

A COOL CARTESIAN CUBICAL TYPE THEORY

CARLO ANGIULI, EVAN CAVALLO, KUEN-BANG HOU (FAVONIA), AND JONATHAN STERLING

We present a semantic version of Cartesian cubical type theory, previously encoded in syntactic form by Angiuli [Ang19], corresponding to an idealized version of `cooltt` [Red20]. By *semantic* we mean that we use ordinary families and functions (as in [OP16]) instead of syntactic substitutions. We differ from the Orton–Pitts-style developments [OP16; Ang+19] in the following ways:

- (1) The developments in the style of Orton–Pitts construct various notions in the internal language of a topos, but do not always carefully check that they may be combined in a coherent way, nor whether they are expressible purely in the language of finite limits (a pre-requisite for defining a type theory). In contrast, we are defining an equational theory for a *general* coercion and composition operator, which can be *modeled* by a carefully executed Orton–Pitts construction.
- (2) The artificial distinction between “cap” and “tube” maintained in the Orton–Pitts developments is replaced by the more semantic use of disjunction. This significantly simplifies both the rules for typing compositions, and for computing them.
- (3) Finally, we are experimentally targeting *weak* universes à la Tarski instead of strict ones (i.e. we do not place equations on the $\text{el}(-)$ operator except those induced by cubical boundary conditions); this is both more natural from a semantic point of view, and apparently advantageous from the side of implementation. The strict equations lose too much information too early, such as the Kan-ness and universe levels. For example, it is difficult to recognize

$$\prod_{i:\mathbb{I}} \text{Sub}(\text{el}(\hat{A}(i)); \partial i; [i = 0 \rightarrow M \mid i = 1 \rightarrow N])$$

as a path type “ $\text{Path}_A(M; N)$ ” and infer its Kan operators. (It will be next to impossible with extension types or other more sophisticated type formers.)

1. Cubical structure

We leave implicit the basic rules of type theory, including π and σ types equipped with η rules. As in [OP16; Ang+19], we expose the cubical nature of our setting through \mathbb{I} , the Yoneda embedding of the interval, and \mathbb{F} , a universe of strict propositions. (Note that A *type* means that A is a *cubical set*, not that A is Kan.)

$\frac{}{\mathbb{I} \text{ type}} \quad \text{\texttt{I-FORMATION}}$	$\frac{}{0, 1 : \mathbb{I}} \quad \text{\texttt{I-INTRO}}$	$\frac{}{\mathbb{F} \text{ type}} \quad \text{\texttt{F-FORMATION}}$
$\frac{r, s : \mathbb{I}}{r = s : \mathbb{F}} \quad \text{\texttt{COFIBRATION INTRO}}$	$\frac{\phi, \psi : \mathbb{F}}{\phi \vee \psi : \mathbb{F}}$	$\frac{\phi, \psi : \mathbb{F}}{\phi \wedge \psi : \mathbb{F}} \quad \frac{\phi : \mathbb{I} \rightarrow \mathbb{F}}{\forall i. \phi(i) : \mathbb{F}}$


$$\begin{array}{c}
\text{PRF FORMATION} \\
\frac{\phi : \mathbb{F}}{[\phi] \text{ type}} \\
\\
\text{PRF INTRO} \\
\frac{\phi : \mathbb{F} \quad \phi \text{ true}}{* : [\phi]} \\
\\
\text{PRF ELIM} \\
\frac{\phi : \mathbb{F} \quad p : [\phi]}{\phi \text{ true}} \\
\\
\text{PRF UNIQUENESS} \\
\frac{\phi : \mathbb{F} \quad p : [\phi]}{p = * : [\phi]} \\
\\
\text{PRF EXTENSIONALITY} \\
\frac{\phi, \psi : \mathbb{F} \quad \phi \text{ true} \vdash \psi \text{ true} \quad \psi \text{ true} \vdash \phi \text{ true}}{\phi = \psi : \mathbb{F}}
\end{array}$$

Truth of each $\phi : \mathbb{F}$ has the expected meaning. In particular:

- (1) under a hypothesis $[r = s]$, we have $r = s : \mathbb{I}$;
- (2) $[0 = 1]$ is the false proposition, whose elimination is the empty case split $[]$; and
- (3) elimination for a disjunction $* : [\phi \vee \psi]$ is a case split $[\phi \rightarrow c_\phi \mid \psi \rightarrow c_\psi]$ whose branches must agree on $\phi \wedge \psi$:

$$\frac{\phi, \psi : \mathbb{F} \quad \phi \vee \psi \text{ true} \quad C \text{ type} \quad \phi \text{ true} \vdash c_\phi : C \quad \psi \text{ true} \vdash c_\psi : C \quad \phi \wedge \psi \text{ true} \vdash c_\phi = c_\psi : C}{[\phi \rightarrow c_\phi \mid \psi \rightarrow c_\psi] : C}$$

We extend the case split notation to n -ary disjunctions as needed.

Cool Remark 1.1. Although this presentation allows for terms $\mathbb{I} \rightarrow \mathbb{F}$, etc., it is important for our implementation that we disallow \mathbb{I} and \mathbb{F} from appearing “on the right” in connectives. That is, while `cooltt` users can write $\mathbb{I} \rightarrow A$, they cannot write $A \rightarrow \mathbb{I}$. This ensures that the equational theory of \mathbb{I} and \mathbb{F} do not depend on the equational theory of general terms. It also allows us to define $\forall i.\phi(i)$ by cases on ϕ . 

