Computational semantics of Cartesian cubical type theory

Carlo Angiuli
joint work with Favonia
and Robert Harper

Carnegie Mellon University

HoTTEST, March 15, 2018

Problem

Suppose I want to define a type theory for reasoning about $\{$homotopy types, ∞-categories, smooth ∞-groupoids, non-terminating programs, probabilistic programs $\ldots\}$.

What rules should I include?

Problem

Suppose I want to define a type theory for reasoning about $\{$homotopy types, ∞-categories, smooth ∞-groupoids, non-terminating programs, probabilistic programs $\ldots\}$.

What rules should I include?

Obvious answer: whatever holds in the intended models.

Problem

In the HoTT Book,

$$\frac{\Gamma \vdash a : \mathbf{1}}{\Gamma \vdash a \equiv \star : \mathbf{1}}\ \textbf{1}\text{-}\textsc{eta}\ ✗$$

$$\frac{\begin{array}{c}\Gamma, x : \mathbf{I} \vdash P\ \mathbf{type} \\ \Gamma \vdash b_0 : P[\mathbf{0_I}/x] \\ \Gamma \vdash b_1 : P[\mathbf{1_I}/x] \\ \Gamma \vdash s : b_0 =^P_{\mathbf{seg}} b_1\end{array}}{\Gamma \vdash \mathbf{ind_I}(x.P, b_0, b_1, s, \mathbf{0_I}) \equiv b_0 : P[\mathbf{0_I}/x]}\ \mathbf{I}\text{-}\textsc{comp-z}\ ✓$$

$$\frac{\vdots}{\Gamma \vdash \mathbf{apd}_{\lambda y.\mathbf{ind_I}(x.P, b_0, b_1, s, y)}(\mathbf{seg}) \equiv s : b_0 =^P_{\mathbf{seg}} b_1}\ \mathbf{I}\text{-}\textsc{comp-s}\ ✗$$

Problem

We might want...

- terms to have unique types.
- judgments (especially $\equiv$) to be decidable.
- existence property:
  if $\cdot \vdash p : (n{:}\mathbf{nat}) \times P(n)$, there is a numeral n such that $P(n)$.
- canonicity property:
  if $\cdot \vdash b : \mathbf{bool}$, b computes to $\mathbf{true}$ or $\mathbf{false}$.

These are all inherently questions of rules and syntax!

All, in practice, require models in which proofs are computations.

Problem

These properties are important in practice.

Brunerie successfully showed $\pi_4(\mathbf{S}^3)$ is $\mathbb{Z}/k\mathbb{Z}$
where $\cdot \vdash k : \mathbf{nat}$ (14 pages, 2013).

In his PhD thesis (129 pages, 2016), showed $k = 2$.

In a computational semantics, k just evaluates to 2!

Overview

When designing a TT, consider its computational semantics!

- Computational semantics of MLTT.
- Cartesian cubical TT and its computational semantics.

Computational semantics

Computational semantics

Canonicity only holds for closed terms.

$$b : \mathbf{bool} \vdash b : \mathbf{bool}$$
$$f : \mathbf{bool} \to \mathbf{bool} \vdash f(\mathbf{true}) : \mathbf{bool}$$
$$x : \mathbf{bool}, f : (\mathbf{bool} \times \mathbf{bool}) \to \mathbf{bool} \vdash f\ \langle x, \mathbf{true} \rangle : \mathbf{bool}$$

Can characterize neutral open terms using a generalization of the tools I discuss; I will focus on properties of closed terms.

Computational semantics

Build a model in which closed terms are regarded as programs.

- Define programming language and operational semantics.
- Define a notion of equality at each type.
- Check this is compatible with desired rules.

Operational semantics

Define syntax of preterms (modulo α-equivalence only).
Includes both "terms" and "types."

$$\begin{aligned}\mathbf{Term} := {} & (a{:}A) \to B \mid \lambda a.M \mid \mathbf{app}(M, N) \\ & \mid (a{:}A) \times B \mid \langle M, N \rangle \mid \mathbf{fst}(M) \mid \mathbf{snd}(M) \\ & \mid \mathbf{Id}_A(M, N) \mid \mathbf{refl}_M \mid \mathbf{J}_{a.b.p.C}(M; a.R) \\ & \mid \mathbf{bool} \mid \mathbf{true} \mid \mathbf{false} \mid \mathbf{if}_{b.A}(M; T, F) \mid \cdots\end{aligned}$$

Each closed term computes to a value.

Operational semantics

$$- \ \mathbf{val} : \mathbf{Term} \to \mathbf{Prop}$$

$$- \longmapsto - : \mathbf{Term} \to \mathbf{Term} \to \mathbf{Prop}$$

$$- \Downarrow - : \mathbf{Term} \to \mathbf{Val} \to \mathbf{Prop}$$

$$\frac{}{(a{:}A) \to B \ \mathbf{val}} \qquad \frac{}{\lambda a.M \ \mathbf{val}} \qquad \frac{M \longmapsto M'}{\mathbf{app}(M, N) \longmapsto \mathbf{app}(M', N)}$$

$$\frac{}{\mathbf{app}(\lambda a.M, N) \longmapsto M[N/a]} \qquad \frac{}{\mathbf{bool\ val}} \qquad \frac{}{\mathbf{true\ val}} \qquad \frac{}{\mathbf{false\ val}}$$

$$\frac{M \longmapsto M'}{\mathbf{if}_{b.A}(M; T, F) \longmapsto \mathbf{if}_{b.A}(M'; T, F)} \qquad \frac{}{\mathbf{if}_{b.A}(\mathbf{true}; T, F) \longmapsto T}$$

$$\frac{}{\mathbf{if}_{b.A}(\mathbf{false}; T, F) \longmapsto F}$$

Booleans

The meanings of non-values are determined by their values.

Definition

- $M \in \mathbf{bool}$ if $M \Downarrow \mathbf{true}$ or $M \Downarrow \mathbf{false}$.
- $M \doteq N \in \mathbf{bool}$ if $M, N \Downarrow \mathbf{true}$ or $M, N \Downarrow \mathbf{false}$.

Booleans

The meanings of non-values are determined by their values.

Definition

- $M \in \mathbf{bool}$ if $M \Downarrow \mathbf{true}$ or $M \Downarrow \mathbf{false}$.
- $M \doteq N \in \mathbf{bool}$ if $M, N \Downarrow \mathbf{true}$ or $M, N \Downarrow \mathbf{false}$.

Definition

- $M \in \mathbf{bool}$ iff $M \doteq M \in \mathbf{bool}$.

Booleans

The meanings of non-values are determined by their values.

Definition

- $M \in \mathbf{bool}$ if $M \Downarrow \mathbf{true}$ or $M \Downarrow \mathbf{false}$.
- $M \doteq N \in \mathbf{bool}$ if $M, N \Downarrow \mathbf{true}$ or $M, N \Downarrow \mathbf{false}$.

