INSTITUT POLYTECHNIQUE DE PARIS

NNT : 2020IPPAX077

Thèse de doctorat

A type theoretic approach to weak ω-categories and related higher structures

Thèse de doctorat de l'Institut Polytechnique de Paris
préparée à l'École polytechnique

École doctorale n°626 École doctorale de l'Institut Polytechnique de Paris (EDIPP)
Spécialité de doctorat : Mathématiques et Informatique

Thèse présentée et soutenue à Palaiseau, le 5 November 2020, par

THIBAUT BENJAMIN

Composition du Jury :

Thierry Coquand Professor, University of Gothemburg	Rapporteur
Nicolas Tabareau Senior researcher, INRIA	Rapporteur
Dimitri Ara Maître de conférences, Université d'Aix-Marseille	Examinateur
Peter LeFanu Lumsdaine Assistant professor, Stockholm University	Examinateur
Emily Riehl Associate professor, Johns Hopkins University	Examinateur
Samuel Mimram Professeur, Ecole Polytechnique	Directeur de thèse
Eric Finster Post-doctoral researcher, University of Birmingham	Co-encadrant de thèse

ECOLE
DOCTORALE

Titre : Catégories faibles et structures supérieures afférentes en théorie des types

Mots clés : Théorie des types, Catégories supérieures, Théorie des catégories, Diagrammes de compositions, Sémantique, Théorie homotopique des types

Résumé : Nous présentons une définition des ω-catégories faibles formulée en théorie des types proposée initialement par Finster et Mimram, suivant des idées provenant à la fois de la théorie homotopique des types et d'une définition des ω-catégories faibles due à Grothendieck et Maltsiniotis. Les avantages d'une telle approche sont multiples : Le langage de la théorie des types permet une définition qui se réduit à quelques règles seulement, et il fournit une syntaxe explicite sur laquelle il est possible de raisonner par induction. Cela donne également un algorithme pour implémenter un assistant de preuve dédié à l'exploration des ω-catégories faibles. Le travail que nous présentons est organisé selon deux axes principaux : Nous étudions les fondements théoriques de cette définition et sa sémantique et établissons l'équivalence avec la définition des ω-catégories faibles de Grothendieck-Maltsiniotis, et nous présentons l'assistant de preuve basé sur cette théorie ainsi que des considérations pour rendre cet outil plus utilisable en pratique. Nous considérons également des généralisations de cette approche à d'autres structures supérieures similaires, les ω-catégories faibles monoidales, et les ω-catégories faibles cubiques.

Title : A type theoretic approach to weak ω-categories and related higher structures

Keywords : Type theory, Higher categories, Category theory, Pasting schemes, Semantics, Homotopy type theory

Abstract : We study a type theoretic definition of weak ω-categories originally introduced by Finster and Mimram, inspired both from ideas coming from homotopy type theory and from a definition of weak ω-category due to Grothendieck and Maltsiniotis. The advantages of such an approach are multiple: The language of type theory allows for a definition restricted to only a few rules, it also provides an explicit syntax on which one can perform inductive reasoning, and gives an algorithm for implementing a proof-assistant dedicated to exploring weak ω-categories. The work we present about this type theory is organized along two main axes: We investigate the theoretical grounds for this definition study its semantics and prove its to the Grothendieck-Maltsiniotis definition of weak ω-categories, and we present the proof-assistant based on this theory together with practical considerations to improve its use. We also consider a generalization of this approach to other related higher structures, the monoidal weak ω-categories and the cubical weak ω-categories.

Institut Polytechnique de Paris
91120 Palaiseau, France

Remerciements

Je tiens en premier lieu à remercier toutes les personnes qui ont directement contribué à la réalisation de ce projet, et sans qui le présent document n'existerait pas. En particulier Samuel Mimram et Eric Finster pour leur encadrement et leurs conseils tout au long de ces trois années de travail, ainsi qu'Assia Mahboubi pour m'avoir toujours soutenu et conseillé dans l'élaboration de ce projet. Je remercie également tous les membres qui ont accepté de faire partie de mon Jury Dimitri Ara, Peter LeFanu Lumsdaine, Emily Riehl et Jamie Vicary, ainsi que les deux rapporteurs Thierry Coquand et Nicolas Tabareau pour leurs retours bienveillants et constructifs. Enfin, je remercie chaleureusement toute l'équipe Cosynus, au LIX pour m'avoir acceuilli au cours de ces trois années, et en particulier Simon et Patrick avec qui j'ai eu le plaisir de partager un bureau, et toutes l'équipe de doctorants de l'IRIF avec qui j'ai eu des interactions aussi enrichissantes humainement que scientifiquement : Pierre, Chaitanya, Antoine, Léonard, Léo.

Je souhaite maintenant remercier les toutes les personnes qui bien que n'ayant pas une contribution directe à ce projet ont joué un rôle essentiel dans sa complétion par leur soutient indéfectible. Tout d'abord mes parents, qui m'ont transmis le goût des sciences, ainsi que mon frère et ma soeur. Egalement ma copine Pallavi avec qui j'ai partagé au quotidien ma vie de doctorant. Finalement tous mes amis qui m'ont toujours soutenu et avec qui j'ai passé d'excellents moments, notamment Quentin, Anuradha, Alexis, Thomas, Thibaut, Loïc, Caroline, Thibault, Matthias, Marine, Constance, Julien, Pauline et Laurent. Finalement je remercie mon chat pour avoir tant bien que mal supporté mon indisponibilité pour jouer lors de la rédaction du présent manuscrit, alors que j'étais confiné à domicile.

Abstract

We study a type theoretic definition of weak ω-categories originally introduced by Finster and Mimram, inspired both from ideas coming from homotopy type theory and from a definition of weak ω-category due to Grothendieck and Maltsiniotis. The advantages of such an approach are multiple: The language of type theory allows for a definition restricted to only a few rules, it also provides an explicit syntax on which one can perform inductive reasoning, and gives an algorithm for implementing a proof-assistant dedicated to exploring weak ω-categories. The work we present about this type theory is organized along two main axes: We investigate the theoretical grounds for this definition and relate it to an other known definition of weak ω-categories, and we present the proof-assistant based on this theory together with practical considerations to improve its use. We also consider a generalization of this approach to other related higher structures.

We start with an introduction to the language of dependent type theory that we rely on to introduce our definitions, presenting both the syntax and the semantics that we study by means of categorical tools. We then present weak ω-categories and a type theory that defines them. We detail the categorical semantics of this theory and our main contribution in this direction establishes an equivalence between its models and the prior definition of weak ω-categories due to Grothendieck and Maltsiniotis. This definition has enabled us to implement a proof-assistant capable of checking whether a given morphism is well-defined in the theory of weak ω-category, and we present this implementation together with a few examples demonstrating both the capabilities of such a tool, and its tediousness in the vanilla version. To improve this issue, we present two main additional features allowing to partially automating its use: The suspension and the functorialization. These two operations are defined by similar techniques of induction on the syntax of the type theory. We then generalize this definition of weak ω-categories and present a type theoretic framework that is both modular enough to allow for defining higher structures, and constrained enough to precisely understand its semantics. This enables us to sketch a connection with the theory of monads with arities. Using this framework, we introduce and study two other definitions of higher structures: Monoidal weak ω-categories and cubical weak ω-categories. By using syntactic reasoning we are able to defines translations back and forth between the type theory defining weak ω-categories and the one describing monoidal weak ω-categories. One of our main result is to show that these translations imply an equivalence at the level of models: It shows that the monoidal ω-categories are equivalent to the ω-categories with a single object thus justifying the correctness of the appellation monoidal. We then give an alternate presentation of the type theory defining monoidal weak ω-categories, which diverges from our framework but is more standalone, and prove it to be equivalent to the previous presentation. We finally introduce in our framework a definition of cubical weak ω-categories and study its semantics, our main result along these lines is to characterize the models of this type theory and extract a mathematical definition equivalent to them.

Contents

Introduction

Type theory and type theories

Along the developments of computer science and the theory of formal languages, type theory has revealed itself to play an important role on a theoretical levels in various topics. It was first used to extend λ-calculus, a formalism introduced by Church to model computation [26], to typed λ-calculus, and thus eliminate from the syntax some terms that were problematic because they could not be normalized. A complete overview of this topic can be found at [10].

Martin-Löf type theory. It was then noticed that the rules for constructing and computing with these types were very similar to the rules for intuitionistic logic, a result known as the Curry-Howard correspondence. This presents type theory as a natural place to do constructive mathematics, this idea was further developed by Martin-Löf [55], and then Coquand and Huet [30] (calculus of constructions), leading to the development of the Coq proof-assistant [1], and ultimately of other type-theory based proof-assistants (such as Agda [2], Lean [3]). These systems introduce the notion of inductive types, among which in particular is the identity type, which expresses the equality between two elements. The construction of this identity is generic and makes it possible to "stack" them, i.e., to consider identity types between two terms which are themselves in an identity type, and so on. The structure generated by the interaction of these identity is very rich and lead in particular to homotopy type theory (that we present in more details later).

Constructive mathematics in Coq. One of the main successes of type theory is to provide a language suited to formalize mathematical proofs and software verification, that can be later computer-checked. This is the principle behind the proof-assistant Coq, in which important results have been developed. Most notably, in mathematics, it has been used by Gonthier and Werner [36] to formalize the demonstration of the four-color theorem: A famous question which had stayed open for a long time and was resolved shortly before. Later, Gonthier and al. [37] also formalized the Feit-Thomson theorem in Coq, which is a fundamental result in group theory, upon which relies the classification of finite groups. On a more computational side, Coq has been used by a team lead by Leroy to implement a completely certified compiler for the language C called CompCert [49]. Other proof assistants are also getting used on a wider scale, such as for instance Lean, with which Buzzard has recently started a project [22] to formalize algebraic geometry. The present document is itself accompanied with a formalization of some of the results, in the proof-assistant Agda [12], our choice for using this particular proof assistant is motivated

[1] `https://coq.inria.fr/`
[2] `https://hackage.haskell.org/package/Agda`
[3] `https://leanprover.github.io/`

by our heavy use of one of its features called inductive-inductive types, that is well-suited to work with our problems.

Type theories as generalized algebraic theories. Since the early days of type theories, Cartmell [23] had noticed that this general guideline could be used as a syntax to express axioms, and he called the logical theories expressed this way *generalized algebraic theories*. With the help of categorical tools, an appropriate notion of semantics was defined to formalize a notion of model of a type theory. There are various categorical settings to perform this, like comprehension categories, display map categories, categories with families, natural models, contextual categories. A survey of these notions can be found at [41]. Note that these tools study type theories in general and see them as generalized algebraic theories, whereas Martin-Löf type theory is a fixed type theory inside of which one can develop constructive mathematics.

Weak ω-groupoids

The more recent development of homotopy type theory unveils a connection between type theory and a particular area of topology called homotopy type theory. To understand this connection we present a brief detour to weak ω-categories. Like all other higher structures, they are particularly difficult to define, and as a result there have been a lot of variations on this definition. In particular, we mention a definition due to Grothendieck [39], since it has strong connections with our work.

The fundamental groupoid of a space. Algebraic topology is the study of topological spaces, by the mean of algebraic tools. A primary example is the one of the so-called *fundamental groupoid*. Given a space together with two points, one can study the space of paths between these points, we consider the data of the set of all points of the space, together with, for all pairs of points, the set of all paths between those points. Given a path from x to y and a path from y to z, one can compose them simply by following the first path first, and then the second path. However, this composition is not associative on the nose, it is only associative *up to* a continuous deformation of the paths (a *homotopy*). To account for this fact, one can quotient the set of paths by the homotopies, and thus get a structure where the composition is associative: the fundamental groupoid. Algebraically, this structure is a groupoid, and it defines an invariant of the space: two homeomorphic spaces have equivalent fundamental groupoids.

The fundamental higher groupoid. Focusing on the fundamental groupoid of a space erases a lot of information about the space: Intuitively, it only keeps the paths, which are all of dimension 1 and forgets all the higher dimensional data. Indeed there are known examples of different spaces such as the 2-dimensional disk and the 2-dimensional sphere (see Figure 1), that have equivalent fundamental groupoids (topologist justify this claim by saying that they are connected and simply connected). A way to address this problem is to remember the homotopies

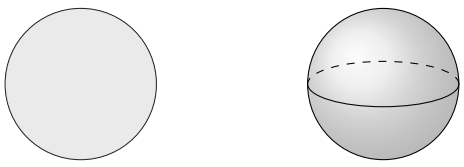


Figure 1: The disk and the sphere

in the fundamental groupoid, as an extra structure instead of quotienting them. In doing so, we make the paths non-associative. Moreover the homotopies themselves can be composed and this composition is associative only up to a continuous deformation (a 2-*homotopy*). One can once again quotient these 2-homotopies to define the *fundamental* 2-*groupoid* of a space. Algebraically, this is an instance of a weak 2-groupoid. We can also iterate this process by taking in account the 2-homotopies, which require 3-homotopies and so on, and we finally obtain a structure with infinitely many levels, each of which is equipped with a composition that is associative only up to the next level. We call this structure the *fundamental* ω-*groupoid* of a space, and it is the subject of study of homotopy theory. The usual way of constructing the fundamental ω-groupoids of all spaces is to endow the category of spaces with a *model structure* and to compute the *homotopy category* of this model structure: It is the category obtained by forcing all the weak equivalences to become isomorphisms. This formalizes the idea of considering the object of the category up to an appropriate notion of equivalence. For the category of spaces, the space up to equivalence are called the *homotopy types*.

Homotopy hypothesis. The homotopy hypothesis states that weak ω-groupoids are Quillen equivalent to the category of homotopy types. Depending on the chosen definition for weak ω-groupoids, it is either a conjecture or a theorem, and it should rather be understood as a guideline for a satisfactory definition of weak ω-groupoid than a theorem: it prescribes that weak ω-groupoids should be a description of the homotopy types. Moreover, most people interested in providing such a description often require it to be algebraic in a sense that is not fully formal but generally means that not only cells do compose, but there is also a chosen witness for their composition. There is at the moment no definition of weak ω-groupoid that is known to satisfy the homotopy hypothesis and recognized as enjoying algebraicity. However, it is mostly admitted that Grothendieck's definition of weak ω-groupoids (that is algebraic) satisfy this hypothesis. We admit this result as well, for the sake of giving the motivation of our work, even though it plays a minor role in this thesis.

Homotopy type theory

One of the major recent development of type theory is homotopy type theory, and it comes from a subtle observation of the role played by the identity types in Martin-Löf type theory.

Identity types. In Martin-Löf type theory, given two terms t, u of the same type A, one can construct the type $t = u$, whose inhabitants are proofs that t and u are equal. If t and u are not equal then the type $t = u$ is not inhabited, and we refer the reader to [63] for a complete presentation of these types, and of other inductive types. The terms of these types have a very interesting properties: they can be composed. Given a term p of type $t = u$ and a term q of type $u = v$, one can define a term $p \cdot q$ of type $t = v$, this property is usually referred to as the transitivity of equality, but since we are working in a constructive setting we prefer see this as a way to compose two proofs in order to produce a new proof. A natural question to ask is whether this composition is associative, and it turns out that it is not: if one compute $(p \cdot q) \cdot r$ and $p \cdot (q \cdot t)$ completely, they define different terms, however, both these terms are of the same type, and as it turns out, the type $(p \cdot q) \cdot r = p \cdot (q \cdot r)$ is inhabited. Hence the composition of equality proofs is associative up to an equality proof. This is reminiscent of our description of the fundamental ω-groupoid of a space. Moreover this argument can be repeated for identity types over identity types, and identity types over identity types over identity types and so on. Making these arguments formal [52, 64, 2] was one of they key progress of homotopy type theory.

It showed that the iterated identity types naturally endow each type with a structure of weak ω-groupoids.

Homotopy type theory and univalent foundations. The idea of making use of Martin-Löf type theory in order to study homotopy theory is originally due to Awodey [8], who is at the origin of the term *homotopy type theory*. This idea was then backed up by the formal proof that the identity types carry a structure of weak ω-groupoid, which was the original motivation behind this correspondence. Independently, Voevodsky introduced the *univalence axiom* [7], which provides a way to characterize the identity types between types themselves, and studied the models of the type theory with univalence within simplicial sets. This initiated the study of the connection between type theory and homotopy theory as an independent and active field of research. Since the special year at the IAS about univalent foundations [63], the term homotopy type theory generally designates a type theory which has both the univalence axioms and *higher inductive types*: a generalization of inductive that allows to introduce generators for the identity types.

Synthetic homotopy theory. Martin-Löf type theory was initially introduced as a setting to perform constructive mathematics, and homotopy type theory takes this approach further and provides a setting for *synthetic homotopy theory*: The types with their iterated identities are identified with weak ω-groupoids, which under the homotopy hypothesis are equivalent to homotopy types. It then becomes possible to translate a result about the structure of the identities of a type to a result about the fundamental ω-groupoid of a space. Along these lines various results of homotopy theory have been formalized in one of the implementations of homotopy type theory, such as the universal cover of the circle [51], some of the homotopy groups of the spheres [20] and the Blackers-Massey theorem [3]. Usual constructive mathematics can be seen as a special case of synthetic homotopy theory manipulating only types that are 1-truncated (i.e., types that correspond to groupoids with no non trivial (> 1)-cells).

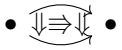
Models of homotopy type theory. Homotopy type theory still has a lot of open problems, among which we are particularly interested in the question of its models. As we have mentioned, a type theory is a generalized algebraic theory, and so is the case for homotopy type theory. A legitimate question for an algebraic theory is then to understand its models: all object of a given category that satisfy the axioms defined by the theory. This line of study is often called *categorical semantics*. In the case of homotopy type theory, one is not simply interested by any model, but specifically by the ones that interpret the identity types as homotopies.

A type theory for the theory of weak ω-groupoids. From the rules that generate the identity types in homotopy type theory, Brunerie [20] has extracted a minimal setting of a type theory that describe weak ω-groupoids. This type theory does not have the type formers of homotopy type theory, and is less powerful than it, as it does not allow for a developing constructive mathematics in general. However it pinpoints exactly the process that spans weak ω-groupoids. He has proved that the algebraic theory corresponding to this theory is the theory of weak ω-groupoids, as defined by Grothendieck [39]. Our work is heavily influenced by this idea, and generalize it to weak ω-categories.

Higher categories

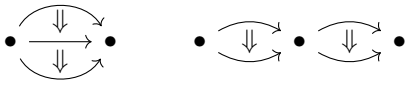
We are interested in a generalization of the structure of weak ω-groupoid which adds a preferred direction to each of the cells of the groupoid. These are called weak ω-categories. By default, all our structures are weak, so unless stated otherwise, when we refer to ω-categories, we always mean weak ω-categories. Similarly to ω-groupoids, ω-categories are higher structures, which make them very difficult to define, and allow for a lot of different definitions. For instance, one can chose the cells to look like higher dimensional disks, higher dimensional triangles, higher dimensional cubes, or to have more exotic shapes like the opetopes. As a result, the literature on ω-categories consists mainly in a lot of definitions, along with various proofs of equivalence between some of these definitions. We refer the reader to the surveys [47] and [25] for a complete overview of this literature, and we focus on giving an intuitive introduction to a particular approach to ω-categories. This approach was proposed by Maltsiniotis [54], as a variation on the definition of ω-groupoid defined by Grothendieck. We give here a intuitive approach to motivate our work, and present a formal definition in Section 2.1. This definition was studied extensively by Ara [4] who has in particular proven it to be equivalent to a definition introduced by by Batanin [11] and Leinster [48].

Cells and compositions. A weak ω-category is a structure that contains n-cells, for each $n \in \mathbb{N}$, that we represent graphically as filling discs in higher dimensions as follows

n	typical n-cell
0	
1	
2	
3	

In addition, the cells are also required to be able to be composed, when adequate requirements are met, as is the case for instance for two 1-cells in the following situation

or for a pair of 2-cells in one of the following situations, that we call respectively the *vertical composition* and the *horizontal composition*.

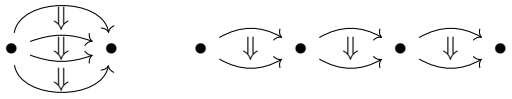


In the second case the composition of the 2-cells relies on the composition of the one cells, as for composing such a diagram, we first compose the two 1-cells on the top together and the two bottom 1-cells together. Cells in higher dimension have more and more ways to be composed: 3-cells can be composed in 3 ways, 4-cells in 4 ways, and so on.

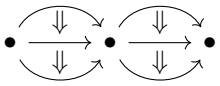
Associativity and exchange. These compositions are required to satisfy axioms that generalize the usual axioms of composition in a category, most notably they must be associative. In

the case of the 1-cells, this is the usual associativity, which is usually expressed by the equation $h \circ (g \circ f) = (h \circ g) \circ f$. Our preferred way of stating this result is by saying that the diagram

defines a composition which is unambiguous. Indeed, one could chose to first compose together the two cells on the left and compose the result with the cell on the right, or the other way around, one could compose the two cells on the right and compose the result with the cell on the left. The associativity requires these two compositions to be equal. Similarly, we require the associativity for the two compositions of 2-cells, that we can state by the fact that the two following diagrams define an unambiguous operation.



