

# Generating Higher Identity Proofs in Homotopy Type Theory

Thibaut Benjamin

December 3, 2024

## Abstract

Finster and Mimram have defined a dependent type theory called **CaTT**, which describes the structure of  $\omega$ categories. Types in homotopy type theory with their higher identity types form weak  $\omega$ groupoids, so they are in particular weak  $\omega$ categories. In this article, we show that this principle makes homotopy type theory into a model of **CaTT**, by defining a translation principle that interprets an operation on the cell of an  $\omega$ category as an operation on higher identity types. We then illustrate how this translation allows to leverage several mechanisation principles that are available in **CaTT**, to reduce the proof effort required to derive results about the structure of identity types, such as the existence of an Eckmann-Hilton cell.

## 1 Introduction

Types in Martin-Löf type theory with intentional equality are weak  $\omega$ -groupoids [22]. This is one of the foundational observation of homotopy type theory [24] (**HoTT**). In this article, we work with a fragment of Martin-Löf type theory that we call **HoTT** to emphasise that we care about higher identity types, even though we do not use the univalence axiom or higher inductive types. Being weak  $\omega$ -groupoids, types in **HoTT** are in particular weak  $\omega$ -categories, by forgetting the preferred direction. On the other hand, Finster and Mimram [15] have introduced the dependent type theory **CaTT**, which is the theory of weak  $\omega$ -categories. **CaTT** and **HoTT** are both dependent type theories, one is the theory of  $\omega$ -categories while in the other one, every type is an  $\omega$ -category. This article establishes a formal connection between the two theories.

**Contributions.** In this article, we define a translation scheme that allows to transport **CaTT** terms onto **HoTT** terms, by viewing higher cells in **CaTT** as higher identity proofs in **HoTT**. We then prove the correctness of this translation. This means that the translation of a well-formed term in context in **CaTT** is a well-formed term in **HoTT**, whose type we characterise as the translation of the type of the original term in **CaTT**. The translation is defined by induction on

the syntax of  $\mathbf{CaTT}$ , and the main difficulty is to translate each of the operation making the  $\omega$ -category structure into successive applications of the  $\mathbf{J}$  rule in  $\mathbf{HoTT}$ . In the terminology of Boulier, Pedrot and Tabareau [9], we show that  $\mathbf{HoTT}$  is a syntactic model of  $\mathbf{CaTT}$ . We also present an implementation of this translation that generates terms in  $\mathbf{Coq}$  and show how it can be used in combination with mechanised reasoning available in  $\mathbf{CaTT}$  to reduce the proof effort required to generate certain complex terms in  $\mathbf{HoTT}$ .

**Related works.** Weak  $\omega$ -groupoids were first defined by Grothendieck [17] with the intent of modelling spaces up to homotopy. The connection between types and groupoids was first discovered by Hofmann and Streicher [18]. Lumsdaine [20], Van den Berg and Garner [25] and Altenkirch and Rypacek [1] then promoted this to  $\omega$ -groupoids, showing that types with their iterated identity types are endowed with the structure of weak  $\omega$ -groupoids. While mathematically weak  $\omega$ -groupoids are described as a collection of cells in every dimension that allow for partial composition operations satisfying associativity, unitality and exchange laws up to higher cells, in Martin-Löf type theory, the entire structure simply emerges from the  $\mathbf{J}$  rule. With a careful examination of how the  $\mathbf{J}$  rule yields a weak  $\omega$ -groupoid structure, Brunerie [11] proposed a definition of  $\omega$ -groupoids formulated as a dependent type theory.

Weak  $\omega$ -categories were first defined by Batanin [3]. The definition was then elaborated upon by Leinster [19]. More recently, Maltiniotis [21] proposed an alternative definition, which is based on Grothendieck’s definition of groupoids and modifies it by enforcing a privileged direction. This definition has been proven equivalent to that of Batanin and Leinster by Ara [2] and Bourke [10].

The definition the theory  $\mathbf{CaTT}$  by Finster and Mimram [15] was inspired by both Brunerie’s formulation of weak  $\omega$ -groupoids in dependent type theory, and Maltiniotis’ definition of weak  $\omega$ -categories from weak  $\omega$ -groupoids. Several works have been conducted around this theory. It was generalised by Dean et al. [14] to give an inductive definition of computads for weak  $\omega$ -categories, it was proven by Benjamin, Finster and Mimram [6] that the models of  $\mathbf{CaTT}$  are precisely Grothendieck-Maltiniotis  $\omega$ -categories, and several meta-operations for  $\mathbf{CaTT}$  have been proposed, such as the suspension and opposites [7], computation of inverses and invertibility witnesses [8], functorialisation [5], allowing for mechanised reasoning in the proof-assistant  $\mathbf{CaTT}$ .

**Overview of the paper.** In Section 2, we present the fragment of Martin-Löf type theory that we work with, which we call  $\mathbf{HoTT}$  in this article to emphasise the fact that we are interested in the structure of higher identity types. Section 3 is dedicated to the presentation of the dependent type theory  $\mathbf{CaTT}$ , which relies on a simpler type theory called  $\mathbf{GSeTT}$ . In Section 4, we present our algorithm to translate terms in  $\mathbf{GSeTT}$  into terms in  $\mathbf{HoTT}$ , and we promote this into a translation of terms in  $\mathbf{CaTT}$  into terms in  $\mathbf{HoTT}$  in Section 5. Both these translation come with proofs of correctness. Finally, in Section 6, we discuss the implementation aspects, and illustrate with the example of the Eckmann-

Hilbert cell how leveraging the mechanisation available in `CaTT` allow for defining complex terms in `HoTT`.

**Acknowledgements.** The author wants to thank Meven Lennon-Bertrand for his helpful guidance on the internal of `Coq`, making the implementation of the plugin possible, as well as Jamie Vicary, Ioannis Markakis, Chiara Sarti and Wilfred Offord, for the interesting discussions around the work presented in this paper.

## 2 Overview of Martin-Löf type theory

This section is dedicated to the presentation of the notation we use to work with type theory, as well as the fragment of Martin-Löf type theory that we place ourselves in and call `HoTT`.

### 2.1 Notations for type theories

We work with dependent type theories, made out of contexts, types, terms and substitutions, as well as definitional equalities. Our theories are presented with an untyped syntax with propositional judgements witnessing well-formedness. We use the following notation for these judgements:

$$\left\{ \begin{array}{ll} \Gamma \vdash \text{ctx} & \Gamma \text{ is a valid context} \\ \Gamma \vdash A \text{ type} & A \text{ is a valid type in } \Gamma \\ \Gamma \vdash u : A & u \text{ is a term of type } A \text{ in } \Gamma \\ \Delta \vdash \gamma :: \Gamma & \gamma \text{ is a substitution from } \Gamma \text{ to } \Delta \end{array} \right.$$

We call pre-contexts, pre-types, pre-terms and pre-substitutions the elements of the untyped syntax, which are not necessarily well-formed, to contrast with contexts, types terms and substitution which are implicitly assumed to be well-formed. We denote  $u = v$  the syntactic equality between pre-terms  $u$  and  $v$ , and  $u \equiv v$  the definitional equality between them. Syntactic equality compares the expressions whereas definitional equality can be subject to equality rules. The empty context is denoted  $\emptyset$ , and substitutions are written as  $\langle x \mapsto u \rangle$  where  $x$  is a variable and  $u$  is the term the variable  $x$  is substituted for. In the rest of this article, we introduce a three dependent type theories with which we work and we assume that they enjoy all of the usual structure, making their syntax into a category with families. This has already been shown for each of these, and more details regarding this structure is given in Appendix A

### 2.2 Martin-Löf type theory

In this article, we consider a fragment of Martin-Löf type theory with universes a la Tarski, identity types and  $\Pi$ -types. Since we are interested in the structure of higher identity types, we call this theory homotopy type theory, or `HoTT`, even

though our work does not require the univalence principle or higher inductive types. For the sake of clarity, we avoid the use of de Bruijn level and assume an countably infinite set of variables, that we denote  $x, y, \dots$ . We use the standard notations for the  $\Pi$ -types as  $\Pi(x: A).B$  and  $\lambda$ -applications as  $\lambda(x: A).u$ .

Our construction does not involve size issues, so we may assume that we are working with a universe of fixed size  $\mathcal{U}$ . The type constructor  $\mathbf{El}$  produces a type out of a term of type  $\mathcal{U}$ , as given by the following rule

$$\frac{\Gamma \vdash A : \mathcal{U}}{\Gamma \vdash \mathbf{El}(A) \text{ type}} \text{ (El-IN)}$$

Given a type  $A$  and terms  $u, v$ , we denote  $\mathbf{ld}_A(u, v)$  the identity type, and given just  $A$  and  $u$ , we denote  $\mathbf{refl}_{A,u}$  the reflexivity term. The introductions rules for identity types and reflexivity are the following:

$$\frac{\Gamma \vdash u : A \quad \Gamma \vdash v : A}{\Gamma \vdash \mathbf{ld}_A(u, v) \text{ type}} \text{ (ld-IN)} \qquad \frac{\Gamma \vdash u : A}{\Gamma \vdash \mathbf{refl}_{A,u} : \mathbf{ld}_A(u, u)} \text{ (refl-IN)}$$

We also use the Paulin-Mohring formulation of the  $\mathbf{J}$  constructor for identity types [23], whose introduction rule is given as follows:

$$\frac{\Gamma \vdash A \text{ type} \quad \Gamma \vdash u : A \quad \Gamma, x: A, y: A, e: \mathbf{ld}_A(x, y) \vdash P(x, y, e) \text{ type} \quad \Gamma \vdash p : P(u, u, \mathbf{refl}_{A,u})}{\Gamma \vdash \mathbf{J}(A, P, p) : \Pi(v: A)(e: \mathbf{ld}_A(u, v)).P(u, v, e)} \text{ (J-IN)}$$

The  $\mathbf{J}$  rule is subject to a computation rule. For the sake of simplicity, we write this as the following linear untyped definitional equality:

$$\mathbf{J}(A, u, P, p) v \mathbf{refl}_{B,w} \equiv p. \tag{\eta_J}$$

Any application of  $(\eta_J)$  in a context where both sides are well-typed must be instantiated with  $u, v, w$ , as well as  $A, B$  being definitionally equal, and thus coincide with the standard computation rule.

For any term  $u$  and any type  $A$  of  $\mathbf{HoTT}$  we define the  $\mathbb{N}$ -indexed family of types  $\mathbf{ld}_A^n(u)$  and the  $\mathbb{N}$ -indexed family of terms  $\mathbf{refl}_{A,u}^n$  by induction on  $n$  as follows:

$$\begin{aligned} \mathbf{ld}_A^0(u) &:= A & \mathbf{refl}_{A,u}^0 &:= u \\ \mathbf{ld}_A^{n+1}(u) &:= \mathbf{ld}_{\mathbf{ld}_A^n(u)}(\mathbf{refl}_{A,u}^n, \mathbf{refl}_{A,u}^n) & \mathbf{refl}_{A,u}^{n+1} &:= \mathbf{refl}_{\mathbf{ld}_A^n(u), \mathbf{refl}_{A,u}^n} \end{aligned}$$

These define valid types and terms in the theory  $\mathbf{HoTT}$ , in the following sense:

**Lemma 1.** *For every  $n \in \mathbb{N}$ , the following rules are admissible in  $\mathbf{HoTT}$ :*

$$\frac{\Gamma \vdash u : A}{\Gamma \vdash \mathbf{ld}_A^n(u) \text{ type}} \qquad \frac{\Gamma \vdash u : A}{\Gamma \vdash \mathbf{refl}_{A,u}^n : \mathbf{ld}_A^n(u)}$$

*Proof.* We prove this result by induction. The validity of the type  $\text{Id}_A^0(u)$  is a standard inversion lemma, and the typing of the term  $\text{refl}_{A,u}^0$  is exactly given by the premise of the rule. The validity of the type  $\text{Id}_A^{n+1}(u)$  is obtained from Rule (ld-IN) and the typing of the term  $\text{refl}_{A,u}^n$ . The typing of the term  $\text{refl}_{A,u}^{n+1}$  is obtained from Rule (refl-IN) and the typing of the term  $\text{refl}_{A,u}^n$ .  $\square$

Given a valid type  $\Gamma \vdash A$  **type** in  $\text{HoTT}$ , we define its  $\Pi$ -*lifting*  $\Pi_\Gamma.A$  to be the iterated  $\Pi$ -type with body  $A$  where all variables of  $\Gamma$  are successively bound by a  $\Pi$  constructor. Similarly, given a valid term  $\Gamma \vdash u : A$ , we define its  $\lambda$ -*lifting*  $\lambda_\Gamma.u$  to be the iterated  $\lambda$ -term with body  $u$  where all variables of  $\Gamma$  are successively bound by a  $\lambda$ -constructor.