We will make particular use of the *boundary* cofibration:

$$\begin{aligned}
\partial \bullet : \mathbb{I} &\rightarrow \mathbb{F} \\
\partial r &:= (r = 0 \vee r = 1)
\end{aligned}$$

The role of \mathbb{F} is to allow us to (partially) specify the boundaries of elements. If M is a partial element of A defined under $\phi \text{ true}$, we write $N : A \mid \phi \rightarrow M$ to express that N is a total element of A that restricts under ϕ to M :

$$\frac{A \text{ type} \quad \phi : \mathbb{F} \quad \phi \text{ true} \vdash M : A \quad N : A \quad \phi \text{ true} \vdash M = N : A}{N : A \mid \phi \rightarrow M}$$

We internalize this judgment as the *cubical subtype* connective.

$$\begin{array}{c}
\text{SUB FORMATION} \\
\frac{A \text{ type} \quad \phi : \mathbb{F} \quad \phi \text{ true} \vdash M : A}{\text{Sub}(A; \phi; M) \text{ type}} \\
\\
\text{SUB INTRO} \\
\frac{A \text{ type} \quad \phi : \mathbb{F} \quad \phi \text{ true} \vdash M : A \quad N : A \mid \phi \rightarrow M}{\uparrow_{\text{sub}}(N) : \text{Sub}(A; \phi; M)} \\
\\
\text{SUB ELIMINATION} \\
\frac{A \text{ type} \quad \phi : \mathbb{F} \quad \phi \text{ true} \vdash M : A \quad N : \text{Sub}(A; \phi; M)}{\downarrow_{\text{sub}}(N) : A \mid \phi \rightarrow M}
\end{array}$$

2. The universe


We consider a weak universe à la Tarski closed under dependent product and sum and, for the sake of simplicity, *itself*; this inconsistent universe would be replaced by a hierarchy of universes.

$$\begin{array}{c}
\begin{array}{c} \text{UNIV FORMATION} \\ \hline \text{Univ type} \end{array} \quad \begin{array}{c} \text{EL FORMATION} \\ \hline \hat{A} : \text{Univ} \\ \hline \text{el}(\hat{A}) \text{ type} \end{array} \quad \begin{array}{c} \text{PI CODE} \\ \hline \hat{A} : \text{Univ} \quad \hat{B} : \text{el}(\hat{A}) \rightarrow \text{Univ} \\ \hline \widehat{\prod}_{x:\hat{A}} \hat{B}(x) : \text{Univ} \end{array} \\
\\
\begin{array}{c} \text{SIGMA CODE} \\ \hline \hat{A} : \text{Univ} \quad \hat{B} : \text{el}(\hat{A}) \rightarrow \text{Univ} \\ \hline \widehat{\sum}_{x:\hat{A}} \hat{B}(x) : \text{Univ} \end{array} \quad \begin{array}{c} \text{UNIVERSE CODE} \\ \hline \widehat{\text{Univ}} : \text{Univ} \end{array} \\
\\
\begin{array}{c} \text{PI DECODING} \\ \hline \hat{A} : \text{Univ} \quad \hat{B} : \text{el}(\hat{A}) \rightarrow \text{Univ} \\ \hline \Downarrow_{\text{el}} : \text{el}(\widehat{\prod}_{x:\hat{A}} \hat{B}(x)) \cong \prod_{x:\text{el}(\hat{A})} \text{el}(\hat{B}(x)) \end{array} \quad \begin{array}{c} \text{SIGMA DECODING} \\ \hline \hat{A} : \text{Univ} \quad \hat{B} : \text{el}(\hat{A}) \rightarrow \text{Univ} \\ \hline \Downarrow_{\text{el}} : \text{el}(\widehat{\sum}_{x:\hat{A}} \hat{B}(x)) \cong \sum_{x:\text{el}(\hat{A})} \text{el}(\hat{B}(x)) \end{array} \\
\\
\begin{array}{c} \text{PATH DECODING} \\ \hline \hat{A} : \mathbb{I} \rightarrow \text{Univ} \quad M : \text{el}(\hat{A}(0)) \quad N : \text{el}(\hat{A}(1)) \\ \hline \Downarrow_{\text{el}} : \text{el}(\widehat{\text{Path}}_{\hat{A}}(M; N)) \cong \prod_{i:\mathbb{I}} \text{Sub}(\text{el}(\hat{A}(i)); \partial i; [i = 0 \rightarrow M \mid i = 1 \rightarrow N]) \end{array} \\
\\
\begin{array}{c} \text{UNIVERSE CODE DECODING} \\ \hline \Downarrow_{\text{el}} : \text{el}(\widehat{\text{Univ}}) \cong \text{Univ} \end{array}
\end{array}$$

Kan operations are a structure on codes of types.

