CANONICITY FOR HOMOTOPY TYPE THEORY WITH AN INTERVAL TYPE

VALERY ISAEV

Abstract. We define a version of homotopy type theory, which we call *homotopy type theory with an interval type*, and give a partial proof of the canonicity property for it.

1. Syntax

We will work in a version of Martin-Löf's type theory (MLTT), which we call *homotopy type theory with an interval type* (HoTT-I). It has all of the usual constructions of MLTT except for the identity types.

1.1. **Basic theory.** The inference and reduction rules for (some of) these construction are listed below.

$$\frac{}{\vdash} \qquad \frac{\Gamma \vdash A}{\Gamma, x : A \vdash}\ , x \notin \Gamma \qquad \frac{\Gamma \vdash}{\Gamma \vdash x : A}\ , x : A \in \Gamma$$

$$\frac{\Gamma \vdash a : A \qquad \Gamma \vdash B}{\Gamma \vdash a : B}\ , A \equiv B$$

$$\frac{\Gamma \vdash A \qquad \Gamma, x : A \vdash B}{\Gamma \vdash \Pi(x : A)B} \qquad \frac{\Gamma \vdash A \qquad \Gamma, x : A \vdash B}{\Gamma \vdash \Sigma(x : A)B}$$

$$\frac{\Gamma, x : A \vdash b : B}{\Gamma \vdash \lambda x.\ b : \Pi(x : A)B} \qquad \frac{\Gamma \vdash f : \Pi(x : A)B \qquad \Gamma \vdash a : A}{\Gamma \vdash f\ a : B[x := a]}$$

$$\frac{\Gamma \vdash a : A \qquad \Gamma \vdash b : B[x := a] \qquad \Gamma \vdash B}{\Gamma \vdash (a, b) : \Sigma(x : A)B}$$

$$\frac{\Gamma \vdash p : \Sigma(x : A)B}{\Gamma \vdash proj_1\ p : A} \qquad \frac{\Gamma \vdash p : \Sigma(x : A)B}{\Gamma \vdash proj_2\ p : B[x := proj_1\ p]}$$

$$\frac{\Gamma \vdash}{\Gamma \vdash \mathbb{N}} \qquad \frac{\Gamma \vdash}{\Gamma \vdash 0 : \mathbb{N}} \qquad \frac{\Gamma \vdash n : \mathbb{N}}{\Gamma \vdash S\ n : \mathbb{N}}$$

$$\frac{\Gamma, x : \mathbb{N} \vdash A \qquad \Gamma \vdash b : A[x := 0] \qquad \Gamma, x : \mathbb{N}, r : A \vdash s : A[x := S\ x] \qquad \Gamma \vdash n : \mathbb{N}}{\Gamma \vdash R_{\lambda x.\ A}\ b\ (\lambda x r.\ s)\ n : A[x := n]}$$

$$\frac{\Gamma \vdash}{\Gamma \vdash U_\alpha} \qquad \frac{\Gamma \vdash A : U_\alpha}{\Gamma \vdash A} \qquad \frac{\Gamma \vdash A : U_\alpha}{\Gamma \vdash A : U_{\alpha+1}} \qquad \frac{\Gamma \vdash}{\Gamma \vdash \mathbb{N} : U_\alpha}$$

$$\frac{\Gamma \vdash A : U_\alpha \qquad \Gamma, x : A \vdash B : U_\alpha}{\Gamma \vdash \Pi(x : A)B : U_\alpha} \qquad \frac{\Gamma \vdash A : U_\alpha \qquad \Gamma, x : A \vdash B : U_\alpha}{\Gamma \vdash \Sigma(x : A)B : U_\alpha}$$

$$(\lambda x.\ b)\ a \Rightarrow_\beta b[x := a]$$
$$proj_1\ (a, b) \Rightarrow_\beta a$$
$$proj_2\ (a, b) \Rightarrow_\beta b$$
$$R_{\lambda x.\ A}\ b\ (\lambda x r.\ s)\ 0 \Rightarrow_\beta b$$
$$R_{\lambda x.\ A}\ b\ (\lambda x r.\ s)\ (S\ n) \Rightarrow_\beta s[x := n, r := R_{\lambda x.\ A}\ b\ (\lambda x r.\ s)\ n]$$

We will write $(x_1 \dots x_n : A) \to B$ for $\Pi(x_1 : A) \dots \Pi(x_n : A)B$.