**Lemma 2.** *The following rules are admissible in HoTT:*

$$\frac{\Gamma \vdash A \text{ type}}{\emptyset \vdash \Pi_\Gamma.A \text{ type}} \qquad \frac{\Gamma \vdash u : A}{\emptyset \vdash \lambda_\Gamma.u : \Pi_\Gamma.A}$$

*Proof.* The proof is a straightforward induction on the length of the context  $\Gamma$ , applying successively the introduction rules for  $\Pi$ -types and  $\lambda$ -terms.  $\square$

### 3 The dependent type theory CaTT

In this section, we present the dependent types theories GSeTT and CaTT. These are type-theoretic formulations of algebraic structures, namely globular sets and weak  $\omega$ -categories, using Cartmell's point of view of dependent type theories as generalised algebraic theories [12]. These type theories do not support any of the usual constructions that one typically encounters with Martin-Löf type theory, in particular they do not have universes, function types,  $\Pi$ -types or  $\Sigma$ -types. The type theories GSeTT and CaTT have the same type constructors, but GSeTT has no term constructor, so that the only terms in GSeTT are variables, and CaTT is an extension of GSeTT with term constructors.

#### 3.1 The type theory GSeTT

The type theories GSeTT and CaTT have two type constructors  $*$  and  $u \rightarrow_A v$  where  $A$  is a pre-type and  $u, v$  are pre-terms. The introduction rules state that  $*$  is always a valid type, while  $u \rightarrow_A v$  is subject to the same conditions as  $\text{Id}_A(u, v)$ . Formally the rules are the following:

$$\frac{\Gamma \vdash \text{ctx}}{\Gamma \vdash * \text{ type}} \text{ (*-IN)} \qquad \frac{\Gamma \vdash u : A \quad \Gamma \vdash v : A}{\Gamma \vdash u \rightarrow_A v \text{ type}} \text{ (\(\rightarrow$$
-IN)}

We define by induction the dimension of a pre-type of GSeTT or CaTT as follows:

$$\dim * = -1 \qquad \dim u \rightarrow_A v = \dim A + 1$$

By convention, the initialisation is at  $-1$ , to be consistent with notations used in topology. The dependent type theory GSeTT is the theory with those two type constructor, and no term constructors.

**Intuition on the semantics.** Contexts in the theory  $\mathbf{GSeTT}$  can be understood as a description of a structure called globular sets, which is a generalisation of a graph, allowing 2-dimensional arrows between the arrows, 3-dimensional arrows between the 2-dimensional ones, and so on. In globular sets,  $n$ -dimensional arrows are called  $n$ -cells. A context in  $\mathbf{GSeTT}$  describes a finite globular sets, by interpreting variables of type  $*$  as vertices, and variables of an arrow type as arrows between the prescribed source and target. The following illustrates an a few contexts and their corresponding globular sets, using a pictorial representation of those:

$$\left( \begin{array}{l} x: *, \\ f: x \rightarrow_* x \end{array} \right) \quad \left( \begin{array}{l} x: *, y: *, z: *, \\ f: x \rightarrow_* y, f': x \rightarrow_* y, g: z \rightarrow_* y, \\ \alpha: f \rightarrow_{x \rightarrow_* y} f' \end{array} \right)$$

A judgement  $\Gamma \vdash \mathbf{x} : A$  then corresponds to a choice of a particular cell in the globular cell corresponding to  $\Gamma$ , together with its iterated sources and targets. Adopting the presheaf view on globular sets, this is equivalent by the Yoneda lemma to a morphism of globular sets from  $D^n$  to the globular set corresponding to  $\Gamma$ , where  $D^n$  is the walking  $n$ -cell globular set, also called  $n$ -dimensional disk. A more in-depth exploration of this facts can be found in the works that directly deals with the semantics of  $\mathbf{CaTT}$  [15, 6].

### 3.2 The type theory $\mathbf{CaTT}$

The type theory  $\mathbf{CaTT}$  extends  $\mathbf{GSeTT}$  by adding term constructors. Intuitively these term constructors correspond to adding the higher categories operations to the globular sets described by the theory  $\mathbf{GSeTT}$ .

**Ps-contexts and their source and target.** The term constructors of the theory  $\mathbf{CaTT}$  are parameterised by a class of contexts of  $\mathbf{GSeTT}$  that we call *ps-contexts*. Those contexts are recognised by a judgement denoted  $\Gamma \vdash_{\text{ps}}$ , which is itself dependent on an auxiliary judgement  $\Gamma \vdash_{\text{ps}} x : A$  where  $x$  is a variable and  $A$  a type in  $\mathbf{GSeTT}$ . The derivation rules for these judgements are the following:

$$\frac{}{\Gamma \vdash_{\text{ps}} x : *} \text{ (PSS)} \quad \frac{\Gamma \vdash_{\text{ps}} x : A}{\Gamma, y : A, f : x \rightarrow_A y \vdash_{\text{ps}} f : x \rightarrow_a y} \text{ (PSE)}$$

$$\frac{\Gamma \vdash_{\text{ps}} f : x \rightarrow_A y}{\Gamma \vdash_{\text{ps}} y : A} \text{ (PSD)} \quad \frac{\Gamma \vdash_{\text{ps}} x : *}{\Gamma \vdash_{\text{ps}}} \text{ (PS)}$$

Additionally, we also require that all variables bound by ps-contexts are different. Intuitively, ps-contexts are the contexts of  $\mathbf{GSeTT}$  that do not present holes, and have a global directionality.

**Lemma 3.** *Ps-contexts are valid GSeTT contexts. More precisely, the following rules are admissible in GSeTT*

$$\frac{\Gamma \vdash_{ps}}{\Gamma \vdash \text{ctx}} \qquad \frac{\Gamma \vdash_{ps} x : A}{\Gamma \vdash x : A}$$

This result proven in [6] shows that ps-context correspond to a particular class of globular sets. They can be recognised by the existence of a total order on their cells [26, 15], and correspond to situations that can be contracted in a directed sense. For instance the following contexts are ps-contexts (illustrated here with their corresponding globular set):

$$\left( \begin{array}{l} x : *, y : *, f : x \rightarrow_* y, \\ z : *, g : y \rightarrow_* z \end{array} \right) \qquad W := \left( \begin{array}{l} x : *, y : *, f : x \rightarrow_* y, \\ g : x \rightarrow_* y, a : f \rightarrow_{x \rightarrow_* y} g, \\ z : *, h : y \rightarrow_* z \end{array} \right)$$

$$x \xrightarrow{f} y \xrightarrow{g} z \qquad x \begin{array}{c} \xrightarrow{f} \\ \Downarrow a \\ \xrightarrow{g} \end{array} y \xrightarrow{h} z$$

Given a ps-context  $\Gamma$  and an integer  $i$ , we define two sub-contexts  $\partial_i^- \Gamma$  and  $\partial_i^+ \Gamma$  called respectively the  $i$ -source and the  $i$ -target of  $\Gamma$  as follows:

$$\begin{aligned} \partial_i^-(x : *) &:= (x : *) \\ \partial_i^-(\Gamma, y : A, f : x \rightarrow_A y) &:= \begin{cases} \partial_i^- \Gamma & \text{if } \dim A + 1 \geq i \\ (\partial_i^- \Gamma, y : A, f : x \rightarrow_A y) & \text{otherwise} \end{cases} \\ \partial_i^+(x : *) &:= (x : *) \\ \partial_i^+(\Gamma, y : A, f : x \rightarrow_A y) &:= \begin{cases} \partial_i^- \Gamma & \text{if } \dim A + 1 > i \\ (\text{tail}(\partial_i^- \Gamma), y : A) & \text{if } \dim A + 1 > i \\ (\partial_i^- \Gamma, y : A, f : x \rightarrow_A y) & \text{otherwise} \end{cases} \end{aligned}$$

where tail is the operator that removes the top first element of the list. This definition illustrates an important principle when reasoning on ps-contexts, only allow to start with a one-element list and add two elements at one time, performing structural induction on a derivation that a context is a pasting scheme amounts to reasoning by induction on the pasting scheme seen as an odd-sized list. This is because there exists at most one derivation than a given context is a ps-context [4]. We illustrate the sources and targets of the ps-context  $W$  introduced above:

$$\begin{array}{cccc} \partial_1^- W & \partial_1^+ W & \partial_0^- W & \partial_0^+ W \\ \left( \begin{array}{l} x : *, y : *, f : x \rightarrow_* y, \\ z : *, h : y \rightarrow_* z \end{array} \right) & \left( \begin{array}{l} x : *, y : *, g : x \rightarrow_* y, \\ z : *, h : y \rightarrow_* z \end{array} \right) & (x : *) & (z : *) \\ x \xrightarrow{f} y \xrightarrow{h} z & x \begin{array}{c} \xrightarrow{g} \\ \xrightarrow{h} \end{array} y & x & z \end{array}$$

For any  $i \geq 2$ , the  $i$ -sources and  $i$ -targets of  $W$  are equal to  $W$ . This is a generic fact: Given a ps-context  $\Gamma$  and  $i \geq \dim \Gamma$ , the  $i$ -sources and targets of  $\Gamma$  are equal to  $\Gamma$ .

**Term constructors.** The theory  $\text{CaTT}$  has a family of term constructors  $\text{coh}_{\Gamma,A}$ , where  $\Gamma$  is a ps-context and  $A$  is a type. Not all pairs  $(\Gamma, A)$  of contexts and types yield a valid term constructor, so we introduce a judgement  $\vdash \text{coh}_{\Gamma,A}$  to signify when this is the case. We sometimes call the *coherences* the valid term constructors  $\vdash \text{coh}_{\Gamma,A}$ , and the following derivation rule characterises these coherences, for  $i \in \{\dim \Gamma - 1, \dim \Gamma\}$ :

$$\frac{\Gamma \vdash_{\text{ps}} \quad \partial_i^- \Gamma \vdash u : A \quad \partial_i^+ \Gamma \vdash v : A}{\vdash \text{coh}_{\Gamma, u \rightarrow_A v}} \text{ (coh-wd)} \quad \begin{cases} \text{Var}(u : A) = \text{Var}(\partial_i^- \Gamma) \\ \text{Var}(v : A) = \text{Var}(\partial_i^+ \Gamma). \end{cases}$$

Here,  $\text{Var}(u : A)$  denotes the union of all variables used by the term  $u$  and the type  $A$ , and  $\text{Var} \Gamma$  denotes all variables bound in context  $\Gamma$ . The term constructors are subject to the following introduction rule:

$$\frac{\vdash \text{coh}_{\Gamma,A} \quad \Delta \vdash \gamma :: \Gamma}{\Delta \vdash \text{coh}_{\Gamma,A}[\gamma] : A[\gamma]} \text{ (coh-IN)}$$

*Remark 1.* Some presentations of  $\text{CaTT}$  use two different term constructors or two different introduction rules for  $\text{coh}$ , and sometimes Rule (coh-wd) and Rule (coh-IN) are combined into a single rule. It is straightforward that those are equivalent, and the presentation we give here is better suited for the work in this article.