$$\begin{array}{c}
\text{HCOM STRUCTURE} \\
\hline
\hat{A} : \text{Univ} \quad r, s : \mathbb{I} \quad \phi : \mathbb{F} \quad M : \prod_{i:\mathbb{I}} [i = r \vee \phi] \rightarrow \text{el}(\hat{A}) \\
\hline
\text{hcom}_{\hat{A}}\{r \rightsquigarrow s; \phi\}(M) : \text{el}(\hat{A}) \mid r = s \vee \phi \rightarrow M(s, *) \\
\\
\text{COE STRUCTURE} \\
\hline
\hat{A} : \mathbb{I} \rightarrow \text{Univ} \quad r, s : \mathbb{I} \quad M : \text{el}(\hat{A}(r)) \\
\hline
\text{coe}_{\hat{A}}\{r \rightsquigarrow s\}(M) : \text{el}(\hat{A}(s)) \mid r = s \rightarrow M \\
\\
\text{COM STRUCTURE} \\
\hline
\hat{A} : \mathbb{I} \rightarrow \text{Univ} \quad r, s : \mathbb{I} \quad \phi : \mathbb{F} \quad M : \prod_{i:\mathbb{I}} [i = r \vee \phi] \rightarrow \text{el}(\hat{A}(i)) \\
\hline
\text{com}_{\hat{A}}\{r \rightsquigarrow s; \phi\}(M) : \text{el}(\hat{A}(s)) \mid r = s \vee \phi \rightarrow M(s, *)
\end{array}$$

In the above, com is definitionally equal to its standard decomposition into hcom and coe.

Cool Remark 2.1. We intend for the bureaucracy of \Downarrow_{el} (and $\uparrow_{\text{sub}}(-)$, $\downarrow_{\text{sub}}(-)$) to be handled by `cooltt`'s elaborator. One might informally imagine $\hat{A} : \text{Univ}$ as a pair of a type $\text{el}(\hat{A})$ and evidence that $\text{el}(\hat{A})$ is Kan. 


Cool Remark 2.2. Prior implementations¹ have inherited the artificial separation of the “cap” ($i = r$) and “tube” (ϕ) parts of the composition data, inspired by the cubical type theoretic literature [CCHM17; AFH17; OP16; Ang+19]. While this is unnecessary and inelegant in general, the situation is particularly worsened in the context of Agda where the comparatively under-developed judgmental support for cubical subtypes leads

¹Namely `cubicaltt`, `RedPRL`, Cubical Agda, and `redtt` [Coh+18; VMA19; Red18b; Red18a].

to a discrepancy between the types of arguments passed to `hcomp` (primitive) and `hfill` (derived). The main design challenge is to express the fact that the cap *fits* onto the tube; one can use cubical subtypes to impose this coherence condition, but cubical subtypes are clumsy without good judgmental support during elaboration.

Consequently, Agda over-approximates the type of the primitive operator `hcomp` and then employs an ad hoc check for the coherence condition. However, the *derived* function `hfill` does not enjoy the special treatment of `hcomp`, and therefore must use cubical subtypes. In other words, the operators are either semantically correct but clumsy (`hfill`) or convenient but semantically broken (`hcomp`). There are at least two ways to fix this:

- (1) Consolidate “cap” and “tube” into one system.
- (2) Good judgmental support for cubical subtypes.

We have done both in `cooltt`. 

2.1. Rules for composite types. We proceed by equipping the universe with composite types and V types, which ensure it is Kan and univalent, respectively. The homogeneous composition structure on the universe corresponds to a code for composite types; we characterize the elements of this code *directly* rather than via some isomorphism (this is simplest in the case of cubically unstable type codes).²

FORMAL HCOM CODE

$$\frac{r, s : \mathbb{I} \quad \phi : \mathbb{F} \quad \hat{A} : \prod_{i:\mathbb{I}} [i = r \vee \phi] \rightarrow \text{Univ}}{\widehat{\text{Hcom}}\{r \rightsquigarrow s; \phi\}(\hat{A}) : \text{Univ} \mid r = s \vee \phi \rightarrow \hat{A}(s, *)}$$

HCOM-UNIV COMPUTATION

$$\frac{\hat{A} : \prod_{i:\mathbb{I}} [i = r \vee \phi] \rightarrow \text{el}(\widehat{\text{Univ}})}{\text{hcom}_{\widehat{\text{Univ}}}\{r \rightsquigarrow s; \phi\}(\hat{A}) = \uparrow_{\text{el}}(\widehat{\text{Hcom}}\{r \rightsquigarrow s; \phi\}(\lambda i, *. \downarrow_{\text{el}}(\hat{A}(i, *)))) : \text{el}(\widehat{\text{Univ}})}$$

COMPOSITE TYPE INTRODUCTION

$$\frac{r, s : \mathbb{I} \quad \phi : \mathbb{F} \quad \hat{A} : \prod_{i:\mathbb{I}} [i = r \vee \phi] \rightarrow \text{Univ} \quad \begin{array}{l} N : \prod_{*:[\phi]} \text{el}(\hat{A}(s, *)) \quad M : \text{el}(\hat{A}(r, *)) \mid \phi \rightarrow \text{coe}_{\lambda i. \hat{A}(i, *)} \{s \rightsquigarrow r\}(N(*)) \end{array}}{\text{box}\{r \rightsquigarrow s; \phi\}(N; M) : \text{el}(\widehat{\text{Hcom}}\{r \rightsquigarrow s; \phi\}(\hat{A})) \mid r = s \rightarrow M \mid \phi \rightarrow N(*)}$$