Apart from the usual types, HoTT-I also has an interval type I, which has two constructors $left$ and $right$ and one eliminator coe. The inference rules for I, $left$ and $right$ are given below, and the inference rule for coe will be described later.

$$\frac{\Gamma \vdash}{\Gamma \vdash I : U_\alpha} \qquad \frac{\Gamma \vdash}{\Gamma \vdash left : I} \qquad \frac{\Gamma \vdash}{\Gamma \vdash right : I}$$

Using the interval type, we can define path types as functions from the interval type. Path types have one constructor $path$ and one eliminator @.

$$\frac{\Gamma, x : I \vdash A \qquad \Gamma \vdash a : A[x := left] \qquad \Gamma \vdash a' : A[x := right]}{\Gamma \vdash Path\ (\lambda x.\ A)\ a\ a'}$$

$$\frac{\Gamma, x : I \vdash a : A}{\Gamma \vdash path\ (\lambda x.\ a) : Path\ (\lambda x.\ A)\ a[x := left]\ a[x := right]}$$

$$\frac{\Gamma \vdash p : Path\ (\lambda x.\ A)\ a\ a' \qquad \Gamma \vdash i : I}{\Gamma \vdash p\ @_{a,a'}\ i : A[x := i]}$$

$$\frac{\Gamma, x : I \vdash A : U_\alpha \qquad \Gamma \vdash a : A[x := left] \qquad \Gamma \vdash a' : A[x := right]}{\Gamma \vdash Path\ (\lambda x.\ A)\ a\ a' : U_\alpha}$$

$$path\ (\lambda x.\ t)\ @_{a,a'}\ i \Rightarrow_\beta t[x := i]$$
$$p\ @_{a,a'}\ left \Rightarrow_\beta a$$
$$p\ @_{a,a'}\ right \Rightarrow_\beta a'$$
$$path\ (\lambda x.\ p@_{a,a'}x) \equiv_\eta p$$

We write $a =_A a'$ (or simply $a = a'$) for $Path\ (\lambda x.\ A)\ a\ a'$ if $x \notin FV(A)$.

There are several equivalent to define an elimination rule for the interval type, but the idea is the same: we have a dependent type A over I and a point in some fibre of A, then we can extend this point to the whole interval. First, we describe the basic version of coe:

$$\frac{\Gamma, x : I \vdash A \qquad \Gamma \vdash a : A[x := left] \qquad \Gamma \vdash j : I}{\Gamma \vdash coe_1\ (\lambda x.\ A)\ a\ j : A[x := j]}$$

$$coe_1\ (\lambda x.\ A)\ a\ left \Rightarrow_\beta a$$

Using coe_1, we can show that path types satisfy the usual J rule. Reflexivity is defined simply as the constant path:

$$refl : (x : A) \to x = x$$
$$refl = \lambda x.\ path\ (\lambda i.\ x)$$

To construct J, first we need to define function $squeeze$, which satisfies the following rules:

$$squeeze : I \to I \to I$$
$$squeeze\ left\ j \equiv left$$
$$squeeze\ right\ j \equiv j$$
$$squeeze\ i\ left \equiv left$$
$$squeeze\ i\ right \equiv i$$

To define $squeeze$, first we define function $squeeze'$ which satisfies all of the equations above except for the last one:

$$squeeze' = \lambda ij.\ coe_1\ (\lambda x.\ left = x)\ (path\ (\lambda x.\ left))\ j\ @\ i$$

Now we can define $squeeze$ as follows:

$$squeeze = \lambda ij.\ coe_1\ (\lambda i.\ Path\ (\lambda j.\ left = squeeze'\ i\ j)\ (path\ (\lambda x.\ left))$$
$$(path\ (\lambda j.\ squeeze'\ i\ j)))\ (path\ (\lambda x.\ path\ (\lambda x.\ left)))\ right\ @\ i\ @\ j$$

The idea is that the right hand sides of the definitions of $squeeze'$ and $squeeze$ describe fillers of certain cubical horns of dimension two and three respectively.