Definition

- $M \in \mathbf{bool}$ iff $M \doteq M \in \mathbf{bool}$.
- $M \doteq N \in \mathbf{bool}$ iff $[\![\mathbf{bool}]\!]^{\Downarrow}(M, N)$, where $[\![\mathbf{bool}]\!] = \{(\mathbf{true}, \mathbf{true}), (\mathbf{false}, \mathbf{false})\}$.

Partial equivalence relations

$[\![\mathbf{bool}]\!] : \mathbf{Val} \to \mathbf{Val} \to \mathbf{Prop}$ is a partial equivalence relation:
a symmetric and transitive relation.

Equivalently, a subset of $\mathbf{Val}$, and an equivalence relation.

Partial equivalence relations

$[\![\mathbf{bool}]\!] : \mathbf{Val} \to \mathbf{Val} \to \mathbf{Prop}$ is a partial equivalence relation: a symmetric and transitive relation.

Equivalently, a subset of $\mathbf{Val}$, and an equivalence relation.

Why not just quotient, and say types are sets of values?
Because rules of TT range over terms, not equivalence classes.

Function types

The meanings of open terms are determined by their behavior as maps from closed terms to closed terms.

Definition

Given $[\![A]\!]$ and $[\![B]\!]$, define $[\![A \to B]\!](\lambda a.N_1, \lambda a.N_2)$ when $P \mapsto N_i[P/a]$ are equal as functions from $[\![A]\!]^{\Downarrow}$ to $[\![B]\!]^{\Downarrow}$: they send equal elements of A to equal elements of B.

The meanings of compound types are determined by the meanings of their constituent types.

Function types

The meanings of open terms are determined by their behavior as maps from closed terms to closed terms.

Definition

Given $[\![A]\!]$ and $[\![B]\!]$, define $[\![A \to B]\!](\lambda a.N_1, \lambda a.N_2)$ when $P \mapsto N_i[P/a]$ are equal as functions from $[\![A]\!]^{\Downarrow}$ to $[\![B]\!]^{\Downarrow}$: they send equal elements of A to equal elements of B.

The meanings of compound types are determined by the meanings of their constituent types.

Type systems

MLTT has five judgments:

$$\Gamma\ \mathbf{ctx}$$
$$\Gamma \vdash A\ \mathbf{type}$$
$$\Gamma \vdash A \equiv B\ \mathbf{type}$$
$$\Gamma \vdash M : A$$
$$\Gamma \vdash M \equiv N : A$$

In the computational semantics, we reduce open judgments to closed judgments, membership judgments to equality judgments. . .

Type systems

$$A \doteq B \ \mathbf{type}$$

$$M \doteq N \in A$$

Type systems

$$A \doteq B \ \mathbf{type}$$
$$M \doteq N \in A$$

...and judgments on non-values to judgments on values.

Definition

$\tau : \mathbf{Val} \to \mathbf{Val} \to (\mathbf{Val} \to \mathbf{Val} \to \mathbf{Prop}) \to \mathbf{Prop}$
is a (semantic) type system if it is

- functional: $\tau(A_0, B_0, \varphi) \wedge \tau(A_0, B_0, \varphi') \implies (\varphi = \varphi')$;
- symmetric: $\tau(A_0, B_0, \varphi) \implies \tau(B_0, A_0, \varphi')$;
- transitive: $\tau(A_0, B_0, \varphi) \wedge \tau(B_0, C_0, \varphi') \implies \tau(A_0, C_0, \varphi'')$;
- PER-valued: $\tau(A_0, B_0, \varphi) \implies (\varphi$ is a PER$)$.

Type systems

We can define all the semantic judgments relative to any τ.

Definition
$\tau \models (A \doteq B\ \mathbf{type})$ when $\tau^{\Downarrow}(A, B, \varphi)$.

In this case, let $[\![A]\!] = [\![B]\!] = \varphi$.

Definition
$\tau \models (M \doteq N \in A)$, presupposing $\tau \models (A\ \mathbf{type})$, when $[\![A]\!]^{\Downarrow}(M, N)$.

Type systems

Define open judgments by induction on the length of the context.

Definition

$\tau \models (a : A \gg B \doteq C \ \mathbf{type})$, presupposing $\tau \models (A \ \mathbf{type})$, when for all M, M' such that $\tau \models (M \doteq M' \in A)$, $\tau \models (B[M/a] \doteq C[M'/a] \ \mathbf{type})$.

Definition

$\tau \models (a : A \gg N \doteq N' \in B)$, presupposing $\tau \models (a : A \gg B \ \mathbf{type})$, when for all M, M' such that $\tau \models (M \doteq M' \in A)$, $\tau \models (N[M/a] \doteq N'[M'/a] \in B[M/a])$.

Type systems

Define open judgments by induction on the length of the context.

Definition

$\tau \models (a : A \gg B \doteq C \ \mathbf{type})$, presupposing $\tau \models (A \ \mathbf{type})$, when for all M, M' such that $\tau \models (M \doteq M' \in A)$, $\tau \models (B[M/a] \doteq C[M'/a] \ \mathbf{type})$.

Definition

$\tau \models (a : A \gg N \doteq N' \in B)$, presupposing $\tau \models (a : A \gg B \ \mathbf{type})$, when for all M, M' such that $\tau \models (M \doteq M' \in A)$, $\tau \models (N[M/a] \doteq N'[M'/a] \in B[M/a])$.

Type systems

Define τ as the least relation such that:

$\tau(\mathbf{bool}, \mathbf{bool}, \varphi)$ when $\varphi = \{(\mathbf{true}, \mathbf{true}), (\mathbf{false}, \mathbf{false})\}$.

$\tau((a{:}A) \to B, (a{:}A') \to B', \varphi)$ when

- $\tau \models (A \doteq A' \ \mathbf{type})$,
- $\tau \models (a : A \gg B \doteq B' \ \mathbf{type})$, and
- $\varphi(\lambda a.M, \lambda a.M')$ when $\tau \models (a : A \gg M \doteq M' \in B)$.

Type systems

Define τ as the least relation such that:

$\tau((a{:}A) \times B, (a{:}A') \times B', \varphi)$ when

- $\tau \models (A \doteq A'\ \mathbf{type})$,
- $\tau \models (a : A \gg B \doteq B'\ \mathbf{type})$, and
- $\varphi(\langle M, N\rangle, \langle M', N'\rangle)$ when $\tau \models (M \doteq M' \in A)$ and $\tau \models (N \doteq N' \in B[M/a])$.

$\tau(\mathbf{Id}_A(M, N), \mathbf{Id}_{A'}(M', N'), \varphi)$ when

- $\tau \models (A \doteq A'\ \mathbf{type})$,
- $\tau \models (M \doteq M' \in A)$,
- $\tau \models (N \doteq N' \in A)$, and
- $\varphi(\mathbf{refl}_M, \mathbf{refl}_N)$ when $\tau \models (M \doteq N \in A)$.

Canonicity

Theorem (Soundness)

If $\Gamma \vdash A \equiv B$ $\mathbf{type}$ *then* $\Gamma \gg A \doteq B$ $\mathbf{type}$.
If $\Gamma \vdash M \equiv N : A$ *then* $\Gamma \gg M \doteq N \in A$.

Proof.

