Implementing a Variation of the Cubical Set Model

Simon Huber
(j.w.w. Cyril Cohen, Thierry Coquand, Anders Mörtberg)

University of Gothenburg

ProgLog Seminar, 26. November 2014

Outline

1. Yet another variant of cubical sets
2. `cubical`: a type checker
3. Demo!

Cubical Sets: Intuition

Consider a topological space X. A (topological) n-cube in X is a continuous map $[0,1]^n \to X$.
For I a finite set, let $X(I)$ be the continuous maps

$$[0,1]^I \to X$$

where the elements in I are called names/dimensions.

We want to represent this combinatorially!

Cubical Sets

Let $\mathrm{D}(I)$ denote the free De Morgan algebra with generators I. "De Morgan algebra = Boolean algebra minus $\varphi \vee \neg\varphi = 1$ and $\varphi \wedge \neg\varphi = 0$"

Let $\square$ be the category with

- objects: finite sets of names $I, J, K, \ldots$, and
- morphisms $I \to J$ given by maps $I \to \mathrm{D}(J)$;

A *cubical set* X is a presheaf on $\square^{\mathrm{op}}$, i.e., $X : \square \to \mathbf{Set}$.

Concretely: X is given by sets $X(I)$ and maps

$$\begin{aligned} X(I) &\to X(J) \\ u &\mapsto u\,f \end{aligned} \qquad \text{for } f : I \to J$$

such that $(u\,f)g = u(f;g)$ and $u\,\mathbf{1} = u$.

Cubical Sets

Intuition: $u \in X(I)$ an element depending on names I, and $u\,f$ as performing the substitution $f : I \to J$.

E.g., $u = u(i,j)$ in $X(i,j)$ and $f = (i \mapsto 0, j \mapsto j \wedge k)$, then
uf is $u(0, j \wedge k)$ in $X(j,k)$.

Think of an element $u = u(i,j)$ in $X(i,j)$ as formally representing a map:

$$[0,1]^{\{i,j\}} \to X$$

$i \wedge j$ corresponds to $\min(i,j)$, $i \vee j$ to $\max(i,j)$, and $\neg i$ to $1 - i$.

Faces and Degeneracies

- For i in I there are two maps $(i0), (i1) \colon I \to I - i$, sending i to 0 (or 1). For u in $X(I)$, $u(i0)$ and $u(i1)$ are *faces* of u

$$u(i0) \xrightarrow{\quad u \quad} u(i1)$$

- For j *not* in I, the inclusion $\iota_j \colon I \to I, j$ induces *degeneracies*.

Connections, Diagonals, and Symmetry

- For $u = u(i)$ in $X(i)$, the cube $u(i \wedge j)$ connects $u(0)$ to $u(i)$

$$\begin{tikzcd} u(0) \arrow[r, "u(i)"] & u(1) \\ u(0) \arrow[u, "u(0)"] \arrow[r, "u(0)"] \arrow[ur, phantom, "u(i \wedge j)"] & u(0) \arrow[u, "u(j)"'] \end{tikzcd}$$

- For $u = u(i, j)$ in $X(i, j)$, $u(k, k)$ is the diagonal of u, connecting $u(0, 0)$ to $u(1, 1)$.
- (For $u = u(i)$ in $X(i)$, $u(\neg i)$ connects $u(1)$ to $u(0)$.)

Path Space

For a cubical set X the path space $\texttt{Path}_X$ given by:

- $\langle i \rangle w \in (\texttt{Path}_X)(I)$ where $i \notin I$ and $w \in X(I, i)$; i is bound in $\langle i \rangle w$;
- $(\langle i \rangle w)f = \langle j \rangle\, wf'$ for $f : I \to J$, and $f' = (f, i = j) \colon I, i \to J, j$, j fresh.

$\texttt{Path}_X$ is $X^{\mathbb{I}}$ where $\mathbb{I}(J) = \mathrm{D}(J)$ the interval.

Fixing the endpoints gives an interpretation for $\texttt{Id}_X(u, v)$.

Kan Operations

To get the interpretation we need to require composition operations: for

$$\begin{aligned} &u_{jb} \text{ in } X(I-j) \\ &u_{i0} \text{ in } X(I-i) \\ \text{s.t. } &u_{jb}(i0) = u_{i0}(jb) \end{aligned}$$

we require

$$u_{i1} = \mathrm{comp}^{i}_{\vec{u}}(u_{i0}) \in X(I-i)$$

where $\vec{u}, u_{i0}$ specifies an "open box" and u_{i1} is its "lid".

