Computational Higher Type Theory (CHTT)

Robert Harper

Lecture Notes of Week 16 by Paul Gölz

1 Defining Kan composition hcom_A for various types A

1.1 The strict Booleans Bool

May 1, 2018

The strict Booleans serve as the type of observables in our system, and they will be used to prove canonicity. They are defined as follows:

$$\mathsf{Bool}\ \mathsf{val}_\Psi \qquad \mathsf{Bool} \doteq \mathsf{Bool}\ \mathsf{type}\ [\Psi]$$

$$\mathsf{tt}\ \mathsf{val}_\Psi \qquad \mathsf{ff}\ \mathsf{val}_\Psi \qquad \mathsf{tt} \doteq \mathsf{tt} \in \mathsf{Bool}\ [\Psi] \qquad \mathsf{ff} \doteq \mathsf{ff} \in \mathsf{Bool}\ [\Psi]$$

$$\frac{M \mapsto_\Psi M'}{\mathsf{if}(M; M_0; M_1) \mapsto_\Psi \mathsf{if}(M'; M_0; M_1)} \qquad \frac{}{\mathsf{if}(\mathsf{tt}; M_0; M_1) \mapsto_\Psi M_0} \qquad \frac{}{\mathsf{if}(\mathsf{ff}; M_0; M_1) \mapsto_\Psi M_1}$$

All lines in Bool are degenerate; each strict Boolean is tt or ff. Therefore, both coercion and composition are trivial:

$$\mathsf{coe}^{r \rightsquigarrow r'}_{\_.\mathsf{Bool}}(M) \mapsto M \qquad \mathsf{hcom}^{r \rightsquigarrow r'}_{Bool}(M; \overrightarrow{\xi_i \hookrightarrow y.\ N_i}) \mapsto M$$

One can check that this defines a valid Kan structure.

1.2 The weak Booleans wBool

For the strict Booleans, we manually encoded the insight that there are only two Booleans. We will now explore the weak Booleans, that are defined as the equivalent to an inductive type, namely as the free (think: "least") Kan type generated by tt and ff. We begin as above:

$$\mathsf{wBool}\ \mathsf{val}_\Psi \qquad \mathsf{wBool} \doteq \mathsf{wBool}\ \mathsf{type}\ [\Psi]$$

$$\mathsf{tt}\ \mathsf{val}_\Psi \qquad \mathsf{ff}\ \mathsf{val}_\Psi \qquad \mathsf{tt} \doteq \mathsf{tt} \in \mathsf{wBool}\ [\Psi] \qquad \mathsf{ff} \doteq \mathsf{ff} \in \mathsf{wBool}\ [\Psi]$$

However, instead of our specialized composition above, we simply declare hcom's to be values:[1]

$$\frac{\text{no } \xi_i \text{ is true} \qquad r \neq r'}{\mathsf{hcom}^{r \rightsquigarrow r'}_{\mathsf{wBool}}(M; \overrightarrow{\xi_i \hookrightarrow y.\ N_i})\ \mathsf{val}_\Psi}$$

As a result, we now have infinitely many Booleans! In particular, this poses the question of how if should be evaluated if its first argument is an hcom: Let us say that we have $M_0 \in A[\mathsf{tt}/a]\ [\Psi]$, $M_1 \in A[\mathsf{ff}/a]\ [\Psi]$ and $a : \mathsf{wBool} \gg A\ \mathsf{type}$; thus,

$$\mathsf{if}_{a.A}(\mathsf{hcom}^{r \rightsquigarrow r'}_{\mathsf{wBool}}(M; \overrightarrow{\xi_i \hookrightarrow y.\ N_i}); M_0; M_1) \in A[\mathsf{hcom}^{r \rightsquigarrow r'}_{\mathsf{wBool}}(M; \overrightarrow{\xi_i \hookrightarrow y.\ N_i})/a]\ [\Psi].$$

But where should this expression step? Even though this problem is reminiscent of the problems that prevent HoTT from having computational meaning, our problem can be solved. The main idea is to "push the composition out" of the conditional.

[1]For technical reasons, the hcom actually steps to an fcom value as seen last week. We will ignore this complication for pedagogical reasons.

Set $M' := \mathsf{if}_{a.A}(M; M_0; M_1) \in A[M]$ and $N_i := \mathsf{if}_{a.A}(N_i; M_0; M_1) \in A[N_i]$. We might want to compose these terms as $\mathsf{hcom}^{r\rightsquigarrow r'}_{?}(M'; \overrightarrow{\xi_i \hookrightarrow y.\ N_i'})$, but it is not clear which type we should choose for ?. The resulting type should be $A[\mathsf{hcom}^{r\rightsquigarrow r'}_{\mathsf{wBool}}(M; \overrightarrow{\xi_i \hookrightarrow y.\ N_i})/a]$, but we cannot use this for ? because then the types of M' and N_i' would not match. The solution is to use a heterogeneous composition, i.e.,

$$\mathsf{if}_{a.A}(\mathsf{hcom}^{r\rightsquigarrow r'}_{\mathsf{wBool}}(M; \overrightarrow{\xi_i \hookrightarrow y.\ N_i}); M_0; M_1) \mapsto \mathsf{com}^{r\rightsquigarrow r'}_{z.A[zline/a]}(M'; \overrightarrow{\xi_i \hookrightarrow y.\ N_i'}).$$