COMPOSITE TYPE ELIMINATION

$$\frac{r, s : \mathbb{I} \quad \phi : \mathbb{F} \quad \hat{A} : \prod_{i:\mathbb{I}} [i = r \vee \phi] \rightarrow \text{Univ} \quad M : \text{el}(\widehat{\text{Hcom}}\{r \rightsquigarrow s; \phi\}(\hat{A}))}{\text{cap}\{r \rightsquigarrow s; \phi\}(\hat{A}; M) : \text{el}(\hat{A}(r, *)) \mid r = s \rightarrow M \mid \phi \rightarrow \text{coe}_{\lambda i. \hat{A}(i, *)} \{s \rightsquigarrow r\}(M)}$$

COMPOSITE TYPE COMPUTATION

$$\frac{r, s : \mathbb{I} \quad \phi : \mathbb{F} \quad \hat{A} : \prod_{i:\mathbb{I}} [i = r \vee \phi] \rightarrow \text{Univ} \quad \begin{array}{l} N : \prod_{*:[\phi]} \text{el}(\hat{A}(s, *)) \quad M : \text{el}(\hat{A}(r, *)) \mid \phi \rightarrow \text{coe}_{\lambda i. \hat{A}(i, *)} \{s \rightsquigarrow r\}(N(*)) \end{array}}{\text{cap}\{r \rightsquigarrow s; \phi\}(\hat{A}; \text{box}\{r \rightsquigarrow s; \phi\}(N; M)) = M : \text{el}(\hat{A}(r, *))}$$

COMPOSITE TYPE UNIQUENESS

$$\frac{r, s : \mathbb{I} \quad \phi : \mathbb{F} \quad \hat{A} : \prod_{i:\mathbb{I}} [i = r \vee \phi] \rightarrow \text{Univ} \quad M : \text{el}(\widehat{\text{Hcom}}\{r \rightsquigarrow s; \phi\}(\hat{A}))}{M = \text{box}\{r \rightsquigarrow s; \phi\}(\lambda *. M; \text{cap}\{r \rightsquigarrow s; \phi\}(\hat{A}; M)) : \text{el}(\widehat{\text{Hcom}}\{r \rightsquigarrow s; \phi\}(\hat{A}))}$$

²A constructor is considered *cubically unstable* when the action of a cubical substitution may change its head constructor: this is the case for the codes of V and composite types, as well as path constructors in higher inductive types.

Warning 2.3. The arguments to box go in the order forced by their typing constraints, but please note that this is the opposite of what appeared in [Ang19].

2.2. Rules for V types. We begin by defining some definitional extensions for equivalences.

$$\begin{aligned}
\widehat{\text{IsContr}} &: \text{Univ} \rightarrow \text{Univ} \\
\widehat{\text{IsContr}}(\hat{C}) &:= \widehat{\sum}_{x:\hat{C}} \widehat{\prod}_{y:\hat{C}} \widehat{\text{Path}}_{\lambda_{\hat{C}}}(x; y) \\
\widehat{\text{Fiber}} &: \prod_{\hat{A}:\text{Univ}} \prod_{\hat{B}:\text{Univ}} (\text{el}(\hat{A}) \rightarrow \text{el}(\hat{B})) \rightarrow \text{el}(\hat{B}) \rightarrow \text{Univ} \\
\widehat{\text{Fiber}}(\hat{A}; \hat{B}; f; y) &:= \widehat{\sum}_{x:\hat{A}} \widehat{\text{Path}}_{\lambda_{\hat{B}}}(f(x); y) \\
\widehat{\text{Equiv}} &: \text{Univ} \rightarrow \text{Univ} \rightarrow \text{Univ} \\
\widehat{\text{Equiv}}(\hat{A}; \hat{B}) &:= \widehat{\sum}_{f:\hat{A} \rightarrow \hat{B}} \widehat{\prod}_{y:\hat{B}} \widehat{\text{IsContr}}(\widehat{\text{Fiber}}(\hat{A}; \hat{B}; \downarrow_{\text{el}}(f); y)) \\
\text{EquivApp} &: \prod_{\hat{A}:\text{Univ}} \prod_{\hat{B}:\text{Univ}} \text{el}(\widehat{\text{Equiv}}(\hat{A}; \hat{B})) \rightarrow \text{el}(\hat{A}) \rightarrow \text{el}(\hat{B}) \\
\text{EquivApp}(\hat{A}; \hat{B}; E) &:= \downarrow_{\text{el}}(\text{fst}(\downarrow_{\text{el}}(E)))
\end{aligned}$$

We may now specify the rules for the V types.