### 3.3 Implementation and intuition on the semantics

We have implemented in OCaml a type checker for the theory  $\text{CaTT}$ , that we also call  $\text{CaTT}^1$ . This plays the role of a proof assistant specifically dedicated to working with a mathematical structure called  $\omega$ -categories [21]. In this proof assistant the user may declare the coherence  $\text{coh}_{\Gamma,A}$  with the following syntax

```
coh [name] [ps-context] : [type-expr]
```

and define the term  $\Gamma \vdash u : A$  using one of the following syntax

```
let [name] [context] = [body]
let [name] [context] : [type-expr] = [body]
```

where **[name]** is a string representing a name given to the coherence of the term, **[context]** and **[ps-context]** are description of the context  $\Gamma$ , **type-expr** is the type expression representing  $A$  and **body** is the term expression representing

<sup>1</sup><https://www.github.com/thibautbenjamin/catt>



t. For instance, consider the following ps-context (represented with the corresponding diagram) and coherence (where the output type), representing the composition of 1-cells

$$\Gamma_2 := \left( \begin{array}{l} x: *, y: *, f: x \rightarrow_* y, \\ z: *, g: y \rightarrow_* z \end{array} \right) \quad \text{comp} := \text{coh}_{\Gamma_2, x \rightarrow_* z}$$

This coherence can be declared with the name `comp` in the proof-assistant `CaTT` using the following input:

```
coh comp (x y : *) (f : x -> y) (z : *) (g : y -> z) : x -> z
```

In the proof arguments, contexts are declared as list of parenthesised items, and in the type  $u \rightarrow_A v$ , the type  $A$  is always left implicit and inferred from  $u$  and  $v$ . Consider now the following context together with the term, representing the way to compose three 1-cells together by first composing the first two then composing the result with the last cell

$$\Gamma_3 := \left( \begin{array}{l} x: *, y: *, f: x \rightarrow_* y, \\ z: *, g: y \rightarrow_* z \\ w: *, g: z \rightarrow_* w \end{array} \right) \quad \Gamma_3 \vdash \text{lcomp} : x \rightarrow_* w$$

$$\text{lcomp} := \text{comp} \left[ \left\langle \begin{array}{l} x \mapsto x, y \mapsto z, f \mapsto \text{comp} \\ z \mapsto w, g \mapsto h \end{array} \right\rangle, \left[ \left\langle \begin{array}{l} x \mapsto x, y \mapsto y, f \mapsto f \\ z \mapsto z, g \mapsto g \end{array} \right\rangle \right] \right]$$

This term can be declared in the proof assistant `CaTT` with the following input:

```
let lcomp (x y z w : *) (f : x -> y) (g : y -> z) (h : z -> w)
  = comp (comp f g) h
```

In this declaration, since the context need not be a ps-context, we grouped together all variables of type `*` for simplification. This also illustrates that substitutions are written like applications, and that a lot of arguments are left implicit. The proof assistant `CaTT` can automatically infer which arguments should be implicit, matching the usual notational conventions where the domains and codomains of functions are implicit. As hinted by these examples, terms in `CaTT` correspond to operations on weak  $\omega$ -categories. Coherences represent primitive operations while terms are cells obtained by using those primitive operations. Here we use the word operation in a liberal sense. Consider the following coherence

```
coh assoc (x : * ) (y : *) (f : x -> y)
          (z : *) (g : x -> y)
          (w : *) (h : y -> z)
  : comp (comp f g) h -> comp f (comp g h)
```

This coherence define a cell known as the associator, which witnesses that the composition is associative, but in a weak sense. That is, the two terms `comp (comp f g) h` and `comp f (comp g h)` are not definitionally equal, but

GSeTT term in context	closed $\lambda$ -term
$x : * \vdash x : *$	$\lambda(\mathbf{B}:\mathcal{U})(x:\mathbf{B}).x$
$x : *, f : x \rightarrow x \vdash f : x \rightarrow x$	$\lambda(\mathbf{B}:\mathcal{U})(x:\mathbf{B})(f:\text{Id}_{\mathbf{B}}(x,x)).f$
$x : *, y : *, f : x \rightarrow y \vdash y : *$	$\lambda(\mathbf{B}:\mathcal{U})(x:\mathbf{B})(y:\mathbf{B})(f:\text{Id}_{\mathbf{B}}(x,y)).y$

Figure 1: Translation from GSeTT terms to HoTT

related by this higher cell, which happens to be invertible in some appropriate sense [13, 16, 8].

A formal connection between the semantics of CaTT and the theory of weak  $\omega$ -categories has been established [6], showing that CaTT allows one to work directly in the theory of weak  $\omega$ -categories. Since types in HoTT are weak  $\omega$ -groupoids [20, 25, 1], they are in particular weak  $\omega$ -categories. The next two sections are dedicated to expanding in the connection between HoTT and CaTT, by showing that HoTT is a syntactic model of CaTT, that is, by defining a translation from the syntax of HoTT to that of CaTT and showing that this translation preserves the typing judgements.

## 4 Translation of GSeTT terms into HoTT

Before presenting the translation of CaTT terms into HoTT, we present that of GSeTT terms into HoTT. Not only this presentation is useful as a buildup towards the translation of CaTT terms in a simpler setting, it is also directly used by that translation, as the coherence rule of CaTT terms depend on derivation in GSeTT through ps-contexts. From now on, in order to simplify the notations, we assume the following on variables:

- Variables names in GSeTT and in CaTT are the same
- Every variable name in CaTT or GSeTT is also a variable name in HoTT
- There are variable names in HoTT that are not valid names in CaTT.

These assumptions are possible: if we assume both sets of variables to be countably infinite, there exists a non-bijective injection from variable names in CaTT to variable names of HoTT. By convention we use the font  $\mathbf{x}$  to denote a variable in GSeTT or CaTT and the font  $x$  to denote a variable in HoTT, and we use the name of the variable to keep track of the names.

We consider a term judgement  $\Gamma \vdash \mathbf{x} : A$  in GSeTT. Our translation interprets the variable  $\mathbf{x}$  as a closed term representing the projection of the interpretation of the context  $\Gamma$  onto the variable  $\mathbf{x}$ . The context  $\Gamma$  is understood as describing the arguments of the function, over a generic type  $\mathbf{B}$ , where variables of type  $*$  represent arguments of type  $\mathbf{B}$  and variables of type  $\mathbf{y} \rightarrow \mathbf{z}$  represent equalities between the corresponding variables. This translation is best understood by considering a few simple examples. Figure 1 illustrates it on a few GSeTT terms, displayed with their corresponding closed  $\lambda$ -terms. As illustrated by this

figure, when no term constructor is involved, all the functions resulting from the translation are projections, they just return one of their arguments unchanged and discard all the other arguments. Although this case does not produce interesting functions, it lets us illustrate a few principles about the translation. First, it shows how the type of objects  $*$  is translated into a reference to a universally quantified type  $\mathbf{B}$ , so the functions resulting from the translations must take  $\mathbf{B}$  as their first argument. Secondly, it shows how a  $\mathbf{GSeTT}$  context can be understood as describing an sequence of arguments that are all of type  $\mathbf{B}$  or of an higher identity type above those.

To define the translation, we first fix a variable  $\mathbf{B}$  in  $\mathbf{HoTT}$  whose not is not a valid name in  $\mathbf{GSeTT}$ , and define translation functions  $\llbracket \_ \rrbracket_{\mathbf{B}}$ , which associate to a context (resp. a type) of  $\mathbf{CaTT}$ , a context (resp. a type) in  $\mathbf{HoTT}$ . These function are defined by induction as follows:

$$\left\{ \begin{array}{ll} \llbracket \emptyset \rrbracket_{\mathbf{B}} & := (\mathbf{B} : \mathcal{U}) \\ \llbracket (\Gamma . \mathbf{x} : A) \rrbracket_{\mathbf{B}} & := (\llbracket \Gamma \rrbracket_{\mathbf{B}} . \mathbf{x} : \llbracket A \rrbracket_{\mathbf{B}}) \end{array} \right. \quad \left\{ \begin{array}{ll} \llbracket * \rrbracket_{\mathbf{B}} & := \mathbf{El}(\mathbf{B}) \\ \llbracket \mathbf{x} \rightarrow_A \mathbf{y} \rrbracket_{\mathbf{B}} & := \mathbf{Id}_{\llbracket A \rrbracket_{\mathbf{B}}}(\mathbf{x}, \mathbf{y}) \end{array} \right.$$

**Lemma 4.** *Translating derivable contexts, types and variables in  $\mathbf{GSeTT}$  yields derivable contexts, types and variables in  $\mathbf{HoTT}$ . More precisely, we have:*

- *For every derivable context judgement  $\Gamma \vdash \text{ctx}$  in  $\mathbf{GSeTT}$ , the judgement  $\llbracket \Gamma \rrbracket_{\mathbf{B}} \vdash \text{ctx}$  is derivable in  $\mathbf{HoTT}$ .*
- *For any type judgement  $\Gamma \vdash A$  type derivable in  $\mathbf{GSeTT}$ , the judgement  $\llbracket \Gamma \rrbracket_{\mathbf{B}} \vdash \llbracket A \rrbracket_{\mathbf{B}}$  type is derivable in  $\mathbf{HoTT}$ .*
- *For any term judgement  $\Gamma \vdash \mathbf{x} : A$  derivable in  $\mathbf{GSeTT}$ , the judgement  $\llbracket \Gamma \rrbracket_{\mathbf{B}} \vdash \mathbf{x} : \llbracket A \rrbracket_{\mathbf{B}}$  is derivable in  $\mathbf{HoTT}$ .*

*Proof.* This can be proved by induction on the derivation tree of the judgement in the theory  $\mathbf{GSeTT}$ . See Appendix B for a complete proof.  $\square$

Given a derivable type judgement  $\Gamma \vdash A$  type in  $\mathbf{GSeTT}$ , we define the closed type  $\llbracket \Gamma \vdash A \text{ type} \rrbracket$  as the  $\Pi$ -lifting of the type judgement  $\llbracket \Gamma \rrbracket_{\mathbf{B}} \vdash \llbracket A \rrbracket_{\mathbf{B}}$  type in  $\mathbf{HoTT}$ . Similarly, for a derivable term judgement  $\Gamma \vdash \mathbf{x} : A$  in  $\mathbf{GSeTT}$ , we define the closed term  $\llbracket \Gamma \vdash \mathbf{x} : A \rrbracket$  to be the  $\lambda$ -lifting of the derivable term judgement  $\llbracket \Gamma \rrbracket_{\mathbf{B}} \vdash \mathbf{x} : \llbracket A \rrbracket_{\mathbf{B}}$  in  $\mathbf{HoTT}$ .

$$\llbracket \Gamma \vdash A \text{ type} \rrbracket := \Pi_{\llbracket \Gamma \rrbracket_{\mathbf{B}}} . \llbracket A \rrbracket_{\mathbf{B}} \quad \llbracket \Gamma \vdash \mathbf{x} : A \rrbracket := \lambda_{\llbracket \Gamma \rrbracket_{\mathbf{B}}} . \mathbf{x}$$

The choice of the variable  $\mathbf{B}$  is irrelevant as any two choices lead to  $\alpha$ -equivalent terms, hence the variable disappears from the notation.

**Proposition 5.** *Consider a judgement  $\Gamma \vdash \mathbf{x} : A$  derivable in  $\mathbf{CaTT}$  where no term constructor appears. Then the closed  $\lambda$ -term  $\llbracket \Gamma \vdash \mathbf{x} : A \rrbracket$  is valid in  $\mathbf{HoTT}$ . More precisely, the following judgement is derivable in  $\mathbf{HoTT}$ :*

$$\emptyset \vdash \llbracket \Gamma \vdash \mathbf{x} : A \rrbracket : \llbracket \Gamma \vdash A \text{ type} \rrbracket.$$

*Proof.* This is a consequence of Lemmas 2 and 4  $\square$

## 5 Translation of CaTT terms into HoTT

In this section, we extend our translation to a translation from CaTT term to closed  $\lambda$ -term in HoTT. Intuitively, a CaTT term in a context is interpreted in HoTT as a function taking arguments as described by the CaTT context, and returning a value determined by the term. As in the previous section, we start by defining functions that perform the translation in contexts, and then we take the  $\lambda$ -lifting. From now on, we consider a variable  $B$  that is a variable in HoTT which is not a name in CaTT. The translation functions  $\llbracket \_ \rrbracket_B$  are now more numerous and more complex. Not only do we consider translations of contexts and types, but also of terms, substitutions and coherences. The translation function on coherences sends a coherence of CaTT onto a closed term in HoTT. These functions are all defined by mutual induction. The cases for the contexts, types, terms and substitutions are straightforward by induction and given by the following:

$$\begin{cases} \llbracket \emptyset \rrbracket_B & := (B : \mathcal{U}) \\ \llbracket (\Gamma . x : A) \rrbracket_B & := (\llbracket \Gamma \rrbracket_B . x : \llbracket A \rrbracket_B) \end{cases} \quad \begin{cases} \llbracket * \rrbracket_B & := \text{El}(B) \\ \llbracket t \rightarrow_A u \rrbracket_B & := \text{Id}_{\llbracket A \rrbracket_B}(\llbracket t \rrbracket_B, \llbracket u \rrbracket_B) \end{cases}$$

$$\begin{cases} \llbracket x \rrbracket_B & := x \\ \llbracket \text{coh}_{\Gamma, A}[\gamma] \rrbracket_B & := (\llbracket \text{coh}_{\Gamma, A} \rrbracket_B) \llbracket \llbracket \gamma \rrbracket_B \rrbracket \end{cases} \quad \begin{cases} \llbracket \langle \rangle \rrbracket_B & := \langle B \mapsto B \rangle \\ \llbracket \langle \gamma, x \mapsto t \rangle \rrbracket_B & := \langle \llbracket \gamma \rrbracket_B, x \mapsto \llbracket t \rrbracket_B \rangle. \end{cases}$$

The definition of  $\llbracket \text{coh}_{\Gamma, A} \rrbracket_B$  is left in order to complete the mutual induction. We define this case as  $\llbracket \text{coh}_{\Gamma, A} \rrbracket_B := \text{elim}_B(\Gamma, A, \Gamma, \text{id}_{\llbracket \Gamma \rrbracket_B})$ , where  $\text{elim}$  is the auxiliary function defined mutually inductively, by induction on the derivation of the ps-context:

$$\begin{cases} \text{elim}_B((x : *), A, \Gamma, \gamma) & := \text{refl}_{B, x}^{\dim A + 1} \\ \text{elim}_B((\Delta, y : C, f : x \rightarrow_C y), A, \Gamma, \gamma) & := J(\llbracket C \rrbracket_B, x, P(x', y', f'), p) y f \end{cases}$$

where  $P$  and  $p$  are respectively defined as follows

$$\begin{aligned} P(x', y', f') &:= \llbracket A \rrbracket_B[\gamma] \langle \text{id}_{\llbracket (\Delta, y : C, f : x \rightarrow_C y) \rrbracket_B}, x \mapsto x', y \mapsto y', f \mapsto f' \rangle \\ p &:= \text{elim}_B(\Delta, A, \Gamma, \gamma \circ \langle \text{id}_{\llbracket \Delta \rrbracket_B}, y \mapsto x, f \mapsto \text{refl}_{\llbracket C \rrbracket_B, x} \rangle). \end{aligned}$$

Intuitively, in the expression  $\text{elim}(\Delta, A, \Gamma, \gamma)$ , the context  $\Gamma$  is meant to represent the ambient ps-context we started from,  $\Delta$  is meant to represent a sub-context of  $\Gamma$  obtained by successively peeling off variables from  $\Gamma$ , and  $\gamma$  a substitution mapping variables of  $\llbracket \Gamma \rrbracket_B$  to variables of  $\llbracket \Delta \rrbracket_B$  or reflexivity witnesses of such.