Check every rule! (Very long.) $\square$

Canonicity

Corollary (Canonicity property)

If $\cdot \vdash M : \mathbf{bool}$ *then* $M \Downarrow \mathbf{true}$ *or* $M \Downarrow \mathbf{false}$*.*

Proof.

Then $M \in \mathbf{bool}$. Unwinding definitions, $[\![\mathbf{bool}]\!]^{\Downarrow}(M, M)$.
Therefore $M \Downarrow M_0$ and $[\![\mathbf{bool}]\!](M_0, M_0)$. $\square$

Canonicity

Corollary (Canonicity property)

If $\cdot \vdash M : \mathbf{bool}$ *then* $M \Downarrow \mathbf{true}$ *or* $M \Downarrow \mathbf{false}$*.*

Proof.

Then $M \in \mathbf{bool}$. Unwinding definitions, $[\![\mathbf{bool}]\!]^{\Downarrow}(M, M)$.
Therefore $M \Downarrow M_0$ and $[\![\mathbf{bool}]\!](M_0, M_0)$. □

Corollary (Consistency)

It is impossible that $\cdot \vdash M : \mathbf{void}$*.*

Proof.

$M \in \mathbf{void}$, so $[\![\mathbf{void}]\!]^{\Downarrow}(M, M)$, but $[\![\mathbf{void}]\!](M_0, M_0)$ never. □

Summary

This is a constructive (“logical relations”) model of types as sets of evaluated programs, modulo semantic equality.

Depending on your aims, this may even be the intended model (e.g., for program extraction).

Summary

This is a constructive ("logical relations") model of types as sets of evaluated programs, modulo semantic equality.

Depending on your aims, this may even be the intended model (e.g., for program extraction).

For better or for worse, it's not the initial model.

Summary

$$\frac{\Gamma \gg a \in \mathbf{unit}}{\Gamma \gg a \doteq \star \in \mathbf{unit}} \text{ ETA } ✓$$

$$\frac{}{\Gamma, a : \mathbf{void} \gg \mathcal{J}} \text{ ETA } ✓$$

$$\frac{\Gamma \gg P \in \mathbf{Id}_A(M, N)}{\Gamma \gg M \doteq N \in A} \text{ REFL } ✓$$

$$\frac{M \Downarrow \mathbf{true}}{M \in \mathbf{bool}} \text{ COMP } ✓$$

Extending the model?

Suppose, for the sake of argument, we want:

$$\frac{\Gamma \vdash A\ \mathbf{type} \qquad \Gamma \vdash B\ \mathbf{type}}{\Gamma \vdash \mathbf{ua}(\cdots) : \mathbf{Id}_{\mathcal{U}}(A \times B, B \times A)}$$

Extending the model?

Suppose, for the sake of argument, we want:

$$\frac{\Gamma \vdash A \ \mathbf{type} \qquad \Gamma \vdash B \ \mathbf{type}}{\Gamma \vdash \mathbf{ua}(\cdots) : \mathbf{Id}_{\mathcal{U}}(A \times B, B \times A)}$$

Computational justification of $\mathbf{J}$ was that every closed element of $\mathbf{Id}$ will be $\mathbf{refl}$. This rule is nonsense!

The end

Cubical type theory

Judgmental paths

Need to equip $[\![A]\!]$ directly with path structure (and composition structure), then define $[\![\mathbf{Id}_A(M, N)]\!]$ and $\mathbf{J}$ in terms of those.

Judgmental paths

Licata and Harper, *Canonicity for 2-Dimensional Type Theory* (2012): Define a judgment for path elements of A.

$\cdot \vdash A$ **type** means $[\![A]\!]$ is a 1-groupoid.

$\cdot \vdash M : A$ means M is an object of $[\![A]\!]$.

$\cdot \vdash P : M \simeq_A N$ means P is a morphism in $[\![A]\!]$ from M to N.

Groupoid structure is axiomatized directly:

$$\frac{\Gamma \vdash M : A}{\Gamma \vdash \mathbf{refl}_M : M \simeq_A N} \qquad \frac{\Gamma \vdash P : M \simeq_A N}{\Gamma \vdash P^{-1} : N \simeq_A M} \qquad \cdots$$

Judgmental paths

Bezem, Coquand, Huber, *A model of type theory in cubical sets* (2014): Constructive cubical set model, uniform Kan condition.

Direct inspiration for both Cohen, Coquand, Huber, Mörtberg, *Cubical Type Theory: a constructive interpretation of the univalence axiom* (2016), and the present work.

Why cubes?

Cubical type theory

Rough idea: $\square^n \vdash M : A$ means M is an n-cube of A.

$$\frac{\square^1 \vdash P : A}{\cdot \vdash P : \mathbf{Path}_A(P_0, P_1)}$$

Cubical type theory

Rough idea: $\square^n \vdash M : A$ means M is an n-cube of A.

$$\frac{\square^n, \square^1 \vdash P : A}{\square^n \vdash P : \mathbf{Path}_A(P_0, P_1)}$$

Cubical type theory

Rough idea: $\square^n \vdash M : A$ means M is an n-cube of A.

$$\frac{\square^n, \square^1 \vdash P : A}{\square^n \vdash P : \mathbf{Path}_A(P_0, P_1)}$$

Cubical type theory

Rough idea: $\square^n \vdash M : A$ means M is an n-cube of A.

$$\frac{\square^{n+1} \vdash P : A}{\square^n \vdash P : \mathbf{Path}_A(P_0, P_1)}$$

Representables are closed under products: $\square^{n+1} = \square^n \times \square^1$.

In contrast, $\Delta^{n+1} \neq \Delta^n \times \Delta^1$.

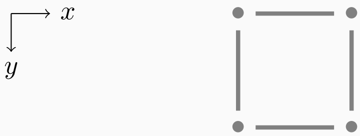
Cubical type theory

$x : \mathbb{I}, y : \mathbb{I} \vdash M$ is a square parametrized by two dimension variables.

We can take degeneracies by weakening by $z : \mathbb{I}$.

We can take faces by instantiating x, y at $0, 1$.

We can take diagonals by substituting x for y.



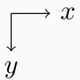
Cubical type theory

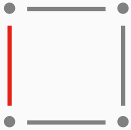
$x : \mathbb{I}, y : \mathbb{I} \vdash M$ is a square parametrized by two dimension variables.

We can take degeneracies by weakening by $z : \mathbb{I}$.

We can take faces by instantiating x, y at $0, 1$.

We can take diagonals by substituting x for y.





$M\langle 0/x \rangle$

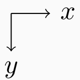
Cubical type theory

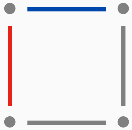
$x : \mathbb{I}, y : \mathbb{I} \vdash M$ is a square parametrized by two dimension variables.

We can take degeneracies by weakening by $z : \mathbb{I}$.

We can take faces by instantiating x, y at $0, 1$.

We can take diagonals by substituting x for y.





$M\langle 0/x\rangle$ $\quad$ $M\langle 0/y\rangle$

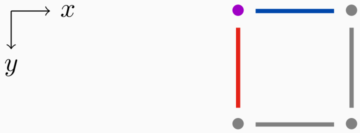
Cubical type theory

$x : \mathbb{I}, y : \mathbb{I} \vdash M$ is a square parametrized by two dimension variables.