V TYPE CODE

$$\frac{r : \mathbb{I} \quad \hat{A} : [r = 0] \rightarrow \text{Univ} \quad \hat{B} : \text{Univ} \quad E : \prod_{*: [r=0]} \text{el}(\widehat{\text{Equiv}}(\hat{A}(*); \hat{B}))}{\widehat{V}\{r\}(\hat{A}; \hat{B}; E) : \text{Univ} \mid r = 0 \rightarrow \hat{A}(\ast) \mid r = 1 \rightarrow \hat{B}}$$

V TYPE INTRODUCTION

$$\frac{r : \mathbb{I} \quad \hat{A} : [r = 0] \rightarrow \text{Univ} \quad \hat{B} : \text{Univ} \quad E : \prod_{*: [r=0]} \text{el}(\widehat{\text{Equiv}}(\hat{A}(*); \hat{B})) \quad M : \prod_{*: [r=0]} \text{el}(\hat{A}(\ast)) \quad N : \text{el}(\hat{B}) \mid r = 0 \rightarrow \text{EquivApp}(\hat{A}(*); \hat{B}; E(*); M(*))}{\text{Vin}\{r\}(E; M; N) : \text{el}(\widehat{V}\{r\}(\hat{A}; \hat{B}; E)) \mid r = 0 \rightarrow M(\ast) \mid r = 1 \rightarrow N}$$

V TYPE ELIMINATION

$$\frac{r : \mathbb{I} \quad \hat{A} : [r = 0] \rightarrow \text{Univ} \quad \hat{B} : \text{Univ} \quad E : \prod_{*: [r=0]} \text{el}(\widehat{\text{Equiv}}(\hat{A}(*); \hat{B})) \quad M : \text{el}(\widehat{V}\{r\}(\hat{A}; \hat{B}; E))}{\text{Vproj}\{r\}(\hat{A}; \hat{B}; E; M) : \text{el}(\hat{B}) \mid r = 0 \rightarrow \text{EquivApp}(\hat{A}(*); \hat{B}; E(*); M) \mid r = 1 \rightarrow M}$$

V TYPE COMPUTATION

$$\frac{r : \mathbb{I} \quad \hat{A} : [r = 0] \rightarrow \text{Univ} \quad \hat{B} : \text{Univ} \quad E : \prod_{*: [r=0]} \text{el}(\widehat{\text{Equiv}}(\hat{A}(*); \hat{B})) \quad M : \prod_{*: [r=0]} \text{el}(\hat{A}(\ast)) \quad N : \text{el}(\hat{B}) \mid r = 0 \rightarrow \text{EquivApp}(\hat{A}(*); \hat{B}; E(*); M(*))}{\text{Vproj}\{r\}(\hat{A}; \hat{B}; E; \text{Vin}\{r\}(E; M; N)) = N : \text{el}(\hat{B})}$$

V TYPE UNIQUENESS

$$\frac{r : \mathbb{I} \quad \hat{A} : [r = 0] \rightarrow \text{Univ} \quad \hat{B} : \text{Univ} \quad E : \prod_{*: [r=0]} \text{el}(\widehat{\text{Equiv}}(\hat{A}(*); \hat{B})) \quad M : \text{el}(\widehat{V}\{r\}(\hat{A}; \hat{B}; E))}{M = \text{Vin}\{r\}(E; \lambda *. M; \text{Vproj}\{r\}(\hat{A}; \hat{B}; E; M)) : \text{el}(\widehat{V}\{r\}(\hat{A}; \hat{B}; E))}$$

3. Composition and coercion algorithms

3.1. Composition in V types.

$$\begin{array}{l}
s : \mathbb{I} \quad \hat{A} : [s = 0] \rightarrow \text{Univ} \quad \hat{B} : \text{Univ} \quad E : \prod_{*:[s=0]} \text{el}(\widehat{\text{Equiv}}(\hat{A}(*); \hat{B})) \\
r, r' : \mathbb{I} \quad M : \prod_{i:\mathbb{I}} [i = r \vee \phi] \rightarrow \text{el}(\widehat{V}\{s\})(\hat{A}; \hat{B}; E) \\
\\
\tilde{O} : \prod_{\hat{X}:\text{Univ}} (\prod_{i:\mathbb{I}} [i = r \vee \phi] \rightarrow \text{el}(\hat{X})) \rightarrow \mathbb{I} \rightarrow \text{el}(\hat{X}) \\
\tilde{O}(\hat{X}, N, i) := \text{hcom}_{\hat{X}}\{r \rightsquigarrow i; \phi\}(N) \\
\\
\tilde{P} : \prod_{i:\mathbb{I}} [i = r \vee \phi \vee \partial s] \rightarrow \text{el}(\hat{B}) \\
\tilde{P}(i, *) := [i = r \vee \phi \rightarrow \text{Vproj}\{s\}(\hat{A}; \hat{B}; E; M(i, *)) \mid s = 0 \rightarrow \text{EquivApp}(\hat{A}(*); \hat{B}; E(*); \tilde{O}(\hat{A}(*), M, i)) \mid s = 1 \rightarrow \tilde{O}(\hat{B}, M, i)] \\
\\
\hline
\text{hcom}_{\widehat{V}\{s\}(\hat{A}, \hat{B}; E)}\{r \rightsquigarrow r'; \phi\}(M) = \text{Vin}\{s\}(E; \lambda *. \tilde{O}(\hat{A}(*), M, r'); \text{hcom}_{\hat{B}}\{r \rightsquigarrow r'; \phi \vee \partial s\}(\tilde{P})) : \text{el}(\widehat{V}\{s\})(\hat{A}; \hat{B}; E)
\end{array}$$