Using $squeeze$ we can defined function $psqueeze$:

$$psqueeze : (x\ y : A) \to (p : x = y) \to (i : I) \to x = p\ @\ i$$
$$psqueeze = \lambda x\ y\ p\ i.\ path\ (\lambda j.\ p\ @\ squeeze\ i\ j)$$

Using $psqueeze$ we can define J which satisfies the following inference rule:

$$\frac{\Gamma, x : A, x' : A, z : x = x' \vdash D \qquad \Gamma, x : A \vdash d : D[x' := x, z := refl\ x] \qquad \Gamma \vdash p : a = a'}{\Gamma \vdash J\ (\lambda x x' z.\ D)\ (\lambda x.\ d)\ a\ a'\ p : D[x := a, x' := a', z := p]}$$

$J\ (\lambda x x' z.\ D)\ (\lambda x.\ d)\ a\ a'\ p$ is defined as follows:

$$coe_1\ (\lambda i.\ D[x := a, x' := p\ @\ i, z := psqueeze\ a\ a'\ p\ i])\ (d[x := a])\ right$$

The usual computation rule does not hold for this version of J. We can fix this by adding the following additional reduction rule for coe_1:

$$coe_1\ (\lambda x.\ A)\ a\ i \Rightarrow_\sigma a, \text{ if } x \notin FV(A)$$

But it is not necessary, since this rule holds propositionally.

1.2. **Other versions of coe.** We can define coe in a different way:

$$\frac{\Gamma, x : I \vdash A \qquad \Gamma \vdash i : I \qquad \Gamma \vdash a : A[x := i] \qquad \Gamma \vdash j : I}{\Gamma \vdash coe_2\ (\lambda x.\ A)\ i\ a\ j : A[x := j]}$$

Obiously, coe_1 is a special case of coe_2, but coe_2 can be constructed from coe_1. Indeed, first let us prove that I is contractible:

$$(left, \lambda x.\ path\ (\lambda i.\ squeeze\ x\ i))$$

It follows that for every $i, j : I$ we have term $pp\ i\ j$ of type $i = j$. Now we can define coe_2 as follows:

$$coe_2\ (\lambda x.\ A)\ i\ a\ j = coe_1\ (\lambda y.\ A[x := pp\ i\ j\ @\ y])\ a\ right$$

Yet another version of coe is the following one:

$$\frac{\Gamma, x : I \vdash A \qquad \Gamma \vdash a : A[x := left]}{\Gamma \vdash coe_0\ (\lambda x.\ A)\ a : A[x := right]}$$

This version seems to be weaker then coe_1, but if we have $squeeze$, then we can define coe_1 as follows:

$$coe_1\ (\lambda x.\ A)\ a\ i = coe_0\ (\lambda y.\ A[x := squeeze\ y\ i])\ a$$

In the rest of this document we will use this version of coe. We will assume that $squeeze$ is defined as a primitive operation which satisfies four rules that we described.

1.3. **Univalence.** The theory that we described does not have the univalence axiom. We can extend the theory in such a way that UA becomes true. We add the following construction:

$$\frac{\begin{array}{llll}\Gamma \vdash A & \Gamma \vdash f : A \to B & \Gamma \vdash p : (a : A) \to g\ (f\ a) =_A a & \\ \Gamma \vdash B & \Gamma \vdash g : B \to A & \Gamma \vdash q : (b : B) \to f\ (g\ b) =_B b & \Gamma \vdash i : I\end{array}}{\Gamma \vdash Iso\ A\ B\ f\ g\ p\ q\ i}$$

$$\frac{\begin{array}{llll}\Gamma \vdash A : U_\alpha & \Gamma \vdash f : A \to B & \Gamma \vdash p : (a : A) \to g\ (f\ a) =_A a & \\ \Gamma \vdash B : U_\alpha & \Gamma \vdash g : B \to A & \Gamma \vdash q : (b : B) \to f\ (g\ b) =_B b & \Gamma \vdash i : I\end{array}}{\Gamma \vdash Iso\ A\ B\ f\ g\ p\ q\ i : U_\alpha}$$

$$Iso\ A\ B\ f\ g\ p\ q\ left \Rightarrow_\beta A$$
$$Iso\ A\ B\ f\ g\ p\ q\ right \Rightarrow_\beta B$$
$$coe_0\ (\lambda x.\ Iso\ A\ B\ f\ g\ p\ q\ x)\ a \Rightarrow_\beta f\ a, \text{ if } x \notin FV(A\ B\ f\ g\ p\ q)$$