In this mutually inductive definition, all the inductive are strictly decreasing, except in the case for  $\text{elim}$  where in the inductive the structural height of the type argument  $A$  is constant, but the first argument is strictly decreasing. The induction is thus well founded, and proceeds by structural induction on the syntax, and upon hitting a term constructor  $\text{coh}_{\Gamma, A}$  temporarily switches to an induction on the length of  $\Gamma$  before resuming the structural induction. This inductive scheme is also used for the proof of correctness of Proposition 8. Note

that in the inductive case defining  $\text{elim}$ , the type  $C$  is the type of a ps-context, and thus a type in  $\text{GSeTT}$ , so the expression  $\llbracket C \rrbracket_{\mathbb{B}}$  denotes the previously defined translation on  $\text{GSeTT}$  and it not an inductive call.

**Lemma 6.** *The proposed translation scheme respects substitution application at the level of the pre-syntax. More precisely, the following results hold:*

- For all pre-type  $A$  and all pre-substitution  $\gamma$  in  $\text{CaTT}$ , we have the following equality of pre-types in  $\text{HoTT}$ :  $\llbracket A[\gamma] \rrbracket_{\mathbb{B}} = \llbracket A \rrbracket_{\mathbb{B}} \llbracket \llbracket \gamma \rrbracket_{\mathbb{B}} \rrbracket_{\mathbb{B}}$ .
- For all pre-term  $t$  and all pre-substitution  $\gamma$  in  $\text{CaTT}$ , we have the following equality of pre-terms in  $\text{HoTT}$ :  $\llbracket t[\gamma] \rrbracket_{\mathbb{B}} = \llbracket t \rrbracket_{\mathbb{B}} \llbracket \llbracket \gamma \rrbracket_{\mathbb{B}} \rrbracket_{\mathbb{B}}$ .
- For all pairs of pre-substitutions  $\delta, \gamma$  in  $\text{CaTT}$ , we have the following equality of pre-substitutions in  $\text{HoTT}$ :  $\llbracket \delta \circ \gamma \rrbracket_{\mathbb{B}} = \llbracket \delta \rrbracket_{\mathbb{B}} \circ \llbracket \gamma \rrbracket_{\mathbb{B}}$ .

*Proof.* One can check this property by structural induction on the pre-syntax of  $\text{CaTT}$ , see Appendix C.  $\square$

**Lemma 7.** *For every term  $(\mathbf{x}:*) \vdash u : A$  in  $\text{CaTT}$ , we have  $\llbracket u \rrbracket_{\mathbb{B}} \equiv \text{ref}_{\mathbb{B}, \mathbf{x}}^{\dim A+1}$ . Similarly, for every type  $(\mathbf{x}:*) \vdash A$  type in  $\text{CaTT}$ , we have  $\llbracket A \rrbracket_{\mathbb{B}} \equiv \text{ld}_{\mathbb{B}, \mathbf{x}}^{\dim A+1}$ .*

*Proof.* The result on term can be proven by induction together with a corresponding for the auxiliary operation  $\text{elim}$ . The case of  $\text{elim}$  is itself an induction on the length of the ps-context given by the first argument, and amounts to successive applications of the computation rule  $(\eta_j)$ . The result on types is a consequence of that of terms. See Appendix C for more details.  $\square$

**Proposition 8.** *The following results hold:*

- For any context  $\Gamma \vdash \text{ctx}$  in  $\text{CaTT}$ , we have  $\llbracket \Gamma \rrbracket_{\mathbb{B}} \vdash \text{ctx}$  in  $\text{HoTT}$ .
- For any type  $\Gamma \vdash A$  type in  $\text{CaTT}$ , we have  $\llbracket \Gamma \rrbracket_{\mathbb{B}} \vdash \llbracket A \rrbracket_{\mathbb{B}}$  type in  $\text{HoTT}$ .
- For any term  $\Gamma \vdash t : A$  in  $\text{CaTT}$ , we have  $\llbracket \Gamma \rrbracket_{\mathbb{B}} \vdash \llbracket t \rrbracket_{\mathbb{B}} : \llbracket A \rrbracket_{\mathbb{B}}$  in  $\text{HoTT}$ .
- For any substitution  $\Delta \vdash \gamma :: \Gamma$  in  $\text{CaTT}$ , we have  $\llbracket \Delta \rrbracket_{\mathbb{B}} \vdash \llbracket \gamma \rrbracket_{\mathbb{B}} :: \llbracket \Gamma \rrbracket_{\mathbb{B}}$  in  $\text{HoTT}$ .
- For any coherence  $\vdash \text{coh}_{\Gamma, A}$ , we have  $\llbracket \Gamma \rrbracket_{\mathbb{B}} \vdash \llbracket \text{coh}_{\Gamma, A} \rrbracket_{\mathbb{B}} : \llbracket A \rrbracket_{\mathbb{B}}$  in  $\text{HoTT}$ .
- For any pair of ps-contexts  $\Gamma, \Delta$  together with a type  $\Gamma \vdash A$  type in  $\text{CaTT}$  and a substitution  $\llbracket \Delta \rrbracket_{\mathbb{B}} \vdash \gamma :: \llbracket \Gamma \rrbracket_{\mathbb{B}}$  in  $\text{HoTT}$ , the following judgement is derivable in  $\text{HoTT}$ :

$$\llbracket \Delta \rrbracket_{\mathbb{B}} \vdash \text{elim}_{\mathbb{B}}(\Delta, A, \Gamma, \gamma) : \llbracket A \rrbracket_{\mathbb{B}}[\gamma]$$

*Proof.* One can prove this result by mutual induction on the derivation and induction on the length of the ps-context for the last case, following the induction scheme defining the translation. The case of elim is the interesting one, where the case of a ps-context with a single object comes from Lemma 7, and for the inductive case, we need to check that the application of the J rule is valid and has the desired type. This is done by induction, and manipulation of the judgements with usual lemmas about the structure of the type theory, such as cut admissibility and the functoriality of the action of substitutions. A complete proof is given in Appendix C.  $\square$

We now proceed take the liftings of the translation of types and terms in order to obtain a translation from CaTT terms to closed terms in HoTT. Given a type  $\Gamma \vdash A \text{ type}$  in CaTT, we denote  $\llbracket \Gamma \vdash A \text{ type} \rrbracket$  the  $\Pi$ -lifting of the type  $\llbracket \Gamma \rrbracket_{\mathbf{B}} \vdash \llbracket A \rrbracket_{\mathbf{B}} : \text{in HoTT}$ , and given a term  $\Gamma \vdash t : A$  in CaTT, we denote  $\llbracket \Gamma \vdash t : A \rrbracket$  the  $\lambda$ -lifting of the term  $\llbracket \Gamma \rrbracket_{\mathbf{B}} \vdash \llbracket t \rrbracket_{\mathbf{B}} : \llbracket A \rrbracket_{\mathbf{B}}$  in HoTT. This is independent from the choice of the variable  $\mathbf{B}$ .

**Theorem 9.** *Given a derivable term  $\Gamma \vdash t : A$  in CaTT, the closed  $\lambda$ -term  $\llbracket \Gamma \vdash t : A \rrbracket$  is valid in HoTT. More precisely, the following judgement is derivable in HoTT:*

$$\emptyset \vdash \llbracket \Gamma \vdash t : A \rrbracket : \llbracket \Gamma \vdash A \text{ type} \rrbracket.$$

*Proof.* This is consequence of Lemma 2 and Proposition 8  $\square$

## 6 Leveraging automation

We have now finished the presentation of our translation scheme, and we conclude with a discussion around the implementation of this translation scheme, and briefly present mechanisation principles available in CaTT, to finish with an example where we generate a proof term in HoTT leveraging this mechanisation, and compare it with a definition of a similar term directly in HoTT.

### 6.1 Implementation

We have implemented the translation algorithm presented in Section 5, in the form of a plugin for the Coq proof assistant, named CaTT plugin<sup>2</sup>. This plugin extends the syntax of Coq with a new primitive `Catt`, that can be used with the following syntax:

```
Catt [names] From File [path]
```

where `[path]` is the path of a CaTT file and `[names]` is a list of space separated string corresponding to names of declared coherences and terms in the CaTT file. Upon execution, this command uses as a backend CaTT to parse and type-check the provided file, and then uses the translation algorithm to generate the terms in Coq corresponding to the specified coherences and terms. The choice of Coq

<sup>2</sup><https://www.github.com/thibautbenjamin/catt>

was motivated by the extensibility of its plugin system, and the ease of tooling, since both `Coq` and our implementation of `CaTT` are written in `OCaml`. In order to generate terms, our implementation interfaces directly with the `Coq` kernel. Following `Coq`'s internals, our implemented algorithm uses pattern-matching of identities against the reflexivity terms instead of application of the `J` rule.

Our implementation is still experimental, and could be improved in many ways. First, as the time of writing, we always expand all applied terms down to coherences. In practice, this means every function application is inline, making the type checking by `Coq` computationally heavy.

## 6.2 Mechanisation principles in `CaTT`

Despite `HoTT` terms interpret `CaTT` terms, there are still three main differences between the work in `CaTT` and the work in `HoTT`. First, `CaTT` is a theory for weak  $\omega$ -categories, and thus is inherently directed, while in `HoTT` the identity types satisfy symmetry. Second, `CaTT` is fully weak, that is terms in `CaTT` never reduce to other terms, while in `HoTT`, it is always the case that the composition of two reflexivity definitionally reduces to a reflexivity. Finally, in `HoTT`, all the operations that we define on identity types are polymorphic, allowing to lift them to identity types on identity types themselves, and so on, while in `CaTT`, the type `*` can only ever be substituted for itself. This may indicate that working in `CaTT` is harder than working in `HoTT`. However, this added complexity is largely offset by the various mechanisation features that implemented in `CaTT`, which are only permitted by the simplicity of the language. The suspension meta-operation in `CaTT` [7] exactly accounts for the lack of polymorphism. The automatic computation of inverses and cancellation witnesses [8] accounts for the lack of symmetry when the cells still happen to be invertible. Other meta-operations, such as the opposites [7] and the functorialisation [5] account for a wide range of phenomena that are not accounted for in `HoTT`. These meta-operations are leveraged in practical uses of `CaTT` to construct complex terms with a minimal proof effort.

## 6.3 Quantitative experiment: the Eckmann-Hilton cell

As an example to illustrate our point, we present here the Eckmann-Hilton cell. This is a cell that formalises the Eckmann-Hilton argument, playing an important role in topology. It can be derived in `HoTT` by considering a term  $x$  of any type  $A$  and two terms  $a, b$  of type  $\text{Id}(\text{refl}_{A,x}, \text{refl}_{A,x})$ , and is a witness that inhabit the identity type between the transitivity of  $a$  and  $b$ , and the transitivity of  $b$  and  $a$ . Deriving this cell is done by purely algebraic manipulation of identity types, and can be done in `CaTT`. Figure 3 illustrates the intuitive idea allowing to define this cell, where the one dimensional cells are all identities. This figure is only for intuition, as it is missing a lot of data, such as the associators and unitors.