3.2. Composition in composite types.

$$\begin{array}{l}
s, s' : \mathbb{I} \quad \psi : \mathbb{F} \quad \hat{A} : \prod_{j:\mathbb{I}} [j = s \vee \psi] \rightarrow \text{Univ} \\
r, r' : \mathbb{I} \quad \phi : \mathbb{F} \quad M : \prod_{i:\mathbb{I}} \prod_{*:[i=r \vee \phi]} \text{el}(\widehat{\text{Hcom}}\{s \rightsquigarrow s'; \psi\}(\hat{A})) \\
\\
\tilde{O} : \prod_{i:\mathbb{I}} \prod_{*:[i=r \vee \phi]} \text{el}(\hat{A}(s, *)) \quad \tilde{P} : \prod_{i:\mathbb{I}} \prod_{*:[s=s' \vee \psi]} \text{el}(\hat{A}(s', *)) \\
\tilde{O}(i, *) := \text{cap}\{s \rightsquigarrow s'; \psi\}(\hat{A}; M(i, *)) \quad \tilde{P}(i, *) = \text{hcom}_{\hat{A}(s', *)}\{r \rightsquigarrow i; \phi\}(M) \\
\\
\tilde{T} : \prod_{i:\mathbb{I}} \prod_{*:[i=r \vee \phi \vee \psi \vee s=s']} \text{el}(\hat{A}(s, *)) \\
\tilde{T}(i, *) := [i = r \vee \phi \rightarrow \tilde{O}(i, *) \mid \psi \rightarrow \text{coe}_{\lambda j. \hat{A}(j, *)}\{s' \rightsquigarrow s\}(\tilde{P}(i, *)) \mid s = s' \rightarrow \tilde{P}(i, *)] \\
\\
\hline
\text{hcom}_{\widehat{\text{Hcom}}\{s \rightsquigarrow s'; \psi\}(\hat{A})}\{r \rightsquigarrow r'; \phi\}(M) = \text{box}\{s \rightsquigarrow s'; \psi\}(\lambda *. \tilde{P}(r', *); \text{hcom}_{\hat{A}(s, *)}\{r \rightsquigarrow r'; \phi \vee \psi \vee s = s'\}(\tilde{T})) : \text{el}(\widehat{\text{Hcom}}\{s \rightsquigarrow s'; \psi\}(\hat{A}))
\end{array}$$

3.3. Coercion in V types.

$$\begin{array}{l}
s_\bullet : \mathbb{I} \rightarrow \mathbb{I} \quad \hat{A}_\bullet : \prod_{i:\mathbb{I}} [s_i = 0] \rightarrow \text{Univ} \quad \hat{B}_\bullet : \mathbb{I} \rightarrow \text{Univ} \quad E_\bullet : \prod_{i:\mathbb{I}} \prod_{*: [s_i=0]} \text{el}(\widehat{\text{Equiv}}(\hat{A}_i(*); \hat{B}_i)) \\
r, r' : \mathbb{I} \quad M : \text{el}(\widehat{V}\{s_r\}(\hat{A}_r; \hat{B}_r; E_r)) \\
\\
\tilde{F} : \prod_{i:\mathbb{I}} [s_i = 0] \rightarrow \text{el}(\hat{A}_i(*)) \rightarrow \text{el}(\hat{B}_i) \quad \tilde{O} : \text{el}(\hat{B}_{r'}) \\
\tilde{F}(i, *) := \text{EquivApp}(\hat{A}_i(*); \hat{B}_i; E_i(*)) \quad \tilde{O} := \text{coe}_{\hat{B}_\bullet} \{r \rightsquigarrow r'\} (\text{Vproj}\{s_r\}(\hat{A}_r; \hat{B}_r; E_r; M)) \\
\\
\widehat{\text{Fiber}}' : [s_{r'} = 0] \rightarrow \text{Univ} \quad \tilde{I} : \prod_{*: [s_{r'}=0]} \text{el}(\widehat{\text{IsContr}}(\widehat{\text{Fiber}}'(*))) \\
\widehat{\text{Fiber}}'(*) := \widehat{\text{Fiber}}(\hat{A}_{r'}(*); \hat{B}_{r'}; \tilde{F}(r', *); \tilde{O}) \quad \tilde{I}(*) := \downarrow_{\text{el}}(\text{snd}(\downarrow_{\text{el}}(E_{r'}(*))))(\tilde{O}) \\