Note that we use quasi-inverses instead of some correct version of inverse functions since Iso does not assert that $\Sigma(f : A \to B)\ (qinv\ f)$ is equivalent to $A = B$, it only implies that there exists a map $\Sigma(f : A \to B)\ (qinv\ f) \to A = B$ which satisfies some properties which imply the univalence axiom.

2. Canonicity

In this section we will add more construction and reduction rules to the theory so that it will have the canonicity property. The problem with the current version of the theory is that terms of the form $coe\ (\lambda x.\ A)\ a$ are never reduce. To fix this we define new reduction rules by recursion on A.

2.1. **Basic theory.** We already saw that we can define fillers using coe. It is convenient to add fillers as separate constructions. But before we do this let us introduce a bit of notation. We will write $\overline{x}$ for a sequence of variables or terms $x_1\ \ldots\ x_n$, and we will write $\overline{x}_{\hat{i}}$ for the sequence $x_1\ \ldots\ x_{i-1}\ x_{i+1}\ \ldots\ x_n$. We use the same notation in other cases. For example, $\overline{x : I}$ denotes $x_1 : I, \ldots x_n : I$, and $A[\overline{x := j}]$ denotes $A[x_1 := j_1] \ldots [x_n := j_n]$. Now we can define the inference rules for fillers:

$\Gamma, \overline{x : I} \vdash A$
$\Gamma, \overline{x : I}_{\hat{k}} \vdash a_k : A[x_k := left]$, $1 \leq k \leq n$
$\Gamma, \overline{x : I}_{\hat{k}} \vdash a'_k : A[x_k := right]$, $2 \leq k \leq n$
$\Gamma \vdash j_k : I$, $2 \leq k \leq n$
$a_{i_1}[x_{i_2} := left] \equiv a_{i_2}[x_{i_1} := left]$, $1 \leq i_1 < i_2 \leq n$
$a'_{i_1}[x_{i_2} := left] \equiv a_{i_2}[x_{i_1} := right]$, $2 \leq i_1 < i_2 \leq n$
$a_{i_1}[x_{i_2} := right] \equiv a'_{i_2}[x_{i_1} := left]$, $1 \leq i_1 < i_2 \leq n$
$a'_{i_1}[x_{i_2} := right] \equiv a'_{i_2}[x_{i_1} := right]$, $2 \leq i_1 < i_2 \leq n$

$\Gamma \vdash Fill^n\ (\lambda \overline{x}.\ A)\ (\lambda \overline{x}_{\hat{n}}.\ a_n)\ (\lambda \overline{x}_{\hat{n}}.\ a'_n)\ \ldots\ (\lambda \overline{x}_{\hat{2}}.\ a_2)\ (\lambda \overline{x}_{\hat{2}}.\ a'_2)\ (\lambda \overline{x}_{\hat{1}}.\ a_1)\ \overline{j}_{\hat{1}} : A[x_1 := right]\overline{[x := j]}_{\hat{1}}$

These terms are defined for every $n \geq 2$. It is actually possible to define these terms using coe, but it is convenient to introduce them as primitive constructions. Note that the rule for $Fill^1$ is the same as for coe_0. So we will write $Fill^1$ for coe_0.