Additionally, we need that if $\vec{u}$ is constant (i.e., degenerate) along the direction i, then $u_{i0} = u_{i1}$.
Moreover: $(\mathrm{comp}^{i}_{\vec{u}}(u_{i0}))f = \mathrm{comp}^{j}_{\vec{u}f'}(u_{i0}f)$ (uniformity conditions)

Kan Fillings

Fillings can be derived using connections:

$$\mathsf{fill}^i_{\vec{u}}(u_{i0}) \in X(I)$$
$$\mathsf{fill}^i_{\vec{u}}(u_{i0}) = \mathsf{comp}^j_{\vec{u}(i=i\wedge j)}(u_{i0}) \qquad (j \text{ fresh})$$

We get a model of type theory with $\Pi, \Sigma, \mathtt{Id}, U$ satisfying univalence.

Overview of Cubical

- Proof assistant based on the cubical set model.
- The Univalence Axiom and functional extensionality are available and compute!
- No indexed-families, but recursive definitions, and `Id`-types are a primitive notion
- Supports Higher Inductive Types (experimental)
- Available at https://github.com/simhu/cubical (branch connections_hsplit)

Terms

$$
\begin{aligned}
r, s, t, A, F ::= {} & x \mid r\,s \mid \lambda x\, t \mid \Pi A F \mid U \\
& \mid c\vec{t} \mid \mathtt{sum}\{c(\vec{x} : \vec{A}) | \dots\} \mid \mathtt{split}\{c\vec{x} \to t; \dots\} \\
& \mid t\, \mathtt{where}\, \vec{x} : \vec{A} = \vec{t} \\
& \dots \\
& \mid \mathsf{PN}
\end{aligned}
$$

Primitive notions

$$PN ::= \mathtt{Id} \mid \mathtt{refl} \mid \mathtt{Ext} \mid \mathtt{TransU} \mid \mathtt{IsoId} \mid \dots$$

Values

$$
\begin{aligned}
u, v, w ::= {} & t\rho \mid u\, v \mid \mathtt{Id}_u(v, w) \mid \Pi\, u\, v \mid U \\
& \mid c\vec{u} \\
& \mid \langle i \rangle u \mid \mathsf{comp}^i_{w,\vec{u}}(u_{i0}) \\
& \mid \mathtt{ext}\ v_0\ v_1\ w\ \varphi \\
& \mid x \mid u\, v \mid u\, \varphi \mid \ldots \qquad \text{(neutral values)} \\
& \ldots
\end{aligned}
$$

On values we can define actions of cubical sets $u(i = i \wedge j)$, $u(i0)$, etc. Any value u depends only on a finite set of names $\mathsf{supp}(u)$.

Evaluation and Operational Semantics

$t\rho$ for $\texttt{eval}\,\rho\,t$

$$
\begin{aligned}
(\texttt{refl}\,A t)\rho &= \langle i\rangle\; t\rho\\
(\texttt{Ext}\,f\,g\,p)\rho &= \langle i\rangle\;\texttt{ext}\,(f\rho)\,(g\rho)\,(p\rho)\,i\\
(\texttt{TransU}_{A,B}\,p\,a)\rho &= \mathsf{comp}^{i}_{(p\rho)\,i}(a\rho)\\
&\;\;\vdots
\end{aligned}
$$

Operational semantics:

$$
\begin{aligned}
(\lambda x t)\rho\,u &\to t(\rho, x = u)\\
\texttt{ext}\,u_0\,u_1\,v\,0 &\to u_0\\
\texttt{ext}\,u_0\,u_1\,v\,1 &\to u_1\\
&\ldots
\end{aligned}
$$

$\mathsf{comp}^{i}_{w,\vec{u}}(u_{i0})$ is explained depending on the shape of w

Structure

Bidirectional type checking:

$\rho, \Gamma \vdash t \uparrow v$	type checking
$\rho, \Gamma \vdash t \downarrow v$	type inference

where: t is a term, v is a value (ρ an environment, Γ assigns values to the types of variables)

$$\frac{\rho, \Gamma \vdash t \downarrow v' \quad v' \equiv v}{\rho, \Gamma \vdash t \uparrow v}$$

($v \equiv v'$ checks conversion of *values*)

Extension: Higher Inductive Types

Example: the circle S^1

```
S1 : U
hdata S1 = base
         | loop @ base ~ base

loop' : Id S1 base base
loop' = loop
```

HITs

Two types of constructors:

- object constructors (e.g., `base`)

$$\mathtt{c}\,(x_1 : A_1) \ldots (x_n : A_n)$$

- path constructors (here `loop`);

$$\mathtt{pc}\,(x_1 : A_1) \ldots (x_n : A_n)\ \mathtt{@}\ e_1 \sim e_2$$

  e_1 and e_2 are the end-points (whose type has to be the data type we are defining)
- no path constructors for higher identities

HITs: hsplit

```
S1rec : (F : S1 -> U) (b : F base)
        (l : IdS S1 F base base loop b b)
        (x : S1) -> F x
S1rec F b l = hsplit F with
  base -> b
  loop -> l
```

IdS is heterogeneous equality:

```
IdS : (A : U) (F : A -> U) (a0 a1 : A)
      (p : Id A a0 a1) ->
      F a0 -> F a1 -> U
```