\\
\tilde{P} : \prod_{*: [s_{r'}=0]} \text{el}(\widehat{\text{Fiber}}'(*)) \quad \tilde{Q} : \prod_{*: [s_{r'}=0]} \prod_{p:\text{el}(\widehat{\text{Fiber}}'(*))} \text{el}(\widehat{\text{Path}}_{\lambda_{\text{Fiber}}'}(\tilde{P}(*) ; p)) \\
\tilde{P}(*) := \text{fst}(\downarrow_{\text{el}}(\tilde{I}(*))) \quad \tilde{Q}(*) := \downarrow_{\text{el}}(\text{snd}(\downarrow_{\text{el}}(\tilde{I}(*)))) \\
\\
\tilde{R} : \prod_{*: [\forall k. s_k = 0]} \text{el}(\widehat{\text{Path}}_{\lambda_{\text{Fiber}}'}(\tilde{F}(r', *, \text{coe}_{\hat{A}_\bullet} \{r \rightsquigarrow r'\}(M)); \tilde{O})) \\
\tilde{R}(*) := \text{coe}_{\lambda_{\text{Fiber}}'}(\widehat{\text{Path}}_{\lambda_{\text{Fiber}}'}(\tilde{F}(i, *, \text{coe}_{\hat{A}_\bullet} \{r \rightsquigarrow i\}(M)); \text{coe}_{\hat{B}_\bullet} \{r \rightsquigarrow i\}(\tilde{F}(r, *, M)))) \{r \rightsquigarrow r'\}(\uparrow_{\text{el}}(\lambda_{\text{Fiber}}'. \uparrow_{\text{sub}}(\tilde{F}(r, *, M)))) \\
\\
\tilde{S} : \prod_{*: [s_{r'}=0 \wedge ((\forall k. s_k=0) \vee r=r')]} \text{el}(\widehat{\text{Path}}_{\lambda_{\text{Fiber}}'}(\tilde{P}(*) ; \uparrow_{\text{el}}[(\forall k. s_k = 0) \rightarrow \dots \text{too long} \dots \mid r = r' \rightarrow \dots \text{too long} \dots])) \\
\tilde{S}(*) := \tilde{Q}(*, \uparrow_{\text{el}}[(\forall k. s_k = 0) \rightarrow \langle \text{coe}_{\hat{A}_\bullet} \{r \rightsquigarrow r'\}(M); \tilde{R}(*) \rangle \mid r = r' \rightarrow \langle M; \uparrow_{\text{el}}(\lambda_{\text{Fiber}}'. \uparrow_{\text{sub}}(\text{Vproj}\{s_r\}(\hat{A}_r; \hat{B}_r; E_r; M))) \rangle]) \\
\\
\tilde{T} : \prod_{*: [s_{r'}=0]} \text{el}(\widehat{\text{Fiber}}'(*)) \\
\tilde{T}(*) := \text{hcom}_{\widehat{\text{Fiber}}'} \{0 \rightsquigarrow 1; (\forall k. s_k = 0) \vee r = r'\}(\lambda j, *. [j = 0 \rightarrow \tilde{P}(*) \mid (\forall k. s_k = 0) \vee r = r' \rightarrow \downarrow_{\text{sub}}(\downarrow_{\text{el}}(\tilde{S}(j)))])) \\
\\
\tilde{U} : \text{el}(\hat{B}_{r'}) \\
\tilde{U} := \text{hcom}_{\hat{B}_{r'}} \{1 \rightsquigarrow s_{r'}; r = r' \vee s_{r'} = 0\}(\lambda k, *. [k = 1 \vee r = r' \rightarrow \tilde{O} \mid s_{r'} = 0 \rightarrow \downarrow_{\text{sub}}(\downarrow_{\text{el}}(\text{snd}(\downarrow_{\text{el}}(\tilde{T}(k))))(k))]) \\
\\
\hline
\text{coe}_{\widehat{V}\{s_\bullet\}(\hat{A}_\bullet; \hat{B}_\bullet; E_\bullet)} \{r \rightsquigarrow r'\}(M) = \text{Vin}\{s_{r'}\}(E_{r'}; \lambda *. \text{fst}(\downarrow_{\text{el}}(\tilde{T}(k)))) ; \tilde{U} : \text{el}(\widehat{V}\{s_{r'}\}(\hat{A}_{r'}; \hat{B}_{r'}; E_{r'}))
\end{array}$$