We can take degeneracies by weakening by $z : \mathbb{I}$.

We can take faces by instantiating x, y at $0, 1$.

We can take diagonals by substituting x for y.



$$M\langle 0/x\rangle\langle 0/y\rangle \;=\; M\langle 0/y\rangle\langle 0/x\rangle$$

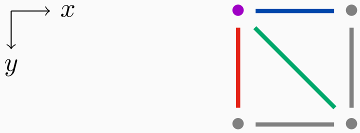
Cubical type theory

$x : \mathbb{I}, y : \mathbb{I} \vdash M$ is a square parametrized by two dimension variables.

We can take degeneracies by weakening by $z : \mathbb{I}$.

We can take faces by instantiating x, y at $0, 1$.

We can take diagonals by substituting x for y.



$$M\langle 0/x\rangle\langle 0/y\rangle \;=\; M\langle 0/y\rangle\langle 0/x\rangle$$

Cubical type theory

CCHM consider a full De Morgan algebra with also

- connections $M\langle(x \wedge y)/x\rangle$, $M\langle(x \vee y)/x\rangle$, and
- reversals $M\langle(1-x)/x\rangle$.

We have only permutations, faces, degeneracies, and diagonals: free finite-product category on $1 \rightrightarrows \mathbb{I}$.

Cartesian cubical computational type theory

Cubical programs

Define a cubical programming language.

$$\mathbf{base} \xrightarrow{\mathbf{loop}_x} \mathbf{base}$$

$$\frac{}{\mathbf{base}\ \mathbf{val}} \qquad \frac{}{\mathbf{loop}_x\ \mathbf{val}}$$

Cubical programs

Define a cubical programming language.

$$\mathbf{loop}_0 \doteq \mathbf{base} \xrightarrow{\mathbf{loop}_x} \mathbf{base} \doteq \mathbf{loop}_1$$

$$\frac{}{\mathbf{base}\ \mathbf{val}} \qquad \frac{}{\mathbf{loop}_x\ \mathbf{val}} \qquad \frac{}{\mathbf{loop}_0 \longmapsto \mathbf{base}} \qquad \frac{}{\mathbf{loop}_1 \longmapsto \mathbf{base}}$$

Cubical computational semantics

Build a model in which closed terms are regarded as programs.

- Define a cubical programming language.
- Types are interpreted as Cartesian cubical sets* of values.

Cubical computational semantics

Build a model in which closed terms are regarded as programs.

- Define a cubical programming language.
- Types are interpreted as Cartesian cubical sets* of values.

Can't consider only dimensionally-closed (0-dimensional) terms: then you wouldn't be able to tell $\mathbf{loop}_x$ and $\mathbf{base}$ apart!

Cubical computational semantics

Essentially, for every dimension context $\Psi = \{x, y, \dots\}$, a type specifies a PER of its $|\Psi|$-dimensional values.

- $[\![\mathbf{S}^1]\!]_\Psi(\mathbf{base}, \mathbf{base})$ for all Ψ,
- $[\![\mathbf{S}^1]\!]_{(\Psi,x)}(\mathbf{loop}_x, \mathbf{loop}_x)$ for all Ψ,
- (and compositions, inverses, $\dots$)

Functorial action is dimension substitution then evaluation:

$$\langle 0/x \rangle : \Psi \to (\Psi, x)$$
$$\langle 0/x \rangle : [\![\mathbf{S}^1]\!]_{(\Psi,x)} \to [\![\mathbf{S}^1]\!]_\Psi$$
$$(\mathbf{loop}_x)\langle 0/x \rangle = \mathbf{loop}_0 \Downarrow \mathbf{base}$$

For each type, must verify this is functorial (up to the PER)!

Cubical computational semantics

$$A \times B \xrightarrow{\mathbf{ua}_x(\cdots)} B \times A$$

$$[\![\mathbf{ua}(\cdots)]\!]_x$$

$$\begin{tikzcd} \{x\} & & \emptyset \arrow[ll, bend right, "\langle 0/x \rangle"'] \arrow[ll, bend left, "\langle 1/x \rangle"] \end{tikzcd}$$

Cubical computational semantics

$$A \times B \xrightarrow{\mathbf{ua}_x(\cdots)} B \times A$$

$$[\![\mathbf{ua}(\cdots)]\!]_x \xrightarrow{\langle 0/x \rangle} [\![A \times B]\!]_\emptyset$$

$$\begin{tikzcd} \{x\} & \emptyset \arrow[l, bend right, "\langle 0/x \rangle"'] \arrow[l, bend left, "\langle 1/x \rangle"] \end{tikzcd}$$

Cubical computational semantics

$$A \times B \xrightarrow{\mathbf{ua}_x(\cdots)} B \times A$$

$$\begin{tikzcd}[column sep=small]
 & & \llbracket A \times B \rrbracket_{\emptyset} \\
\llbracket \mathbf{ua}(\cdots) \rrbracket_x \arrow[urr, "\langle 0/x \rangle"] \arrow[drr, "\langle 1/x \rangle"'] & & \\
 & & \llbracket B \times A \rrbracket_{\emptyset}
\end{tikzcd}$$

$$\begin{tikzcd}
\{x\} & & \emptyset \arrow[ll, bend right, "\langle 0/x \rangle"'] \arrow[ll, bend left, "\langle 1/x \rangle"]
\end{tikzcd}$$

Cubical computational semantics

$$A \times B \xrightarrow{\mathbf{ua}_x(\cdots)} B \times A$$

Types are "dependent cubical sets" $(\mathbb{C}/\Psi)^{\mathbf{op}} \to \mathbf{Set}$.

$$\begin{tikzcd}
& {[\![A \times B]\!]_\emptyset} \\
{[\![\mathbf{ua}(\cdots)]\!]_x} \arrow[ur, "\langle 0/x \rangle"] \arrow[dr, "\langle 1/x \rangle"'] & \\
& {[\![B \times A]\!]_\emptyset}
\end{tikzcd}$$

$$\begin{tikzcd}
\{x\} & \emptyset \arrow[l, bend right, "\langle 0/x \rangle"'] \arrow[l, bend left, "\langle 1/x \rangle"]
\end{tikzcd}$$

Cubical computational semantics

Definition

$\tau : \mathbf{DimCtx} \to \mathbf{Val} \to \mathbf{Val} \to (\mathbf{Val} \to \mathbf{Val} \to \mathbf{Prop}) \to \mathbf{Prop}$ is a (semantic) cubical type system if it is functional, symmetric, transitive, PER-valued, and $\Psi \mapsto \{(A_0, B_0) \mid \tau(\Psi, A_0, B_0, \varphi)\}$ forms a cubical set.

$$A \doteq B \ \mathbf{type_{pre}} \ [\Psi]$$
$$M \doteq N \in A \ [\Psi]$$

Definition

$\tau \models (A \doteq B \ \mathbf{type_{pre}} \ [\Psi])$ when $\tau^{\Downarrow}(\Psi, A, B, \varphi)$???

