hypothetico-general judgment

$$a : A \gg N \in B\ [\Psi].$$

A first attempt at a definition would be as follows:

$$\text{if } M \doteq M' \in A\ [\Psi], \quad \text{then} \quad [M/a]N \doteq [M'/a]N \in [M/a]B \doteq [M'/a]B\ [\Psi].$$

This is not enough though. We need more than just ensuring that equal things are sent to equal things: we also need to ensure that the definition works at all accessible dimensions (for all aspects of A) and respects paths. This gives us the following definition.

Definition. *We write* $a : A \gg N \in B\ [\Psi]$ *whenever for any substitution* $\psi : \Psi' \to \Psi$,

$$\textit{if } M \doteq M' \in A\psi\ [\Psi'] \textit{ then } [M/a]N\psi \doteq [M'/a]N\psi \in [M/a]B\psi \doteq [M'/a]B\psi\ [\Psi'].$$

In particular, if we take ψ to be the weakening map $\Psi, x \to \Psi$ ($x \notin \Psi$), then we get that N must respect x-lines in A, taking them to heterogeneous lines spanning $[M/a]B$ and $[M'/a]B$. Therefore, we recover the `apd` behavior of HoTT.

The semantics of the hypothetical judgment corresponds to that of a mapping in a presheaf, wherein a mapping from A to B at Ψ is not merely a function mapping the elements of A at Ψ to elements of B at Ψ, but, rather, must also account for arguments that arise in all possible worlds relative to Ψ: given any $\psi : \Psi' \to \Psi$, it must map elements of A at Ψ' to elements of B at Ψ', a strong uniformity condition. The motivation for this requirement is to ensure that elements and mappings are preserved on passage to accessible worlds.

In the case of the hypothetical judgment, we wish to ensure that if

$$a : A \gg N \in B\ [\Psi]$$

and $\psi : \Psi' \to \Psi$, then

$$a : A\psi \gg N\psi \in B\psi\ [\Psi'].$$

In particular if ψ is a weakening $\Psi, x \to \Psi$, which acts silently, then we require that if

$$a : A \gg N \in B\ [\Psi],$$

then

$$a : A \gg N \in B\ [\Psi, x].$$

This says that N must not only take *points* in A at dimension Ψ to points in B at the same dimension, it must also take *lines* in A at dimension Ψ, x to lines in B at that extended dimension. More succinctly, N must map points in such a way that it also preserves paths between them.

References

Carlo Angiuli, Robert Harper, and Todd Wilson. Computational higher-dimensional type theory. POPL 2017, 2017a.

Carlo Angiuli, Kuen-Bang Hou (Favonia), and Robert Harper. Computational higher type theory III: Univalent universes and exact equality. `https://arxiv.org/abs/1712.01800`, 2017b.

Daniel R. Licata and Michael Shulman. Calculating the fundamental group of the circle in homotopy type theory. In *Proceedings of the 2013 28th Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '13, pages 223–232, Washington, DC, USA, 2013. IEEE Computer Society. ISBN 978-0-7695-5020-6. doi: 10.1109/LICS.2013.28. URL `http://dx.doi.org/10.1109/LICS.2013.28`.

Per Martin-Löf. Truth of a proposition, evidence of a judgement, validity of a proof. *Synthese*, 73(3):407–420, 1987a.

Per Martin-Löf. The logic of judgements. Workshop on General Logic, Laboratory for Foundations of Computer Science, University of Edinburgh, 1987b.

Per Martin-Löf. Analytic and synthetic judgements in type theory. In Paolo Parrini, editor, *Kant and Contemporary Epistemology*, pages 87–99. Springer Netherlands, Dordrecht, 1994. ISBN 978-94-011-0834-8. doi: 10.1007/978-94-011-0834-8_5.

Per Martin-Löf. On the meanings of the logical constants and the justifications of the logical laws. *Nordic Journal of Philosophical Logic*, 1(1):11–60, 1996.

The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. `https://homotopytypetheory.org/book`, Institute for Advanced Study, 2013.