There are a few β rules for $Fill^n$:

$$Fill^n\ (\lambda \overline{x}.\ A)\ (\lambda \overline{x}_{\hat{n}}.\ a_n)\ (\lambda \overline{x}_{\hat{n}}.\ a'_n)\ \ldots\ (\lambda \overline{x}_{\hat{2}}.\ a_2)\ (\lambda \overline{x}_{\hat{2}}.\ a'_2)\ (\lambda \overline{x}_{\hat{1}}.\ a_1)\ j_2\ \ldots j_{i-1}\ left\ j_{i+1}\ \ldots\ j_n \Rightarrow_\beta$$
$$a_i[x_1 := right][x_2 := j_2]\ldots[x_{i-1} := j_{i-1}][x_{i+1} := j_{i+1}]\ldots[x_n := j_n]$$
$$Fill^n\ (\lambda \overline{x}.\ A)\ (\lambda \overline{x}_{\hat{n}}.\ a_n)\ (\lambda \overline{x}_{\hat{n}}.\ a'_n)\ \ldots\ (\lambda \overline{x}_{\hat{2}}.\ a_2)\ (\lambda \overline{x}_{\hat{2}}.\ a'_2)\ (\lambda \overline{x}_{\hat{1}}.\ a_1)\ j_2\ \ldots j_{i-1}\ right\ j_{i+1}\ \ldots\ j_n \Rightarrow_\beta$$
$$a'_i[x_1 := right][x_2 := j_2]\ldots[x_{i-1} := j_{i-1}][x_{i+1} := j_{i+1}]\ldots[x_n := j_n]$$

We can also define the sigma rule for every $Fill^n$:

$$Fill^n\ (\lambda \overline{x}.\ A)\ (\lambda \overline{x}_{\hat{n}}.\ a_n)\ (\lambda \overline{x}_{\hat{n}}.\ a'_n)\ \ldots\ (\lambda \overline{x}_{\hat{2}}.\ a_2)\ (\lambda \overline{x}_{\hat{2}}.\ a'_2)\ (\lambda \overline{x}_{\hat{1}}.\ a_1)\ j_2\ \ldots\ j_n \Rightarrow_\sigma$$
$$a_1[x_2 := j_2]\ldots[x_n := j_n] \text{ if } x_1 \notin FV(A) \cup FV(a_n) \cup FV(a'_n) \cup \ldots \cup FV(a_2) \cup FV(a'_2)$$

But as in the case of coe_0 this rule is not necessary. To make the theory computational, we need to add new rules, which we call τ, for every type construction that we have: $Path$, Σ, Π, $\mathbb{N}$ and U_α. It is easy to define τ rules for $\mathbb{N}$ and path types. For every $n \geq 1$, we add the following rules:

$$Fill^n\ (\lambda \overline{x}.\ \mathbb{N})\ (\lambda \overline{x}_{\hat{n}}.\ 0)\ (\lambda \overline{x}_{\hat{n}}.\ 0)\ \ldots\ (\lambda \overline{x}_{\hat{2}}.\ 0)\ (\lambda \overline{x}_{\hat{2}}.\ 0)\ (\lambda \overline{x}_{\hat{1}}.\ 0)\ \overline{j}_{\hat{1}}\ \Rightarrow_\tau 0$$
$$Fill^n\ (\lambda \overline{x}.\ \mathbb{N})\ (\lambda \overline{x}_{\hat{n}}.\ S\ a_n)\ (\lambda \overline{x}_{\hat{n}}.\ S\ a'_n)\ \ldots\ (\lambda \overline{x}_{\hat{2}}.\ S\ a_2)\ (\lambda \overline{x}_{\hat{2}}.\ S\ a'_2)\ (\lambda \overline{x}_{\hat{1}}.\ S\ a_1)\ \overline{j}_{\hat{1}}\ \Rightarrow_\tau$$
$$S\ (Fill^n\ (\lambda \overline{x}.\ \mathbb{N})\ (\lambda \overline{x}_{\hat{n}}.\ a_n)\ (\lambda \overline{x}_{\hat{n}}.\ a'_n)\ \ldots\ (\lambda \overline{x}_{\hat{2}}.\ a_2)\ (\lambda \overline{x}_{\hat{2}}.\ a'_2)\ (\lambda \overline{x}_{\hat{1}}.\ a_1)\ \overline{j}_{\hat{1}})$$