Cubical computational semantics

Must be closed under both evaluation and dimension substitution.

$$\frac{M \in A\ [\Psi] \qquad M \Downarrow M_0}{M_0 \in A\ [\Psi]} \qquad\qquad \frac{M \in A\ [\Psi] \qquad \psi : \Psi' \to \Psi}{M\psi \in A\psi\ [\Psi']}$$

Must require that each instance of M evaluates to an element of $[\![A]\!]$, and coherently.

Cubical computational semantics

Definition

$\tau \models (A\ \mathbf{type_{pre}}\ [\Psi])$ when for all $\Psi_2 \xrightarrow{\psi_2} \Psi_1 \xrightarrow{\psi_1} \Psi$,
$A\psi_1 \Downarrow A_1$ and $\tau^{\Downarrow}(\Psi_2, A\psi_1\psi_2, A_1\psi_2, \varphi)$.

Let $[\![A]\!]_{\psi} := \varphi$ for each $\Psi' \xrightarrow{\psi} \Psi$, where $\tau^{\Downarrow}(\Psi', A\psi, A\psi, \varphi)$.

Definition

$\tau \models (M \in A\ [\Psi])$ when for all $\Psi_2 \xrightarrow{\psi_2} \Psi_1 \xrightarrow{\psi_1} \Psi$,
$M\psi_1 \Downarrow M_1$ and $[\![A]\!]_{\psi_1\psi_2}(M\psi_1\psi_2, M_1\psi_2)$.

Cubical computational semantics

Open judgments:

Definition
$a : A \gg B \doteq B'$ $\mathbf{type_{pre}}$ $[\Psi]$, presupposing A $\mathbf{type_{pre}}$ $[\Psi]$, when for any $\psi : \Psi' \to \Psi$ and $N \doteq N' \in A\psi$ $[\Psi']$,
$B\psi[N/a] \doteq B'\psi[N'/a]$ $\mathbf{type_{pre}}$ $[\Psi']$.

Definition
$a : A \gg M \doteq M' \in B$ $[\Psi]$, presupposing $a : A \gg B$ $\mathbf{type_{pre}}$ $[\Psi]$, when for any $\psi : \Psi' \to \Psi$ and $N \doteq N' \in A\psi$ $[\Psi']$,
$M\psi[N/a] \doteq M'\psi[N'/a] \in B\psi[N/a]$ $[\Psi']$.

Pi types

Many familiar principles hold at every dimension.

$$\frac{a : A \gg B\ \mathbf{type_{pre}}\ [\Psi]}{(a{:}A) \to B\ \mathbf{type_{pre}}\ [\Psi]} \qquad \frac{a : A \gg M \in B\ [\Psi]}{\lambda a.M \in (a{:}A) \to B\ [\Psi]}$$

$$\frac{M \in (a{:}A) \to B\ [\Psi] \qquad N \in A\ [\Psi]}{\mathbf{app}(M, N) \in B[N/a]\ [\Psi]}$$

$$\frac{a : A \gg M \in B\ [\Psi] \qquad N \in A\ [\Psi]}{\mathbf{app}(\lambda a.M, N) \doteq M[N/a] \in B[N/a]\ [\Psi]}$$

Path types

$$\frac{A\ \mathbf{type_{pre}}\ [\Psi, x] \qquad P_0 \in A\langle 0/x\rangle\ [\Psi] \qquad P_1 \in A\langle 1/x\rangle\ [\Psi]}{\mathbf{Path}_{x.A}(P_0, P_1)\ \mathbf{type_{pre}}\ [\Psi]}$$

$$\frac{M \in A\ [\Psi, x]}{\langle x\rangle M \in \mathbf{Path}_{x.A}(M\langle 0/x\rangle, M\langle 1/x\rangle)\ [\Psi]}$$

$$\frac{M \in \mathbf{Path}_{x.A}(P_0, P_1)\ [\Psi]}{M@r \in A\langle r/x\rangle\ [\Psi]} \qquad \frac{M \in \mathbf{Path}_{x.A}(P_0, P_1)\ [\Psi]}{M@\varepsilon \doteq P_\varepsilon \in A\langle \varepsilon/x\rangle\ [\Psi]}$$

$$\frac{M \in A\ [\Psi, x]}{(\langle x\rangle M)@r \doteq M\langle r/x\rangle \in A\langle r/x\rangle\ [\Psi]}$$

Path types

$$\frac{A\ \mathbf{type_{pre}}\ [\Psi, x] \qquad P_0 \in A\langle 0/x\rangle\ [\Psi] \qquad P_1 \in A\langle 1/x\rangle\ [\Psi]}{\mathbf{Path}_{x.A}(P_0, P_1)\ \mathbf{type_{pre}}\ [\Psi]}$$

$$\frac{M \in A\ [\Psi, x]}{\langle x\rangle M \in \mathbf{Path}_{x.A}(M\langle 0/x\rangle, M\langle 1/x\rangle)\ [\Psi]}$$

$$\frac{M \in \mathbf{Path}_{x.A}(P_0, P_1)\ [\Psi]}{M@r \in A\langle r/x\rangle\ [\Psi]} \qquad \frac{M \in \mathbf{Path}_{x.A}(P_0, P_1)\ [\Psi]}{M@\varepsilon \doteq P_\varepsilon \in A\langle \varepsilon/x\rangle\ [\Psi]}$$

$$\frac{M \in A\ [\Psi, x]}{(\langle x\rangle M)@r \doteq M\langle r/x\rangle \in A\langle r/x\rangle\ [\Psi]}$$

Path types

$$\frac{A\ \mathbf{type_{pre}}\ [\Psi, x] \qquad P_0 \in A\langle 0/x\rangle\ [\Psi] \qquad P_1 \in A\langle 1/x\rangle\ [\Psi]}{\mathbf{Path}_{x.A}(P_0, P_1)\ \mathbf{type_{pre}}\ [\Psi]}$$

$$\frac{M \in A\ [\Psi, x]}{\langle x\rangle M \in \mathbf{Path}_{x.A}(M\langle 0/x\rangle, M\langle 1/x\rangle)\ [\Psi]}$$

$$\frac{M \in \mathbf{Path}_{x.A}(P_0, P_1)\ [\Psi]}{M@r \in A\langle r/x\rangle\ [\Psi]} \qquad \frac{M \in \mathbf{Path}_{x.A}(P_0, P_1)\ [\Psi]}{M@\varepsilon \doteq P_\varepsilon \in A\langle \varepsilon/x\rangle\ [\Psi]}$$

$$\frac{M \in A\ [\Psi, x]}{(\langle x\rangle M)@r \doteq M\langle r/x\rangle \in A\langle r/x\rangle\ [\Psi]}$$