3.4. Coercion in composite types.

$$\begin{array}{l}
s_{\bullet}, s'_{\bullet} : \mathbb{I} \rightarrow \mathbb{I} \quad \phi_{\bullet} : \mathbb{I} \rightarrow \mathbb{F} \quad \hat{A}_{\bullet} : \prod_{i:\mathbb{I}} \prod_{j:\mathbb{I}} \prod_{*:[j=s_i \vee \phi_i]} \text{Univ} \\
r, r' : \mathbb{I} \quad M : \text{el}(\widehat{\text{Hcom}}\{s_r \rightsquigarrow s'_r; \phi_r\}(\hat{A}_r)) \\
\\
\tilde{N} : \prod_{i,j:\mathbb{I}} \prod_{*:[\forall i.\phi_i]} \text{el}(\hat{A}_i(j, *)) \\
\tilde{N}(i, j, *) := \text{coe}_{\hat{A}_i(\bullet, *)} \{s'_i \rightsquigarrow j\} (\text{coe}_{\hat{A}_{\bullet}(s'_{\bullet}, *)} \{r \rightsquigarrow i\}(M)) \\
\\
\tilde{O} : \mathbb{I} \rightarrow \text{el}(\hat{A}_r(s_r, *)) \\
\tilde{O}(j) := \text{hcom}_{\hat{A}_r(s_r, *)} \{s'_r \rightsquigarrow j; \phi_r\} (\lambda k, *. [k = s'_r \rightarrow \text{cap}\{s_r \leftarrow s'_r; \phi_r\}(\hat{A}_r; M) \mid \phi_r \rightarrow \text{coe}_{\hat{A}_r(\bullet, *)} \{k \rightsquigarrow s_r\} (\text{coe}_{\hat{A}_r(\bullet, *)} \{s'_r \rightsquigarrow k\}(M))]) \\
\\
\tilde{P} : \text{el}(\hat{A}_{r'}(s_{r'}, *)) \\
\tilde{P} := \text{com}_{\hat{A}_{\bullet}(s_{\bullet}, *)} \{r \rightsquigarrow r'; \forall i.\phi_i \vee \forall i.(s_i = s'_i)\} (\lambda i, *. [i = r \rightarrow \tilde{O}(s_r) \mid \forall i.\phi_i \rightarrow \tilde{N}(i, s_i, *) \mid \forall i.(s_i = s'_i) \rightarrow \text{coe}_{\hat{A}_{\bullet}(s_{\bullet}, *)} \{r \rightsquigarrow i\}(M)]) \\
\\
\tilde{Q} : \prod_{j:\mathbb{I}} \prod_{*:[\phi_{r'}]} \text{el}(\hat{A}_{r'}(j, *)) \\
\tilde{Q}(j, *) := \text{com}_{\hat{A}_{r'}(\bullet, *)} \{s_{r'} \rightsquigarrow j; r = r' \vee \forall i.\phi_i\} (\lambda k, *. [k = s_{r'} \rightarrow \tilde{P} \mid r = r' \rightarrow \text{coe}_{\hat{A}_{r'}(\bullet, *)} \{s'_{r'} \rightsquigarrow k\}(M) \mid \forall i.\phi_i \rightarrow \tilde{N}(r', k, *)]) \\
\\
\tilde{H} : \text{el}(\hat{A}_{r'}(s_{r'}, *)) \\
\tilde{H} := \text{hcom}_{\hat{A}_{r'}(s_{r'}, *)} \{s_{r'} \rightsquigarrow s'_{r'}; \phi_{r'} \vee r = r'\} (\lambda j, *. [j = s_{r'} \rightarrow \tilde{P} \mid \phi_{r'} \rightarrow \text{coe}_{\hat{A}_{r'}(\bullet, *)} \{j \rightsquigarrow s_{r'}\}(\tilde{Q}(j, *)) \mid r = r' \rightarrow \tilde{O}(j)]) \\
\\
\hline
\text{coe}_{\lambda i.\widehat{\text{Hcom}}\{s_i \rightsquigarrow s'_i; \phi_i\}(\hat{A}_i)} \{r \rightsquigarrow r'\}(M) = \text{box}\{s_{r'} \rightsquigarrow s'_{r'}; \phi_{r'}\} (\lambda *. \tilde{Q}(s'_{r'}, *); \tilde{H}) : \text{el}(\widehat{\text{Hcom}}\{s_{r'} \rightsquigarrow s'_{r'}; \phi_{r'}\}(\hat{A}_{r'}))
\end{array}$$

References

- [AFH17] Carlo Angiuli, Kuen-Bang Hou (Favonia), and Robert Harper. *Computational Higher Type Theory III: Univalent Universes and Exact Equality*. 2017. arXiv: 1712.01800. URL: <https://arxiv.org/abs/1712.01800> (cit. on p. 3).
- [Ang+19] Carlo Angiuli, Guillaume Brunerie, Thierry Coquand, Kuen-Bang Hou (Favonia), Robert Harper, and Daniel R. Licata. *Syntax and Models of Cartesian Cubical Type Theory*. Preprint. Feb. 2019. URL: <https://github.com/dlicata335/cart-cube> (cit. on pp. 1, 3).
- [Ang19] Carlo Angiuli. “Computational Semantics of Cartesian Cubical Type Theory”. PhD thesis. Carnegie Mellon University, 2019 (cit. on pp. 1, 5).
- [CCHM17] Cyril Cohen, Thierry Coquand, Simon Huber, and Anders Mörtberg. “Cubical Type Theory: a constructive interpretation of the univalence axiom”. In: *IfCoLog Journal of Logics and their Applications* 4.10 (Nov. 2017), pp. 3127–3169. URL: <http://www.collegepublications.co.uk/journals/ifcolog/?00019> (cit. on p. 3).
- [Coh+18] Cyril Cohen, Thierry Coquand, Simon Huber, and Anders Mörtberg. *cubicaltt: Experimental implementation of Cubical Type Theory*. 2018. URL: <https://github.com/mortberg/cubicaltt> (cit. on p. 3).
- [OP16] Ian Orton and Andrew M. Pitts. “Axioms for Modelling Cubical Type Theory in a Topos”. In: *25th EACSL Annual Conference on Computer Science Logic, CSL 2016, August 29 - September 1, 2016, Marseille, France*. 2016, 24:1–24:19 (cit. on pp. 1, 3).
- [Red18a] The RedPRL Development Team. *RedPRL – the People’s Refinement Logic*. 2018. URL: <http://www.redprl.org/> (cit. on p. 3).
- [Red18b] The RedPRL Development Team. *redtt*. 2018. URL: <http://www.github.com/RedPRL/redtt> (cit. on p. 3).
- [Red20] The RedPRL Development Team. *cooltt*. 2020. URL: <http://www.github.com/RedPRL/cooltt> (cit. on p. 1).
- [VMA19] Andrea Vezzosi, Anders Mörtberg, and Andreas Abel. “Cubical Agda: A Dependently Typed Programming Language with Univalence and Higher Inductive Types”. In: *Proceedings of the 24th ACM SIGPLAN International Conference on Functional Programming*. ICFP ’19. Boston, Massachusetts, USA: ACM, 2019. DOI: 10.1145/3341691. URL: <http://www.cs.cmu.edu/~amoertbe/papers/cubicalagda.pdf> (cit. on p. 3).