We also require an additional condition imposing that the two compositions that we have defined on the two cells interact nicely. This condition is called the *exchange law*, and can also be expressed in the same way with diagrams. It states that the following diagram defines an unambiguous operation



Intuitively, there are two strategies to compose this diagram: either one can compose horizontally the two 2-cells on the top together, and the two 2-cells on the bottom together, and then compose the results vertically, or the other way around, one can compose vertically the two 2-cells on the left together and the two 2-cells on the right together, and then compose the results horizontally. The exchange law states that these two ways should be equal. The conditions and associativity requirements become harder to express as the dimension increases, but can still be fully expressed.

Weak axioms. The presentation that we have sketched until now gives rise to *strict* ω-categories. Indeed, for every axiom like associativity, we have required an equality between two composition. Although these are important structure in their own right, our objective is to study the weak variant of this structure, since it is the right setting to describe situations arising from homotopy theory or from type theory. Intuitively, to obtain a weak structure, we need to shift perspective for what it means for a diagram to compose unambiguously: From the strict point of view that we have adopted until now, it means that there is only one operation that compose the diagram. In a weaker setting, like from the point of view of homotopy theory, one would typically only require that there is a contractible space of ways of composing these diagrams. From our more algebraic perspective, we use a variant of this requirement to be contractible, that we express as follows. We require any two ways of composing such a diagram to be related by a higher cell that we call a *witness*. For instance in the case of the associativity, we require that for any two ways f and g of composing the following diagram

there exists a 2-cell that relates f to g as follows (and since this is for any two ways of composing, we can switch f and g to get a witness the other way around)

Weak ω-categories are obtained by relaxing all the conditions that we have expressed about diagrams defining a composition in an unambiguous way the same way as we have just illustrated for the associativity. Note that the new witnesses that are obtained by these relaxed compositions are now cells in the categories, and as such, they must compose like any other cell, and these composition have to themselves satisfy (weak) axioms and so on.

Unbiased categories. This discussion about weak ω-categories illustrates how intricate they are, and shows how defining them with a list of axiom can quickly get out of hands, hence the necessity to find a more homogeneous way to describe them. One solution is to work with an *unbiased* version of weak ω-categories, that is, for instance, instead of requiring the existence of a composition of two 1-cells, we require the existence of such a composition for arbitrarily long sequences of composable 1-cells. Since one can apply successively the composition of 1-cells several times, there was already a way to compose such arbitrarily long sequence. Hence requiring a new way of composing them should lead to equivalent (in a informal weak sense) structures, if we also require this new composition and the ones that were already there to be equal (again in a weak sense). Unbiased weak ω-categories are obtained by applying this line of thinking to every dimension and every operation, and defining as primitive any composition of previously defined operations. It may seem that doing so we add a lot of unnecessary data, but it also allows for a more homogeneous description of weak ω-categories. In fact the ω-categories we manipulate both in type theory and in our corresponding mathematical description are always unbiased.

A type theory for weak ω-categories. The definition of weak ω-categories due to Maltsiniotis [54] is a variation on Grothendieck's approach to define weak ω-groupoids [39]. Using a type-theoretic approaches, Brunerie has given [20] a reformulation of Grothendieck's definition of weak ω-categories. Combining both of these approach, Finster and Mimram have carried over [33] Maltsiniotis' variation in a type theoretic style similar to Brunerie's work. This is the starting point of our approach to weak ω-categories, using this type-theoretic definition as a tool. In their article [33], Finster and Mimram have left several points about this approach unanswered, most notably about the models of this theory. One of our goals is to answer these questions. We also discuss about the practical implications of such a theory and the implementation of a proof-assistant based on it, as well as generalization to other variations of weak ω-categories.

Directed type theory. A very common approach in type theory is to use a type theory as an axiomatic framework in which one can perform mathematics internally. This is the approach of Martin-Löf type theory and its variants, in which one encounters various constructions such as the Σ-types, and the Π-types, or the identity types, whose behaviour turn out to define the structure of ω-groupoids. There have been several attempts to define a similar type theory where the identity types carry a direction [50, 58, 56] however none of these achieve at the moment a structure that adding a direction in all the dimension, that would encompass every type with the structure of a weak ω-category. Our approach is orthogonal to this one, we introduce a type theory in order to encode the combinatorics of weak ω-categories within its rules. This theory does not have the usual Σ-types and Π-types, and does not intend to be a suitable framework to perform mathematics or replace set theory.

Structure of the thesis

We first introduce the generic background about type theory, in order to recall the common notations and conventions of this field. Along with this introduction, we present a fairly generic

and modular framework we rely on to introduce and study type theories. We also specify the tools from categorical semantics that we use all along the thesis. In the second chapter, we give a formal definition of weak ω-categories, and present the type theory CaTT that describes these ω-categories and upon which most of our work relies. We then use our categorical semantics tools to prove the equivalence between the mathematical definition and the type theory we have introduced. In the third chapter, we study the practical implications of CaTT, and in particular how it can be used to implement a proof-assistant. We then study two features that allow for partially automating the usage of this proof assistant. In the fourth chapter we briefly describe a much more general framework to present higher dimensional theories, using type theory, and establish connections with similar notions from categorical logic. Finally, in the last two chapters, we use this general framework to introduce and study two new type theories describing variations of weak ω-categories: monoidal weak ω-categories (chapter 5) and cubical weak ω-categories (chapter 6).

Chapter 1

Introduction to type theory

1.1 Introduction to type theories

We first present the notion of type theory that we extensively use to describe all our theories. There are many variations and flavors of type theory, as it is more a generic guideline and style for introducing a theory than a fully defined notion. For our purposes, we are interested in dependent type theories with weakening, exchange and contraction, and we only present a framework that implements these features as is usually the case for dependent types. We present two ways for introducing type theories, as they both provide a different insight, and both have their respective drawbacks. Our presentation of type theory is heavily influenced by Shulman [59].

1.1.1 Conventions and notations

For our purposes, a type theory is a framework that manipulates four kind of objects, that we introduce here together with the respective notation conventions we adopt.

- *Types* : $A, B, \ldots$
- *Terms* : $t, u, \ldots$
- *Contexts* : $\Gamma, \Delta, \ldots$
- *Substitutions* : $\gamma, \delta, \ldots$

In the different styles, these objects will not be implemented in the same way, hence we do not provide yet a specific implementation for them.

Judgments. These objects are related to each other by the means of *judgments*, which can be thought of as the statements that are expressible in the theory. These judgments also come in four kinds, one for each kind of object the theory manipulates.

$\Gamma \vdash$	Γ is a well-formed context
$\Gamma \vdash A$	A is a well-formed type in the context Γ
$\Gamma \vdash t : A$	t is a term of type A in the context Γ
$\Gamma \vdash \delta : \Delta$	δ is a substitution from Γ to Δ

The intuition is that in type theory, the objects we study do not make sense on their own, they only make sense in relation to one another, and these judgments are our way to express the

relations: Their are too many expressions and the judgments are a way to select only some of them. The primary concept that we want to study are types, they can be thought of as describing the kind of things that a term can be. For instance mathematicians often consider the type of natural numbers, or the types of real-valued functions. Types are allowed to depend on arguments, as for example does the type of square real matrices of size n, however this only makes sense under the assumption that n is a natural number (which we state in type theory as n is of the type of natural numbers). It is the role of the contexts to encode those assumed types of the variables used in the types and in the terms. Thus the judgment $\Gamma \vdash$ expresses the fact that Γ defines a valid set of such assumptions, the judgment $\Gamma \vdash A$ expresses the fact that under the assumptions defined by the context Γ, A defines a valid type, the judgment $\Gamma \vdash t : A$ expresses the fact that under the assumptions defined by Γ the term t is of type A. Finally substitutions can be thought of as ways to relate these assumptions: The judgment $\Delta \vdash \gamma : \Gamma$ expresses the fact that the assumptions Δ and Γ are related in such a way that for everything happening under the assumptions Γ there is a corresponding thing happening under the assumptions Δ.

Derivations. Type theories are usually formulated in terms of *inference rules*, these are expressions of the form

$$\frac{\mathcal{J}_1 \qquad \cdots \qquad \mathcal{J}_n}{\mathcal{J}}$$

where $\mathcal{J}, \mathcal{J}_1, \ldots, \mathcal{J}_n$ are all judgments of the theory. The judgment $\mathcal{J}$ is called the *conclusion* of the rule, while the judgments $\mathcal{J}_1, \ldots, \mathcal{J}_n$ are called its *premises*. These rules can be assembled into *derivation trees*, by plugging in, for each premise a rule whose conclusion match this premise. We write these trees as indicated in the following example

$$\dfrac{\dfrac{\mathcal{J}_1 \qquad \mathcal{J}_2 \qquad \mathcal{J}_3}{\mathcal{J}_4} \qquad \dfrac{}{\mathcal{J}_5}}{\mathcal{J}}$$

A derivation tree whose leaves are all rules with no premises and whose conclusion is $\mathcal{J}$ is called a *derivation* of $\mathcal{J}$, and a judgment in a type theory which has a derivation using only the rules of the type theory is called *derivable*, they are the judgments we are interested in. In a given type theory, a rule

$$\frac{\mathcal{J}_1 \qquad \cdots \qquad \mathcal{J}_n}{\mathcal{J}}$$

that is not part of the theory is said to be *derivable* if there exists a derivation tree whose premises are $\mathcal{J}_1, \ldots, \mathcal{J}_n$ and whose conclusion is $\mathcal{J}$. The rule is said to be *admissible* if provided a derivation of each of the judgments $\mathcal{J}_1, \ldots, \mathcal{J}_n$, there is a derivation of $\mathcal{J}$.

1.1.2 Cut-full type theory

We present our first formulation of the rule of a type theory, following [32], that we call a *cut-full* type theory. This style is the closest to the categorical semantics we provide but present drawbacks for the practical use and implementation of a type theory. It is an *intrinsic* approach, in the sense that there is no syntax independently from the rules, and the rules both define the syntax and specify its behavior at the same time.

Structural rules. Among the rules that define a theory, we distinguish between the *structural rules* and the rules associated to types and terms constructors: The structural rules are shared by all the type theories we consider, and indicate how the theory is structured around the types and the terms, in particular they allow for constructing the contexts and substitutions. On the contrary, the rules for types and terms are specific to each type theory. The term 'structural rule" is usually used to designate only some of these rules, along with contractions and weakening; we use this terminology here which diverges slightly from the norm as these rules are the ones that generate all the structure of a dependent type theory.

For contexts:

$$\frac{}{\varnothing \vdash} \qquad \frac{\Gamma \vdash A}{\Gamma, A \vdash}$$

For types:

$$\frac{\Delta \vdash A \qquad \Gamma \vdash \delta : \Delta}{\Gamma \vdash A[\delta]}$$

For terms:

$$\frac{\Gamma, A \vdash}{\Gamma, A \vdash p_{\Gamma,A} : A[\pi_{\Gamma,A}]} \qquad \frac{\Delta \vdash t : A \qquad \Gamma \vdash \delta : \Delta}{\Gamma \vdash t[\delta] : B[\delta]}$$

For substitutions:

$$\frac{\Gamma \vdash}{\Gamma \vdash \langle\rangle : \varnothing} \qquad \frac{\Gamma, A \vdash}{\Gamma, A \vdash \pi_{\Gamma,A} : \Gamma}$$

$$\frac{\Gamma \vdash}{\Gamma \vdash \mathrm{id}_\Gamma : \Gamma} \qquad \frac{\Xi \vdash \delta : \Delta \qquad \Delta \vdash \gamma : \Gamma}{\Xi \vdash \gamma \circ \delta : \Gamma}$$

$$\frac{\Gamma \vdash \delta : \Delta \qquad \Delta, A \vdash \qquad \Gamma \vdash t : A[\delta]}{\Gamma \vdash \langle \delta, t \rangle : (\Delta, A)}$$

The entities introduced in these rules have to interact with each other and satisfy some axioms, called *definitional equalities*. We express these axioms in the form of inference rules, and they have to be understood as, under the conditions in the premise of the rules, the equality between

the two entities in the conclusion of the rule must hold.

$$\frac{\Psi \vdash \xi : \Xi \qquad \Xi \vdash \delta : \Delta \qquad \Delta \vdash \gamma : \Gamma}{\Psi \vdash (\gamma \circ \delta) \circ \xi \equiv \gamma \circ (\delta \circ \xi) : \Gamma}$$

$$\frac{\Delta \vdash \gamma : \Gamma}{\Delta \vdash \gamma \circ \mathrm{id}_\Delta \equiv \gamma : \Gamma} \qquad \frac{\Delta \vdash \gamma : \Gamma}{\Delta \vdash \mathrm{id}_\Gamma \circ \gamma \equiv \gamma : \Gamma}$$

$$\frac{\Xi \vdash \delta : \Delta \qquad \Delta \vdash \gamma : \Gamma \qquad \Gamma \vdash A}{\Xi \vdash A[\gamma \circ \delta] \equiv A[\gamma][\delta]} \qquad \frac{\Gamma \vdash A}{\Gamma \vdash A[\mathrm{id}_\Gamma] \equiv A}$$

$$\frac{\Xi \vdash \delta : \Delta \qquad \Delta \vdash \gamma : \Gamma \qquad \Gamma \vdash t : A}{\Xi \vdash t[\gamma \circ \delta] \equiv t[\gamma][\delta] : A[\gamma \circ \delta]} \qquad \frac{\Gamma \vdash t : A}{\Gamma \vdash t[\mathrm{id}_\Gamma] \equiv t : A}$$

$$\frac{\Gamma \vdash \gamma : \varnothing}{\Gamma \vdash \gamma \equiv \langle\rangle : \varnothing} \qquad \frac{}{\Gamma, A \vdash \mathrm{id}_{\Gamma,A} \equiv \langle \pi_{\Gamma,A}, p_{\Gamma,A} \rangle}$$

$$\frac{\Delta \vdash \langle \gamma, t \rangle : (\Gamma, t)}{\Delta \vdash \pi_{\Gamma,A} \circ \langle \gamma, t \rangle \equiv \gamma : \Gamma} \qquad \frac{\Delta \vdash \langle \gamma, t \rangle : (\Gamma, A)}{\Delta \vdash p_{\Gamma,A}[\langle \gamma, t \rangle] \equiv t : A[\gamma]}$$

$$\frac{\Delta \vdash \langle \gamma, t \rangle : (\Gamma, A) \qquad \Xi \vdash \delta : \Delta}{\Xi \vdash \langle \gamma, t \rangle \circ \delta \equiv \langle \gamma \circ \delta, t[\delta] \rangle : (\Gamma, A)}$$

We denote $\mathrm{wk}_{\Gamma,A}(B)$ the type $B[\pi_{\Gamma,A}]$, and think of it as a *weakening*: Given a type B in a context Γ it produces a corresponding type in the context Γ, A. With this notation, a type theory as presented above supports the weakening, that is the following rule is derivable

$$\frac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma, A \vdash \mathrm{wk}_{\Gamma,A}(B)}$$

These structural rules for type theory are very close to the categorical semantics that we present in Section 1.2. However, from a computational standpoint they have some drawbacks: Since we introduce definitional equalities, a same judgment may be derived in various ways. Hence it is unclear how a type checker could be defined for this system. Indeed, checking the equality between two expressions, which is essential for type checking, relies on the previously introduced definitional equalities and requires an algorithm two determine whether two expressions belong to the same equivalence class, for the equivalence relation generated by those equalities. The problem of finding such an algorithm is known as a word, and it is known in the general case to be undecidable. It is complicated, when providing a type theory in this style , to show that its associated word problem and hence its type checking is decidable.

Type and term constructors. We add new features to introduce types and terms in the theory that we call *type constructor* and *term constructor*. Both of them come with an associated *introduction rules*, which explains under which premises it should be used. We illustrate this concept by defining the introduction for the unique type in a simple type theory with only one type (by "simple type theory" we mean that the type constructors do not depend on the terms,). This theory has only one type constructor, that we denote $\star$. Intuitively, we want to describe a system in which there is exactly one type in each valid context. Our first intuition is then to

give the following rule

$$\frac{\Gamma \vdash}{\Gamma \vdash \star}$$

However, this rule is problematic, as it clashes with the presentation we have given. For instance, there are now two derivable types in the context $(x : \star)$, namely

$$\star \vdash \star \qquad \text{and} \qquad \star \vdash \mathrm{wk}_\star(\star)$$

This comes from the fact that the rule we have given has "built-in" weakening, while the structural rule also has weakening, hence in order to perform a weakening, one has the choice to either use the structural rule for weakening applied to the introduction rule, or to immediately use the introduction rule with the appropriate context, thus skipping the explicit weakening altogether. In order to correct this problem, a solution is to modify the introduction rule in order to allow it only in the "minimal context" in which it applies, shaving off the built-in weakening of it. In our example, the correct rule is

$$\frac{}{\varnothing \vdash \star}$$

Taking this example into the more complicated territory of dependent type theory makes us realize that a general pattern for the introduction rules for types of a theory presented in this form is

$$\frac{}{\Gamma \vdash C}$$

where Γ is a specific context of the theory, and C is a type constructor. For instance, consider the theory with the constructor $\star$ as above, and a constructor $\to$, which given two terms t, u of type $\star$ in a context produces a type $t \to u$, then the introduction rule for this constructor is

$$\frac{}{(x : \star, y : \star) \vdash x \to y}$$

Since types then depend on these "minimal contexts", which themselves depend on types, this forces us to define these "minimal contexts" mutually inductively with the rules of the theory and thus each theory has to be defined and studied separately, as one needs to show every time that these mutually inductive definition are well-formed. We call a cut-full type theory any theory implemented in this style, which has all the structural rules as well as introduction rules for types and terms.

1.1.3 Cut-free type theory

We give a more tractable presentation of type theory. This presentation is closer to the syntactic aspects, and makes explicit use of the variables (not just as convenience for writing). Moreover, this setting can be seen as less theoretically robust as the previous one, for reasons we shall explain after with the presentation of the structural rules. However, it also makes the introduction rules for types and terms a lot more easier to describe, and matches more the intuition of arity of operations. We present here an introduction for this style of type theory, and state many of its properties, without proofs. In Section 4.1, we introduce a precise framework for manipulating a large number of type theories that we study in a cut-free style, and we have formalized this framework along with all the properties we state here in the proof-assistant Agda [12]. We thus postpone the proofs to Section 4.1, where we present our formal proofs for the aforementioned

framework. This is an extrinsic presentation of a type theory: the syntax exists in its own right and the rules only allow to specify which of the preexisting syntactic expression have to be taken as valid from the point of view of the theory.

Signature. In this style of presentation, we make a more heavy use of the syntactic aspect of type theory, and consider that the contexts, types, terms, and substitutions all have an underlying *expression*, which has an existence in its own right. We call *pre-contexts* (resp. *pre-types*, *pre-terms* and *pre-substitutions*) the set of all expressions of all contexts (resp. types, terms and substitutions). We axiomatize these sets, and understand the judgments in a slightly different way, for instance, we conceive the judgment $\Gamma \vdash$ as statement that makes sense for every pre-context Γ, but that is only derivable for the pre-contexts Γ that are in fact contexts.

In order to present this style of theory, we start with a infinitely countable set of *variables* $\mathcal{V}$ (whose elements we denote $x, y, \ldots$). We also assume type constructors and term constructors, each of them being equipped with an arity that is a natural number. We then define by induction

- Pre-contexts to be lists of couples $(x : A)$ where x is a variable and A a pre-type. The empty list is denoted $\varnothing$, and concatenation is denoted $(\Gamma, x : A)$
- Pre-types to be the expressions formed by $C(t_1, \ldots, t_n)$ where C is a type constructor of arity n and $t_1, \ldots, t_n$ are pre-terms.
- Pre-terms to be either variables or expressions of the form $T(t_1, \ldots, t_n)$ where T is a term constructor of arity n and $t_1, \ldots, t_n$ are pre-terms
- Pre-substitutions to be lists of couples $(x \mapsto t)$ where x is a variable and t is a pre-term. The empty list is denoted $\langle\rangle$ and concatenation is denoted $\langle \gamma, x \mapsto t \rangle$

Given a syntactical expression, we define its *set of variables*, by induction on the syntax as follows

$$\begin{aligned}
\mathrm{Var}(\varnothing) &= \emptyset & \mathrm{Var}(\Gamma, x : A) &= \mathrm{Var}(\Gamma) \cup \{x\} \\
& & \mathrm{Var}(C(t_1, \ldots, t_n)) &= \bigcup_{i=1}^{n} \mathrm{Var}(t)_i \\
\mathrm{Var}(x) &= \{x\} & \mathrm{Var}(T(t_1, \ldots, t_n)) &= \bigcup_{i=1}^{n} \mathrm{Var}(t)_i \\
\mathrm{Var}(\langle\rangle) &= \{\} & \mathrm{Var}(\gamma, x \mapsto t) &= \mathrm{Var}(\gamma) \cup \mathrm{Var}(t)
\end{aligned}$$

Intuitively, they are all the variables needed to write the corresponding expression. For a type or a term, this set is often referred to as the set of *free variables*. For substitutions, we do not consider the variables appearing on the left of the mappings $x \mapsto t$ as they are merely placeholders for defining the action.

We say that x appears with type A in Γ and write $(x : A) \in \Gamma$ if either $\Gamma = (\Gamma', x : A)$ or $\Gamma = (\Gamma', y : B)$ with $(x : A) \in \Gamma$ and $y \neq x$. Similarly, we say that the mapping $(x \mapsto t)$ appears in γ or that γ sends x to t and write $(x \mapsto t) \in \gamma$ if either $\gamma = \langle \gamma', x \mapsto t \rangle$ or $\gamma = \langle \gamma', y \mapsto u \rangle$ with $(x \mapsto t) \in \gamma'$ and $x \neq y$.