Path types

$$\frac{A\ \mathbf{type}_{\mathbf{pre}}\ [\Psi, x] \qquad P_0 \in A\langle 0/x\rangle\ [\Psi] \qquad P_1 \in A\langle 1/x\rangle\ [\Psi]}{\mathbf{Path}_{x.A}(P_0, P_1)\ \mathbf{type}_{\mathbf{pre}}\ [\Psi]}$$

$$\frac{M \in A\ [\Psi, x]}{\langle x\rangle M \in \mathbf{Path}_{x.A}(M\langle 0/x\rangle, M\langle 1/x\rangle)\ [\Psi]}$$

$$\frac{M \in \mathbf{Path}_{x.A}(P_0, P_1)\ [\Psi]}{M@r \in A\langle r/x\rangle\ [\Psi]} \qquad \frac{M \in \mathbf{Path}_{x.A}(P_0, P_1)\ [\Psi]}{M@\varepsilon \doteq P_\varepsilon \in A\langle \varepsilon/x\rangle\ [\Psi]}$$

$$\frac{M \in A\ [\Psi, x]}{(\langle x\rangle M)@r \doteq M\langle r/x\rangle \in A\langle r/x\rangle\ [\Psi]}$$

Path types

$$\frac{A\ \mathbf{type}_{\mathbf{pre}}\ [\Psi, x] \qquad P_0 \in A\langle 0/x\rangle\ [\Psi] \qquad P_1 \in A\langle 1/x\rangle\ [\Psi]}{\mathbf{Path}_{x.A}(P_0, P_1)\ \mathbf{type}_{\mathbf{pre}}\ [\Psi]}$$

$$\frac{M \in A\ [\Psi, x]}{\langle x\rangle M \in \mathbf{Path}_{x.A}(M\langle 0/x\rangle, M\langle 1/x\rangle)\ [\Psi]}$$

$$\frac{M \in \mathbf{Path}_{x.A}(P_0, P_1)\ [\Psi]}{M@r \in A\langle r/x\rangle\ [\Psi]} \qquad \frac{M \in \mathbf{Path}_{x.A}(P_0, P_1)\ [\Psi]}{M@\varepsilon \doteq P_\varepsilon \in A\langle \varepsilon/x\rangle\ [\Psi]}$$

$$\frac{M \in A\ [\Psi, x]}{(\langle x\rangle M)@r \doteq M\langle r/x\rangle \in A\langle r/x\rangle\ [\Psi]}$$

Exact equality types

Can define exact equality types, with equality reflection.

$$\frac{A\ \mathbf{type_{pre}}\ [\Psi] \qquad M \in A\ [\Psi] \qquad N \in A\ [\Psi]}{\mathbf{Eq}_A(M, N)\ \mathbf{type_{pre}}\ [\Psi]}$$

$$\frac{M \doteq N \in A\ [\Psi]}{\star \in \mathbf{Eq}_A(M, N)\ [\Psi]} \qquad \frac{E \in \mathbf{Eq}_A(M, N)\ [\Psi]}{M \doteq N \in A\ [\Psi]}$$

Univalence

Licata: univalence follows from $\mathbf{Equiv}(A, B) \to \mathbf{Path}_{\mathcal{U}}(A, B)$, provided transport applies the equivalence, up to a path.

$$
\begin{tikzcd}
M \arrow[d, mapsto, "F"'] \arrow[dr, dashed, "\mathbf{Vin}_x(M, N)"] & \\
\mathbf{app}(F, M) \arrow[r, "N"'] & N\langle 1/x \rangle
\end{tikzcd}
\quad \in \quad
\begin{tikzcd}
A \arrow[d, "F"'] \arrow[dr, dashed, "\mathbf{V}_x(A, B, \langle F, \_ \rangle)"] & \\
B\langle 0/x \rangle \arrow[r, "B"'] & B\langle 1/x \rangle
\end{tikzcd}
$$

(Special instance of CCHM “Glue” types.)

Kan operations

Speaking of transport...

Equip types with two Kan operations:

- Coercion (generalized transport)
- Homogeneous Kan composition (generalized box filling)

This is a structure, not a property, and must be stable under dimension substitution.

We have multiple universe hierarchies, $\mathcal{U}_i^{\mathsf{pre}}$ and $\mathcal{U}_i^{\mathsf{Kan}}$.

Coercion

$$\frac{A\ \mathbf{type}_{\mathbf{Kan}}\ [\Psi, x] \qquad M \in A\langle r/x\rangle\ [\Psi]}{\mathbf{coe}_{x.A}^{r\rightsquigarrow r'}(M) \in A\langle r'/x\rangle\ [\Psi]}$$

$$\begin{tikzcd} M \arrow[d, phantom, sloped, "\in"] \\ A\langle 0/x\rangle \arrow[rrr, "A"'] & & & A\langle 1/x\rangle \end{tikzcd}$$

Coercion

$$\frac{A\ \mathbf{type}_{\mathbf{Kan}}\ [\Psi, x] \qquad M \in A\langle r/x\rangle\ [\Psi]}{\mathbf{coe}_{x.A}^{r\rightsquigarrow r'}(M) \in A\langle r'/x\rangle\ [\Psi]}$$

$$\begin{tikzcd}
M \arrow[r, dashed] \arrow[d, phantom, "\in" sloped] & \mathbf{coe}_{x.A}^{\color{red}{0\rightsquigarrow 1}}(M) \arrow[d, phantom, "\in" sloped] \\
A\langle 0/x\rangle \arrow[r, "A"'] & A\langle 1/x\rangle
\end{tikzcd}$$

Coercion

$$\frac{A\ \mathbf{type_{Kan}}\ [\Psi, x] \qquad M \in A\langle r/x\rangle\ [\Psi]}{\mathbf{coe}_{x.A}^{r \rightsquigarrow r'}(M) \in A\langle r'/x\rangle\ [\Psi]}$$

$$\begin{tikzcd}
M \arrow[rr, "\mathbf{coe}_{x.A}^{0 \rightsquigarrow x}(M)"] & & \mathbf{coe}_{x.A}^{0 \rightsquigarrow 1}(M) \\
\rotatebox{90}{$\in$} & \rotatebox{90}{$\in$} & \rotatebox{90}{$\in$} \\
A\langle 0/x\rangle \arrow[rr, "A"'] & & A\langle 1/x\rangle
\end{tikzcd}$$

Coercion

$$\frac{A\ \mathbf{type}_{\mathbf{Kan}}\ [\Psi, x] \qquad M \in A\langle r/x\rangle\ [\Psi]}{\begin{array}{c}\mathbf{coe}_{x.A}^{r\rightsquigarrow r'}(M) \in A\langle r'/x\rangle\ [\Psi] \\ \mathbf{coe}_{x.A}^{r\rightsquigarrow r}(M) \doteq M \in A\langle r/x\rangle\ [\Psi]\end{array}}$$

$$\begin{tikzcd}
M \arrow[rr, "\mathbf{coe}_{x.A}^{0\rightsquigarrow x}(M)"] \arrow[d, phantom, "\rotatebox{-90}{$\in$}"] & \arrow[d, phantom, "\rotatebox{-90}{$\in$}"] & \mathbf{coe}_{x.A}^{0\rightsquigarrow 1}(M) \arrow[d, phantom, "\rotatebox{-90}{$\in$}"] \\
A\langle 0/x\rangle \arrow[rr, "A"'] & & A\langle 1/x\rangle
\end{tikzcd}$$