$$Fill^n\ (\lambda \overline{x}.\ Path\ (\lambda x_{n+1}.A)\ a_{n+1}\ a'_{n+1})\ (\lambda \overline{x}_{\hat{n}}.\ path\ (\lambda x_{n+1}.\ a_n))\ (\lambda \overline{x}_{\hat{n}}.\ path\ (\lambda x_{n+1}.\ a'_n))\ \ldots$$
$$(\lambda \overline{x}_{\hat{2}}.\ path\ (\lambda x_{n+1}.\ a_2))\ (\lambda \overline{x}_{\hat{2}}.\ path\ (\lambda x_{n+1}.\ a'_2))\ (\lambda \overline{x}_{\hat{1}}.\ path\ (\lambda x_{n+1}.\ a_1))\ j_2\ \ldots\ j_n \Rightarrow_\tau$$
$$path\ (\lambda j_{n+1}.\ Fill^{n+1}\ (\lambda \overline{x}.\ \lambda x_{n+1}.\ A)\ (\lambda \overline{x}.\ a_{n+1})\ (\lambda \overline{x}.\ a'_{n+1})\ \ldots$$
$$(\lambda \overline{x}_{\hat{2}}.\ \lambda x_{n+1}.\ a_2)\ (\lambda \overline{x}_{\hat{2}}.\ \lambda x_{n+1}.\ a'_2)\ \ldots\ (\lambda \overline{x}_{\hat{1}}.\ \lambda x_{n+1}.\ a_1)\ j_2\ \ldots\ j_{n+1})$$

As in the case of coe_0, there are versions of $Fill^n$ that define filler for the whole box and not only for the lid. We can define them as follows:

$$Fill^n_1\ (\lambda \overline{x}.\ A)\ (\lambda \overline{x}_{\hat{n}}.\ a_n)\ (\lambda \overline{x}_{\hat{n}}.\ a'_n)\ldots(\lambda \overline{x}_{\hat{2}}.\ a_2)\ (\lambda \overline{x}_{\hat{2}}.\ a'_2)\ (\lambda \overline{x}_{\hat{1}}.\ a_1)\ \overline{j} =$$
$$Fill^n\ (\lambda \overline{x}.\ A[x_1 := squeeze\ x_1\ j_1])\ (\lambda \overline{x}_{\hat{n}}.\ a_n[x_1 := squeeze\ x_1\ j_1])\ (\lambda \overline{x}_{\hat{n}}.\ a'_n[x_1 := squeeze\ x_1\ j_1])\ldots$$
$$(\lambda \overline{x}_{\hat{2}}.\ a_2[x_1 := squeeze\ x_1\ j_1])\ (\lambda \overline{x}_{\hat{2}}.\ a'_2[x_1 := squeeze\ x_1\ j_1])\ (\lambda \overline{x}_{\hat{1}}.\ a_1)\ \overline{j}_{\hat{1}}$$

Now, we can define τ rules for Σ types:

$$Fill^n\ (\lambda \overline{x}.\ \Sigma(y : A)B)\ (\lambda \overline{x}_{\hat{n}}.\ (a_n, b_n))\ (\lambda \overline{x}_{\hat{n}}.\ (a'_n, b'_n))\ \ldots\ (\lambda \overline{x}_{\hat{2}}.\ (a_2, b_2))\ (\lambda \overline{x}_{\hat{2}}.\ (a'_2, b'_2))\ (\lambda \overline{x}_{\hat{1}}.\ (a_1, b_1))\ \overline{j}_{\hat{1}}\ \Rightarrow_\tau$$