We have a `CaTT` definition of the Eckmann-Hilton cell using all the mechanisation features mentioned above (see Appendix D), using 786 characters. The





## References

- [1] Thorsten Altenkirch and Ondrej Rypacek. A syntactical approach to weak omega-groupoids. Schloss-Dagstuhl-Leibniz Zentrum für Informatik.
- [2] Dimitri Ara. Sur les infini-groupoïdes de Grothendieck et une variante infini-catégorique.
- [3] Michael A Batanin. Monoidal globular categories as a natural environment for the theory of weakn-categories. 136(1):39–103.
- [4] Thibaut Benjamin. Formalization of dependent type theory: The example of CaTT.
- [5] Thibaut Benjamin. A type theoretic approach to weak  $\omega$ -categories and related higher structures.
- [6] Thibaut Benjamin, Eric Finster, and Samuel Mimram. Globular weak  $\omega$ -categories as models of a type theory.
- [7] Thibaut Benjamin and Ioannis Markakis. Hom  $\omega$ -categories of a computad are free.
- [8] Thibaut Benjamin and Ioannis Markakis. Invertible cells in  $\omega$ -categories.
- [9] Simon Boulier, Pierre-Marie Pédrot, and Nicolas Tabareau. The next 700 syntactical models of type theory. pages 182–194.
- [10] John Bourke. Iterated algebraic injectivity and the faithfulness conjecture.
- [11] Guillaume Brunerie. On the homotopy groups of spheres in homotopy type theory.
- [12] John Cartmell. Generalised algebraic theories and contextual categories. 32:209–243.
- [13] Eugenia Cheng. An  $\omega$ -category with all duals is an  $\omega$ -groupoid. 15:439–453.
- [14] Christopher J. Dean, Eric Finster, Ioannis Markakis, David Reutter, and Jamie Vicary. Computads for weak  $\omega$ -categories as an inductive type.
- [15] Eric Finster and Samuel Mimram. A type-theoretical definition of weak  $\omega$ -categories. pages 1–12. IEEE.
- [16] Soichiro Fujii, Keisuke Hoshino, and Yuki Maehara.  $\omega$ -weak equivalences between weak  $\omega$ -categories.
- [17] Alexander Grothendieck. Pursuing stacks.
- [18] Martin Hofmann and Thomas Streicher. The groupoid interpretation of type theory. 36:83–111.

- [19] Tom Leinster. *Higher Operads, Higher Categories*. Number 298. Cambridge University Press.
- [20] Peter LeFanu Lumsdaine. Weak omega-categories from intensional type theory. Volume 6, Issue 3:1062.
- [21] Georges Maltsiniotis. Grothendieck  $\infty$ -groupoids, and still another definition of  $\infty$ -categories.
- [22] Per Martin-Löf and Giovanni Sambin. *Intuitionistic Type Theory*, volume 9. Bibliopolis Naples.
- [23] Christine Paulin-Mohring. Inductive definitions in the system Coq rules and properties. pages 328–345. Springer.
- [24] The {Univalent Foundations Program}. *Homotopy Type Theory: Univalent Foundations of Mathematics*. [\url{https://homotopytypetheory.org/book}](https://homotopytypetheory.org/book).
- [25] Benno Van Den Berg and Richard Garner. Types are weak  $\omega$ -groupoids. 102(2):370–394.
- [26] Mark Weber. Generic morphisms, parametric representations and weakly Cartesian monads. 13(14):191–234.

## A Structural rules of dependent type theories

We consider dependent type theories that always satisfy some amount of structure, namely that which makes their syntax into a category with families. We recall here part of this structure that we use, and which is satisfied by all the types theories we consider. First, the variables are pre-terms, and we want there to be an action of substitutions on types and terms. On term type and term constructors, this action is defined for every type and term constructor, but is usually the unique sensible choice. On variables, this action associates the pre-term associated to the variable in the substitution, or leaves the variable unchanged if there is no association. We can express this inductively as follows

$$x[\langle \rangle] = \text{undefined} \quad x[\langle \gamma, y \mapsto t \rangle] = \begin{cases} t & \text{if } y = x \\ x[\gamma] & \text{otherwise} \end{cases}$$

There is also a composition of substitutions defined inductively from the action on terms as follows

$$\langle \rangle \circ \gamma = \langle \rangle \quad \langle \delta, x \mapsto t \rangle \circ \gamma = \langle \delta \circ \gamma, x \mapsto t[\gamma] \rangle.$$

All the substitutions that we consider here are implicitly well-scoped, and so in particular, when we write  $t[\gamma]$ , (reps.  $A[\gamma]$ ,  $\delta \circ \gamma$ ), we implicitly assume that all free variables of  $t$ , (resp.  $A, \delta$ ) are bound in  $\gamma$ . Given a pre-context  $\Gamma$ , we denote  $\text{id}_\Gamma$  the identity substitution on  $\Gamma$  which maps every variable to itself. It is required that the action of substitutions on pre-types and on pre-terms is compatible with the substitution, and that the identity is a neutral element for this action in the sense that the following equations hold:

$$\begin{aligned} A[\delta \circ \gamma] &= A[\delta][\gamma] & t[\delta \circ \gamma] &= t[\delta][\gamma] \\ A[\text{id}_\Gamma] &= A & t[\text{id}_\Gamma] &= t & \gamma \circ \text{id}_\Gamma &= \gamma. \end{aligned} \tag{1}$$

The derivation rules for contexts, substitutions and terms that are variables are standard, and given by the following:

$$\begin{aligned} & \frac{}{\emptyset \vdash \text{ctx}} \text{(\emptyset-CTX)} & \frac{\Gamma \vdash \text{ctx} \quad \Gamma \vdash A \text{ type}}{(\Gamma, x: A) \vdash \text{ctx}} \text{(+CTX)} \\ & \frac{\Gamma}{\Gamma \vdash \langle \rangle :: \emptyset} \text{(\emptyset-SUB)} & \frac{\Delta \vdash \gamma :: \Gamma \quad \Gamma \vdash A \text{ type} \quad \Delta \vdash t : A[\gamma]}{\Delta \vdash \langle \gamma, x \mapsto t \rangle :: (\Gamma, x: A)} \text{(+SUB)} \\ & \frac{\Gamma \vdash \text{ctx} \quad \text{assoc}(\Gamma, x) = A}{\Gamma \vdash x : A} \text{(VAR-IN)} \end{aligned}$$

where  $\text{assoc}(\Gamma, x)$  is the partial function that returns the last element to which  $x$  is associated if it exists, in the context  $\Gamma$  seen as an association list. We also require our type theories to satisfy cut-admissibility for terms and types, that

is we require the following rules to be admissible

$$\frac{\Gamma \vdash A \text{ type} \quad \Delta \vdash \gamma :: \Gamma}{\Delta \vdash A[\gamma] \text{ type}} \quad \frac{\Gamma \vdash t : A \quad \Delta \vdash \gamma :: \Gamma}{\Delta \vdash t[\gamma] : A[\gamma]} \quad \frac{\Gamma \vdash \vartheta :: \Theta \quad \Delta \vdash \gamma :: \Gamma}{\Delta \vdash \vartheta \circ \gamma : \Theta} \quad (2)$$

It is standard that Martin-Löf type theory, homotopy type theory or any reasonable fragment of those respect all of the structure described above. The dependent type theories GSeTT and CaTT also do, as witnessed by a formalisation in Agda [4]. In the case of the CaTT, the action of a substitution  $\delta$  on a pre-term of the form  $\text{coh}_{\Gamma, A}[\gamma]$  is given by the formula

$$\text{coh}_{\Gamma, A}[\gamma][\delta] = \text{coh}_{\Gamma, A}[\gamma \circ \delta].$$

The dependent type theory HoTT is also subject to a non-trivial definitional equality, for which we assume that we have a confluent rewriting system, and we assume that this equality respects the typing, in the sense that the following rules are admissible

$$\frac{\Gamma \vdash A \text{ type} \quad A \equiv B}{\Gamma \vdash B \text{ type}} \quad \frac{\Gamma \vdash u : A \quad A \equiv B}{\Gamma \vdash u : B} \quad \frac{\Gamma \vdash u : A \quad u \equiv v}{\Gamma \vdash v : A} \quad (3)$$

## B Proof of Section 4

This section is dedicated to the proof of Lemma 4, which shows the correctness of the translation principle from GSeTT to HoTT.

**Lemma 4.** *Translating derivable contexts, types and variables in GSeTT yields derivable contexts, types and variables in HoTT. More precisely, we have:*

- For every derivable context judgement  $\Gamma \vdash \text{ctx}$  in GSeTT, the judgement  $\llbracket \Gamma \rrbracket_{\mathcal{B}} \vdash \text{ctx}$  is derivable in HoTT.
- For any type judgement  $\Gamma \vdash A \text{ type}$  derivable in GSeTT, the judgement  $\llbracket \Gamma \rrbracket_{\mathcal{B}} \vdash \llbracket A \rrbracket_{\mathcal{B}} \text{ type}$  is derivable in HoTT.
- For any term judgement  $\Gamma \vdash x : A$  derivable in GSeTT, the judgement  $\llbracket \Gamma \rrbracket_{\mathcal{B}} \vdash x : \llbracket A \rrbracket_{\mathcal{B}}$  is derivable in HoTT.

*Proof.* We proceed by mutual induction on the derivation tree of the judgements.

- If the derivation tree ends with Rule ( $\emptyset$ -CTX), then it is a derivation of the judgement  $\emptyset \vdash \text{ctx}$ . We then have  $\llbracket \emptyset \rrbracket_{\mathcal{B}} = (\mathcal{B} : \mathcal{U})$ , and the judgement  $(\mathcal{B} : \mathcal{U}) \vdash \text{ctx}$  is derivable in HoTT.
- If the derivation tree ends with Rule (+-CTX), then it is a derivation of the judgement  $(\Gamma, x : A)$ , produced from derivations for the two following judgements in GSeTT:

$$\Gamma \vdash \text{ctx} \qquad \Gamma \vdash A \text{ type}$$

by induction, these give derivations for the following judgements in HoTT

$$\llbracket \Gamma \rrbracket_{\mathbf{B}} \vdash \text{ctx} \qquad \llbracket \Gamma \rrbracket_{\mathbf{B}} \vdash \llbracket A \rrbracket_{\mathbf{B}} \text{ type}$$

which by application of Rule (+-CTX) give a derivation of the judgement  $(\llbracket \Gamma \rrbracket_{\mathbf{B}}, \mathbf{x} : \llbracket A \rrbracket_{\mathbf{B}}) \vdash \text{ctx}$  in HoTT.

- A derivation ending with Rule (\*-IN) is a derivation of  $\Gamma \vdash * \text{ type}$  obtained from a derivation of  $\Gamma \vdash \text{ctx}$ . By induction, this gives a derivation of the judgement  $\llbracket \Gamma \rrbracket_{\mathbf{B}} \vdash \text{ctx}$ . Moreover, this contexts starts with the association  $(\mathbf{B} : \mathcal{U})$ , and since the variable  $\mathbf{B}$  is not a valid variable name in GSeTT, this association cannot be overridden in  $\llbracket \Gamma \rrbracket_{\mathbf{B}}$ . Thus, Rule (VAR-IN) gives a derivation of the judgement  $\Gamma \vdash \mathbf{B} : \mathcal{U}$  in HoTT, which in terms let us get by Rule (El-IN) a derivation of  $\Gamma \vdash \text{El}(\mathbf{B}) \text{ type}$ .
- A derivation ending with Rule ( $\rightarrow$ -IN) is a derivation of  $\Gamma \vdash \mathbf{x} \rightarrow_A \mathbf{y} \text{ type}$  obtained from derivations of the following three judgements in GSeTT:

$$\Gamma \vdash A \text{ type} \qquad \Gamma \vdash \mathbf{x} : A \qquad \Gamma \vdash \mathbf{y} : A$$

By induction, each of them gives a derivation for the corresponding judgement in HoTT

$$\llbracket \Gamma \rrbracket_{\mathbf{B}} \vdash \llbracket A \rrbracket_{\mathbf{B}} \text{ type} \qquad \llbracket \Gamma \rrbracket_{\mathbf{B}} \vdash \mathbf{x} : \llbracket A \rrbracket_{\mathbf{B}} \qquad \llbracket \Gamma \rrbracket_{\mathbf{B}} \vdash \mathbf{y} : \llbracket A \rrbracket_{\mathbf{B}}$$

Rule (ld-IN) then gives a derivation of the judgement  $\llbracket \Gamma \rrbracket_{\mathbf{B}} \vdash \text{ld}_{\llbracket A \rrbracket_{\mathbf{B}}}(\mathbf{x}, \mathbf{y}) \text{ type}$  in HoTT.