It remains to choose *zline* such that $(zline)$ equals M at $z = r$ and equals $\mathsf{hcom}^{r\rightsquigarrow r'}_{\mathsf{wBool}}(M; \overrightarrow{\xi_i \hookrightarrow y.\ N_i})$ at $z = r'$. Choosing *zline* as $\mathsf{hcom}_{\mathsf{wBool}^{r\rightsquigarrow z}}(M; \overrightarrow{\xi_i \hookrightarrow y.\ N_i})$ works.

It is worth pointing out that the motive $a.A$ of the conditional cannot be erased for evaluation: If we unfold the definition of com, we rely on coercions that are governed by the type family A.

1.3 Pi types

For Pi types, defining hcom works without complications:

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

1.4 Sigma types

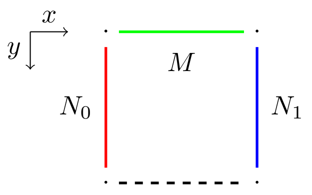
For Sigma types, dependency slightly complicates the picture, by requiring a heterogeneous composition for the second component:

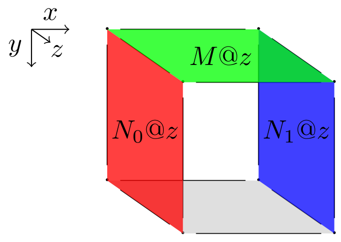
$$\begin{aligned}
&\mathsf{hcom}^{r\rightsquigarrow r'}_{a:A\times B(a)}(M; \overrightarrow{\xi_i \hookrightarrow y.\ N_i}) \\
&\qquad \mapsto \langle \mathsf{hcom}^{r\rightsquigarrow r'}_{A}(\mathsf{fst}(M); \overrightarrow{\xi_i \hookrightarrow y.\ \mathsf{fst}(N_i)}), \\
&\qquad\qquad \mathsf{com}^{r\rightsquigarrow r'}_{z.B[\mathsf{hcom}^{r\rightsquigarrow z}_{A}(\mathsf{fst}(M); \overrightarrow{\xi_i \hookrightarrow y.\ \mathsf{fst}(N_i)})/a]}(\mathsf{snd}(M); \overrightarrow{\xi_i \hookrightarrow y.\ \mathsf{snd}(N_i)})\rangle
\end{aligned}$$

1.5 Path types

May 3, 2018

Where should $\mathsf{hcom}^{r\rightsquigarrow r'}_{\mathsf{Path}[z.A](P_0;P_1)}(M; \overrightarrow{\xi_i \hookrightarrow y.\ N_i})$ step to? M, N_0 and N_1 are quoted lines and we can orient them in some direction z, as illustrated below for the two-dimensional case:





It suffices to do the composition on the unquoted lines and then quote the result:

$$\begin{aligned}
\mathsf{hcom}^{r\rightsquigarrow r'}_{\mathsf{Path}[z.A](P_0;P_1)}(M; \overrightarrow{\xi_i \hookrightarrow y.\ N_i}) \mapsto \langle z.\ \mathsf{hcom}_A(M@z;\ & z = 0 \hookrightarrow \_.\ P_0, \\
& z = 1 \hookrightarrow \_.\ P_1, \\
& \xi_i \hookrightarrow y.\ N_i@z)\rangle
\end{aligned}$$

2 The lay of the land

So far, we have looked at cubical types: We saw how they must behave in different dimensions and that they furthermore have to satisfy the Kan conditions (coercion and composition). Now, we would like to look at the multiverse of types[2], which itself has a cubical structure. In particular, the multiverse has a form of Kan composition: if A and B_i are cubes of types, $\mathsf{HCOM}^{r \rightsquigarrow r'}_{(multiverse)}(A; \overrightarrow{\xi_i \hookrightarrow y.\ B_i})$ is itself a cube of types.

Considered at dimension 0, this gives us our usual types. But even at higher dimensions, the HCOM's can be given a Kan type structure by defining appropriate $\mathsf{coe}^{r \rightsquigarrow r'}_{x.\mathsf{HCOM}}(A)$ and $\mathsf{hcom}^{r \rightsquigarrow r'}_{\mathsf{HCOM}}(H; \overrightarrow{\xi_i \hookrightarrow y.\ H_i})$. Especially the composition of compositions is challenging. While we will not present the solution here, the key idea is to introduce constraints ξ_i of the shape $y = y'$, representing diagonals. Whereas part II of the Computational Higher Type Theory papers [Angiuli and Harper, 2016] did not yet incorporate these constraints, they appear in part III [Angiuli et al., 2017].