$$(Fill^n\ (\lambda\overline{x}.\ A)\ (\lambda\overline{x}_{\hat{n}}.\ a_n)\ (\lambda\overline{x}_{\hat{n}}.\ a'_n)\ \ldots\ (\lambda\overline{x}_{\hat{2}}.\ a_2)\ (\lambda\overline{x}_{\hat{2}}.\ a'_2)\ (\lambda\overline{x}_{\hat{1}}.\ a_1)\ \overline{j}_{\hat{1}},$$
$$Fill^n\ (\lambda\overline{x}.\ B[y := Fill^n_1\ (\lambda\overline{x}.\ A)\ (\lambda\overline{x}_{\hat{n}}.\ a_n)\ (\lambda\overline{x}_{\hat{n}}.\ a'_n)\ \ldots\ (\lambda\overline{x}_{\hat{2}}.\ a_2)\ (\lambda\overline{x}_{\hat{2}}.\ a'_2)\ (\lambda\overline{x}_{\hat{1}}.\ a_1)\ \overline{x}])$$
$$(\lambda\overline{x}_{\hat{n}}.\ b_n)\ (\lambda\overline{x}_{\hat{n}}.\ b'_n)\ \ldots\ (\lambda\overline{x}_{\hat{2}}.\ b_2)\ (\lambda\overline{x}_{\hat{2}}.\ b'_2)\ (\lambda\overline{x}_{\hat{1}}.\ b_1)\ \overline{j}_{\hat{1}})$$

To define τ rules for Π types, we need an operation that transports a point in a fibre of a type over a cube to any other fibre. We already have such operation for the interval, namely coe_2. We can define the general one in terms of coe_2:

$$coe^0\ A\ a = a$$
$$coe^{n+1}\ (\lambda\overline{x}x_{n+1}.\ A)\ \overline{i}\ i_{n+1}\ a\ \overline{j}\ j_{n+1} =$$
$$coe_2\ (\lambda x_{n+1}.\ A[\overline{x := j}])\ i_{n+1}\ (coe^n\ (\lambda\overline{x}.\ A[x_{n+1} := i_{n+1}])\ \overline{i}\ a\ \overline{j})\ j_{n+1}$$

Now, for every $n \geq 1$, we can define τ rules for Π:

$$Fill^n\ (\lambda\overline{x}.\ \Pi(y : A)B)\ (\lambda\overline{x}_{\hat{n}}.\ \lambda y.\ b_n)\ (\lambda\overline{x}_{\hat{n}}.\ \lambda y.\ b'_n)\ \ldots\ (\lambda\overline{x}_{\hat{2}}.\ \lambda y.\ b_2)\ (\lambda\overline{x}_{\hat{2}}.\ \lambda y.\ b'_2)\ (\lambda\overline{x}_{\hat{1}}.\ \lambda y.\ b_1)\ \overline{j}_{\hat{1}} \Rightarrow_\tau$$
$$\lambda y'.\ Fill^n\ (\lambda\overline{x}.\ B[y := coe^n\ (\lambda\overline{x}.\ A)\ right\ j_2\ \ldots\ j_n\ y'\ x_1\ \ldots\ x_n])$$
$$(\lambda\overline{x}_{\hat{n}}.\ b_n[y := coe^n\ (\lambda\overline{x}.\ A)\ right\ j_2\ \ldots\ j_n\ y'\ x_1\ \ldots\ x_{n-1}\ left])$$
$$(\lambda\overline{x}_{\hat{n}}.\ b'_n[y := coe^n\ (\lambda\overline{x}.\ A)\ right\ j_2\ \ldots\ j_n\ y'\ x_1\ \ldots\ x_{n-1}\ right])\ \ldots$$
$$(\lambda\overline{x}_{\hat{2}}.\ b_2[y := coe^n\ (\lambda\overline{x}.\ A)\ right\ j_2\ \ldots\ j_n\ y'\ x_1\ left\ x_3\ \ldots\ x_n])$$
$$(\lambda\overline{x}_{\hat{2}}.\ b'_2[y := coe^n\ (\lambda\overline{x}.\ A)\ right\ j_2\ \ldots\ j_n\ y'\ x_1\ right\ x_3\ \ldots\ x_n])$$
$$(\lambda\overline{x}_{\hat{1}}.\ b_1[y := coe^n\ (\lambda\overline{x}.\ A)\ right\ j_2\ \ldots\ j_n\ y'\ left\ x_2\ \ldots\ x_n])\ j_2\ \ldots\ j_n$$