- A derivation finishing with an application of Rule (VAR-IN) is a derivation of the judgement  $\Gamma \vdash \mathbf{x} : A$  obtained from a derivation of  $\Gamma \vdash \text{ctx}$ , such that  $\text{assoc}(\Gamma, \mathbf{x}) = A$ . By induction, we get a derivation for the rule  $\llbracket \Gamma \rrbracket_{\mathbf{B}} \vdash \text{ctx}$  in HoTT. Moreover, from the definition of  $\llbracket \Gamma \rrbracket_{\mathbf{B}}$ , one can see that  $\text{assoc}(\llbracket \Gamma \rrbracket_{\mathbf{B}}, \mathbf{x}) = \llbracket A \rrbracket_{\mathbf{B}}$ . Hence, we get a definition of the judgement  $\llbracket \Gamma \rrbracket_{\mathbf{B}} \vdash \mathbf{x} : \llbracket A \rrbracket_{\mathbf{B}}$ .

□

## C Proof of Section 5

This section is dedicated to proving correctness of the translation scheme of CaTT term to closed  $\lambda$ -terms in HoTT.

**Lemma 6.** *The proposed translation scheme respects substitution application at the level of the pre-syntax. More precisely, the following results hold:*

- For all pre-type  $A$  and all pre-substitution  $\gamma$  in CaTT, we have the following equality of pre-types in HoTT:  $\llbracket A[\gamma] \rrbracket_{\mathbf{B}} = \llbracket A \rrbracket_{\mathbf{B}} \llbracket \llbracket \gamma \rrbracket_{\mathbf{B}} \rrbracket$ .
- For all pre-term  $t$  and all pre-substitution  $\gamma$  in CaTT, we have the following equality of pre-terms in HoTT:  $\llbracket t[\gamma] \rrbracket_{\mathbf{B}} = \llbracket t \rrbracket_{\mathbf{B}} \llbracket \llbracket \gamma \rrbracket_{\mathbf{B}} \rrbracket$ .

- For all pairs of pre-substitutions  $\delta, \gamma$  in  $\text{CaTT}$ , we have the following equality of pre-substitutions in  $\text{HoTT}$ :  $\llbracket \delta \circ \gamma \rrbracket_{\mathbf{B}} = \llbracket \delta \rrbracket_{\mathbf{B}} \circ \llbracket \gamma \rrbracket_{\mathbf{B}}$ .

*Proof.* These results hold syntactically regardless of any well-foundedness principle. We fix a pre-substitution  $\gamma$ , and proceed by mutual induction.

- Considering the pre-type  $*$ , we have  $\llbracket * \rrbracket_{\mathbf{B}} = \mathbf{B}$ . On the other hand,  $\llbracket \gamma \rrbracket_{\mathbf{B}}$  starts by defining the mapping  $\mathbf{B} \mapsto \mathbf{B}$ , and since by assumption, the variable  $\mathbf{B}$  is not a variable in  $\text{CaTT}$ , this mapping is never overridden in  $\llbracket \gamma \rrbracket_{\mathbf{B}}$ . Thus we have  $\mathbf{B}[\llbracket \gamma \rrbracket_{\mathbf{B}}] = \mathbf{B}$ . This shows the equality

$$\llbracket * \rrbracket_{\mathbf{B}} = (\llbracket * \rrbracket_{\mathbf{B}})[\llbracket \gamma \rrbracket_{\mathbf{B}}].$$

- Considering the pre-type  $u \rightarrow_A v$ , by the inductive case for types, we have the equality  $\llbracket A \rrbracket_{\mathbf{B}} = (\llbracket A \rrbracket_{\mathbf{B}})[\llbracket \gamma \rrbracket_{\mathbf{B}}]$ . By the mutually inductive case for terms, we have  $\llbracket u \rrbracket_{\mathbf{B}} = (\llbracket u \rrbracket_{\mathbf{B}})[\llbracket \gamma \rrbracket_{\mathbf{B}}]$ , and similarly for  $v$ . This shows the equality

$$\llbracket (u \rightarrow_A v) \rrbracket_{\mathbf{B}} = (\llbracket u \rightarrow_A v \rrbracket_{\mathbf{B}})[\llbracket \gamma \rrbracket_{\mathbf{B}}].$$

- Considering a variable  $\mathbf{x}$  in  $\text{CaTT}$ , denote  $u = \mathbf{x}[\gamma]$  the term corresponding to the last binding of  $\mathbf{x}$  in  $\gamma$ . By definition, this binding defines the binding  $\mathbf{x} \mapsto \llbracket u \rrbracket_{\mathbf{B}}$  in  $\llbracket \gamma \rrbracket_{\mathbf{B}}$ . Moreover, the binding  $\mathbf{x} \mapsto u$  being the last binding of  $\mathbf{x}$  in  $\gamma$  implies that this binding of  $\mathbf{x}$  is never overridden in  $\llbracket \gamma \rrbracket_{\mathbf{B}}$ . Hence, we have  $\mathbf{x}[\llbracket \gamma \rrbracket_{\mathbf{B}}] = \llbracket u \rrbracket_{\mathbf{B}}$ . This proves the equality

$$\llbracket \mathbf{x} \rrbracket_{\mathbf{B}} = (\llbracket \mathbf{x} \rrbracket_{\mathbf{B}})[\llbracket \gamma \rrbracket_{\mathbf{B}}]$$

- Considering the pre-term  $\text{coh}_{\Gamma, A}[\delta]$ , we have by definition

$$\begin{aligned} \llbracket \text{coh}_{\Gamma, A}[\delta] \rrbracket_{\mathbf{B}} &= (\llbracket \text{coh}_{\Gamma, A} \rrbracket_{\mathbf{B}})[\llbracket \delta \circ \gamma \rrbracket_{\mathbf{B}}] \\ \llbracket \text{coh}_{\Gamma, A}[\delta] \rrbracket_{\mathbf{B}}[\llbracket \gamma \rrbracket_{\mathbf{B}}] &= (\llbracket \text{coh}_{\Gamma, A} \rrbracket_{\mathbf{B}})[\llbracket \delta \rrbracket_{\mathbf{B}}][\llbracket \gamma \rrbracket_{\mathbf{B}}] \end{aligned}$$

By the inductive case for substitutions, we have  $\llbracket \delta \circ \gamma \rrbracket_{\mathbf{B}} = \llbracket \delta \rrbracket_{\mathbf{B}} \circ \llbracket \gamma \rrbracket_{\mathbf{B}}$ . Then using the definition of the action of substitution on coherences as well as (1) shows that

$$\llbracket \text{coh}_{\Gamma, A}[\delta] \rrbracket_{\mathbf{B}} = \llbracket \text{coh}_{\Gamma, A}[\delta] \rrbracket_{\mathbf{B}}[\llbracket \gamma \rrbracket_{\mathbf{B}}]$$

- Considering the empty pre-substitution  $\langle \rangle$ , since  $\mathbf{B}[\llbracket \gamma \rrbracket_{\mathbf{B}}] = \mathbf{B}$ , we have the following equalities

$$\begin{aligned} \llbracket \langle \rangle \circ \gamma \rrbracket_{\mathbf{B}} &= \llbracket \langle \rangle \rrbracket_{\mathbf{B}} = \langle \mathbf{B} \mapsto \mathbf{B} \rangle \\ \llbracket \langle \rangle \rrbracket_{\mathbf{B}} \circ \llbracket \gamma \rrbracket_{\mathbf{B}} &= \langle \mathbf{B} \mapsto \mathbf{B}[\llbracket \gamma \rrbracket_{\mathbf{B}}] \rangle = \langle \mathbf{B} \mapsto \mathbf{B} \rangle. \end{aligned}$$

- Considering the pre-substitution  $\langle \delta, \mathbf{x} \mapsto u \rangle$ , the inductive case shows that  $\llbracket \delta \circ \gamma \rrbracket_{\mathbf{B}} = \llbracket \delta \rrbracket_{\mathbf{B}} \circ \llbracket \gamma \rrbracket_{\mathbf{B}}$ . The mutually inductive case for terms shows that  $\llbracket u \rrbracket_{\mathbf{B}} = (\llbracket u \rrbracket_{\mathbf{B}})[\llbracket \gamma \rrbracket_{\mathbf{B}}]$ . Together, they prove the equality

$$\llbracket \langle \delta, \mathbf{x} \mapsto u \rangle \circ \gamma \rrbracket_{\mathbf{B}} = \llbracket \langle \delta, \mathbf{x} \mapsto u \rangle \rrbracket_{\mathbf{B}} \circ \llbracket \gamma \rrbracket_{\mathbf{B}}. \quad \square$$

Before proving Lemma 7, we first show the following result as an intermediate step

**Lemma 10.** *Given two ps-contexts  $\Gamma$  and  $\Delta$ , a type  $\Gamma \vdash A$  type and a substitution  $\Delta \vdash \gamma :: \Gamma$  in  $\text{CaTT}$ , the free variables of the term  $\text{elim}_B(\Delta, A, \Gamma, \gamma)$  are all bound in the context  $\llbracket \Delta \rrbracket_B$ .*

*Proof.* We can verify this by induction on the ps-context  $\Delta$ .

- For the context  $(\mathbf{x}:*)$ , the free variables of the term

$$\text{elim}_B((\mathbf{x}:*), A, \Gamma, \gamma) = \text{refl}_{B,\mathbf{x}}$$

are  $B$  and  $\mathbf{x}$  and both are bound in the context  $\llbracket (\mathbf{x}:*) \rrbracket_B = (B:\mathcal{U}, \mathbf{x}: \text{El}(B))$ .

- For the context  $(\Delta, \mathbf{y}:C, \mathbf{f}: \mathbf{x} \rightarrow_C \mathbf{y})$ , recall that we have

$$\text{elim}_B((\Delta, \mathbf{y}:C, \mathbf{f}: \mathbf{x} \rightarrow_C \mathbf{y}), A, \Gamma, \gamma) = J(\llbracket C \rrbracket_{B,\mathbf{x}}, P(\mathbf{x}', \mathbf{y}', \mathbf{f}'), p) \mathbf{y} \mathbf{f}$$

where

$$\begin{aligned} P(\mathbf{x}', \mathbf{y}', \mathbf{f}') &:= \llbracket A \rrbracket_B[\gamma](\langle \text{id}_{\llbracket (\Delta, \mathbf{y}:C, \mathbf{f}: \mathbf{x} \rightarrow_C \mathbf{y}) \rrbracket_B}, \mathbf{x} \mapsto \mathbf{x}', \mathbf{y} \mapsto \mathbf{y}', \mathbf{f} \mapsto \mathbf{f}' \rangle) \\ p &:= \text{elim}_B(\Delta, A, \Gamma, \gamma \circ \langle \text{id}_{\llbracket \Delta \rrbracket_B}, \mathbf{y} \mapsto \mathbf{x}, \mathbf{f} \mapsto \text{refl}_{\llbracket C \rrbracket_{B,\mathbf{x}}} \rangle). \end{aligned}$$

Hence, the free variables of the term  $\text{elim}_B((\Delta, \mathbf{y}:C, \mathbf{f}: \mathbf{x} \rightarrow_C \mathbf{y}), A, \Gamma, \gamma)$  are the union of the free variables of the term  $p$  and the set  $\{\mathbf{x}, \mathbf{y}, \mathbf{f}\}$ . By induction, all free variables of  $p$  are bound in  $\llbracket \Delta \rrbracket_B$ , thus, all the above variables are bound in  $\llbracket (\Delta, \mathbf{y}:C, \mathbf{f}: \mathbf{x} \rightarrow_{\llbracket C \rrbracket_B} \mathbf{y}) \rrbracket_B$

□

**Lemma 7.** *For every term  $(\mathbf{x}:*) \vdash u : A$  in  $\text{CaTT}$ , we have  $\llbracket u \rrbracket_B \equiv \text{refl}_{B,\mathbf{x}}^{\dim A+1}$ . Similarly, for every type  $(\mathbf{x}:*) \vdash A$  type in  $\text{CaTT}$ , we have  $\llbracket A \rrbracket_B \equiv \text{ld}_{B,\mathbf{x}}^{\dim A+1}$*

*Proof.* We first prove the result for terms, that we prove mutually inductively with the following proposition: for every ps-contexts  $\Gamma, \Delta$ , type  $\Gamma \vdash A$  type and substitution  $\llbracket \Delta \rrbracket_B \vdash \gamma :: \llbracket \Gamma \rrbracket_B$ , as well as a substitution  $(\mathbf{x}:*) \vdash \delta :: \Delta$ , we have

$$(\text{elim}_B(\Delta, A, \Gamma, \gamma))(\llbracket \delta \rrbracket_B) \equiv \text{refl}_{B,\mathbf{x}}^{\dim A+1}.$$

We proceed by structural induction on the term, and induction on the length of the ps-context.