Note that dimensions in the multiverse are not related to the indexing of universes in HoTT ($\mathcal{U}_0 \subseteq \mathcal{U}_1 \subseteq \dots$). We can choose our multiverse to contain the entire hierarchy of universes at dimension 0, but this is not essential for our considerations and we could choose differently.

3 Univalence

We would like to incorporate univalence into our type theory, obviously without giving up its computational meaning. We do so by populating the higher-dimensional structure of the multiverse with lines that correspond to (or: are given by) equivalences.

3.1 Equivalences

Recall the definition of an equivalence from Week 11:

Definition 1 (Equivalences)**.** *Let A and B be types. The equivalences between A and B are functions from A to B whose fibers are contractible:*

$$\mathsf{Equiv}(A, B) :\equiv (f : A \to B) \times (b : B \to \mathsf{isContractible}(f^{-1}(b))),$$

where $f^{-1}(b)$ is short for $a : A \times \mathsf{Path}_B(f(a), b)$ and

$$\mathsf{isContractible}(\mathsf{C}) :\equiv \mathsf{c} : \mathsf{C} \times (\mathsf{c}' : \mathsf{C} \to \mathsf{Path}(\mathsf{c}, \mathsf{c}')).$$

3.2 V-types

Let A type $[\Psi \mid r = 0]$ ("a valid type on the left"), B type $[\Psi]$ ("valid throughout") and $E \in \mathsf{Equiv}(A, B)$ $[\Psi \mid r = 0]$. Then, we introduce

$$V_r(A, B, E)\ \mathsf{type}\ [\Psi].$$

This V-type completes the following diagram:

$$\begin{tikzcd}
A_0 \arrow[d, "E"', "\wr"] \arrow[dr, dashed, no head, "V_x(A,B,E)"] & \\
B_0 \arrow[r, no head, "B_x"] & B_1
\end{tikzcd}$$

[2]The multiverse cannot by a type, as the contrary would give rise to Russel's paradox.

Note that $V_x(A, B, E)$ is a line of types. Let us see how the V-type is evaluated:

$$V_0(A_0, \_, \_) \mapsto A_0 \qquad V_1(\_, B_1, \_) \mapsto B_1$$

In particular,

$$V_x(A_x, B_x, E_x)\langle 0/x\rangle = V_0(A_0, \_, \_) \mapsto A_0 \qquad V_x(A_x, B_x, E_x)\langle 1/x\rangle \mapsto B_1.$$

How does this relate to univalence? Consider the case where B is a degeneracy. Then, $V_x(A, B, E)$ packages E composed with reflexivity, thus the equivalence, as a line. This line allows us to coerce based on the equivalence. But the V-type does even more: It converts lines!

What are the elements of the V_r's? If $M \in A\ [\Psi \mid r = 0]$ and $\mathsf{fst}(E)(M) \doteq N \in B\ [\Psi \mid r = 0]$, then

$$\mathsf{Vin}_r(M, N) \in V_r(A, B, E).$$

$$\begin{tikzcd}
M_0 \arrow[d, mapsto, "\mathsf{fst}(E)"'] \arrow[dr, dashed, dash, "\mathsf{Vin}_x(M,N)"] & \\
N_0 \arrow[r, dash, "N_x"] & N_1
\end{tikzcd}
\qquad
\begin{tikzcd}
A_0 \arrow[d, "E"', "\wr"] \arrow[dr, dashed, dash, "V_r(A,B,E)"] & \\
B_0 \arrow[r, dash, "B_x"] & B_1
\end{tikzcd}$$

Correspondingly,

$$\mathsf{Vin}_0(M, N) \mapsto M \qquad \mathsf{Vin}_1(M, N) \mapsto N.$$

When $V \in V_r(A, B, E)$ and F represents the equivalence ("$\mathsf{fst}(E)$"), the elimination form $\mathsf{Vproj}(V, F)$ recovers N.

Coercion along $V_x(A, B, E))$ from 0 to 1 can easily be implemented by applying the equivalence, then coercing along B. Coercing from 1 to 0 is a little trickier since it involves some fiddling to reverse E. Coercions of the form $\mathsf{coe}_{x.V_x}^{y \rightsquigarrow r'}(-)$ are very tricky and involve diagonal constraints.

While we do not have time to cover it, composition is particularly interesting and elegant.

4 Hacking with cubical types

There are three implementations of cubical type theory that one might want to experiment with:

cubicaltt: Focuses on evaluation, developed in Sweden. `https://github.com/mortberg/cubicaltt`.

yacctt: Focuses on evaluation.

redprl: An interactive proof editor in the tradition of Nuprl, developed at CMU. `http://www.redprl.org/`, `https://github.com/RedPRL/sml-redprl`.

References

Carlo Angiuli and Robert Harper. Computational higher type theory II: Dependent cubical realizability. `https://arxiv.org/abs/1606.09638v2`, 2016.

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`, 2017.