Action of substitution. In this style of presentation, we can combine together syntactic entities to get another syntactic entity, which gives a computation directly on the syntax of the theory, regardless of the derivation rules. More precisely, the pre-substitutions act on the pre-types and on the pre-terms: Given a pre-substitution γ and a pre-type A (resp. a pre-term t),

we define the pre-type $A[\gamma]$ (resp. the pre-term $t[\gamma]$). This action can be defined by induction on the syntax with the following formulas

– For each type constructor C of arity n:

$$C(t_1, \dots, t_n)[\gamma] = C(t_1[\gamma], \dots, t_n[\gamma])$$

– For each variable x:

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

– For each term constructor T of arity n:

$$T(u_1, \dots, u_n)[\gamma] = T(u_1[\gamma], \dots, u_n[\gamma])$$

This action also lets us define a composition of pre-substitutions: Given two pre-substitutions γ and δ, it produces a new pre-substitution $\delta \circ \gamma$, that we define inductively as follows.

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

It is sometimes convenient to think of this composition as the pre-substitution γ acting on the right on the pre-substitution δ, by analogy with how it acts on types and terms. These actions are compatible in the following sense

Proposition 1. *Given two pre-substitution δ, γ, for any pre-type A, we have $A[\delta \circ \gamma] = A[\delta][\gamma]$ and for any pre-term t we have $t[\delta \circ \gamma] = t[\delta][\gamma]$*

We also define a particular pre-substitution id_Γ associated to a context Γ that we call the *identity substitution* of Γ by induction as follows

$$\mathrm{id}_\varnothing = \langle\rangle \qquad\qquad \mathrm{id}_{\Gamma, x:A} = \langle \mathrm{id}_\Gamma, x \mapsto x\rangle$$

By definition of the action of substitutions, this substitution satisfies for all pre-type A, $A[\mathrm{id}_\Gamma] = A$ and for all pre-term t, $t[\mathrm{id}_\Gamma] = t$, and thus for all substitution δ, $\delta \circ \mathrm{id}_\Gamma = \delta$.

Structural rules. In a cut-free type theory, the judgments are subject to the following structural rules.

For contexts:

$$\frac{}{\varnothing \vdash}(\text{EC}) \qquad\qquad \frac{\Gamma \vdash A}{\Gamma, x : A \vdash}(\text{CE}) \quad \text{Where } x \notin \mathrm{Var}(\Gamma)$$

For terms:

$$\frac{\Gamma \vdash \qquad (x : A) \in \Gamma}{\Gamma \vdash x : A}(\text{VAR})$$

For substitutions:

$$\frac{\Delta \vdash}{\Delta \vdash \langle\rangle : \varnothing}(\text{ES}) \qquad\qquad \frac{\Delta \vdash \gamma : \Gamma \qquad \Gamma, x : A \vdash \qquad \Delta \vdash t : A[\gamma]}{\Delta \vdash \langle \gamma, x \mapsto t\rangle : (\Gamma, x : A)}(\text{SE})$$

In this presentation, most of the rules that were primitive in the previous one become computable operations on the objects of the syntax. We thus define the substitutions $\pi_{\Gamma,A}$ and id_Γ, as well as the composition of substitutions $\circ$ and the action of substitutions on terms $A[\gamma]$ and $t[\gamma]$ from

the rules that we have presented here, and then show that these definitions satisfy on the nose the definitional equalities that we have previously introduced. Hence this presentation enjoys the uniqueness of derivations: Each derivable judgment has only one derivation. However, note that the rule (VAR) allows for the derivation of the judgment $\Gamma \vdash x : A$ without ensuring that the judgment $\Gamma \vdash A$ is also derivable. It relies on the fact that weakening for types can be derived from the introduction rules for types and terms. So in this presentation, not all introduction rules give a valid type theory, only the ones that allow for derivability of the rule of weakening for types in order to ensure the important properties that we want type theories to satisfy, presented in Proposition 2.

Types and term constructors. In practice, in order to ensure that weakening for types is valid, and hence that the whole theory enjoys the properties of Proposition 2, we only allow for introduction rules for types and terms that have "built-in" cut. Considering again our example of a simple type theory with only one type $\star$, in this style, the introduction rule for $\star$ is now

$$\frac{\Gamma \vdash}{\Gamma \vdash \star}$$

In general, in this setting, a type constructor C or arity n obeys a rule of the form

$$\frac{\Gamma \vdash t_1 : A_1 \qquad \cdots \qquad \Gamma \vdash t_n : A_n}{\Gamma \vdash C(t_1, \ldots, t_n)}$$

whereas a term constructor T of arity n obeys a rule of the form

$$\frac{\Gamma \vdash t_1 : A_1 \qquad \cdots \qquad \Gamma \vdash t_n : A_n \qquad \Gamma \vdash B}{\Gamma \vdash T(t_1, \ldots, t_n) : B}$$

The reader familiar with variations of Martin-Löf type theory may notice that this particular form for the rules does not capture Σ-types and Π-types - In fact we do not allow for any variable bindings: All the variables in a term are free. The rules allowing for variable bindings are also expressible in a way that builds-in the cuts, and we need not restrict ourselves here to rules that disallow them for the rest of our presentation of cut-free type theories. However, since we only use rules that prevent variable bindings, we have proved and formalized all our results in a framework that only allows these rules.

Proposition 2. *In a cut-free type theory, all the entities that appear in a derivable judgment are also derivable, and these theories support the weakening. Moreover, valid expression only use variables declared in the context. More precisely, the following hold*

- *For every derivable judgment $\Delta \vdash A$, the judgment $\Delta \vdash$ is also derivable.*
- *For every derivable judgment $\Delta \vdash t : A$, the judgments $\Delta \vdash$ and $\Delta \vdash A$ are also derivable.*
- *For every derivable judgments $\Delta \vdash \gamma : \Gamma$, the judgments $\Delta \vdash$ and $\Gamma \vdash$ are also derivable.*
- *For every derivable judgment $(\Delta, x : A) \vdash$, if the judgment $\Delta \vdash B$ is derivable then so is $(\Delta, x : A) \vdash B$.*
- *For every derivable judgment $(\Delta, x : A) \vdash$, if the judgment $\Delta \vdash t : B$ is derivable then so is $(\Delta, x : A) \vdash t : B$.*

– *For every derivable judgment* $(\Delta, x : A)$*, if the judgment* $\Delta \vdash \gamma : \Gamma$ *is derivable then so is* $(\Delta, x : A) \vdash \gamma : \Gamma$*.*

– *For every derivable judgment* $\Delta \vdash A$*, we have* $\mathrm{Var}(A) \subset \mathrm{Var}(\Delta)$*.*

– *For every derivable judgment* $\Delta \vdash t : A$*, we have* $\mathrm{Var}(t) \subset \mathrm{Var}(\Delta)$*.*

– *For every derivable judgment* $\Delta \vdash \gamma : \Gamma$*, we have* $\mathrm{Var}(\gamma) \subset \mathrm{Var}(\Delta)$*, and moreover writing* $\gamma = \langle x_i \mapsto t_i \rangle_{0 \leq i \leq n}$ *and* $\Gamma = (y_i : A_i)_{0 \leq i \leq m}$ *we necessarily have* $n = m$ *and for all* i*,* $x_i = y_i$*.*

Cut admissibility. The action of pre-substitution that we have defined is compatible with the rule and hence lifts on substitutions.

Proposition 3. *The following rules are admissible*

$$\frac{\Delta \vdash \gamma : \Gamma \qquad \Gamma \vdash A}{\Delta \vdash A[\gamma]} \qquad \frac{\Delta \vdash \gamma : \Gamma \qquad \Gamma \vdash t : A}{\Delta \vdash t[\gamma] : A[\gamma]}$$

Composition, identity and projections. The composition of substitution and the identity substitution that we have defined on the pre-syntax is also compatible with the structural rules of the theory

Proposition 4. *The following rules are admissible*

$$\frac{\Delta \vdash \gamma : \Gamma \qquad \Gamma \vdash \delta : \Xi}{\Xi \vdash \delta \circ \gamma : \Delta} \qquad \frac{\Gamma \vdash}{\Gamma \vdash \mathrm{id}_\Gamma : \Gamma}$$

Note that Proposition 2 combined with this result also shows the admissibility of the rule

$$\frac{\Gamma, x : A \vdash}{\Gamma, x : A \vdash \mathrm{id}_\Gamma : \Gamma}$$

In order to distinguish between the two, we denote $\pi_{\Gamma,A}$ the identity pre-substitution seen in the judgment as above. Hence all the structural rules of the cut-full style type theory are valid in the cut-free style type theory, either as structural rules or only as admissible rules.

Equalities. All the definitional equalities that we have introduced in the cut-full version also hold our the cut-free presentation computationally. We have already mentioned some of these that hold regardless of the derivability:

$$A[\mathrm{id}_\Gamma] = A \qquad t[\mathrm{id}_\gamma] = t \qquad \delta \circ \mathrm{id}_\Gamma = \delta$$
$$A[\delta \circ \gamma] = A[\delta][\gamma] \qquad t[\delta \circ \gamma] = t[\delta][\gamma]$$
$$\langle \gamma, x \mapsto t \rangle \circ \delta = \langle \gamma \circ \delta, x \mapsto t[\gamma] \rangle$$

Others only hold under the assumption that they are derivable, and we have:

– Given three substitutions $\Psi \vdash \xi : \Xi$, $\Xi \vdash \delta : \Delta$ and $\Delta \vdash \gamma : \Gamma$, the following equality holds

$$\gamma \circ (\delta \circ \xi) = (\gamma \circ \delta) \circ \xi$$

– For any substitution $\Delta \vdash \delta : \Gamma$, the following equality holds (Note that we have already proved the other unit, as it holds directly on the syntax)

$$\mathrm{id}_\Gamma \circ \delta = \delta$$

– For any substitution $\Delta \vdash \langle \gamma, x \mapsto t \rangle : (\Gamma, x : A)$, the following equalities hold

$$\pi_{\Gamma,A} \circ \langle \gamma, x \mapsto t \rangle = \gamma \qquad\qquad x[\langle \gamma, x \mapsto t \rangle] = t$$

– For any substitution $\Delta \vdash \gamma : \varnothing$, we have $\gamma = \langle\rangle$

– For any context $(\Gamma, x : A) \vdash$, we have $\mathrm{id}_{\Gamma,x:A} = \langle \pi_{\Gamma,A}, x \mapsto x \rangle$

Hence a cut-free type theory implements completely a cut-full one, but enjoys better computational properties: Whereas the equalities were assumed in the cut-full type theory, they simply hold on the syntax in the cut-free type theory. More precisely, the following result holds

Proposition 5. *For any cut-free type theory T, there exists a cut-full type theory T' such that T every (derivable) context (resp. type, term, substitution) in the theory T translates to a context (resp. type, term, substitution) in the theory T', and conversely. The composition of the two translations gives the identity on the theory T and produces on the theory T' an expression which, while not necessarily syntactically equal, is always definitionally equal to the one we started with.*

This proposition relies on the particular form of the types and terms constructors, and proving completely such a result is beyond the scope of this thesis, however we can provide an intuition of how this correspondence works, in particular for substitutions: a non-empty substitution is always of the form $\Delta \vdash \langle \gamma, x \mapsto t \rangle : (\Gamma, x : A)$ in the theory T, thus we can send it, in the theory T' onto the substitution obtained by $\langle \gamma', t' \rangle$, where γ' is the translation of γ and t', the translation of t. A substitution in T' however may be of several forms: If it is of the form $\mathrm{id}_{\Gamma'}$, we can send it on the substitution id_Γ, where Γ is the translation in T of Γ', this is allowed since id_Γ is defined in the theory T. Similarly, if it is of the form $\delta' \circ \gamma'$, we can send it on the substitution $\delta \circ \gamma$ where γ is the translation of γ' and δ the translation of δ', and similarly for all the other possible forms for a substitution in T'. From these associations, we can see that taking a substitution from T into T' and then back to T yields the same substitution we started with. Conversely, starting from a substitution of T' and taking it into T and then back to T', we may change the term: If for example the original term of T' is of the form $\delta' \circ \gamma'$, then it gets translated as a composite in T, which is an operation on the syntax and thus can be computed to yield a substitution of the form $\langle x_i \mapsto t_i \rangle$; this substitution is translated in T onto a substitution obtained by iterating the $\langle \_, \_ \rangle$ construction. Since the definitional equalities in T' are also satisfied in T, the new substitution that we get is definitionally equal to the substitution we started with, but has been *reduced* in the process. In fact this two ways translations between the two theories gives a *normalization procedure* on the theory T': Any expression of T' can be written in a unique way as an expression from T, thus defining a preferred orientation for the definitional equalities. This entire discussion holds for cut-free type theory that do not assume any definitional equality. In the case of cut-free type theories that assume definitional equalities, the equalities may interact in complicated ways, and there may not exist a normal form, hence there may not exists a unique expression of T corresponding to an expression of T'. However, the structural part of the theory T' can still be reduced, and while not presenting a normal form, the theory T still presents a reduced version of the theory T': It has less assumed definitional equalities.

Substitution extensionality. All the cut-free type theories also satisfy the following result, that is not theoretically very meaningful, but very important for reasoning

Lemma 6. *Given two substitutions* $\Delta \vdash \gamma : \Gamma$ *and* $\Delta \vdash \gamma' : \Gamma$ *such that for all variable* x *in* Γ*, we have the equality* $x[\gamma] = x[\gamma']$*, then* $\gamma = \gamma'$*.*

Proof. We prove this by induction on the context Γ

- For the context $\varnothing$, we necessarily have $\gamma = \gamma' = \langle\rangle$.
- For a context of the form $\Gamma = (\Gamma_0, x : A)$, we necessarily have $\gamma = \langle \gamma_0, x \mapsto t\rangle$ and $\gamma' = \langle \gamma'_0, x \mapsto u\rangle$, with $\Delta \vdash \gamma_0 : \Gamma$ and $\Delta \vdash \gamma'_0 : \Gamma$. Then for all variable y in Γ, either $y = x$, or y is a variable of Γ_0. Hence our assumption shows that $x[\langle \gamma_0, x \mapsto t\rangle] = x[\langle \gamma'_0, x \mapsto u\rangle]$ which translates to $t = u$, and for all variable y in Γ_0, $y[\gamma_0] = y[\gamma'_0]$. By induction this shows that $\gamma_0 = \gamma'_0$. □

Uniqueness of derivation. Cut-free type theories provide a setting which may enjoy uniqueness of derivations i.e., every derivable judgment has exactly one derivation, when this was impossible for cut-full type theory. Indeed, the structural rules of the type theory do not clash with each other: Every syntactic construction corresponds to a unique rule. This property may however depend on the type and term constructors along with the assumed definitional equalities between them. We prove and formalize that this property holds for a specific framework in Section 4.1 and Section 4.3, and admit for now that it holds for the theory we introduce.

1.1.4 Categorical structure of a type theory

The general framework that we have given both in a cut-free and in cut-full style for type theories ensures that, no matter what the terms and type constructors are, the theories expressed by this framework satisfy some properties and are endowed with a general structure. The aim of the categorical semantics that we introduce in Section 1.2 is to characterize precisely this structure, but we motivate it by the notion of *syntactic category*.

Syntactic category In both styles of type theory, the existence of identity substitutions and of composition of substitutions, as well as the associativity of the composition, and its left and right unitality show that the contexts and the substitutions can be assembled into a category. Given a type theory T, we call its *syntactic category*, and we denote $\mathcal{S}_T$ the category whose objects are the (well-formed) contexts $\Gamma \vdash$ in T, and whose morphisms $\Delta \to \Gamma$ are the (well-formed) substitutions $\Delta \vdash \delta : \Gamma$ in T. Since both styles of presentations also satisfy other properties, such as existence of types and terms, together with an action of substitutions on both, the syntactic category of a type theory actually carries more structure than a mere category, characterizing precisely this structure is the aim of Section 1.2

Our convention. For our purpose, we always present type theories in a cut-free style. We also implicitly work at the level of the syntax on a theory, and sometimes emphasize that we work on the pre-syntax by using the words "expression" or "syntactical". In Section 1.2, we introduce the categorical semantics of a type theory in a formalism closer to the cut-full version of the type theory, and use our previous discussion showing that the cut-free type theories also implement the structure of a cut-full type theory to ensure that this formalism encompasses the explicit type theories we work with.

Variables in the syntactic category. When constructing the syntactic category of a type theory presented in a cut-free style, we consider all the judgments of the theory up to α-*equivalence*, that is we identify two judgments that differ only by the names of their variables, but have the same inductive structure. For instance, given a type constructor $\star$ as above, the contexts $(x : \star, y : \star)$ and $(y : \star, z : \star)$ define the same objects in the category. There are various ways of achieving this. Firstly, we can replace in the type theory "contexts" by "equivalence classes of contexts" and "substitutions" by "equivalence classes of substitutions". This has the drawback that the entities in the syntactic category do not correspond to syntactical entities, but to equivalence classes of them. Another solution is to add conditions on the variables that a context is allowed to use, to ensure that only one of the possible order is allowed. This amounts to choosing a syntax that gives a specific representative for each of the aforementioned equivalence classes. Examples of such conditions are the *de Bruijn indices* or the *de Bruijn levels*, which both require the variables of a context to be consecutive natural numbers starting from 0. The drawback of this approach is that one is that one cannot for instance consider the context $(0 : \star, 1 : \star, 2 : \star)$ and remove the middle variable to obtain the context $(0 : \star, 2 : \star)$, as the latter is ill-formed, and forces us to carry the identifications of variables with substitution, instead of having simply the same name. In our presentation, we extensively rely on the names of variables for the sake of readability. However, we assume that there is an equivalent formulation of the theory using de Bruijn levels, and implicitly rely on it to keep a correspondence between the entities in the syntactic category and the syntax.

1.2 Semantics of type theory

We study additional structure that the syntactic category of a type theory is endowed with, and use this to give a framework for defining the notion of model of a type theory. We also give a connection with other notions of theory formulated in categorical logic, hence justifying partially the fact that a type theory indeed corresponds to a notion of theory.

1.2.1 Categories with families

We now give a categorical description of the type theories that we use to interpret a type theory in a category (especially in the category of sets). This gives the *semantics* of the theory. Interpreting the theory can be thought of as giving an incarnation of its axioms in the category, and thus corresponds to the notion of *model* of a theory. We work specifically in the formalism of *categories with families*, introduced by Dybjer[32], but other categorical models have been considered for type theory, such as categories with attributes [23], comprehension categories [42], display maps categories [62], and natural models [6], we refer the reader to [41] for a survey on these notions. We use categories with families to describe our specific case of type theory, but it is a much more general formalism and can encompass a lot of other features one may require a type theory to have, like Σ-types or Π-types, and they have been studied extensively by Clairembault and Dybjer [28, 27, 24].

Categories with families. We write **Fam** for the category of families, where an object is a family $(A_i)_{i\in I}$ of sets A_i indexed in a set I and a morphism $f : (A_i)_{i\in I} \to (B_j)_{j\in J}$ is a pair consisting of a function $f : I \to J$ and a family of functions $(f_i : A_i \to B_{f(i)})_{i\in I}$.

Suppose given a category $\mathcal{C}$ equipped with a functor $T : \mathcal{C}^{\mathrm{op}} \to \mathbf{Fam}$. Given an object Γ of $\mathcal{C}$, its image will be denoted

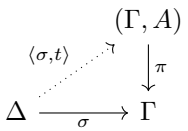
$$T\Gamma = \left(\mathrm{Tm}_A^\Gamma\right)_{A\in \mathrm{Ty}^\Gamma}$$

i.e., we write Ty^Γ for the index set and Tm^Γ_A for the elements of the family. By analogy with a type theory, for a morphism $\gamma : \Delta \to \Gamma$ an element $A \in \mathrm{Ty}^\Gamma$ and an element $t \in \mathrm{Tm}^\Gamma_A$, we write $A[\gamma] = T\gamma(A)$ the image of A in Ty^Δ, and $t[\gamma] = T_A\gamma(t)$ the image of t in $\mathrm{Tm}^\Delta_{A[\gamma]}$. With those notations, the functoriality of T can be written as

$$A[\sigma \circ \delta] = A[\sigma][\delta] \qquad t[\sigma \circ \delta] = t[\sigma][\delta]$$
$$A[\mathrm{id}] = A \qquad t[\mathrm{id}] = t$$

for composable morphisms of $\mathcal{C}$.

A *category with families* (or *CwF*) consists of a category $\mathcal{C}$ equipped with a functor as above $T : \mathcal{C}^{\mathrm{op}} \to \mathbf{Fam}$, such that $\mathcal{C}$ has a terminal object, denoted $\varnothing$, and that there is a *context comprehension* operation: given a context Γ and type $A \in \mathrm{Ty}^\Gamma$, there is a context (Γ, A), together with a projection morphism $\pi : (\Gamma, A) \to \Gamma$ and a term $p \in \mathrm{Tm}^{(\Gamma,A)}_{A[\pi]}$, such that for every morphism $\sigma : \Delta \to \Gamma$ in $\mathcal{C}$ together with a term $t \in \mathrm{Tm}^\Delta_{A[\sigma]}$, there exists a unique morphism $\langle \sigma, t \rangle : \Delta \to (\Gamma, A)$ such that $p[\langle \sigma, t \rangle] = t$:



In a category with families, the class of *display maps* is the smallest class of morphisms containing the projection morphisms $\pi : (\Gamma, A) \to \Gamma$ and closed under composition and identities.