Coercion

Generalizes transport in a type family: if

$$B \in A \to \mathcal{U} \ [\Psi]$$
$$P \in \mathbf{Path}_{\_.A}(P_0, P_1) \ [\Psi]$$
$$M \in \mathbf{app}(B, P_0) \ [\Psi]$$

then

$$\mathbf{coe}^{0 \rightsquigarrow 1}_{x.\mathbf{app}(B, P@x)}(M) \in \mathbf{app}(B, P_1) \ [\Psi].$$

Homogeneous Kan composition

Homogeneous: the type remains constant, unlike in coercion.

$$\begin{tikzcd} {} \arrow[r, "x"] \arrow[d, "y"'] & {} \\ {} & \end{tikzcd} \qquad \begin{tikzcd} \cdot \arrow[r, "M"] \arrow[d, "N_0"'] & \cdot \arrow[d, "N_1"] \\ N_0\langle 1/y\rangle \arrow[r, dashed, "\mathbf{hcom}_A^{0\rightsquigarrow 1}(M;\, x=0 \hookrightarrow y.N_0,\, x=1 \hookrightarrow y.N_1)"'] & N_1\langle 1/y\rangle \end{tikzcd}$$

Given compatible faces of an (x, y)-square:

- at $y = 0$, M
- at $x = 0$, N_0
- at $x = 1$, N_1

we obtain the $y = 1$ face.

Homogeneous Kan composition

Homogeneous: the type remains constant, unlike in coercion.

$$
\begin{tikzcd}
\cdot \arrow[rrr, "M"] \arrow[d, "N_0"'] & & & \cdot \arrow[d, "N_1"] \\
N_0\langle 1/y\rangle \arrow[rrr, dashed, "\mathbf{hcom}_A^{0\rightsquigarrow 1}(M; x=0 \hookrightarrow y.N_0, x=1 \hookrightarrow y.N_1)"'] & & & N_1\langle 1/y\rangle
\end{tikzcd}
$$

Given compatible faces of an (x, y)-square:

- at $y = 0$, M
- at $x = 0$, N_0
- at $x = 1$, N_1

we obtain the $y = 1$ face.

Homogeneous Kan composition

Homogeneous: the type remains constant, unlike in coercion.

$$
\begin{tikzcd}
{} \arrow[r, "x"] \arrow[d, "y"'] & {} \\
{} &
\end{tikzcd}
\qquad
\begin{tikzcd}
\cdot \arrow[rr, "M"] \arrow[d, "N_0"'] & & \cdot \arrow[d, "N_1"] \\
N_0\langle 1/y\rangle \arrow[rr, dashed, "{\mathbf{hcom}_A^{0\rightsquigarrow 1}(M; x=0 \hookrightarrow y.N_0, x=1 \hookrightarrow y.N_1)}"'] & & N_1\langle 1/y\rangle
\end{tikzcd}
$$

Given compatible faces of an (x, y)-square:

- at $y = 0$, M
- at $x = 0$, N_0
- at $x = 1$, N_1

we obtain the $y = 1$ face.

Homogeneous Kan composition

Homogeneous: the type remains constant, unlike in coercion.

$$
\begin{tikzcd}
\cdot \arrow[r, "M"] \arrow[d, "N_0"'] & \cdot \arrow[d, "N_1"] \\
N_0\langle 1/y \rangle \arrow[r, dashed, "\mathbf{hcom}_A^{0 \rightsquigarrow 1}(M; x = 0 \hookrightarrow y.N_0, x = 1 \hookrightarrow y.N_1)"'] & N_1\langle 1/y \rangle
\end{tikzcd}
$$

Given compatible faces of an (x, y)-square:

- at $y = 0$, M
- at $x = 0$, N_0
- at $x = 1$, N_1

we obtain the $y = 1$ face.

Homogeneous Kan composition

Homogeneous: the type remains constant, unlike in coercion.

$$
\begin{tikzcd}
{} \arrow[r, "x"] \arrow[d, "y"'] & {} \\
{} & {}
\end{tikzcd}
\qquad
\begin{tikzcd}[column sep=huge]
\cdot \arrow[r, "M"] \arrow[d, "N_0"'] & \cdot \arrow[d, "N_1"] \\
N_0\langle 1/y\rangle \arrow[r, dashed, "\mathbf{hcom}_A^{0\rightsquigarrow 1}(M; x=0 \hookrightarrow y.N_0, x=1 \hookrightarrow y.N_1)"'] & N_1\langle 1/y\rangle
\end{tikzcd}
$$

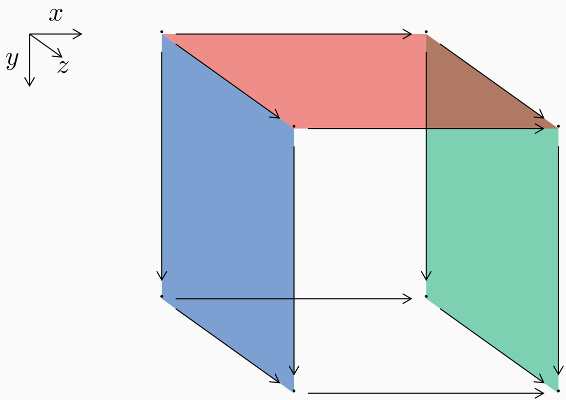
Given compatible faces of an (x, y)-square:

- at $y = 0$, M
- at $x = 0$, N_0
- at $x = 1$, N_1

we obtain the $y = 1$ face.

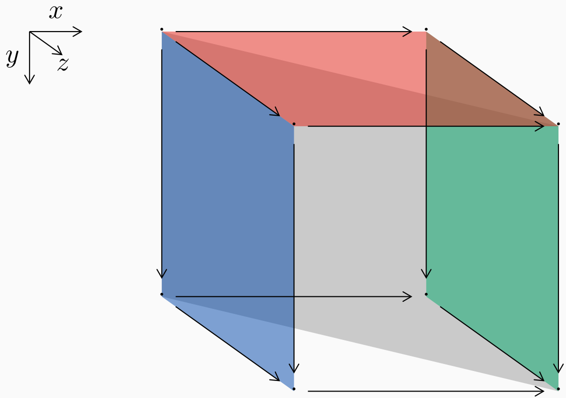
Homogeneous Kan composition

- Need not provide all $(2n-1)$ other sides of the n-cube.
- Can also attach along diagonal maps. (Crucial!)



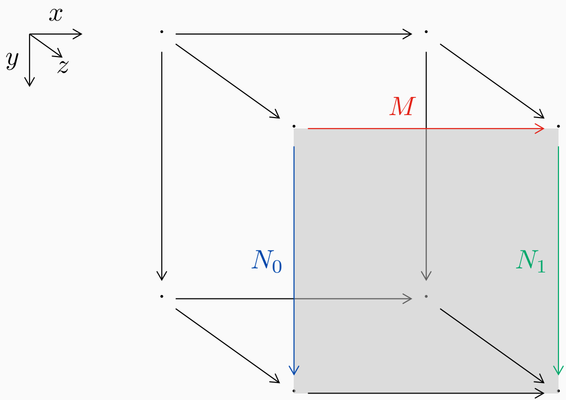
Homogeneous Kan composition

- Need not provide all $(2n-1)$ other sides of the n-cube.
- Can also attach along diagonal maps. (Crucial!)