- For the term  $\mathbf{x}$  which is the unique variable of the context, by definition, we have  $\llbracket \mathbf{x} \rrbracket_B = \mathbf{x} = \text{refl}_{B,\mathbf{x}}^0$ .
- For a coherence term of the form  $\text{coh}_{\Gamma,C}[\gamma]$  with  $C[\gamma] = A$ . Then

$$\llbracket \text{coh}_{\Gamma,C}[\gamma] \rrbracket_B = (\text{elim}_B(\Gamma, C, \Gamma, \text{id}_\Gamma))(\llbracket \gamma \rrbracket_B),$$

so by the mutually inductive proposition, and using that  $\dim A = \dim C$ , we get

$$\begin{aligned} \llbracket \text{coh}_{\Gamma,C}[\gamma] \rrbracket_B &= \text{refl}_{B,\mathbf{x}}^{\dim C+1} \\ &= \text{refl}_{B,\mathbf{x}}^{\dim A+1}. \end{aligned}$$

- Given a ps-context  $\Gamma$  with a substitution  $\llbracket (y:*) \rrbracket_B \vdash \gamma :: \llbracket \Gamma \rrbracket_B$  and a substitution  $(x:*) \vdash \delta :: (y:*)$ , the only term of type  $*$  derivable in the context  $(x:*)$  is  $x$  thus we have  $\delta = \langle y \mapsto x \rangle$ . Then, we have

$$\begin{aligned} \text{elim}((y:*), A, \Gamma, \gamma) \llbracket \delta \rrbracket_B &= \text{refl}_{B,y}^{\dim A+1} [B \mapsto B, y \mapsto x] \\ &= \text{refl}_{B,x}^{\dim A+1}. \end{aligned}$$

- Given a ps-context  $\Gamma$  and a ps-context of the form  $\Delta' = (\Delta, z: D, f: y \rightarrow_D z)$ , with a substitution  $\llbracket \Delta' \rrbracket_B \vdash \gamma :: \llbracket \Gamma \rrbracket_B$  and a substitution  $(x:*) \vdash \delta' :: \Delta'$ , the substitution  $\delta'$  decomposes as  $\delta' = \langle \delta, z \mapsto v, f \mapsto w \rangle$ . We have

$$\text{elim}(\Delta', A, \Gamma, \gamma) \llbracket \delta' \rrbracket_B \equiv J(\llbracket C \rrbracket_{B,y}, P(y', z', f'), p) \llbracket \delta' \rrbracket_B (\llbracket v \rrbracket_B) (\llbracket w \rrbracket_B)$$

where

$$\begin{aligned} P(y', z', f') &:= \llbracket C \rrbracket_B [\gamma] \langle \text{id}_{\llbracket \Delta' \rrbracket_B}, y \mapsto y', z \mapsto z', f \mapsto f' \rangle \\ p &:= \text{elim}_B(\Delta, A, \Gamma, \gamma \circ \langle \text{id}_{\llbracket \Delta \rrbracket_B}, z \mapsto y, f \mapsto \text{refl}_{\llbracket C \rrbracket_{B,y}} \rangle). \end{aligned}$$

By the structural induction case for terms, we have

$$\llbracket v \rrbracket_B \equiv \text{refl}_{B,x}^{\dim D+1} \quad \llbracket w \rrbracket_B \equiv \text{refl}_{B,x}^{\dim D+2},$$

and since  $\dim D + 2 \geq 1$ ,  $\text{refl}_{B,x}^{\dim D+2}$  is a reflexivity term the computation rule  $(\eta_J)$  shows

$$\text{elim}(\Delta', A, \Gamma, \gamma) \llbracket \delta' \rrbracket_B \equiv p \llbracket \delta' \rrbracket_B.$$

By Lemma 10, we have  $p \llbracket \delta' \rrbracket_B \equiv p \llbracket \delta \rrbracket_B$ . By (2) we have

$$\llbracket \Delta \rrbracket_B \vdash \gamma \circ \langle \text{id}_{\llbracket \Delta \rrbracket_B}, y \mapsto x, f \mapsto \text{refl}_{\llbracket C \rrbracket_{B,x}} \rangle :: \llbracket \Gamma \rrbracket_B,$$

and by inversion, we have  $(x:*) \vdash \delta :: \Delta$ , thus by induction, this shows that  $p \llbracket \delta \rrbracket_B \equiv \text{refl}_{B,x}^{\dim A+1}$ .

The result of type is a consequence of the one on terms. Indeed, we proceed by structural induction on the type. For the type  $*$ , since we have  $\llbracket * \rrbracket_B = B = \text{Id}_{B,x}^0$ , and considering the type  $(x:*) \vdash u \rightarrow_A v$  type in  $\text{CaTT}$ , we have derivations of the judgements  $(*) \vdash u : A$  and  $(x:*) \vdash v : A$ , and so by induction and by the result on terms, we get the following equalities

$$\llbracket A \rrbracket_B \equiv \text{Id}_{B,x}^{\dim A+1} \quad \llbracket u \rrbracket_B \equiv \text{refl}_{B,x}^{\dim A+1} \quad \llbracket v \rrbracket_B \equiv \text{refl}_{B,x}^{\dim A+1}$$

This shows that we have

$$\llbracket u \rightarrow_A v \rrbracket_B \equiv \text{Id}_{\text{Id}_{B,x}^{\dim A+1} (\text{refl}_{B,x}^{\dim A+1}, \text{refl}_B^{\dim A+1}, x)} = \text{Id}_{B,x}^{\dim A+2}. \quad \square$$

**Proposition 8.** *The following results hold:*

- For any context  $\Gamma \vdash \text{ctx}$  in  $\text{CaTT}$ , we have  $\llbracket \Gamma \rrbracket_B \vdash \text{ctx}$  in  $\text{HoTT}$ .



- For any type  $\Gamma \vdash A$  type in  $\text{CaTT}$ , we have  $[\Gamma]_{\mathbf{B}} \vdash [A]_{\mathbf{B}}$  type in  $\text{HoTT}$ .
- For any term  $\Gamma \vdash t : A$  in  $\text{CaTT}$ , we have  $[\Gamma]_{\mathbf{B}} \vdash [t]_{\mathbf{B}} : [A]_{\mathbf{B}}$  in  $\text{HoTT}$ .
- For any substitution  $\Delta \vdash \gamma :: \Gamma$  in  $\text{CaTT}$ , we have  $[\Delta]_{\mathbf{B}} \vdash [\gamma]_{\mathbf{B}} :: [\Gamma]_{\mathbf{B}}$  in  $\text{HoTT}$ .
- For any coherence  $\vdash \text{coh}_{\Gamma, A}$ , we have  $[\Gamma]_{\mathbf{B}} \vdash [\text{coh}_{\Gamma, A}]_{\mathbf{B}} : [A]_{\mathbf{B}}$  in  $\text{HoTT}$ .
- For any pair of ps-contexts  $\Gamma, \Delta$  together with a type  $\Gamma \vdash A$  type in  $\text{CaTT}$  and a substitution  $[\Delta]_{\mathbf{B}} \vdash \gamma :: [\Gamma]_{\mathbf{B}}$  in  $\text{HoTT}$ , the following judgement is derivable in  $\text{HoTT}$ :

$$[\Delta]_{\mathbf{B}} \vdash \text{elim}_{\mathbf{B}}(\Delta, A, \Gamma, \gamma) : [A]_{\mathbf{B}}[\gamma]$$

*Proof.* We prove this result by mutual induction, the induction scheme following that of the definition of the translation function  $[\_ ]_{\mathbf{B}}$  and of  $\text{elim}_{\mathbf{B}}$ . The cases of contexts, types and variables are similar to those of the proof of Lemma 4.

- For the empty context  $\emptyset \vdash \text{ctx}$  in  $\text{CaTT}$ , obtained by Rule ( $\emptyset$ -CTX), we have  $[\emptyset]_{\mathbf{B}} = (\mathbf{B}:\mathcal{U})$ , and the rules of  $\text{HoTT}$  allow to build a derivation of  $(\mathbf{B}:\mathcal{U}) \vdash \text{ctx}$ .
- For the context  $(\Gamma, x : A) \vdash \text{ctx}$  in  $\text{CaTT}$  obtained by application of Rule (+-CTX) from derivations of  $\Gamma \vdash \text{ctx}$  and  $\Gamma \vdash A$  type, by induction we have derivations for the two following judgements in  $\text{HoTT}$

$$[\Gamma]_{\mathbf{B}} \vdash \text{ctx} \qquad [\Gamma]_{\mathbf{B}} \vdash [A]_{\mathbf{B}} \text{ type}$$

- For the type  $\Gamma \vdash *$  type in  $\text{CaTT}$  obtained by Rule (\*-IN) from a derivation of  $\Gamma \vdash \text{ctx}$ , by induction we have a derivation of  $[\Gamma]_{\mathbf{B}} \vdash \text{ctx}$  in  $\text{HoTT}$ . Moreover, the first assignation in this context is  $(\mathbf{B}:\mathcal{U})$  and since the name  $\mathbf{B}$  is not a valid variable in  $\text{CaTT}$ , it cannot be overridden in  $[\Gamma]_{\mathbf{B}}$ , thus showing that  $\text{assoc}([\Gamma]_{\mathbf{B}}) = \mathcal{U}$ . This gives by applying Rule (VAR-IN) and then Rule (El-IN), a derivation of

$$\Gamma \vdash \text{El}(\mathbf{B}) \text{ type.}$$

- For the type  $\Gamma \vdash u \rightarrow_A v$  type in  $\text{CaTT}$ , obtained by Rule ( $\rightarrow$ -IN) from derivations for the following three judgements

$$\Gamma \vdash A \text{ type} \qquad \Gamma \vdash u : A \qquad \Gamma \vdash v : A$$

we obtain by the induction cases for types and terms, a derivation for the three corresponding judgements in  $\text{HoTT}$

$$[\Gamma]_{\mathbf{B}} \vdash [A]_{\mathbf{B}} \text{ type} \qquad [\Gamma]_{\mathbf{B}} \vdash [u]_{\mathbf{B}} : [A]_{\mathbf{B}} \qquad [\Gamma]_{\mathbf{B}} \vdash [v]_{\mathbf{B}} : [A]_{\mathbf{B}}$$

Then we obtain by Rule (Id-IN), a derivation of  $[\Gamma]_{\mathbf{B}} \vdash \text{Id}_{[A]_{\mathbf{B}}}([u]_{\mathbf{B}}, [v]_{\mathbf{B}})$  type.

- For a term of  $\text{CaTT}$  given as a variable  $\Gamma \vdash \mathbf{x} : A$  obtained by Rule (VAR-IN) from a derivation of  $\Gamma \vdash \text{ctx}$  such that  $\text{assoc}(\Gamma, \mathbf{x}) = A$ , we obtain by induction a derivation of  $\llbracket \Gamma \rrbracket_{\mathbf{B}} \vdash \text{ctx}$ . Moreover, by definition of  $\llbracket \Gamma \rrbracket_{\mathbf{B}}$ , we have  $\text{assoc}(\llbracket \Gamma \rrbracket_{\mathbf{B}}, \mathbf{x}) = \llbracket A \rrbracket_{\mathbf{B}}$ , which lets us derive by Rule (VAR-IN) a derivation of the following judgement in  $\text{HoTT}$

$$\Gamma \vdash \mathbf{x} : \llbracket A \rrbracket_{\mathbf{B}}.$$

- For a term of the form  $\Delta \vdash \text{coh}_{\Gamma, A}[\gamma] : A[\gamma]$  in  $\text{CaTT}$  obtained by application of Rule (coh-IN) from derivations of  $\vdash \text{coh}_{\Gamma, A}$  and  $\Delta \vdash \gamma :: \Gamma$ , we get by the induction cases for coherences and substitutions derivations for the corresponding judgements in  $\text{HoTT}$ :

$$\llbracket \Gamma \rrbracket_{\mathbf{B}} \vdash \llbracket \text{coh}_{\Gamma, A} \rrbracket_{\mathbf{B}} : \llbracket A \rrbracket_{\mathbf{B}} \quad \llbracket \Delta \rrbracket_{\mathbf{B}} \vdash \llbracket \gamma \rrbracket_{\mathbf{B}} :: \llbracket \Gamma \rrbracket_{\mathbf{B}}$$

and we then obtain, by (2), Lemma 6 and (3) a derivation of the judgement

$$\llbracket \Delta \rrbracket_{\mathbf{B}} \vdash \llbracket \text{coh}_{\Gamma, A}[\gamma] \rrbracket_{\mathbf{B}} : \llbracket A[\gamma] \rrbracket_{\mathbf{B}}.$$

- For the empty substitution  $\Gamma \vdash \langle \rangle :: \emptyset$  in  $\text{CaTT}$  obtained by Rule ( $\emptyset$ -SUB) from a derivation of  $\Gamma \vdash \text{ctx}$ , we have by induction a derivation of  $\llbracket \Gamma \rrbracket_{\mathbf{B}} \vdash \text{ctx}$ . Moreover, as proven in the case for the type  $\Gamma \vdash * \text{ type}$ , we can construct a derivation of  $\llbracket \Gamma \rrbracket_{\mathbf{B}} \vdash \mathbf{B} : \mathcal{U}$ , which lets us use Rule ( $\emptyset$ -SUB) and Rule (+-SUB) to produce a derivation of

$$\llbracket \Gamma \rrbracket_{\mathbf{B}} \vdash \langle \mathbf{B} \mapsto \mathbf{B} \rangle :: (\mathbf{B} : \mathcal{U})$$