Proposition 7. *The syntactic category of a type theory is endowed with a structure of category with families, where for every context* Γ, Ty^Γ *is the set of derivable types in* Γ *and* Tm^Γ_A *is the set of derivable terms of type* A *in* Γ

Proof. For a cut-full type theory, this is a direct translation of the structural rules of the theory, together with the equation it satisfies. Since we have asserted that a cut-free type theory completely implements a cut-full type theory, and satisfies the same equations, it gives the same structure to the syntactic category of a cut-free type theory. □

Given a category with families $\mathcal{C}$, we define its *presheaf of types* $\mathrm{Ty} : \mathcal{C}^{\mathrm{op}} \to \mathbf{Set}$ which associates to each object Γ the set Ty^Γ, as well as its *presheaf of terms* $\mathrm{Tm} : \mathcal{C}^{\mathrm{op}} \to \mathbf{Set}$ which associates to each object Γ the set $\bigsqcup_{A \in \mathrm{Ty}^\Gamma} \mathrm{Tm}^\Gamma_A$. There is a natural transformation $\mathrm{Tm} \Rightarrow \mathrm{Ty}$ associating for each term in Tm^Γ_A the type $A \in \mathrm{Ty}^\Gamma$. The axioms of category with families can be expressed in terms of representability of this natural transformation (c.f. [6]).

Models. A *morphism* between two categories with families $(\mathcal{C}, T)$ and $(\mathcal{C}', T')$, is a functor $F : \mathcal{C} \to \mathcal{C}'$ together with a natural transformation $\phi : T \to T' \circ F$, such that F preserves the terminal object and the context comprehension operation. A 2-*morphism* θ between two morphisms $(F, \phi) : T \to T'$ and $(F', \phi') : T \to T'$ is a natural transformation $\theta : F_1 \to F_2$ such that $T\theta \circ \phi = \phi'$.

We define a large category with families in a similar way, as a large category equipped with a functor into families of large sets indexed by a large set, and satisfying the exact same properties. Note that a category with families can be seen as a large category with families. There is a structure of category with large families on the category **Set**, where, given a set X, Ty^X is the (large) set of all function $f : Y \to X$ with codomain X and given such a function $f : Y \to X$, Tm^X_f is the (large) set of all sections of f. We define the (large) category of *models* of a category

with families $\mathcal{C}$ to be the category whose objects are the morphisms of categories with families from $\mathcal{C}$ to **Set**, and whose morphisms are the 2-morphisms of categories with families. We denote $\mathbf{Mod}(\mathcal{C})$ the category of models associated to a category with families $\mathcal{C}$.

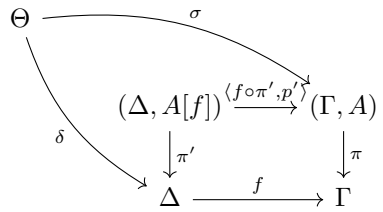
Pullbacks along display maps. The structure of category with families encompasses a compatibility condition between context comprehension and the action of morphisms on the type, expressed by the following lemma. In particular, it states that all pullbacks along display maps exist and that they can be explicitly computed from the given structure.

Lemma 8. *In a category with families $\mathcal{C}$, for every morphism $f : \Delta \to \Gamma$ in $\mathcal{C}$ and $A \in \mathrm{Ty}^\Gamma$, the square*

$$\begin{array}{ccc} (\Delta, A[f]) & \xrightarrow{\langle f\circ\pi', p'\rangle} & (\Gamma, A) \\ {\scriptstyle \pi'}\downarrow & & \downarrow{\scriptstyle \pi} \\ \Delta & \xrightarrow[f]{} & \Gamma \end{array}$$

is a pullback, where $\pi' : (\Delta, A[f]) \to \Delta$ and $p' \in \mathrm{Tm}^{(\Delta, A[f])}_{A[f][\pi']}$ are obtained by context comprehension.

Proof.



Consider the term $p \in \mathrm{Tm}^{(\Gamma, A)}_{A[\pi]}$, then $p[\sigma] \in \mathrm{Tm}^{\Theta}_{A[\pi][\sigma]} = \mathrm{Tm}^{\Theta}_{A[f][\delta]}$. By context extension, we get a map $\langle \delta, p[\sigma]\rangle : \Theta \to (\Delta, A[f])$ such that $\pi' \circ \langle \delta, p[\sigma]\rangle = \delta$ and $p'[\langle \delta, p[\sigma]\rangle] = p[\sigma]$. Since moreover $p' = p[\langle f \circ \pi', p'\rangle]$, the term equality gives in fact $p[\sigma] = p[\langle f \circ \pi', p'\rangle \circ \langle \delta, p[\sigma]\rangle]$, which is a necessary condition for the upper triangle to commute, thus proving uniqueness of the map. We just have to show that this map makes the upper triangle commute. Notice that $\pi \circ \langle f \circ \pi', p'\rangle \circ \langle \delta, p[\sigma]\rangle = \pi \circ \sigma$, and $p[\sigma] = p[\langle f \circ \pi', p'\rangle \circ \langle \delta, p[\sigma]\rangle]$, by universal property of the extension for morphisms, this implies the commutativity of upper triangle. □

Lemma 9. *Let $\mathcal{C}$ and $\mathcal{D}$ be two categories with families, together with a morphism $(F, \phi) : \mathcal{C} \to \mathcal{D}$, then for any object Γ in $\mathcal{C}$ together with an element $A \in \mathrm{Ty}^\Gamma$ and for any morphism $\gamma : \Delta \to \Gamma$ in $\mathcal{C}$, the following equation is satisfied*

$$F(\Delta, A[\gamma]) = (F\Delta, (\phi_\Gamma A)[F\gamma])$$

Proof. By definition of a morphism of category with families, we have

$$F(\Delta, A[\gamma]) = (F(\Delta), (\phi_\Delta(A[\gamma])))$$

Moreover by naturality of ϕ, the following square commutes

$$\begin{array}{ccc} Ty^\Gamma & \xrightarrow{\phi_\Gamma} & Ty^{F(\Gamma)} \\ {\scriptstyle \_[f]}\downarrow & & \downarrow{\scriptstyle \_[F\gamma]} \\ Ty^\Delta & \xrightarrow{\phi_\Delta} & Ty^{F(\Delta)} \end{array}$$

thus $\phi_\Delta(A[\gamma]) = (\phi_\Gamma A)[F\gamma]$. □

Note that Lemma 8 allows to understand this result as the fact that F preserves the pullbacks along the display maps. In fact one can understand the formalism of category with families as a way to define a category with a choice of pullbacks along a certain class of maps, while also ensuring that this choice of pullback is split: the composition of two pullbacks is not only isomorphic to the pullback of the composition, but is equal on the nose. Syntactically, this translates into the equality $(\Delta, A[\gamma \circ \delta]) = (\Delta, A[\gamma][\delta])$.

A characterization of the models. The pullbacks along the display give a nice characterization of the models of a category with families

Lemma 10. *The category of models of a category with families $\mathcal{C}$ is isomorphic to category of functors $\mathcal{C} \to$ **Set** that preserve the terminal object and the morphisms along the display maps.*

Proof. Lemma 9, the underlying functor of a morphism of category with families preserves the pullbacks along the display maps, and by definition, such functor has to preserve the initial object as well. So it suffices to prove that a functor $F : \mathcal{C} \to \mathbf{Set}$ preserving the initial object and the pullbacks along display maps gives rise to a unique model. Consider such a functor F, together with an object Γ in $\mathcal{C}$ and a type $A \in \mathrm{Ty}^\Gamma$. Suppose defined ϕ such that (F, ϕ) is a model of $\mathcal{C}$, then necessarily $F(\Gamma, A) = (F\Gamma, \phi_\Gamma A) = \phi_\Gamma A$ by definition of the context comprehension in **Set**. Thus necessarily $\phi_\Gamma(A) = F(\Gamma, A)$. Consider a term $t \in \mathrm{Tm}_A^\Gamma$, then there is a morphism $\langle \mathrm{id}_\Gamma, t\rangle : \Gamma \to (\Gamma, A)$, and by definition of the category with families structure of **Set**, we then have $F(\langle \mathrm{id}_\Gamma, t\rangle) = \langle \mathrm{id}_{F\Gamma}, \phi_{\Gamma,A}(t)\rangle = t$, which proves that necessarily $\phi_{\Gamma,A}(t) = F(\langle \mathrm{id}_\Gamma, t\rangle)$. Conversely, these assignments define a natural transformation ϕ, which make (F, ϕ) into a model of F. □

This condition relies on the specific structure of category with families of **Set**: It may not be true in general that the morphisms of categories with families between two arbitrary categories with families $\mathcal{C}$ and $\mathcal{D}$ are isomorphic to the functors preserving the display maps and the pullbacks along them. It also justifies retrospectively not to be too precise about the size issues with **Set**, as one may as well ignore the structure of category with families on **Set** altogether, and define a model as a functor $\mathcal{C} \to$ **Set** that preserves the terminal object and the pullback along the display maps.

Contextual categories. In order to carry some inductive constructions that we can perform on the syntax on a theory, and treat them in full generality, we introduce the notion of *contextual category*. These are precisely the categories with families with extra structure making those inductive construction possible, they were introduced by Cartmell [23] and studied later on by Streicher [61] and Voevodsky [65] under the name of *C-system*.

Definition 11. A contextual category is a category with families $\mathcal{C}$ together with a map ℓ associating to each object Γ of $\mathcal{C}$ a natural number $\ell(\Gamma)$ called its *length*, such that

- the terminal object $\varnothing$ is the unique object such that $\ell(\varnothing) = 0$,
- for every object Γ and type $A \in \mathrm{Ty}^\Gamma$, $\ell(\Gamma, A) = \ell(\Gamma) + 1$,
- for every object Γ such that $\ell(\Gamma) > 0$, there is a unique object Γ' together with a type $A \in \mathrm{Ty}^{\Gamma'}$ such that $\Gamma = (\Gamma', A)$.

Note that a contextual category is usually defined to be a category with attributes satisfying such properties. However, since categories with families and categories with attributes are equivalent, we will also refer to these as contextual categories. Also note that, the notion of contextual category is not invariant by equivalence of categories. Their use is justified by the following proposition

Proposition 12. *The syntactic category of a type theory is naturally endowed with a structure of contextual category.*

Proof. In both the cut-full and the cut-free styles, the objects of the syntactic category are contexts, that are formally lists. Hence they are naturally equipped with a notion of length, which by definition satisfies all the expected properties. □

Given a contextual category $\mathcal{C}$, an object Γ whose length is strictly positive is obtained in a unique way as Γ', A, and we simply write $\pi_\Gamma : \Gamma \to (\Gamma', A)$ (or even π) instead of $\pi_{\Gamma',A}$. We also write x_Γ for the term $p_{\Gamma',A}$ in $\mathrm{Tm}^{\Gamma}_{A[\pi]}$, thought of as a variable. More generally, we declare that a term is a *variable* when it is of the form $x_\Gamma[\pi]$ where π is a display map. Note that in a contextual category, if $\pi : \Delta \to \Gamma$ is a display map, then necessarily $l(\Delta) > l(\Gamma)$. This implies that the variables of a non-empty context (Γ, A) are either $x_{(\Gamma,A)}$, or of the form $x[\pi_{(\Gamma,A)}]$ where x is a variable of Γ.

The following lemma shows that a map in a contextual category is entirely characterized by its action on variables in its target context, it is a categorical analogue of Lemma 6 that we have stated on the syntax.

Lemma 13. *Consider two maps $\gamma, \delta : \Delta \to \Gamma$, in a contextual category, such that for every variable x in Γ, $x[\gamma] = x[\delta]$. Then $\gamma = \delta$.*

Proof. We will prove this result by induction on the length of the context Γ :

- If Γ is of length 0, then necessarily, $\Gamma = \varnothing$ is the terminal object, and thus $\gamma = \delta$.
- If Γ is of length $l + 1$, then it is of the form (Γ', A), and there is a substitution $\pi : \Gamma \to \Gamma'$. Suppose that there are two substitutions $\gamma, \delta : \Delta \to \Gamma$, such that for all variables x of Γ, we have $x[\gamma] = x[\delta]$. Note that we necessarily have $\gamma = \langle \pi \circ \gamma, x_\Gamma[\gamma] \rangle$ and $\delta = \langle \pi \circ \delta, x_\Gamma[\delta] \rangle$, as it is the case for every substitutions. Then for the variable x_Γ, we have $x_\Gamma[\gamma] = x_\Gamma[\delta]$. Moreover, for every variable x of Γ', $x[\pi]$ is a variable of Γ, and thus $x[\pi][\gamma] = x[\pi][\delta]$, which proves $x[\pi \circ \gamma] = x[\pi \circ \delta]$, and y induction hypothesis, $\pi \circ \gamma = \pi \circ \delta$. We thus have proved that $\langle \pi \circ \gamma, x_\Gamma[\gamma] \rangle = \langle \pi \circ \delta, x_\Gamma[\delta] \rangle$, i.e., $\gamma = \delta$.

□

Looking back at the cut-full versus cut-free. This discussion gives more insight about the different presentations of the cut-full and the cut-free type theories. Both can be seen as describing a universal category with families: Roughly speaking the syntactic category of a type theory is the initial category with families that satisfy all the rules for types and terms of the theory (i.e., the non-structural rules). This a bold claim, as in most cases, it is just a conjecture (known as the *initiality conjecture*), whose quest for a proof is still a very active project [66, 5]. However, in the cases we are interested in, this claim is fairly reasonable to check, and for the sake of this discussion, we assume that it is the case, if not strictly at least up to some technical details that do not hamper the main idea. The two styles of type theory can be understood as two presentations of this universal category with family (and moreover, both these presentations enjoy a notion of length for object and hence give a structure of contextual category). The

cut-full type theory is then akin to a *standard presentation*, where all the structure is in the generators, and all the relations between them are required as relations, whereas the cut-free type theory is comparable to a *reduced presentation*.

1.2.2 Most general unifiers

Let $\mathcal{C}$ be a category with families and Γ an object of $\mathcal{C}$. Given a type $A \in \mathrm{Ty}^\Gamma$ and two terms $t, u \in \mathrm{Tm}^\Gamma_A$ of the same type A, we call a *most general unifier* of t and u an object $\Gamma/_{t=u}$ together with a map $\varepsilon : \Gamma/_{t=u} \to \Gamma$ such that $t[\varepsilon] = u[\varepsilon]$ and every map $\sigma : \Delta \to \Gamma$ in $\mathcal{C}$ such that $t[\sigma] = u[\sigma]$ factors uniquely as $\sigma = \varepsilon\tilde{\sigma}$:

$$\begin{array}{ccc} \Delta & \xrightarrow{\sigma} & \Gamma \\ & \overset{\tilde{\sigma}}{\searrow} & \uparrow{\scriptstyle \varepsilon} \\ & & \Gamma/_{t=u} \end{array}$$

Given two morphisms $\Delta \overset{f}{\underset{g}{\rightrightarrows}} \Gamma$ we also denote $\Delta/_{f=g}$ the equalizer of f and g in Δ, to emphasize the similitude between the two universal properties.

Theorem 14. *A contextual category $\mathcal{C}$ which has most general unifiers for every pair of terms of the same type has all finite limits.*

Proof. By definition of a category with families, $\mathcal{C}$ has a terminal element, so it suffices to prove that it has all pullbacks. Consider a cospan in $\mathcal{C}$

$$\begin{array}{ccc} & & \Theta \\ & & \downarrow{\scriptstyle \theta} \\ \Delta & \xrightarrow[\delta]{} & \Gamma \end{array}$$

We prove by induction on the length of Γ that this cospan has a pullback.

- If Γ is of length 0, then it is the terminal object, and by uniqueness, $\theta = \pi^{\ell(\Theta)}$. Since the successive pullbacks exist along all the display maps π, this cospan admits a pullback.
- If Γ is of length $l+1$, then it is of the form (Γ', A) with Γ' of length l, and $A \in Ty^{\Gamma'}$. Then by induction, the following cospan has a pullback, denoted Σ

$$\begin{array}{ccc} \Sigma & \overset{\sigma_1}{\dashrightarrow} & \Theta \\ {\scriptstyle \sigma_2}\downarrow & \lrcorner & \downarrow{\scriptstyle \pi_{\Gamma',A}\theta} \\ \Delta & \xrightarrow[\pi_{\Gamma',A}\delta]{} & \Gamma' \end{array} \tag{1.1}$$

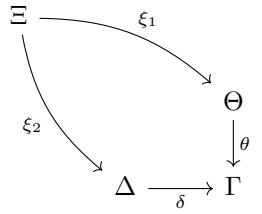
  From this pullback, we will construct the pullback of the original cospan, the following way. Take the universal term $p = p_{\Gamma',A} \in \mathrm{Tm}^{\Gamma'}_{A[\pi_{\Gamma',A}]}$, it gives rise to two terms in Σ, $t = p[\theta\sigma_1]$ and $u = p[\delta\sigma_2]$. These two terms have respective types $A[\theta\sigma_1\pi_{\Gamma',A}]$ and $A[\delta\sigma_2\pi_{\Gamma',A}]$, thus by commutation of the diagram (1.1), these two terms have the same type. Thus we can construct their most general unifier $e : \Sigma/_{t=u} \to \Sigma$, and we will now check that the following diagram is a pullback

$$\begin{array}{ccc} \Sigma/_{t=u} & \xrightarrow{\sigma_1 e} & \Theta \\ {\scriptstyle \sigma_2 e}\downarrow & & \downarrow{\scriptstyle \theta} \\ \Delta & \xrightarrow[\delta]{} & \Gamma \end{array}$$

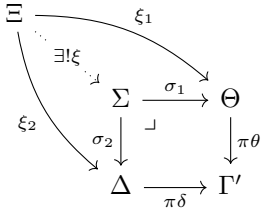
First we will show that this square commutes. First note that by commutation of the diagram (1.1) we have the equality

$$\pi(\theta\sigma_1 e) \quad = \quad \pi(\delta\sigma_2 e)$$

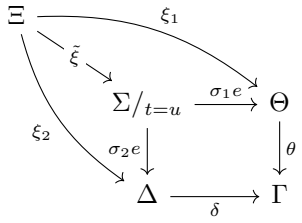
Moreover, by definition of the most general unifier, we have that $t[e] = u[e]$, which rewrites as $v_\Gamma[\theta\sigma_1 e] = v_\Gamma[\delta\sigma_2 e]$, so by Lemma 13, it follows that the above square commutes. Now take another commutative square of the form



Then composing with π, and using the property of the pullback we get a unique map



Moreover, by commutation of this diagram, $t[\xi] = v_\Gamma[\theta\sigma_1\xi] = v_\Gamma[\theta\xi_1]$ and $u[\xi] = v_\Gamma[\delta\xi_2]$. By commutation of the initial diagram, this proves that $t[\xi] = u[\xi]$. By definition of the most general unifier, this implies that ξ factors as $\xi = e\tilde{\xi}$. This gives a unique factorization as



□

Most general unifiers and equalizers. Consider a category with families $\mathcal{C}$, two morphisms in the following form

$$\Delta \underset{\langle\gamma',t'\rangle}{\overset{\langle\gamma,t\rangle}{\rightrightarrows}} (\Gamma, A)$$

Proposition 15. *Suppose that the equalizer $e : \Delta/_{\gamma=\gamma'} \to \Delta$ exists. Then the equalizer $\Delta/_{\langle\gamma,t\rangle=\langle\gamma',t'\rangle}$ exists if and only if the most general unifier of $t[e]$ and $t'[e]$ exists in $\Delta/_{\gamma=\gamma'}$, and if so, they are isomorphic.*

Proof. Suppose that the equalizer $e_+ : \Delta/_{\langle\gamma,t\rangle=\langle\gamma',t'\rangle} \to \Delta$ exists, then by definition, we always have that $\langle\gamma, t\rangle e_+ = \langle\gamma', t'\rangle e_+$, so composing both sides by the projection $\pi : (\Gamma, A) \to \Gamma$ yields

the equality $\gamma e_+ = \gamma' e_+$. This proves by universal property of e that e_+ factors in a unique way as $e_+ = eu$, with $u : \Delta/_{\langle\gamma,t\rangle=\langle\gamma',t'\rangle} \to \Delta/_{\gamma=\gamma'}$. We show that u satisfies the universal property of the most general unifier of $t[e]$ and $t'[e]$. Consider a map $f : X \to \Delta/_{\gamma=\gamma'}$ such that $t[e][f] = t'[e][f]$, then we have by definition of e, the equality $\gamma ef = \gamma' ef$, and by definition of f, $t[ef] = t'[ef]$, hence by universality of the operation $\langle \_, \_ \rangle$, this shows that $\langle \gamma, t\rangle ef = \langle \gamma', t'\rangle ef$, so by definition of e_+, there exists a unique map $\tilde{f}$ such that $ef = e_+\tilde{f}$. Using our decomposition of e_+, we have $ef = eu\tilde{f}$, and since e is an equalizer, this shows $f = u\tilde{f}$. Thus we have proved that u satisfies the universal property of the most general unifier. Conversely, consider the most general unifier of $t[e]$ and $t'[e]$ in $\Delta/_{\gamma=\gamma'}$, in order to simplify the notations we denote it $u : U \to \Delta/_{\gamma=\gamma'}$. We show that the map eu is the equalizer of $\langle \gamma, t\rangle$ and $\langle \gamma', t'\rangle$. Consider a map $f : X \to \Delta$ such that $\langle \gamma, t\rangle f = \langle \gamma', t'\rangle f$, then composing with the projection on the two sides yields $\gamma f = \gamma' f$, and hence, since e is the equalizer f factors in a unique way in $f = e\bar{f}$. Moreover, we have $t[f] = t'[f]$, which now translates to $t[e][\bar{f}] = t'[e][\bar{f}]$, and since u is the most general unifier of $t[e]$ and $t'[e]$ there is a unique factorization of $\bar{f}$ into $u\tilde{f}$, which gives a unique factorization of f in $f = eu\tilde{f}$. Hence U is the equalizer of $\langle \gamma, t\rangle$ and $\langle \gamma', t'\rangle$. □

Under the assumption that enough equalizers exist, we can iterate this result and express any equalizer as a sequence of most general unifiers. A sufficient condition for this would be that whenever the equalizer of $\langle \gamma, t\rangle$ and $\langle \gamma', t'\rangle$ exists, so does the equalizer of γ and γ', and we conjecture that this is the case in all the type theories we consider. Under this assumption, the most general unifiers can be seen as the generators of the equalizers of the category.