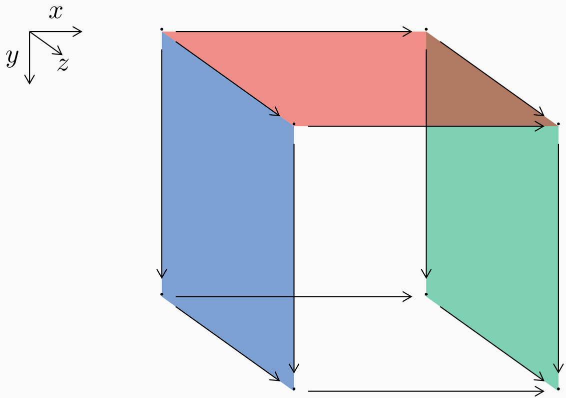
Homogeneous Kan composition

- Composing to a diagonal (y from $0 \rightsquigarrow z$) yields the filler.
- As with coercion, composition $r \rightsquigarrow r$ must be identity.



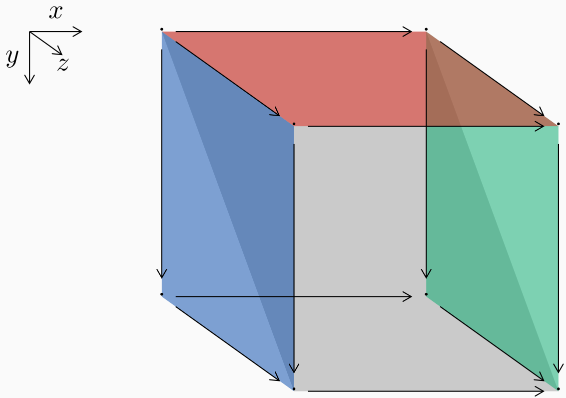
Homogeneous Kan composition

- Composing to a diagonal (y from $0 \rightsquigarrow z$) yields the filler.
- As with coercion, composition $r \rightsquigarrow r$ must be identity.



Homogeneous Kan composition

- Composing to a diagonal (y from $0 \rightsquigarrow z$) yields the filler.
- As with coercion, composition $r \rightsquigarrow r$ must be identity.



Kan operations

Implement at every type, using operations of constituent types.

$$\mathbf{hcom}^{r \rightsquigarrow r'}_{(a:A)\to B}(M; \overrightarrow{\xi_i \hookrightarrow y.N_i}) \longmapsto$$

$$\lambda a.\mathbf{hcom}^{r \rightsquigarrow r'}_{B}(\mathbf{app}(M,a); \overrightarrow{\xi_i \hookrightarrow y.\mathbf{app}(N_i,a)})$$

$$\mathbf{coe}^{r \rightsquigarrow r'}_{x.(a:A)\to B}(M) \longmapsto$$

$$\lambda a.\mathbf{coe}^{r \rightsquigarrow r'}_{x.B[\mathbf{coe}^{r' \rightsquigarrow x}_{x.A}(a)/a]}(\mathbf{app}(M, \mathbf{coe}^{r' \rightsquigarrow r}_{x.A}(a)))$$

$$\mathbf{hcom}^{r \rightsquigarrow r'}_{\mathbf{Path}_{x.A}(P_0,P_1)}(M; \overrightarrow{\xi_i \hookrightarrow y.N_i}) \longmapsto$$

$$\langle x\rangle \mathbf{hcom}^{r \rightsquigarrow r'}_{A}(M@x; \overrightarrow{x = \varepsilon \hookrightarrow \_.P_\varepsilon}, \overrightarrow{\xi_i \hookrightarrow y.N_i@x})$$

$$\mathbf{coe}^{r \rightsquigarrow r'}_{y.\mathbf{Path}_{x.A}(P_0,P_1)}(M) \longmapsto$$

$$\langle x\rangle \mathbf{com}^{r \rightsquigarrow r'}_{y.A}(M@x; \overrightarrow{x = \varepsilon \hookrightarrow y.P_\varepsilon})$$

Kan operations

Equip HITs and universe with free Kan composition structure.

What are the elements and Kan operations of compositions of types? (Very involved.)

$$
\begin{tikzcd}
\mathbf{coe}^{1 \rightsquigarrow 0}_{y.B_0}(N_0) \arrow[r, "M"] \arrow[d, dashed] & \mathbf{coe}^{1 \rightsquigarrow 0}_{y.B_1}(N_1) \arrow[d, dashed] \\
N_0 \arrow[r, dashed] & N_1
\end{tikzcd}
\quad \in \quad
\begin{tikzcd}
\cdot \arrow[r, "A"] \arrow[d, "B_0"'] & \cdot \arrow[d, "B_1"] \\
B_0\langle 1/y \rangle \arrow[r, dashed] & B_1\langle 1/y \rangle
\end{tikzcd}
$$

Kan operations

This is where the "diagonal cofibrations" are needed.

$$\mathbf{hcom}^{r \rightsquigarrow r'}_{\mathbf{hcom}^{s \rightsquigarrow s'}_{\mathcal{U}}(A;\cdots)}(M;\cdots) \longmapsto$$
$$\cdots \mathbf{hcom}^{s \rightsquigarrow s'}_{A}(\cdots ; r = r' \hookrightarrow \cdots) \cdots$$

Need $\mathbf{hcom}^{r \rightsquigarrow r'}_{A}(M;\cdots)$ when $s = s'$, and M when $r = r'$.

Kan operations

Weak $\mathbf{J}$ can be defined for the $\mathbf{Path}$ type using $\mathbf{hcom}$ and $\mathbf{coe}$.

Separately, an $\mathbf{Id}$ indexed higher inductive type, generated by $\mathbf{refl}$, satisfies strict $\mathbf{J}$. (Cavallo, Harper, arXiv:1801.01568)

Summary

A cubical type theory, based on Cartesian (not De Morgan) cubes, whose terms are programs, satisfying canonicity.

- Cartesian cubes suffice!
- A computational model of Book HoTT.
- A "two-level" type theory (à la HTS) with both paths and exact equality. Some equality types are fibrant!

Implementations

By design, suitable for implementation! Two in progress:

- RedPRL proof assistant (`redprl.org`)
  - Proofs of full univalence, J, groupoid laws...
  - Definition of semi-simplicial types
- `yacctt` type-checker (Angiuli, Mörtberg)

References

Angiuli, Favonia, Harper. *Cartesian Cubical Computational Type Theory: A Constructive Formulation of Two-Level Type Theory*. Preprint.

Angiuli, Favonia, Harper. *Computational Higher Type Theory III: Univalent Universes and Exact Equality*. arXiv:1712.01800.

Angiuli, Brunerie, Coquand, Favonia, Harper, Licata. *Cartesian Cubical Type Theory*. Preprint.

`http://www.cs.cmu.edu/~cangiuli/`