- For a substitution of the form  $\Delta \vdash \langle \gamma, x \mapsto t \rangle :: (\Gamma, x : A)$  in  $\text{CaTT}$  obtained by Rule (+-SUB) from derivations of the following judgements

$$\Delta \vdash \gamma :: \Gamma \quad \Gamma \vdash A \text{ type} \quad \Delta \vdash t : A[\gamma]$$

we get by the induction cases for substitutions, contexts and terms, a derivation of the corresponding judgements in  $\text{HoTT}$

$$\llbracket \Delta \rrbracket_{\mathbf{B}} \vdash \llbracket \gamma \rrbracket_{\mathbf{B}} :: \llbracket \Gamma \rrbracket_{\mathbf{B}} \quad \llbracket \Gamma \rrbracket_{\mathbf{B}} \vdash \llbracket A \rrbracket_{\mathbf{B}} \text{ type} \quad \llbracket \Delta \rrbracket_{\mathbf{B}} \vdash \llbracket t \rrbracket_{\mathbf{B}} : \llbracket A[\gamma] \rrbracket_{\mathbf{B}}$$

Applying Lemma 6 then allows us to apply Rule (+-SUB) to obtain a derivation of the judgement

$$\llbracket \Delta \rrbracket_{\mathbf{B}} \vdash \langle \llbracket \gamma \rrbracket_{\mathbf{B}}, \mathbf{x} \mapsto \llbracket t \rrbracket_{\mathbf{B}} \rangle :: (\llbracket \Gamma \rrbracket_{\mathbf{B}}, \mathbf{x} : \llbracket A \rrbracket_{\mathbf{B}})$$

- For a coherence  $\vdash \text{coh}_{\Gamma, A}$  in  $\text{CaTT}$  obtained by Rule (coh-WD), we have derivations of  $\Gamma \vdash_{\text{ps}}$  and  $\Gamma \vdash A \text{ type}$ , thus, we get by the inductive case for  $\text{elim}_{\mathbf{B}}$  as well as (1) and the fact that  $\llbracket \text{id}_{\Gamma} \rrbracket_{\mathbf{B}} = \text{id}_{\llbracket \Gamma \rrbracket_{\mathbf{B}}}$  a derivation of

$$\llbracket \Gamma \rrbracket_{\mathbf{B}} \vdash \llbracket \text{coh}_{\Gamma, A} \rrbracket_{\mathbf{B}} : \llbracket A \rrbracket_{\mathbf{B}}$$

- Given a ps-context  $\Gamma$ , a type  $\Gamma \vdash A$  type and a substitution  $\Gamma \vdash \gamma :: (\mathbf{x}:*)$  in  $\text{CaTT}$ , by an argument similar to that of the case of context, by Lemma 4 applied to the term  $(\mathbf{x}:*) \vdash \mathbf{x} : *$ , we have a derivation of  $\llbracket (\mathbf{x}:*) \rrbracket_{\mathbb{B}} \vdash \mathbf{x} : \text{El}(\mathbb{B})$ . By Lemma 1, we get a derivation of the judgement

$$\llbracket (\mathbf{x}:*) \rrbracket_{\mathbb{B}} \vdash \text{refl}_{\mathbb{B}, \mathbf{x}}^{\dim A+1} : \text{ld}_{\mathbb{B}, \mathbf{x}}^{\dim A+1}.$$

Then, applying (2) provides a derivation of  $(\mathbf{x}:*) \vdash A[\gamma]$  type, and thus Lemma 7 then shows  $\llbracket A[\gamma] \rrbracket_{\gamma} \equiv \text{ld}_{\mathbb{B}, \mathbf{x}}^{\dim A+1}$ , which by (3) gives a derivation of the judgement

$$\llbracket (\mathbf{x}:*) \rrbracket_{\mathbb{B}} \vdash \text{refl}_{\mathbb{B}, \mathbf{x}}^{\dim A+1} : \llbracket A[\gamma] \rrbracket_{\mathbb{B}}.$$

which is syntactically equal to

$$\llbracket (\mathbf{x}:*) \rrbracket_{\mathbb{B}} \vdash \text{elim}_{\mathbb{B}}(\llbracket (\mathbf{x}:*) \rrbracket_{\mathbb{B}}, A, \Gamma, \gamma) : \llbracket A[\gamma] \rrbracket_{\mathbb{B}}.$$

- Given a ps-context  $\Gamma$ , a type  $\Gamma \vdash A$  type a ps-context and a substitution respectively of the following forms

$$\begin{aligned} \Delta' &= (\Delta, \mathbf{z}: C, \mathbf{f}: \mathbf{y} \rightarrow_C \mathbf{z}) \\ \delta' &= \langle \delta, \mathbf{z} \mapsto v, \mathbf{f} \mapsto w \rangle \end{aligned}$$

satisfying  $\Gamma \vdash \delta' :: \Delta'$ , recall that we have

$$\text{elim}_{\mathbb{B}}(\llbracket (\Delta, \mathbf{y}: C, \mathbf{f}: \mathbf{x} \rightarrow_C \mathbf{y}), A, \Gamma, \gamma \rrbracket := \text{J}(\llbracket C \rrbracket_{\mathbb{B}, \mathbf{y}}, P(\mathbf{y}', \mathbf{z}', \mathbf{f}'), p) \mathbf{z} \mathbf{f}$$

where  $P$  and  $p$  are respectively defined as follows

$$\begin{aligned} P(\mathbf{y}', \mathbf{z}', \mathbf{f}') &:= \llbracket A \rrbracket_{\mathbb{B}}[\gamma](\langle \text{id}_{\llbracket \Delta' \rrbracket_{\mathbb{B}}}, \mathbf{y} \mapsto \mathbf{y}', \mathbf{z} \mapsto \mathbf{z}', \mathbf{f} \mapsto \mathbf{f}' \rangle) \\ p &:= \text{elim}_{\mathbb{B}}(\Delta, A, \Gamma, \gamma \circ \langle \text{id}_{\llbracket \Delta \rrbracket_{\mathbb{B}}}, \mathbf{z} \mapsto \mathbf{y}, \mathbf{f} \mapsto \text{refl}_{\llbracket C \rrbracket_{\mathbb{B}, \mathbf{y}}} \rangle). \end{aligned}$$

We first show that this application of the J rule is well-formed in the context  $\llbracket \Delta' \rrbracket_{\mathbb{B}}$ , that is that it follows the premises of Rule (J-IN) and of the function application rule. We first note that by Lemma 3 we have derivation for the following judgements

$$\begin{array}{ll} \Delta' \vdash C \text{ type} & \Delta' \vdash \mathbf{y} : C \\ \Delta' \vdash \mathbf{z} : C & \Delta' \vdash \mathbf{f} : \mathbf{y} \rightarrow_C \mathbf{z} \end{array}$$

By Lemma 4, those provide derivations for each of the corresponding judgements

$$\begin{array}{ll} \llbracket \Delta' \rrbracket_{\mathbb{B}} \vdash \llbracket C \rrbracket_{\mathbb{B}} \text{ type} & \llbracket \Delta' \rrbracket_{\mathbb{B}} \vdash \mathbf{y} : \llbracket C \rrbracket_{\mathbb{B}} \\ \llbracket \Delta' \rrbracket_{\mathbb{B}} \vdash \mathbf{z} : \llbracket C \rrbracket_{\mathbb{B}} & \llbracket \Delta' \rrbracket_{\mathbb{B}} \vdash \mathbf{f} : \text{ld}_{\llbracket C \rrbracket_{\mathbb{B}}}(\mathbf{y}, \mathbf{z}) \end{array}$$

So it suffices to show that  $P$  and  $p$  satisfy the premises of Rule (J-IN). First, define the following context and substitution in  $\text{HoTT}$ :

$$\begin{aligned} \Theta &:= (\llbracket \Delta' \rrbracket_{\mathbb{B}}, \mathbf{y}' : \llbracket C \rrbracket_{\mathbb{B}}, \mathbf{z}' : \llbracket C \rrbracket_{\mathbb{B}}, \llbracket \mathbf{f}' \rrbracket : \text{ld}_{\llbracket C \rrbracket_{\mathbb{B}}}(\mathbf{y}', \mathbf{z}')) \\ \vartheta &:= \langle \text{id}_{\llbracket \Delta' \rrbracket_{\mathbb{B}}}, \mathbf{y} \mapsto \mathbf{y}', \mathbf{z} \mapsto \mathbf{z}', \mathbf{f} \mapsto \mathbf{f}' \rangle. \end{aligned}$$

The judgement  $\Theta \vdash \vartheta :: \llbracket \Delta' \rrbracket_{\mathbb{B}}$  is derivable in  $\text{HoTT}$ , and by the inductive for types, so is the judgement  $\llbracket \Gamma \rrbracket_{\mathbb{B}} \vdash \llbracket A \rrbracket_{\mathbb{B}}$  type, so applying (2) twice gives a derivation of the judgement

$$\Theta \vdash P(y', z', f') \text{ type.}$$

Finally, we note that by (2), we have a derivation of

$$\llbracket \Delta \rrbracket_{\mathbb{B}} \vdash \gamma \circ \langle \text{id}_{\llbracket \Delta \rrbracket_{\mathbb{B}}}, z \mapsto y, f \mapsto \text{refl}_{\llbracket C \rrbracket_{\mathbb{B}}, y} \rangle :: \llbracket \Gamma \rrbracket_{\mathbb{B}}.$$

Thus, by the inductive case for  $\text{elim}_{\mathbb{B}}$ , decreasing on the first argument  $\Delta$ , we get a derivation of

$$\Delta \vdash p : \llbracket A \rrbracket_{\mathbb{B}}[\gamma \circ \langle \text{id}_{\llbracket \Delta \rrbracket_{\mathbb{B}}}, z \mapsto y, f \mapsto \text{refl}_{\llbracket C \rrbracket_{\mathbb{B}}, y} \rangle]$$

By functoriality of the action of substitution, we have the equality

$$\llbracket A \rrbracket_{\mathbb{B}}[\gamma \circ \langle \text{id}_{\llbracket \Delta \rrbracket_{\mathbb{B}}}, z \mapsto y, f \mapsto \text{refl}_{\llbracket C \rrbracket_{\mathbb{B}}, y} \rangle] = P(y, y, \text{refl}_{\llbracket C \rrbracket_{\mathbb{B}}, y}).$$

This shows that the application of  $\text{J}$  is valid, and provides with Rule (J-IN) a derivation of the judgement

$$\llbracket \Delta' \rrbracket_{\mathbb{B}} \vdash \text{elim}(\Delta', A, \Gamma, \llbracket \gamma \rrbracket_{\mathbb{B}}) : P(y, z, f)$$

Since we have  $P(y, z, f) = \llbracket A \rrbracket_{\mathbb{B}}[\gamma]$ , this in facts gives a derivation of

$$\llbracket \Delta' \rrbracket_{\mathbb{B}} \vdash \text{elim}(\Delta', A, \Gamma, \llbracket \gamma \rrbracket_{\mathbb{B}}) : \llbracket A \rrbracket_{\mathbb{B}}[\gamma]$$

□

## D The Eckmann-Hilton cell in $\text{CaTT}$

```

coh unitl (x(f)y) : comp (id _) f -> f
coh unit (x) : comp (id x) (id x) -> id x
coh lsimp (x) : (unitl (id x)) -> unit x
coh llsimp (x) : I (unitl (id x)) -> I (unit x)
coh exch (x(f(a)g)y(h(b)k)z) :
  comp (comp _ [b]) (id (comp f k)) (comp [a] _) -> comp [a] [b]

coh eh1 (x(f(a)g(b)h)y) :
comp a b -> comp (I (unitl f))
  (comp (comp _ [a])
    (comp (unitl g) (I (op { 1 } (unitl g))))
    (comp [b] _))
  (op { 1 } (unitl h))

let eh2 (x : *) (a : id x -> id x) (b : id x -> id x) =
comp [llsimp _]
  [comp (comp _
    [comp
      (comp [lsimp _] [op { 1 } (llsimp _)])
      (U (unit _))]
    _)]
  (exch b a)]
  [op { 1 } (lsimp _)]

let eh (x : *) (a : id x -> id x) (b : id x -> id x) =
comp (eh1 a b)
  (eh2 a b)
  (I (op { 1 } (eh2 b a)))
  (I (op { 1 } (eh1 b a)))

```

Figure 3: Definition of the Eckmann-Hilton cell in CaTT