Primitive most general unifiers. We now define a class of most general unifiers that we call *primitive*: these are the ones that have to exist because of the structure of category with family. Let $\mathcal{C}$ be a category with families (it needs not be a contextual category), and consider an object of the form (Γ, A), and a term $u \in \mathrm{Tm}^{(\Gamma,A)}_{A[\pi_{\Gamma,A}]}$ the most general unifier $(\Gamma, A)/_{p_{\Gamma,A}=u}$ always exists and is given by

$$\langle \mathrm{id}_\Gamma, u\rangle : \Gamma \to (\Gamma, A)$$

Indeed, consider a map $f : \Delta \to (\Gamma, A)$ such that $u[f] = p_{\Gamma,A}[f]$, then f factors uniquely through $\langle \mathrm{id}_\Gamma, u\rangle$ as

$$\begin{array}{ccc} \Gamma & \xrightarrow{\langle \mathrm{id}_\Gamma, u\rangle} & (\Gamma, A) \\ {\scriptstyle \pi_{\Gamma,A} f}\,\vdots\,\wedge & \nearrow_{f} & \\ \Delta & & \end{array}$$

Suppose that Γ is an object and t and u are two terms with the same type A, such that the most general unifier $\Gamma/_{t=u} \to \Gamma$ exists. Consider Δ to be a sequence of context comprehensions, such that Γ, Δ is a well defined object, and π_Δ be the corresponding display map obtained by composing all the projection maps. Then the most general unifier of $t[\pi_\Delta]$ and $u[\Delta]$ in the context Γ exists, and it is computed as the following pullback (which exists since π_Δ is a display map)

$$\begin{array}{ccc} \Gamma, \Delta/_{t[\pi_\Delta]=u[\pi_\Delta]} & \overset{e}{\dashrightarrow} & \Gamma, \Delta \\ \vdots\,\wedge & \lrcorner & \downarrow{\scriptstyle \pi_\Delta} \\ \Gamma/_{t=u} & \xrightarrow[e]{} & \Gamma \end{array}$$

Consider an object Γ and a term t in Γ along with two maps $f, g : \Delta \to \Gamma$. Then the most general unifier of $t[f]$ and $t[g]$ exists if and only the most general unifier of f and g exists, and when this is the case, they are equal.

Define a *primitive* most general unifier to be a most general unifier of one of the following form

- $(\Gamma, A)/_{p_{\Gamma,A}=u}$ where u is a term of type $A[\pi_{\Gamma,A}]$ in (Γ, A)
- $\Gamma, \Delta/_{t[\pi_\Delta]=u[\pi_\Delta]}$ where $\Gamma/_{t=u}$ is a primitive most general unifier
- $\Gamma/_{t[f]=t[g]}$ where $\Gamma/_{f=g}$ is a primitive most general unifier

And for substitutions a primitive most general unifier is one of either of the following form

- $\Gamma/_{\langle\rangle=\langle\rangle}$
- $\Gamma/_{\langle f,t\rangle=\langle g,u\rangle}$ where both $\Delta = \Gamma/_{f=g}$ and $\Delta/_{t[f]=u[g]}$ are primitive most general unifiers

Lemma 16. *Any category with families has all primitive most general unifiers.*

Proof. By construction, we have defined the primitive most general unifiers from the pullbacks along the display maps, which exist in any category with families. □

Models preserve primitive most general unifiers.

Theorem 17. *If F is a model of a contextual category $\mathcal{C}$, then F preserves the primitive most general unifiers.*

Proof. Since all the primitive most general unifiers are constructed only with context comprehension and morphism extension constructions, and that a model preserves these on the nose, it has to preserve all the primitive most general unifiers. □

Corollary 18. *If F is a model of a contextual category $\mathcal{C}$ such that all most general unifiers are primitive, and such that whenever the most general unifier of $\langle\gamma, t\rangle$ and of $\langle\gamma', t'\rangle$ exists, so does the most general unifier of γ and γ', then F preserves all the finite limits that exist in $\mathcal{C}$.*

Proof. Since F is a model, it preserves all the pullbacks along display maps, and in particular it preserves the products (since for any object Γ, the map $\Gamma \to \varnothing$ is a display map). Moreover, any equalizer e can be written as a series of most general unifiers, that are all preserved by F, and hence can be iterated in **Set** and define $F(e)$ as the image equalizer. Hence F preserves the products and the equalizers, so it preserves all the finite limits in $\mathcal{C}$. □

In particular, in a type theory that does not postulate definitional equality, all the most general unifiers are primitive. This will be the case for our theories where the definitional equality of terms will correspond to their syntactic equality.

1.2.3 Categorical notions of theories

We now introduce tools from categorical logic that formalize the notion of a theory, and of models of a theory by the means of categories. This gives comparisons with the tools we have introduced to study type theory, and allows us to understand a type theory as a language to manipulate axioms of a theory.

Lawvere theories. Category theory provides a framework in which it is possible to formulate some axiomatic theories in a very elegant way, with the notion of product only. Indeed, consider a category $\mathcal{C}$ whose objects are the natural numbers $0, 1, \ldots$, and such that $n + m$ is always the categorical product of n and m (with 0 the terminal object), together with a functor F that preserves the finite products. Then F defines a set $X = F(1)$, and for all other n, we have $F(n) = X^n$, the morphisms of $\mathcal{C}$ then give operations, and in particular a morphism $\mu : 2 \to 1$ in $\mathcal{C}$ provides an binary operation $m = F(\mu) : X^2 \to X$. Finally the equalities between morphisms in $\mathcal{C}$ provide equalities between operations, in particular, from a commutative square

$$\begin{array}{ccc} 3 & \xrightarrow{\mu\times 1} & 2 \\ {\scriptstyle 1\times\mu}\downarrow & & \downarrow{\scriptstyle \mu} \\ 2 & \xrightarrow[\mu]{} & 1 \end{array}$$

we get an equation written as $\forall a, b, c \in X, \mu(a, \mu(b, c)) = \mu(\mu(a, b), c)$. Other equalities could give other equations, but the functor F always corresponds to a set equipped with a specific structure. Conversely, given an axiomatic theory, and under appropriate restriction, there exists a category $\mathcal{C}$ as above such that the functors $\mathcal{C} \to \mathbf{Set}$ preserving the finite products are exactly the sets equipped with these axioms. This lets us interpret such a category $\mathcal{C}$ as being an axiomatic theory, and with this interpretation, the functors preserving the finite products are exactly the models of the theory. This construction has been introduced by Lawvere [45] and is now known as a *Lawvere theory*, the axiomatic theories that correspond to a Lawvere theory are called *algebraic theories*.

Gabriel-Ülmer duality. A more powerful notion of theory expressed in a categorical language is given by the small finitely complete categories. These are the categories that have not only all the finite products, but also all the finite limits. Such a category $\mathcal{C}$ defines an *essentially algebraic theory*, i.e., a notion of theory similar to the algebraic theories, but allowing partially defined operations, and its models are the functors $\mathcal{C} \to \mathbf{Set}$ that preserve the finite limits. Gabriel and Ülmer [35] have showed that these categories assemble into a bicategory, and the functor which associates its models to each essentially algebraic theory is a bi-equivalence. This characterizes exactly the structures that can be expressed by an essentially algebraic theory, which are called the *locally finitely presentable categories*.

Generalized algebraic theories. As motivated by the two previous examples, the notion of a theory can be presented categorically by the existence and preservation of finite limits. The case of Lawvere theory restricts only to finite products, whereas Gabriel-Ülmer duality requires all finite limits. Contextual categories can be seen as a sort of intermediate between these two, where the limits required are the pullbacks along display maps. In contextual categories all the unique maps to the terminal object $! : \Gamma \to \varnothing$ are display maps, thus a contextual category has all products, but it does not necessarily have all limits. Hence the intuition is that contextual categories lie in between Lawvere theories and finitely complete categories. This justifies the use of the terminology *generalized algebraic theory* introduced by Cartmell [23] to denote a theory that is categorically expressed as a contextual category. This comparison can even be taken a step further: multi-sorted Lawvere theories need to have objects freely generated by a finite set. One can understand the types and the length of objects in a contextual category as a convenient way to express a condition of the set of objects being freely generated, and the terms as a convenient way to express the existence of additional morphisms between two objects, that don't come from the limit structure. We illustrate how contextual categories can be understood as generalizations of Lawvere theories with the following results

Proposition 19. *Contextual categories which only have a single type associated to each object are equivalent to Lawvere theories.*

Proof. Consider a contextual category $\mathcal{C}$ which has only one type associated to every context, to simplify the notations, we denote $\star$ the unique type in any object, and note that any two objects Γ and Δ of the same length are necessarily equal. Indeed, the only object of length n in the category is given by $(\star, \dots, \star)$. We thus simply denote n this object. Using this notation, we see that 0 is the terminal object, and the map $n \to 0$ is a display map by definition of a contextual category. Thus we can consider the pullback of $1 \to 0$ along this map, which computes the categorical product $n \times 1$ as the object $(n + 1)$. By iterating this construction, we prove that the categorical product $n \times m$ is given by the object $(n + m)$, hence $\mathcal{C}$ is a Lawvere theory. Conversely, given a Lawvere theory $\mathcal{C}$, we ca define the length of the object n to be n for every object, and define the set of type associated to each object to be a singleton. We then define the set of terms of the unique type associated to the object n to be $\mathcal{C}(n, 1)$, so that a morphism $f : m \to n$ naturally acts on a term $t : n \to 1$ as $t \circ f : m \to 1$. We can check that this defines a contextual category. □

Morita equivalence. Thinking of contextual categories as a generalization of Lawvere theories, allows us to interpret our notion of type theory: it provides a syntax to work internally to a contextual category. Note that it would be wrong to conceive a type theory as a generalized algebraic theory, because different syntaxes may yield the same contextual category (this is commonly referred to as a Morita equivalence). Rather we think of a type theory as a particular *presentation* of a generalized algebraic theory, the same way a set of axioms gives a presentation of a theory but different sets of axioms may define the same theory.

Chapter 2

A type theory for globular ω-categories

2.1 The Grothendieck-Maltsiniotis definition of ω-categories

This entire section is a quick presentation of the definition of weak ω-categories given by Maltsiniotis [54], relying on the ideas for defining weak ω-groupoids introduced by Grothendieck [39]. The aim is to introduce the notions that the type theory CaTT relies on, as well as the notations we will use for these notions. For a more in-depth study of this definition, one can refer to the original article by Maltsiniotis [54] or by a full account of this definition by Ara [4]

2.1.1 The category of globes and globular sets.

Similar to how the regular categories are supported by graphs, ω-categories are supported by a structure called *globular sets*. These are generalization of graphs that do not only include vertices and arrows, but also arrows between the arrows (2-*cells*), arrows between the 2-cells (3-*cells*) and so on. A convenient way to define the category of globular set is as a presheaf category.

The category of globes. We denote $\mathcal{G}$ the category whose objects are the natural numbers $0, 1, \dots$ and whose morphisms are generated by the graph

$$0 \underset{\tau_0}{\overset{\sigma_0}{\rightrightarrows}} 1 \underset{\tau_1}{\overset{\sigma_1}{\rightrightarrows}} 2 \underset{\tau_2}{\overset{\sigma_2}{\rightrightarrows}} \cdots$$

subject to following *coglobular relations*:

$$\sigma_{i+1} \circ \sigma_i = \tau_{i+1} \circ \sigma_i \qquad \sigma_{i+1} \circ \tau_i = \tau_{i+1} \circ \tau_i$$

The category of *globular sets* $\mathbf{GSet} = \widehat{\mathcal{G}}$ is the presheaf category over the category $\mathcal{G}$. Given a globular set G, we write G_n instead of $G\,n$. Equivalently, a globular set is a family of sets $(G_n)_{n\in\mathbb{N}}$ equipped with maps $\partial_i^-, \partial_i^+ : G_{i+1} \to G_i$ satisfying the *globular relations*, dual to the coglobular relations

$$\partial_i^- \circ \partial_{i+1}^- = \partial_i^- \circ \partial_{i+1}^+ \qquad \partial_i^+ \circ \partial_{i+1}^- = \partial_i^+ \circ \partial_{i+1}^+$$

When it is non-ambiguous, we only write ∂^- and ∂^+, leaving the dimension implicit. Globular sets may also be characterized by coinduction: a globular set is a set X equipped with, for all

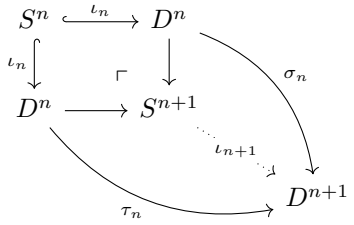
pairs $x, y \in X$, a globular set. Given an object n, the associated representable $\mathrm{Y}(n)$ is called the *n-disk* and is usually written D^n. It can be explicitly described by

$$(D^n)_i = \begin{cases} \{\bullet_-, \bullet_+\} \text{ if } i < n \\ \{\bullet\} \text{ if } i = n \\ \emptyset \text{ if } i > n \end{cases}$$

with $\partial^-(\_) = \bullet_-$ and $\partial^+(\_) = \bullet_+$. We also denote $\sigma_n : D^n \to D^{n+1}$ the image of the map σ_n by the Yoneda embedding. Explicitly, this map is the identity in all the dimensions $i < n$, and it is characterized in dimension n by $\sigma_n(\bullet) = \bullet_-$. Similarly, we denote $\tau_n : D^n \to D^{n+1}$ the image of τ by the Yoneda embedding, and it characterized by its action in dimension n given by $\tau_n(\bullet) = \bullet_+$. As is the case for any presheaf category, the Yoneda embedding $\mathcal{G} \to \mathbf{GSet}$ realizes **GSet** as the free cocompletion of $\mathcal{G}$.

The n-sphere. Given $n \in \mathbb{N}$, the *n-sphere* S^n is the globular set, equipped with an inclusion $\iota^n : S^n \hookrightarrow D^n$, defined by

- $S^{-1} = \emptyset$ is the initial object, and $\emptyset \hookrightarrow D^1$ is the unique arrow,
- S^{n+1} and ι^{n+1} are obtained by the pushout



Finite globular sets. A globular set G is *finite* if it can be obtained as a finite colimit of representable objects. It can be shown that this is the case precisely when the set $\coprod_{i \in \mathbb{N}} G_i$ is finite, because all representables themselves satisfy this property. We write **FinGSet** for the full subcategory of **GSet** whose objects are the finite presheaves. We sometimes call a finite globular set a *diagram*, and describe it using a diagrammatic notation. For instance, the diagram

$$x \underset{g}{\overset{f}{\Downarrow \alpha}} y \xrightarrow{h} z$$

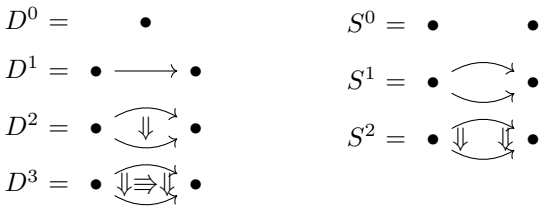
denotes the finite globular set G, whose only non-empty cell sets are

$$G_0 = \{x, y, z\} \qquad G_1 = \{f, g, h\} \qquad G_2 = \{\alpha\}$$

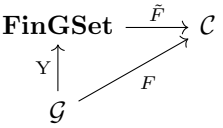
and whose the sources and targets are defined by

$$\begin{array}{ll} \partial^-(f) = x & \partial^+(f) = y \\ \partial^-(g) = x & \partial^+(g) = y \\ \partial^-(h) = y & \partial^+(h) = z \\ \partial^-(\alpha) = f & \partial^+(\alpha) = g \end{array}$$

Disks and spheres are finite globular sets. In small dimensions, they can be depicted as



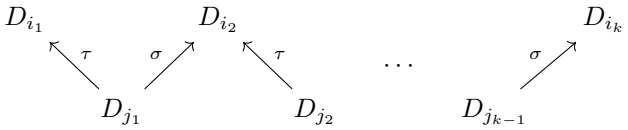
FinGSet is the free cocompletion of $\mathcal{G}$ by all finite colimits since all the representables have a finite number of elements. This means that **FinGSet** has all the finite limits, the Yoneda embedding corestricts to $\mathrm{Y} : \mathcal{G} \to \mathbf{FinGSet}$, and for all category $\mathcal{C}$ with all finite limits equipped with a functor $F : \mathcal{G} \to \mathcal{C}$, there is an essentially unique functor $\tilde{F} : \mathbf{FinGSet} \to \mathcal{C}$ preserving the finite limits such that $\tilde{F}\,\mathrm{Y} = F$.



2.1.2 Globular extensions

In the following, we consider a category $\mathcal{C}$ equipped with a functor $F : \mathcal{G} \to \mathcal{C}$, such a functor is called a *globular structure* on $\mathcal{C}$. In this case, we denote D^n the object $F(n)$ of $\mathcal{C}$, and we still denote σ_n and τ_n the images by F of the morphisms σ_n and τ_n. When there is no ambiguity, we may write σ and τ, leaving the index implicit, moreover, we write also σ (resp. τ) to indicate a composite of maps of the form σ (resp. τ). Dually, for a category $\mathcal{C}$, a functor $F : \mathcal{G}^{\mathrm{op}} \to \mathcal{C}$ is called a *coglobular structure*, and we denote respectively by D^n, ∂_n^+ and ∂_n^+ the images by F of $[n]$, σ_n and τ_n. We thus always denote σ and τ families of maps satisfying the coglobular relations and ∂^- and ∂^+ families of maps satisfying the globular relations.

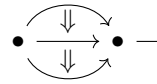
Globular sums. In the category $\mathcal{C}$ equipped with a globular structure, a *globular sum* is a colimit of a diagram of the form



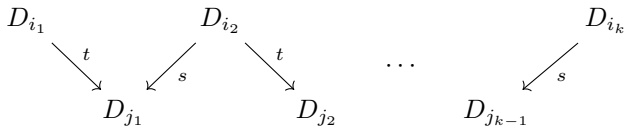
It will be useful to encode such a colimit by its *table of dimensions*

$$\begin{pmatrix} i_1 & & i_2 & & \cdots & & i_k \\ & j_1 & & j_2 & \cdots & j_{k-1} & \end{pmatrix}$$

We give a few examples in the category **FinGSet** of finite globular sets that can be described as a globular sum, using the diagrammatic notation, together with their associated dimension table

globular set	dimension table
	$\begin{pmatrix} 1 & & 1 \\ & 0 & \end{pmatrix}$
	$\begin{pmatrix} 2 & & 2 & & 1 \\ & 1 & & 0 & \end{pmatrix}$

Intuitively, the globular sets that can be written as a globular sum are the ones that are well ordered and that do not have a hole. We sometimes write $D^{i_1} \coprod_{D^{j_1}} D^{i_2} \coprod_{D^{j_2}} \cdots \coprod_{D^{j_{k-1}}} D^{i_k}$ the globular sum with a dimension table as above, leaving implicit that the coproduct is taken with the iterated source and iterated target maps. Similarly, in a category equipped with a coglobular structure, a *globular product* is a limit of the diagram of the form



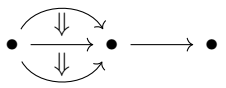
It will also be convenient to denote it by its table of dimensions, we use the same notation and distinguish between the two by the variance of the globular structure.

$$\begin{pmatrix} i_1 & & i_2 & & \cdots & & i_k \\ & j_1 & & j_2 & \cdots & j_{k-1} & \end{pmatrix}$$

If $\mathcal{C}$ has a globular structure and $\mathcal{D}$ has a contravariant globular structure, we will say that a globular sum in $\mathcal{C}$ and a globular product in $\mathcal{D}$ are *dual* to each other if they share the same table of dimensions.

Globular extensions. A category $\mathcal{C}$ with a globular structure F is called a *globular extension* when all the globular sums exist in $\mathcal{C}$. Given two globular extensions $F : \mathcal{G} \to \mathcal{C}$ and $G : \mathcal{G} \to \mathcal{D}$, a morphism of globular extensions is a functor $H : \mathcal{C} \to \mathcal{D}$ such that $H \circ F = G$, and preserving globular sums. Dually, a category with a contravariant globular structure that has all globular products is called a *coglobular extension*, and the opposite notion of morphisms defines morphisms of coglobular extensions.