2.2. **Universes.** It is easy to define τ rule for universes for coe:

$$coe_0\ (\lambda x.\ U_\alpha)\ A \Rightarrow_\tau A$$

Terms of the form $Fill^n\ (\lambda x.\ U_\alpha)\ \ldots$ for $n \geq 2$ can be handled in two ways. First option is that we can try to reduce this terms using univalence (which is defined below). It is easy to do for $n = 2$, and it may be possible to do for $n > 2$. Second option is that we can say that terms of this form are canonical. It is not a problem, since there are no normalized terms of this form in the empty context. But then we get terms of the following forms:

$$coe_0\ (\lambda y.\ Fill^{n_1}\ (\lambda\overline{x}.\ U_\alpha)\ \ldots)\ a$$
$$coe_0\ (\lambda y.\ Fill^{n_1}\ (\lambda\overline{x}.\ Fill^{n_2}\ (\lambda\overline{x'}.\ U_\alpha)\ \ldots)\ \ldots)\ a$$
$$\ldots$$

We can define τ rules at least for terms of the first form. We do not know how to define τ rules for other cases.

2.3. **Univalence.** To define τ rules for terms of the form $coe_0\ (\lambda x.\ Iso\ A\ B\ f\ g\ p\ q\ k)\ a$, we need to add the following construction:

$$\frac{\begin{array}{cccc}\Gamma, x : I \vdash A & \Gamma, x : I \vdash f : A \to B & \Gamma, x : I \vdash p : (a : A) \to g\ (f\ a) =_A a & \Gamma \vdash i_1 : I \\ \Gamma, x : I \vdash B & \Gamma, x : I \vdash g : B \to A & \Gamma, x : I \vdash q : (b : B) \to f\ (g\ b) =_B b & \Gamma \vdash i_2 : I \\ \multicolumn{4}{c}{\Gamma \vdash a : Iso\ (A[x := left])\ (B[x := left])\ (f[x := left])\ (g[x := left])\ (p[x := left])\ (q[x := left])\ i_1}\end{array}}{\begin{array}{l}\Gamma \vdash iso^1\ (\lambda x.\ A)\ (\lambda x.\ B)\ (\lambda x.\ f)\ (\lambda x.\ g)\ (\lambda x.\ p)\ (\lambda x.\ q)\ i_1\ a\ i_2 : \\ Iso\ (A[x := right])\ (B[x := right])\ (f[x := right])\ (g[x := right])\ (p[x := right])\ (q[x := right])\ i_2\end{array}}$$

$iso^1\ (\lambda x.\ A)\ (\lambda x.\ B)\ (\lambda x.\ f)\ (\lambda x.\ g)\ (\lambda x.\ p)\ (\lambda x.\ q)\ left\ a\ left \Rightarrow_\beta coe_0\ (\lambda x.\ A)\ a$

$iso^1\ (\lambda x.\ A)\ (\lambda x.\ B)\ (\lambda x.\ f)\ (\lambda x.\ g)\ (\lambda x.\ p)\ (\lambda x.\ q)\ left\ a\ right \Rightarrow_\beta coe_0\ (\lambda x.\ B)\ (f[x := left]\ a)$

$iso^1\ (\lambda x.\ A)\ (\lambda x.\ B)\ (\lambda x.\ f)\ (\lambda x.\ g)\ (\lambda x.\ p)\ (\lambda x.\ q)\ right\ a\ left \Rightarrow_\beta coe_0\ (\lambda x.\ A)\ (g[x := left]\ a)$

$iso^1\ (\lambda x.\ A)\ (\lambda x.\ B)\ (\lambda x.\ f)\ (\lambda x.\ g)\ (\lambda x.\ p)\ (\lambda x.\ q)\ right\ a\ right \Rightarrow_\beta coe_0\ (\lambda x.\ B)\ a$

Now we can define the following τ rule:

$$coe_0\ (\lambda x.\ Iso\ A\ B\ f\ g\ p\ q\ k)\ a \Rightarrow_\tau$$

$iso^1\ (\lambda x.\ A)\ (\lambda x.\ B)\ (\lambda x.\ f)\ (\lambda x.\ g)\ (\lambda x.\ p)\ (\lambda x.\ q)\ (k[x := left])\ a\ (k[x := right])$

We also need to define τ rules for terms of the form $Fill^n\ (\lambda x.\ Iso\ A\ B\ f\ g\ p\ q\ k)\ \ldots$, $n \geq 2$. We could try to do this in the same way as for coe_0.