The category Θ_0. There is a universal globular extension Θ_0, which is called a *globular completion*. It is the initial object in the category of globular extensions, or equivalently it is characterized by the fact that for any globular extension $\mathcal{G} \to \mathcal{C}$, there is a unique morphism of globular extensions $\Theta_0 \to \mathcal{C}$. Note that if Θ_0 is a globular completion, then Θ_0^{op} is a *globular cocompletion*, that is for every coglobular extension $\mathcal{G}^{\mathrm{op}} \to \mathcal{C}$, there is a unique morphism $\Theta_0^{\mathrm{op}} \to \mathcal{C}$. The objects of the category Θ_0 are called *pasting schemes*, and it can be characterized as the full subcategory of **GSet** whose objects are globular sums. Note that globular sums are always finite globular sets, and thus, using our diagrammatic notation, we give a few examples of pasting schemes, together with the dimension table of the corresponding globular sum.

pasting scheme	globular sum
	$\begin{pmatrix} 1 & & 1 \\ & 0 & \end{pmatrix}$
	$\begin{pmatrix} 2 & & 2 & & 1 \\ & 1 & & 0 & \end{pmatrix}$

Combinatorial description of Θ_0. The pasting schemes can be defined inductively. For this we need an operation Σ which given a globular set X produces the globular set ΣX defined by $(\Sigma X)_0 = \{\bullet_-, \bullet_+\}$ and $(\Sigma X)_{n+1} = X_n$, with for all $x \in X_0$, $\partial^-(x) = \bullet_-$ and $\partial^+(x) = \bullet_+$ in

ΣX. This operation is to be compared with the topological notion of suspension. Given two globular sets of the form ΣX and ΣY, we define the globular set $\Sigma X \bowtie \Sigma Y$ as the disjoint union of ΣX and ΣY, where we have identified the object $\bullet_+$ from ΣX with the object $\bullet_-$ from ΣY. Categorically, it is a pushout of the following form

$$\begin{array}{ccc} D^0 & \xrightarrow{\bullet_+} & \Sigma X \\ {\scriptstyle \bullet_-}\downarrow & \ulcorner & \downarrow \\ \Sigma Y & \longrightarrow & \Sigma X \bowtie \Sigma Y \end{array}$$

This lets us formulate the following equivalent characterization of the objects of Θ_0:

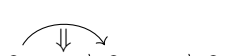
Lemma 20. *A pasting scheme is either the globular set D^0, or is obtained in a unique way as a sequence of the form $\Sigma X_1 \bowtie \Sigma X_2 \bowtie \ldots \bowtie \Sigma X_n$, where $X_1, \ldots, X_n$ is a list of pasting schemes, and every such sequence defines a valid pasting scheme.*

Proof. We sketch an idea of the proof, and mainly show how to unravel this correspondence. First note that for all i, $\Sigma D^i = D^{i+1}$, so an expression of the form $\Sigma^{i_1} D^0 \bowtie \Sigma^{i_2} D^0 \bowtie \ldots \bowtie \Sigma^{i_n} D^0$ is by definition the globular sum whose dimension table is

$$\begin{pmatrix} i_1 & & i_2 & \ldots & & i_n \\ & 0 & & \ldots & 0 & \end{pmatrix}$$

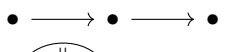
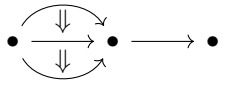
It now suffices to show that for a pasting scheme X the globular set ΣX is again a pasting scheme, whose dimension table is obtained from the dimension table of X by adding one to every number. □

In practice, we can switch in between two notations as follows: the top row in the dimension table indicates the total number of Σ that are applied to every of the D^0 whereas the bottom row indicates the total number of Σ that are applied to every $\bowtie$ operation. The previous example can now be completed with this additional notation

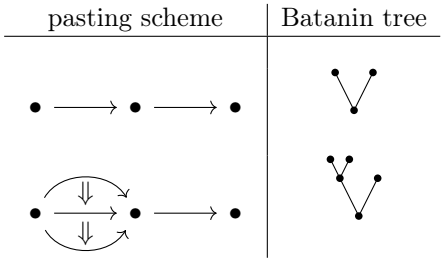
pasting scheme	globular sum	decomposition
	$\begin{pmatrix} 1 & & 1 \\ & 0 & \end{pmatrix}$	$\Sigma D^0 \bowtie \Sigma D^0$
	$\begin{pmatrix} 2 & & 2 & & 1 \\ & 1 & & 0 & \end{pmatrix}$	$\Sigma(\Sigma D^0 \bowtie \Sigma D^0) \bowtie \Sigma D^0$

There are a lot of equivalent combinatorial characterizations of the pasting schemes, among which we distinguish

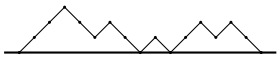
– *Dyck word:*: They are words on the alphabet $\{(,)\}$ which correspond to good parenthesizing of an expression, for instance (() is not a Dyck work since there is a mismatch, whereas (()(())) is a Dyck word. To use this representation in practice, we associate to D^0 the empty word, to the globular set ΣX, the word (w) where w is the Dyck word associated to X, and we associate to $\Sigma X \bowtie \Sigma Y$ the concatenation of w and w', where w is the word associated to ΣX and w' is the word associated to ΣY. This provides the following correspondence

pasting scheme	Dyck word
	()()
	(()())()

– *Batanin Trees*: [11] There is a bijection between the object of Θ_0 and the trees. This essentially comes down to the fact that trees satisfy the same inductive definition as pasting schemes : a tree is either a leaf or a list of trees. The correspondence between trees and pasting schemes is well known, and stronger results than a mere bijection have been proved [4, 18, 43]. In practice, we associate a leaf to the globular set D^0, and we associate to the globular set ΣX the tree obtained by adding a single vertex as the new root, and a single edge from this vertex to the root associated to ΣX. Similarly, we associate to $\Sigma X \bowtie \Sigma Y$ the tree obtained by adding a single vertex as the new root and by adding two new edges from this vertex, one that goes to the root of the tree associated to X and the other one to the tree associated to Y. With our previous example, this gives the following correspondence



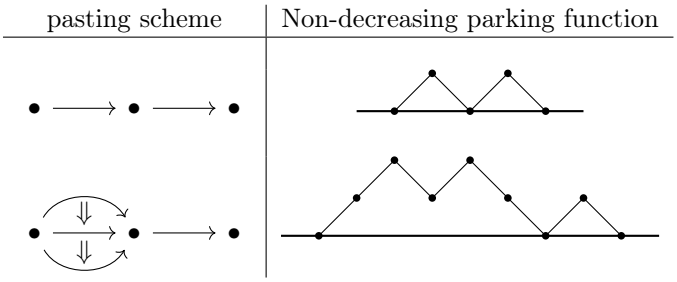
– *Non decreasing parking functions*: These are non decreasing functions over an integer interval $f : \{0, \dots, n\} \to \{0, \dots, n\}$ such that $f(0) = 0$, $f(n) = n$ and for all i, $f(i) \geq i$. We can picture these as diagrams like the following, plotting only the over diagonal part of the function.



This is our preferred description as it is visual and makes the dimension table easily readable: the successive heights of the peaks give the first row of the dimension table, whereas the height of the valleys in between the peaks gives its second row. The example we have given thus corresponds to the dimension table

$$\begin{pmatrix} 3 & & 2 & & 1 & & 2 & & 2 \\ & 1 & & 0 & & 0 & & 1 & \end{pmatrix}$$

and out previous examples give the following correspondence



In fact, pasting schemes entertain close relation with Catalan numbers, which are one of the most ubiquitous sequence of numbers, and thus have a lot of equivalent definitions.

Source and target of pasting schemes. A pasting scheme X naturally comes equipped with a source and a target, that are two distinguished globular subsets of X which are also pasting schemes. Since the source and target are isomorphic globular sets, we will define a unique object ∂X along with the two inclusions which identify ∂X as a subobject of X

$$\sigma_X, \tau_X : \partial X \to X$$

For a pasting scheme X defined by a table as above, we first define the pasting scheme ∂X obtained from X by lowering all the cells of maximal dimension by one dimension. It is thus given by the table

$$\begin{pmatrix} \overline{i_1} & & \overline{i_2} & & \cdots & & \overline{i_k} \\ & j_1 & & j_2 & \cdots & j_{k-1} & \end{pmatrix} \qquad \text{where } \overline{i_k} = \begin{cases} i_k \text{ if } i_m < i \\ i-1 \text{ if } i_m = i \end{cases}$$

Note that this definition may produce tables that do not strictly fall under the scope of globular sums, as presented before, since it is possible to have the equality

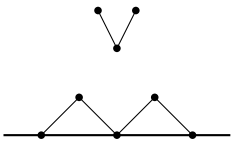
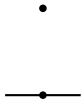
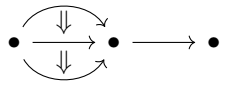
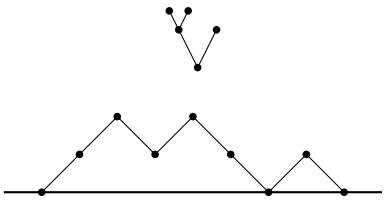
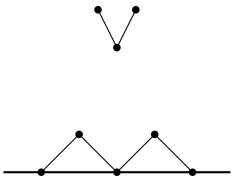
$$\overline{i_m} \quad = \quad j_m \quad = \quad \overline{i_{m+1}} \quad = \quad i-1$$

However when it is the case we will choose the corresponding iterated sources and target to be the identity maps (i.e., the map iterated 0 times). We can then introduce the following rewriting rule, that does not change the colimit and thus exhibits ∂X as a pasting scheme

$$\begin{pmatrix} \cdots & i-1 & & i-1 & \cdots \\ \cdots & & i-1 & & \cdots \end{pmatrix} \quad \rightsquigarrow \quad \begin{pmatrix} \cdots & i-1 & \cdots \\ \cdots & & \cdots \end{pmatrix}$$

We illustrate this notions with our running examples, combining all the six equivalent representations of pasting schemes we have given. In both of these examples, we indicate first the wrong dimension table, with possibly repeating numbers, and then simplify them to the actual

dimension tables.

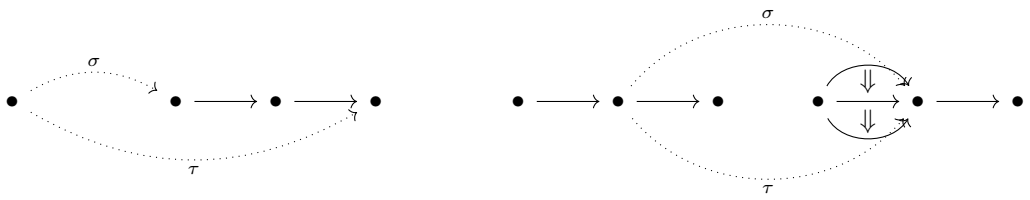
pasting scheme X	border ∂X
$\begin{pmatrix} 1 & & 1 \\ & 0 & \end{pmatrix}$	$\begin{pmatrix} 0 & & 0 \\ & 0 & \end{pmatrix} \rightsquigarrow \begin{pmatrix} 0 & \end{pmatrix}$
$\Sigma D^0 \bowtie \Sigma D^0$	D^0
$()()$	.
	
$\begin{pmatrix} 2 & & 2 & & 1 \\ & 1 & & 0 & \end{pmatrix}$	$\begin{pmatrix} 1 & & 1 & & 1 \\ & 1 & & 0 & \end{pmatrix} \rightsquigarrow \begin{pmatrix} 1 & & 1 \\ & 0 & \end{pmatrix}$
	
$\Sigma(\Sigma D^0 \bowtie \Sigma D^0) \bowtie \Sigma D^0$	$\Sigma D^0 \bowtie \Sigma D^0$
$((())())()$	$()()$
	

Now we can define the two inclusion maps σ_X and τ_X to induced by the families

$$\overline{\sigma_{i_m}} : D_{\overline{i_m}} \longrightarrow D_{i_m} \qquad \overline{\tau_{i_m}} : D_{\overline{i_m}} \longrightarrow D_{i_m}$$

$$\overline{\sigma_{i_m}} = \begin{cases} \mathrm{id}_{D_{i_m}} & \text{if } i_m < i \\ \sigma : D_{i-1} \to D_i & \text{if } i_m = i \end{cases} \qquad \overline{\tau_{i_m}} = \begin{cases} \mathrm{id}_{D_{i_m}} & \text{if } i_m < i \\ \tau : D_{i-1} \to D_i & \text{if } i_m = i \end{cases}$$

We provide a visualization of the inclusions maps σ_X and τ_X for our two running examples, using the diagrammatic representation of pasting schemes



Globular theories. Let $\mathcal{G} \to \mathcal{C}$ be a globular extension, then by universality of the globular completion, there exists a unique morphism of globular extension $F : \Theta_0 \to \mathcal{C}$. The globular extension $\mathcal{G} \to \mathcal{C}$ is called a *globular theory* if the functor induced by F is faithful and is an isomorphism on the isomorphism classes of objects. Whenever it is the case, we can up to equivalence identify Θ_0 as a (in general non-full) subcategory of $\mathcal{C}$, all the objects of $\mathcal{C}$ are equivalent to an object of this subcategory. Up to equivalence, the globular theories are thus the categories obtained by adding morphisms to Θ_0, without adding isomorphisms: It is a category whose objects can be described as the pasting schemes. A *morphism of globular theories* is a morphism of the underlying globular extensions. A morphism f of a globular theory $\mathcal{C}$ is said to be *globular* if it is in the image of Θ_0. Dually a coglobular extension $\mathcal{G}^{\mathrm{op}} \to \mathcal{C}$ is called a *coglobular theory* if $\mathcal{C}^{\mathrm{op}}$ is a globular theory.

2.1.3 Weak ω-categories

The notion of a globular theory formalizes a theory in the globular sets, for which the operations we introduce may have generalized arities that have all the shapes allowed by the pasting schemes. For instance, consider the following pasting scheme C that we describe as a diagram and as a globular sum

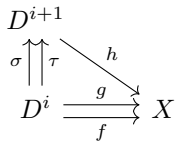
together with a globular theory $\mathcal{C}$ in which there is a morphism $\mu : C \to D^1$. Then a functor $F : \mathcal{C}^{\mathrm{op}} \to \mathbf{Set}$ induces a coglobular structure on $\mathbf{Set}$, which is the same as a globular set X; suppose that F preserves the globular products of $\mathcal{C}^{\mathrm{op}}$ for this coglobular structure. The functor F then sends the morphism μ onto an operation $m : X(1) \times_{X(0)} X(1) \to X(1)$, that is given two 1-cells $f, g \in X(1)$ such that $\partial^+(f) = \partial^-(g)$, the operation m gives a 1-cell $m(f, g) \in X(1)$. Moreover, the interaction of μ with σ and τ in $\mathcal{C}$ specify the source and target of the cell obtained by applying m. This shows that a globular theory expresses a notion of theory allowing the inputs to have shapes specified by the pasting schemes. This situation is analogue to Lawvere theories: The objects are freely generated by a class of colimits (sums for Lawvere theories, globular sums for globular theories), the additional morphisms define operations, and the equality between the morphisms give axioms. Weak ω-categories fall precisely under this scope, and thus can be defined as a particular globular theory.

Admissible pairs of arrows. Let $\mathcal{G} \to \mathcal{C}$ be a globular extension, two arrows $f, g : D^i \to X$ in $\mathcal{C}$ are said to be *parallel* when

$$f \circ \sigma_i = g \circ \sigma_i \qquad f \circ \tau_i = g \circ \tau_i$$

If $\mathcal{C}$ is a globular theory, then a morphism f of $\mathcal{C}$ is said to be *algebraic*, when for every decomposition $f = gf'$, with g globular, then g is an identity. Intuitively, an algebraic morphism is a morphism $f : D^i \to X$ of $\mathcal{C}$ that is "full" on its source, i.e., it cannot be decomposed as $f' : D^i \to X'$ followed by an inclusion $X' \hookrightarrow X$ They are the primitive morphisms of the form $D^i \to X$, in the sense all the other morphisms of this form are obtained as an inclusion of an algebraic morphism. A pair of parallel arrows $f, g : D^i \to X$ is called an *admissible pair* if either both f and g are algebraic, or there exists a decomposition $f = \sigma_X f'$ and $g = \tau_X g'$, with f' and g' algebraic. A *lift* for an admissible pair $f, g : D^i \to X$ is a morphism $h : D^{i+1} \to X$ such that

$h\sigma = f$ and $h\tau = g$

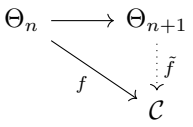


Dually, in a coglobular theory $\mathcal{C}$, an arrow is said to be *coalgebraic* if its opposite is algebraic in the globular theory $\mathcal{C}^{\mathrm{op}}$, and a pair of arrows is called a *coadmissible pair* if the opposite pair is admissible in $\mathcal{C}^{\mathrm{op}}$.

Cat-coherator. The definition of weak ω-categories relies on the notion of *cat-coherator*. Intuitively a cat-coherator defines a globular theory obtained by freely adding lifts to Θ_0 until every pair of admissible arrows has a lift. The definition is a bit technical since every lift added creates new pairs of admissible arrows, for which there needs to be lift, and hence requires taking the limit of an iterative construction. We refer the reader to [54] for the formal definition of a cat-coherator and only introduce the *Batanin-Leinster cat-coherator* Θ_∞. Since it is the only one that we use, we usually refer to the Batanin-Leintser Cat-coherator as "the cat-coherator"; it is defined to be the colimit

$$\Theta_\infty \simeq \mathrm{colim}(\Theta_0 \to \Theta_1 \to \Theta_2 \to \cdots \to \Theta_n \to \cdots)$$

Where Θ_n is given by induction on n. Define E_n to be the set of all pairs of admissible arrows of Θ_n that are not in E'_n for any $n' < n$. Then we can define Θ_{n+1} to be the universal globular extension of Θ_n obtained by formally adding a lift for each pairs in E_n. In other words, for each globular extension $f : \Theta_n \to \mathcal{C}$ such that the image by f of all pairs of arrows in E_n has a lift in $\mathcal{C}$, there is an essentially unique globular extension $\tilde{f}$, which preserves the chosen lifts and makes the following triangle commute



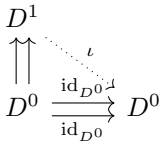
Weak ω-categories. We define a weak ω-category to be functor $F : \Theta_\infty^{\mathrm{op}} \to \mathbf{Set}$ which preserves the globular products in $\Theta_\infty^{\mathrm{op}}$, for the globular structure on **Set** induced by F. The category of weak ω-categories is the full subcategory of $\widehat{\Theta_\infty}$ whose objects are exactly the presheaves that are weak ω-categories. Under a mild conjecture, Ara has proved [4] this definition to be equivalent to a definition originally proposed by Batanin [11] and reformulated by Leinster [48].

2.1.4 Identities and compositions

We work out the definition of the identity 1-cells and the composition of 1-cells in weak ω-categories. They are the most basic constructions and we reserve more advanced examples for Section 2.4.2 where they are given in a type theoretic style. We refer the reader to [54, 4] for more examples in this style.

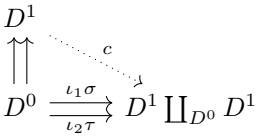
– Identity of a 0-cell: The pair of maps $(\mathrm{id}_{D^0}, \mathrm{id}_{D^0})$ is an admissible pair of arrows $D^0 \to D^0$,

hence there exists a lift



For every weak ω-category $\mathcal{F} : \Theta_\infty^{\mathrm{op}} \to \mathbf{Set}$ together with an element $x \in \mathcal{F}(D^0)$, this allows us to define its *identity* 1*-cell* $i(x) \in \mathcal{F}(D^1)$ to be $i(x) = \mathcal{F}(\iota)(x)$. Moreover, by definition, $\partial^-(i(x)) = \partial^+(i(x)) = x$ as expected for the identity 1-cell on x.

– Composition of 1-cells: We consider the globular sum given as $D^1 \coprod_{D^0} D^1$. Then there are two canonical maps $\iota_1, \iota_2 : D^1 \to D^1 \coprod_{D^0} D^1$, and we consider the following admissible pair $(\iota_1\sigma, \iota_2\tau) : D^0 \to D^1 \coprod_{D^0} D^1$. This provides the lift



For every weak ω-category $\mathcal{F} : \Theta_\infty^{\mathrm{op}} \to \mathbf{Set}$, a pair of composable 1-cells is the same as an element $(f, g) : \mathcal{F}(D^1 \coprod_{D^0} D^1)$, and the element $f \cdot g := \mathcal{F}(c)(f, g) \in \mathcal{F}(D^1)$ defines its composition. By definition, $\partial^-(f \cdot g) = \partial^-(f)$ and $\partial^+(f \cdot g) = \partial^+(g)$, as expected for the composition.

2.2 A type theory for globular sets

We first introduce and study a type theory to study globular sets. We later on build the type theory for weak ω-categories on top of this theory.

2.2.1 The type theory GSeTT

We call our type theory to work with globular sets GSeTT and introduce it in a cut-free style. After studying its properties, we discuss briefly on how to describe the same theory in a cut-full style.

Type constructors and introduction rules. Since we have already introduced the structural rules for type theories in a cut-free style, we only need to specify the type and term constructors, along with their introduction rules. This theory has no term constructors, hence the only terms are variables, and we introduce infinitely countably many type constructors: the constructor $\star$ of arity 0, and for all $n \in \mathbb{N} \setminus \{0\}$, the constructor $\to_n$ of arity 2. These are subject to the following introduction rules

$$\frac{\Gamma \vdash}{\Gamma \vdash \star}(\star\text{-INTRO}) \qquad \frac{\Gamma \vdash t : \star \qquad \Gamma \vdash u : \star}{\Gamma \vdash t \to_1 u}(\to_1\text{-INTRO})$$

$$\frac{\Gamma \vdash t : x \to_n y \qquad \Gamma \vdash u \ : x \to_n y}{\Gamma \vdash t \to_{n+1} u}(\to_{n+1}\text{-INTRO})$$

This theory can be presented in a much more efficient way, taking advantage of the uniformity of all the rules ($\to_n$-INTRO), to encompass all of them in one and only rule. For this we replace

all the type constructors $\to_n$ by a single type constructor $\to$ which takes as arguments one type and two terms, subject to the following introduction rule

$$\frac{\Gamma \vdash A \qquad \Gamma \vdash t : A \qquad \Gamma \vdash u : A}{\Gamma \vdash t \underset{A}{\to} u}(\to\text{-INTRO})$$

Note that this constructor does not strictly correspond to the framework given in Section 1.1, as it takes a type as an argument. However, it can be viewed as a short-hand for the previous family of constructors $\to_n$, which does correspond to this framework. For this reason, we still consider this theory as part of the framework. In order to relate these two formulations, we introduce the notion of *dimension* of a type, that we define as follows

$$\dim \star = -1 \qquad\qquad \dim(t \underset{A}{\to} u) = 1 + \dim A$$

We also define the dimension of a term $\Gamma \vdash t : A$ in a context to be $1 + \dim A$, and the dimension of a context to be the maximal dimension among its variables. Then, replacing the type $t \underset{A}{\to} u$ by the type $t \to_n u$ where $n = \dim A + 1$ everywhere it appears shows the equivalence between the two theories. Formally, it defines an equivalence of categories with families between their syntactic categories. Under this correspondence, the action of substitutions on types in the theory with the constructor $\to$ is given by $(t \underset{A}{\to} u)[\gamma] = (t[\gamma]) \underset{A[\gamma]}{\longrightarrow} (u[\gamma])$. This ensures automatically that all the properties that we checked in Section 1.1 are also satisfied by this type theory. From now on, are only interested about the formulation with only two type constructors $\star$ and $\to$, and we call the resulting type theory, with no term constructors GSeTT. We refer the reader to Appendix A.1 for a complete description of all the rules of this theory in a single place.

2.2.2 Formalization

We have fully implemented this type theory in our Agda formalization[1] as well as proved most of its properties. More specifically, we have proved all the syntactic properties that we have mentioned, but we have not proved the categorical facts, to avoid the use of categories with Agda which is not straightforward. We refer the reader to Appendix A.7 for a summary of all the result we have formalized.

Conventions and structure. For the formalization we have chosen to use de Bruijn levels to encode the variables, so it slightly differ with the presentation we give here. All the formalization relative to the type theory GSeTT is located in the folder called `GSeTT/`. We define the syntactic expressions corresponding to contexts, types, terms and substitutions in the file `Syntax.agda`, along with the syntactic properties: variables of an expression, action of substitutions. In the file `Rules.agda` we state all the rules of the theory, and prove that they satisfy the Proposition 2 that we have stated for all type theories. In the file `CwF-structure.agda` we have formalized all the properties showing that this theory satisfies the defining equations of a cut-full type theory, and in the file `Uniqueness-Derivations.agda` we have proved that every derivable judgment is derivable in a unique way. Additionally, we have also proved in the file `Dec-Type-Checking.agda` that the derivability for all the judgments of the theory is decidable. We present a part of our formalization here, in particular the definition and the statements of the results that we prove.

[1] `https://github.com/ThiBen/catt-formalization/`

Definition of the syntax. The pre-syntax, where we define the expressions for contexts, types terms and substitutions is declared in the file `Syntax.agda` as follows

```
data Pre-Ty : Set
data Pre-Tm : Set

data Pre-Ty where
  ∗ : Pre-Ty
  ⇒ : Pre-Ty → Pre-Tm → Pre-Tm → Pre-Ty

data Pre-Tm where
  Var : ℕ → Pre-Tm

Pre-Ctx : Set₁
Pre-Ctx = list (ℕ × Pre-Ty)

Pre-Sub : Set₁
Pre-Sub = list (ℕ × Pre-Tm)
```

and we define the action of substitutions on types and terms

```
_[_]Pre-Ty : Pre-Ty → Pre-Sub → Pre-Ty
_[_]Pre-Tm : Pre-Tm → Pre-Sub → Pre-Tm

∗ [ γ ]Pre-Ty = ∗
⇒ A t u [ γ ]Pre-Ty = ⇒ (A [ γ ]Pre-Ty) (t [ γ ]Pre-Tm) (u [ γ ]Pre-Tm)
Var x [ nil ]Pre-Tm = Var x
Var x [ γ :: (v , t) ]Pre-Tm = if x ≡ v then t else ((Var x) [ γ ]Pre-Tm)

_∘_ : Pre-Sub → Pre-Sub → Pre-Sub
nil ∘ γ = nil
(γ :: (x , t)) ∘ δ = (γ ∘ δ) :: (x , (t [ δ ]Pre-Tm))
```

Judgments and rules of the theory. We then define the judgments in the file `Rules.agda`, as inductive inductive types, following, where the inference rules give the generators

```
data _⊢C : Pre-Ctx → Set
data _⊢T_ : Pre-Ctx → Pre-Ty → Set
data _⊢t_#_ : Pre-Ctx → Pre-Tm → Pre-Ty → Set
data _⊢S_>_ : Pre-Ctx → Pre-Sub → Pre-Ctx → Set

data _⊢C where
  ec : nil ⊢C
  cc : ∀ {Γ A} → Γ ⊢C → Γ ⊢T A → (Γ :: ((length Γ) , A)) ⊢C

data _⊢T_ where
  ob : ∀ {Γ} → Γ ⊢C → Γ ⊢T ∗
  ar : ∀ {Γ A t u} → Γ ⊢t t # A → Γ ⊢t u # A → Γ ⊢T ⇒ A t u

data _⊢t_#_ where
  var : ∀ {Γ x A} → Γ ⊢C → x # A ∈ Γ → Γ ⊢t (Var x) # A
```

```
data _⊢S_>_ where
  es : ∀ {Δ} → Δ ⊢C → Δ ⊢S nil > nil
  sc : ∀ {Δ Γ γ x A t} → Δ ⊢S γ > Γ → (Γ :: (x , A)) ⊢C
                       → Δ ⊢t t # (A [ γ ]Pre-Ty)
                       → Δ ⊢S (γ :: (x , t)) > (Γ :: (x , A))
```

Note that in this theory, the judgment `_⊢S_>_` is not mutually inductive with the others and could be defined separately, however in more complicated theories, we define it by mutual induction and thus proceed the same way here. We then prove in this file the easier properties satisfied by these rules, and particular we prove cut admissibility

```
[]T : ∀ {Γ A Δ γ} → Γ ⊢T A
                  → Δ ⊢S γ > Γ
                  → Δ ⊢T (A [ γ ]Pre-Ty)
[]t : ∀ {Γ A t Δ γ} → Γ ⊢t t # A
                    → Δ ⊢S γ > Γ
                    → Δ ⊢t (t [ γ ]Pre-Tm) # (A [ γ ]Pre-Ty)
```

Structure of category with families. In the file `CwF-Structure.agda`, we proceed with showing all the equalities that define the syntactic category and endow it with a structure of category with families. We have already proved cut admissibility, we now prove in particular the functoriality of the application of substitutions and the associativity and unitality of the composition

```
[∘]T : ∀ {Γ Δ Θ A γ δ} → Γ ⊢T A
                       → Δ ⊢S γ > Γ
                       → Θ ⊢S δ > Δ
                       → ((A [ γ ]Pre-Ty) [ δ ]Pre-Ty) == (A [ γ ∘ δ ]Pre-Ty)
[∘]t : ∀ {Γ Δ Θ A t γ δ} → Γ ⊢t t # A
                         → Δ ⊢S γ > Γ
                         → Θ ⊢S δ > Δ
                        → ((t [ γ ]Pre-Tm) [ δ ]Pre-Tm) == (t [ γ ∘ δ ]Pre-Tm)
∘-admissibility : ∀ {Γ Δ Θ γ δ} → Δ ⊢S γ > Γ
                                → Θ ⊢S δ > Δ
                                → Θ ⊢S (γ ∘ δ) > Γ
∘-associativity : ∀ {Γ Δ Θ Ξ γ δ θ} → Δ ⊢S γ > Γ
                                     → Θ ⊢S δ > Δ
                                     → Ξ ⊢S θ > Θ
                                     → ((γ ∘ δ) ∘ θ) == (γ ∘ (δ ∘ θ))
∘-left-unit : ∀{Γ Δ γ} → Δ ⊢S γ > Γ
                       → (Pre-id Γ ∘ γ) == γ
∘-right-unit : ∀ {Δ γ} →  (γ ∘ Pre-id Δ) == γ
```

Uniqueness of derivations. In the file `Uniqueness-Derivations.agda` we prove by mutual induction that every derivable judgment is derivable in a unique way, by showing that the type of derivations of this judgment is contractible [63], using a terminology from homotopy type theory. We can simplify the statement even further by showing that every judgment defines a proposition, again in the sense of homotopy type theory

```
is-prop-⊢C : ∀ Γ → is-prop (Γ ⊢C)
is-prop-⊢T : ∀ Γ A → is-prop (Γ ⊢T A)
is-prop-⊢t : ∀ Γ A t → is-prop (Γ ⊢t t # A)
is-prop-⊢S : ∀ Δ Γ γ → is-prop (Δ ⊢S γ > Γ)
```

Decidability of type checking. We show that every judgment defines a decidable type in the file `Dec-Type-Checking.agda`

```
dec-⊢C : ∀ Γ → dec (Γ ⊢C)
dec-⊢T : ∀ Γ A → dec (Γ ⊢T A)
dec-⊢t : ∀ Γ A t → dec (Γ ⊢t t # A)
dec-⊢S : ∀ Δ Γ γ → dec (Δ ⊢S γ > Γ)
```

These are the most involved proofs and they in particular give a certified implementation of a type checker for this theory. Since the theory GSeTT is not very relevant in practice, this is not so important here, however for more complicated theories formally proving the decidability in Agda gives a certified implementation of the theory, of which one could extract a code that computes.

2.2.3 Yoneda embedding and nerve functor

We now study a construction that is very useful, and specific to the theories whose type constructors are the same as GSeTT: They have particular contexts, the disks and the sphere contexts which classify the types and the terms of the theory.

Disks and sphere contexts. Our objective is now to introduce specific contexts in this theory and show that they play an important role. We have completely defined them and proved their properties in the file `Disks.agda` of our formalization. We define two families of contexts, that we call the *disks* (denoted D^n, $n \in \mathbb{N}$) and the *spheres* (denoted S^n, $n \in \mathbb{N} \cup \{-1\}$). We start by choosing once and for all a family of distinct variables $(x_n)_{n\in\mathbb{N}}$, and define a family of types A_n by induction with the following formulas

$$A_{-1} = \star$$
$$A_{n+1} = x_{2n} \underset{A_n}{\longrightarrow} x_{2n+1}$$

In such a way that $\dim A_n = n$. This lets us define the disks and sphere contexts by induction as follows.

$$S^{-1} = \varnothing \qquad\qquad D^n = (S^{n-1}, x_{2n} : A_{n-1})$$
$$S^n = (D^n, x_{2n+1} : A_{n-1})$$

For instance, in low dimensions, the disks and sphere contexts are the following

$$S^{-1} = \varnothing$$
$$D^0 = (x_0 : \star) \qquad S^0 = (x_0 : \star, x_1 : \star)$$
$$D^1 = (x_0 : \star, x_1 : \star, x_2 : x_0 \underset{\star}{\to} x_1) \qquad S^1 = (x_0 : \star, x_1 : \star, x_2 : x_0 \underset{\star}{\to} x_1, x_3 : x_0 \underset{\star}{\to} x_1)$$

The Agda definition of the disks and sphere contexts that we give follows exactly these definitions.

```
𝕊 : ℕ → Pre-Ctx
𝔻 : ℕ → Pre-Ctx

𝕊 0 = nil
𝕊 (S n) = (𝔻 n) :: (length (𝔻 n) , n⇒ n)
𝔻 n = (𝕊 n) :: (length (𝕊 n) , n⇒ n)
```

Lemma 21. *The following rules are derivable*

$$\frac{}{S^n \vdash} \qquad \frac{}{S^n \vdash A_n}$$

$$\frac{}{D^n \vdash} \qquad \frac{}{D^n \vdash x_{2n} : A_{n-1}}$$

Proof. We construct a derivation for each of these judgments by induction on n.

- For $n = -1$, the context D^{-1} is not defined, the context $S^{-1} = \varnothing$ is derivable by the rule (EC), and the judgment $\varnothing \vdash \star$ is derivable by the rule ($\star$-INTRO).
- Suppose these judgments derivable for n, then the rule (CE) applies from $S^n \vdash A_n$ to give a derivation of $D^{n+1} \vdash$,and the rule (VAR) then applies to show $D^{n+1} \vdash x_{2n+2} : A_n$. By weakening (c.f. Proposition 2) this gives a derivation of $D^{n+1} \vdash A_n$ which allows for applying the rule (CE) to get a derivation of $S^{n+1} \vdash$. Since we now have $(x_{2n+2} : A_n) \in S^{n+1}$ and $(x_{2n+3} : A_n) \in S^{n+1}$, the rule (VAR) gives a derivation of $S^{n+1} \vdash x_{2n+2} : A_n$ and of $S^{n+1} \vdash x_{2n+3} : A_n$. We can then construct a derivation of $S^{n+1} \vdash A_{n+1}$ as follows

$$\frac{S^{n+1} \vdash x_{2n+2} : A_n \qquad S^{n+1} \vdash x_{2n+3} : A_n}{S^{n+1} \vdash x_{2n+2} \underset{A_n}{\longrightarrow} x_{2n+3}} (\to\text{-INTRO})$$

Although we have formalized this proof in the file `Disks.agda` we still present here as it is the first proof by mutual induction on the syntax that we show, and is very typical of the kind of proofs we use when working syntactically on a type theory. □

In the Agda formalization, the statements corresponding to this result are the following (where we denote `n⇒` the type A_n).

```
𝕊⊢ : ∀ n → 𝕊 n ⊢C
𝔻⊢ : ∀ n → 𝔻 n ⊢C
n⇒ : ℕ → Pre-Ty
```

Familial representability of the type functor. The particular importance of the disks and sphere contexts is justified by the following property, which is the central structural fact about the category with families $\mathcal{S}_{\mathsf{GSeTT}}$.

Theorem 22. *For every $n \in \mathbb{N}_{-1}$, the canonical map*

$$\begin{array}{rcl} \mathcal{S}_{GSeTT}(\Gamma, S^n) & \to & \mathrm{Ty}^\Gamma \\ \gamma & \mapsto & A_n[\Gamma] \end{array}$$

induces a bijection between the substitutions $\Gamma \to S^n$ and the types in Γ of dimension n.

Intuitively a type distinct from $\star$ in Γ is of the form $t \to u$ and hence is given by a pair of terms of the same types, such pairs correspond to substitutions to a sphere context.

Proof. This can be proved by induction on the derivation trees, and we have formalized the argument in the file `Disks.agda`. In our argument this is proved by mutual induction together with Corollary 23. The induction essentially combines the inductive characterization of the sphere as a pullback

$$\begin{array}{ccc} S^n & \xrightarrow{\chi_{x_{2n+1}}} & D^n \\ {\scriptstyle \chi_{x_{2n}}}\downarrow & \lrcorner & \downarrow{\scriptstyle \pi} \\ D^n & \xrightarrow[\pi]{} & S^{n-1} \end{array}$$

with the property defining the extension of substitutions $\langle \_, \_ \rangle$. □

By aggregating all these dimension wise bijections together, this result may be restated as a natural isomorphism between two functors

$$\mathrm{Ty} \simeq \bigcup_{n \in \mathbb{N}_{-1}} \mathcal{S}_{\mathsf{GSeTT}}(\_, S^n)$$

The category $\mathcal{S}_{\mathsf{GSeTT}}$ is a category with families whose Ty is familially representable and the sphere contexts and the family that represents this presheaf.

Corollary 23. *For every $n \in \mathbb{N}$, the canonical map*

$$\begin{array}{rcl} \mathcal{S}_{\mathit{GSeTT}}(\Gamma, D^n) & \to & \mathrm{Tm}^\Gamma \\ \gamma & \mapsto & x_{2n}[\Gamma] \end{array}$$

induces a bijection between the substitutions $\gamma : \Gamma \to D^n$ and the terms of type classified by $\pi \circ \gamma$ in Γ.

Intuitively, a term in Γ is of the form $\Gamma \vdash t : u \to v$ with u and v two parallel terms. Thus u and v define a substitution to a sphere, and the term defines a substitution to the disk that fills this sphere.

Proof. This can be proved by mutual induction together with Theorem 22, and we have formalized it in the file `Disks.agda`. □

We give the second result the status of a corollary as it is merely a consequence of the theorem together with the structure of category with families: Any category with families whose presheaf of type is familially representable automatically inherits of a familial representation of its presheaf of terms. In our Agda formalization, we do not formulate this correspondence using diagrams, but we are able to define the function that associates a type to each substitution to a sphere and show that it defines an equivalence of types

```
Ty-n : ∀ {Γ} → Σ (ℕ × Pre-Sub) (λ {(n , γ) →  Γ ⊢S γ > 𝕊 n})
            → Σ Pre-Ty (λ A → (Γ ⊢T A))
Ty-classifier : ∀ Γ → is-equiv (Ty-n {Γ})
```

Generating display maps. In order to highlight the idea of seeing the sphere contexts as a family of type classifiers and the disks contexts as a family of term classifiers, we denote χ_A the substitution $\Gamma \vdash \chi_A : S^{\dim A}$ associated to a type A, and χ_t the substitution $\Gamma \vdash \chi_t : D^{\dim t}$ associated to a term t. The typing judgment $\Gamma \vdash t : A$ can then be interpreted directly on the arrows of the syntactic category, as the commutation of the following triangle.

$$\begin{array}{ccc} \Gamma & \xrightarrow{\chi_t} & D^{\dim A\ +1} \\ & \searrow^{\chi_A} & \downarrow \pi \\ & & S^{\dim A} \end{array}$$

Moreover, the structure of category with families implies that for a context extension $(\Gamma, x : A)$, with A of dimension $n-1$, the following square is always a pullback

$$\begin{array}{ccc} (\Gamma, x : A) & \xrightarrow{\chi_x} & D^n \\ \pi \downarrow & \lrcorner & \downarrow \pi \\ \Gamma & \xrightarrow[\chi_A]{} & S^{n-1} \end{array}$$

And any context can be obtained as a finite succession of such pullbacks. We draw many important consequences from this fact. It allows us to understand contexts in the theory as a formal analogue to finite CW-complexes (see for instance [40] for a definition of those), which are one of the central object of algebraic topology.
We call *generating display map* a display map of the form $D^n \to S^{n-1}$, and the display maps are the closure by pullback and composition of the generating display maps. In practice the properties we work with all respect pullbacks and composition, thus in order to check a property for all display maps, it suffices to show it only for generating display maps.

The cut-full style GSeTT. This theorem also gives an interesting insight for a cut-full formulation of the type theory GSeTT. We just sketch here the construction, and do not check that it is well-formed. Such a proof could be done by mutual induction, however formalizing this construction in a language such as Agda leads to a coherence problem: For instance, for a term $\Gamma \vdash t : A$ proving that $t[\gamma][\delta] = t[\gamma \circ \delta]$ is not straightforward, as both of these terms do not have the same type, so one needs to transport along a proof that $\Gamma \vdash A[\gamma][\delta] = A[\gamma \circ \delta]$, but this proof itself depends inductively on the proof that $t[\gamma][\delta] = t[\gamma \circ \delta]$. This minimalist example illustrates how in such a formulation all the properties depend on each other, in a way that Agda's termination checking cannot handle.
We introduce the type constructors $\to_n$, which now do not carry any argument, and give the following introduction rules for these type constructors

$$\frac{}{S^n \vdash \to_n}$$

Of course, the context S^n cannot be defined beforehand and has to be defined together with these introduction rules, by the following formulas

$$S^{-1} = \varnothing \qquad S^{n+1} = (S^n, \to_n, \to_n [\pi])$$

This highlights why this style of presentation is not our preferred one to work with type theory in practice. The sphere contexts are defined using types, which are themselves introduce by a

rule using the sphere contexts, and this kind of mutually inductive situations make proofs very complicated on paper as well as in a proof-assistant like Agda. Altenkirch and Kaposi [1] have proposed a framework to introduce type theories in this style in Agda, using quotient inductive inductive types, but those are not yet a native feature of Agda and have to defined completely manually, hence although this method gives an important theoretical insight, it is still hard to use in practice. Lafont [44] has also introduced a method using heterogeneous equality to handle the problem, but it requires deactivating Agda's termination checker.

2.2.4 The syntactic category $\mathcal{S}_{\mathsf{GSeTT}}$

As we have already illustrated with the familial representability of the Ty and Tm presheaves, some properties of our type theory that are syntactic directly translate as categorical properties on its syntactic category. Our goal is now to completely characterize the syntactic property of this theory.

Coglobular extension. In the category $\mathcal{S}_{\mathsf{GSeTT}}$ we have introduced specific objects D^n for each $n \in \mathbb{N}$. Moreover, Corollary 23 shows that a morphism $D^n \to D^m$ corresponds to a term of dimension m in D^n. Since all terms are variables, we can list explicitly all these morphisms

$$\mathcal{S}_{\mathsf{GSeTT}}(D^n, D^m) = \begin{cases} \{\chi_{x_{2m}}, \chi_{x_{2m+1}}\} & \text{if } m < n \\ \{\chi_{x_{2n}}\} & \text{if } m = n \\ \emptyset & \text{if } m > n \end{cases}$$

Moreover, we can check that these substitutions satisfy the globular relations, and hence this explicit description exhibits $\mathcal{G}^{\mathrm{op}}$ as the full subcategory of $\mathcal{S}_{\mathsf{GSeTT}}$ whose objects are the disk contexts. We denote $D : \mathcal{G}^{\mathrm{op}} \to \mathcal{S}_{\mathsf{GSeTT}}$ this inclusion functor. This gives in particular a coglobular extension on the category $\mathcal{S}_{\mathsf{GSeTT}}$, and we denote $\partial^- : D^{n+1} \to D^n$ and $\partial^- : D^{n+1} \to D^n$ the maps exhibited from this extension. For a term $\Gamma \vdash t : u \to v$ of dimension non-zero classified by the substitution $\Gamma \vdash \chi_t$, we have $\partial^- \circ \chi_t = \chi_u$ and $\partial^+ \circ \chi_t = \chi_v$. With a slight abuse of notation, we also denote $\partial^-(t) = u$ and $\partial^+(t) = v$ in this case.

The nerve functor We denote ν the *nerve functor* associated to the functor D

$$\nu : \mathcal{S}^{\mathrm{op}}_{\mathsf{GSeTT}} \to \mathbf{GSet} \qquad \nu(\Gamma) = \mathcal{S}_{\mathsf{GSeTT}}(\Gamma, D-)$$

Explicitly, an element of the set $\nu(\Gamma)_n$ is the same as a term in Γ of dimension n. Since the only terms are variables, and each context has only finitely many variables, the functor ν factors as $\nu : \mathcal{S}^{\mathrm{op}}_{\mathsf{GSeTT}} \to \mathbf{FinGSet}$, and for simplicity, we also denote $\nu : \mathcal{S}_{\mathsf{GSeTT}} \to \mathbf{FinGSet}^{\mathrm{op}}$ the opposite functor. We denote ν this functor, by opposition to Tm the presheaf of terms, to emphasize that ν equips the set of terms with a structure of globular sets, whereas in an arbitrary category with families, the set of terms has no extra structure. The explicit description of $\nu(D^k)$ shows that it is equivalent to the representable $Y(k)$, or in other words, ν fits into the following commutative triangle

$$\begin{array}{ccc} \mathbf{FinGSet}^{\mathrm{op}} & \xleftarrow{\ \nu\ } & \mathcal{S}_{\mathsf{GSeTT}} \\ {\scriptstyle \mathrm{Y}}\uparrow & \nearrow_{D-} & \\ \mathcal{G}^{\mathrm{op}} & & \end{array}$$

This is equivalent to the fact that the functor $D-$ is fully faithful. By continuity of the hom-functor with respect to its first variable, and the pointwise computation of colimits in **FinGSet**, $\nu : \mathcal{S}^{\mathrm{op}}_{\mathsf{GSeTT}} \to \mathbf{FinGSet}$ sends limits in $\mathcal{S}_{\mathsf{GSeTT}}$ to colimits in **FinGSet**. With our abuse of notation, this shows that $\nu : \mathcal{S}_{\mathsf{GSeTT}} \to \mathbf{FinGSet}^{\mathrm{op}}$ preserves the limits.

Lemma 24. *For all context Δ, and all $n \in \mathbb{N}$, there are two natural isomorphisms*

$$\mathcal{S}_{\mathsf{GSeTT}}(\Delta, D^n) \simeq \mathbf{GSet}(\nu(D^n), \nu(\Delta))$$
$$\mathcal{S}_{\mathsf{GSeTT}}(\Delta, S^n) \simeq \mathbf{GSet}(\nu(S^n), \nu(\Delta))$$

Proof. First note that in the case of a context D^n, by definition of ν, we have $\nu(D^n) = \mathrm{Y}(n)$, hence, for any context Δ, by the Yoneda lemma,

$$\mathbf{GSet}(\nu(D^n), \nu(\Delta)) \simeq \nu(\Delta)_n = \mathcal{S}_{\mathsf{GSeTT}}(\Delta, D^n)$$

We check the second bijection by induction: for $n = -1$ it is satisfied since S^{-1} is the empty context which is terminal, and $\nu(S^{-1})$ is the empty globular set which is initial. Suppose that the equality holds for a given $n \in \mathbb{N} \cup \{-1\}$, then the context S^{n+1} is obtained as the following pullback which by continuity of the hom-functor gives the following pullback in **Set**

$$\begin{array}{ccc} S^{n+1} & \longrightarrow & D^n \\ \downarrow & \lrcorner & \downarrow \\ D^n & \longrightarrow & S^n \end{array} \qquad \begin{array}{ccc} \mathcal{S}_{\mathsf{GSeTT}}(\Delta, S^{n+1}) & \longrightarrow & \mathcal{S}_{\mathsf{GSeTT}}(\Delta, D^n) \\ \downarrow & \lrcorner & \downarrow \\ \mathcal{S}_{\mathsf{GSeTT}}(\Delta, D^n) & \longrightarrow & \mathcal{S}_{\mathsf{GSeTT}}(\Delta, S^n) \end{array}$$

Moreover, since ν sends the limits to the colimits, we have the following pushout in **FinGSet**, which by continuity of the hom-functor gives the pullback in **Set**.

$$\begin{array}{ccc} \nu(S^n) & \longrightarrow & \mathrm{Y}(n) \\ \downarrow & \ulcorner & \downarrow \\ \mathrm{Y}(n) & \longrightarrow & \nu(S^{n+1}) \end{array} \qquad \begin{array}{ccc} \mathbf{GSet}(\nu(S^{n+1}), \nu(\Delta)) & \longrightarrow & \nu(\Delta)_n \\ \downarrow & \lrcorner & \downarrow \\ \nu(\Delta)_n & \longrightarrow & \mathbf{GSet}(\nu(S^n), \nu(\Delta)) \end{array}$$

The induction hypothesis exhibits $\mathcal{S}_{\mathsf{GSeTT}}(\Delta, S^{n+1})$ and $\mathbf{GSet}(\nu(S^{n+1}), \nu(\Delta))$ as pullbacks over the same diagram in **Set**, hence they are isomorphic. □

Lemma 25. *The inclusion functor $D_- : \mathcal{G} \to \mathcal{S}_{\mathsf{GSeTT}}$ is codense, or equivalently the associated nerve ν is fully faithful*

Proof. Consider two contexts Δ and Γ, we prove by induction on the context Γ the bijection

$$\mathcal{S}_{\mathsf{GSeTT}}(\Delta, \Gamma) \simeq \mathbf{FinGSet}(\nu(\Gamma), \nu(\Delta))$$

The proof is very similar to the second part of Lemma 24. If Γ is is the empty context $\varnothing$ which is terminal, since no term is derivable in $\varnothing$, $\nu(\varnothing)$ is the empty globular set, which is initial, this proves the bijection. Suppose $\Gamma = (\Gamma', x : A)$ with the bijection holding for Γ', then by construction Γ is obtained as the following pullback, which by continuity of the hom-functor gives the following pullback in **Set**

$$\begin{array}{ccc} \nu(S^{n-1}) & \longrightarrow & \nu(\Gamma') \\ \downarrow & \ulcorner & \downarrow \\ \mathrm{Y}(n) & \longrightarrow & \nu(\Gamma) \end{array} \qquad \begin{array}{ccc} \mathcal{S}_{\mathsf{GSeTT}}(\Delta, \Gamma) & \longrightarrow & \mathcal{S}_{\mathsf{GSeTT}}(\Delta, D^n) \\ \downarrow & \lrcorner & \downarrow \\ \mathcal{S}_{\mathsf{GSeTT}}(\Delta, \Gamma') & \longrightarrow & \mathcal{S}_{\mathsf{GSeTT}}(\Delta, S^n) \end{array}$$

Moreover since ν sends limits to colimits, we have the following pushout, which by continuity of the hom-functor yields the following pullback in **Set**.

$$\begin{array}{ccc} \Gamma & \longrightarrow & D^n \\ \downarrow & \lrcorner & \downarrow \\ \Gamma' & \xrightarrow[A]{} & S^{n-1} \end{array} \qquad \begin{array}{ccc} \mathbf{GSet}(\nu(\Gamma), \nu(\Delta)) & \longrightarrow & \mathbf{GSet}(\nu(\Delta)_n) \\ \downarrow & \lrcorner & \downarrow \\ \mathbf{GSet}(\nu(\Gamma'), \nu(\Delta)) & \longrightarrow & \mathbf{GSet}(\nu(S^{n-1}), \nu(\Delta)) \end{array}$$

and the induction hypothesis together with Lemma 24 then exhibits the sets $\mathcal{S}_{\mathsf{GSeTT}}(\Delta, \Gamma)$ and $\mathbf{GSet}(\nu(\Gamma), \nu(\Delta))$ as pullbacks over the same diagram, hence they are isomorphic. □

Although the above lemma is expressed in a categorical language, it has a very clear type theoretic translation, indeed, a natural transformation between $\nu(\Gamma)$ and $\nu(\Delta)$ can be described as the data of, for each term of Δ, a term of Γ such that this association respects source and target. Since terms and variables are the same in this theory, this can be understood as associating a term t_x of Δ for each variable x of Γ, such that if x is of type $y \to z$ in Γ, t_x is of type $t_y \to t_z$ in Δ. This lemma states that such a data is equivalent to a substitution $\Delta \to \Gamma$, which is simply a reformulation of the rules that generate substitutions.

Finite globular sets. The nerve functor closely connects the syntactic category of the theory GSeTT to the category of finite globular sets.

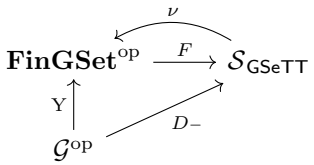
Lemma 26. *The syntactic category $\mathcal{S}_{\mathsf{GSeTT}}$ has all finite limits and its models are equivalent to the functors $\mathcal{S}_{\mathsf{GSeTT}} \to \mathbf{Set}$ preserving the finite limits.*

Proof. This is a consequence of the fact that all the terms in this theory are variables, which immediately implies that $\mathcal{S}_{\mathsf{GSeTT}}$ as all most general unifiers, and they are all primitive. An equivalent way to state this is to notice that since all the terms are variables, all the maps in $\mathcal{S}_{\mathsf{GSeTT}}$ are display maps, and hence $\mathcal{S}_{\mathsf{GSeTT}}$ has all pullbacks. As it also has a terminal object, this shows that it has all finite limits. Models are then equivalent to the functor preserving the terminal object and the pullbacks, which are the functors preserving all finite limits. □

Since the category $\mathcal{S}_{\mathsf{GSeTT}}$ has all finite limits, the universal property of the free completion gives a lift of the inclusion functor $\mathcal{G} \to \mathcal{S}_{\mathsf{GSeTT}}$ as a functor $F : \mathbf{FinGSet}^{\mathrm{op}} \to \mathcal{S}_{\mathsf{GSeTT}}$ which preserves finite limits.

Theorem 27. *The pair of functors $\nu : \mathcal{S}_{\mathsf{GSeTT}} \rightleftarrows \mathbf{FinGSet}^{\mathrm{op}} : F$ is an equivalence of categories.*

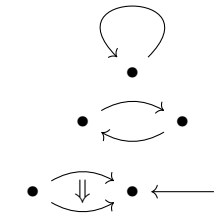
Proof. We have already proved in Lemma 25 that ν is fully faithful, in order to prove that equivalence is adjoint, it suffices to prove that it is essentially surjective, and in fact we show that $\nu \circ F \simeq \mathrm{id}$. The functors that we have defined fit into the following diagram



Note that $\nu F \, \mathrm{Y} = \nu D_- = \mathrm{Y}$, and since both ν and F preserve finite limits, so does νF. By essentially uniqueness in the free completion by finite limits, this proves that $\nu F \simeq \mathrm{id}$. □

One could give more explicit characterizations of the functors F and ν, as in this simple case they are easy to describe, and a proof of the equivalence can be derived from this concrete description. However we still prefer the more categorical proof as it gives a more general view and generalizes better to more involved cases, where explicit definitions tend to become quickly intractable. We refer the reader to [33, 16] for the complete combinatorial description along with a proof of the

equivalence in this style, and simply give examples of correspondence between contexts and finite globular sets using diagrammatic notations

Finite globular set	Context
	$(x : \star, f : x \underset{\star}{\to} x)$
	$(x : \star, y : \star, f : x \underset{\star}{\to} y, g : y \underset{\star}{\to} x)$
	$(x : \star, y : \star, z : \star, f_1 : x \underset{\star}{\to} y, f_2 : x \underset{\star}{\to} y, g : z \underset{\star}{\to} y, \alpha : f_1 \xrightarrow[x \underset{\star}{\to} y]{} f_2)$

The choice of names for the variables in a context and of a particular order among the valid ones are arbitrary. This reflect the fact that the functor F was defined up to isomorphism, by setting its values on limits.

2.2.5 Models of $\mathcal{S}_{\mathsf{GSeTT}}$

Our characterization of the syntactic category $\mathcal{S}_{\mathsf{GSeTT}}$ enables us to compute its category of models. The characterization is simple in our theory without term constructors, but generalizes in a very useful way for theories that have term constructors.

Theorem 28. *The category* $\mathbf{Mod}(\mathcal{S}_{\mathsf{GSeTT}})$ *of models of* $\mathcal{S}_{\mathsf{GSeTT}}$ *is equivalent to the category of globular sets* **GSet**

Proof. By Lemma 10, the models of GSeTT are equivalent to the functors $\mathcal{S}_{\mathsf{GSeTT}} \to \mathbf{Set}$ preserving the pullbacks along the display maps, and since all the maps are display maps, those are the functors that preserve finite limits. Under the equivalence given by Theorem 27 these are equivalent to the functors $\mathbf{FinGSet}^{\mathrm{op}} \to \mathbf{Set}$ that preserve finite limits. Since $\mathbf{FinGSet}^{\mathrm{op}}$ is the free completion of $\mathcal{G}^{\mathrm{op}}$ by finite limits, these are equivalent to the functors $\mathcal{G}^{\mathrm{op}} \to \mathbf{Set}$, which are exactly the globular sets. □

2.2.6 Coglobular structure on categories with families

The syntactic category $\mathcal{S}_{\mathsf{GSeTT}}$ is equipped with a functor $D- : \mathcal{G}^{\mathrm{op}} \to \mathcal{S}_{\mathsf{GSeTT}}$, such functors constitute a coglobular structure on the category. Moreover, in this case all the morphisms of GSeTT are sent onto display maps in the category $\mathcal{S}_{\mathsf{GSeTT}}$ (since all its maps are display maps). We thus call a *coglobular structure on a category with family* $\mathcal{C}$ a coglobular structure on its underlying category which sends all maps onto display maps. We call a *coglobular structured category with families* a category with families equipped with a coglobular structure.

Globular limits. Given a category with a coglobular structure $F : \mathcal{G}^{\mathrm{op}} \to \mathcal{C}$, we call a *globular diagram* a diagram that factors through F as $\mathcal{I} \to \mathcal{G}^{\mathrm{op}} \to \mathcal{C}$, and we call *globular limits* the limits of the globular diagrams.

Lemma 29. *A coglobular structured category with families has all the finite globular limits.*

Proof. All the maps in $\mathcal{G}$ are sent onto display maps, and by assumption $\mathcal{C}$ has pullbacks along those. Hence $\mathcal{C}$ has a terminal object and the pullbacks long the images of the maps in $\mathcal{G}$, so it has all the finite globular limits. □

Morphisms of coglobular structures category with families. Given two coglobular structured categories with families, a *morphism* between them is a morphism of the underlying categories with families that commutes with the globular structures.

Lemma 30. *A morphism of coglobular structured categories with families preserve the finite globular limits.*

Proof. By definition a morphism of category with families preserves the terminal objects and the pullback along display maps. Since the morphism commutes with the coglobular structures, it then sends a pullback along a globular map onto a pullback along a globular map, and hence preserve all finite globular limits. □

Induced coglobular structure. Given a coglobular structure on a category with families $\mathcal{C}$, and a morphism of categories with families $F : \mathcal{C} \to \mathcal{D}$, F induces a unique coglobular structure on $\mathcal{D}$ such that it is a morphism of coglobular structured category with families, since as a morphism of categories with families, F preserves the display maps. In particular $D- : \mathcal{G}^{\mathrm{op}} \to \mathcal{S}_{\mathsf{GSeTT}}$ is a coglobular structured category with families, and any morphism of categories with families $F : \mathcal{S}_{\mathsf{GSeTT}} \to \mathcal{D}$ induces a coglobular structure on the category with families $\mathcal{D}$.

2.3 Pasting schemes as contexts

In Section 2.1 we have presented a definition of weak ω-categories that heavily relies on some specific finite globular sets that we call pasting schemes. In order to reproduce this construction, we give a characterization of the pasting schemes, seen as contexts in the theory GSeTT as allowed by Theorem 27.

2.3.1 Ps-contexts

We introduce a new judgment $\Gamma \vdash_{\mathsf{ps}}$ that we understand as "Γ is a normalized context representation of a pasting scheme". We also introduce an auxiliary judgment $\Gamma \vdash_{\mathsf{ps}} x : A$, whose meaning we infer from the inference rules. These two judgments are subject to the following rules

$$\frac{}{x : \star \vdash_{\mathsf{ps}} x : \star}(\text{PSS}) \qquad \frac{\Gamma \vdash_{\mathsf{ps}} x : A}{\Gamma, y : A, f : x \underset{A}{\longrightarrow} y \vdash_{\mathsf{ps}} f : x \underset{A}{\longrightarrow} y}(\text{PSE})$$

$$\frac{\Gamma \vdash_{\mathsf{ps}} f : x \underset{A}{\longrightarrow} y}{\Gamma \vdash_{\mathsf{ps}} y : A}(\text{PSD}) \qquad \frac{\Gamma \vdash_{\mathsf{ps}} x : \star}{\Gamma \vdash_{\mathsf{ps}}}(\text{PS})$$

We call a context that satisfies $\Gamma \vdash_{\mathsf{ps}}$ a *ps-context*.

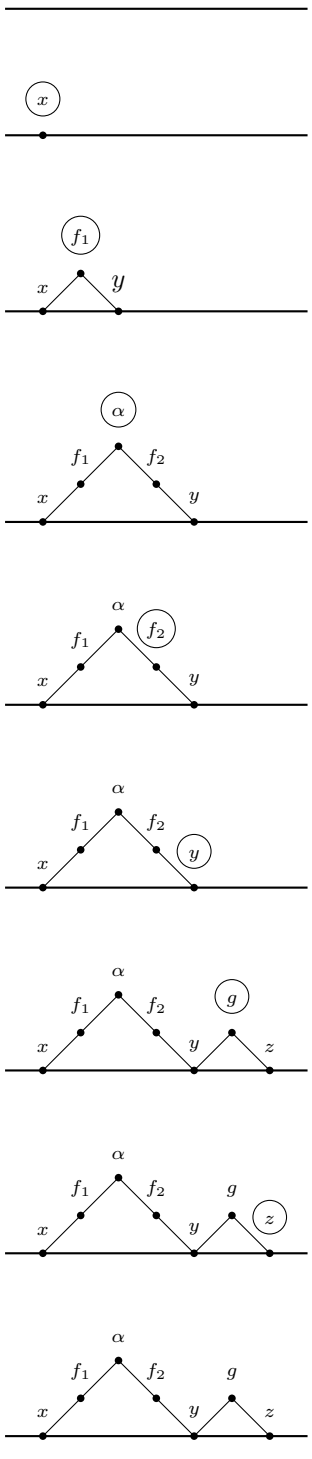
Computation on an example. In order to understand how a derivation of this judgment works, we give a graphical representation of its run: We derive the judgment asserting that the following context Γ_w satisfies $\Gamma_w \vdash_{\mathsf{ps}}$.

$$\Gamma_w = (x : \star, y : \star, f_1 : x \to y, f_2 : x \to y, \alpha : f_1 \to f_2, z : \star, g : y \to z)$$

This context describes the globular set

$$x \underset{f_2}{\overset{f_1}{\Downarrow \alpha}} y \xrightarrow{g} z$$

We give a graphical representation using the description of pasting schemes as non-decreasing parking functions, in which the dangling variable is circled, together with the corresponding judgments, and applications of the rules, describing the entire derivation tree.



$x : \star \vdash_{\mathsf{ps}} x : \star$

$(x : \star, y : \star, f_1 : x \to y) \vdash_{\mathsf{ps}} f_1 : x \to y$

$(x : \star, y : \star, f_1 : x \to y,$
$f_2 : x \to y, \alpha : f_1 \to f_2) \vdash_{\mathsf{ps}} \alpha : f_1 \to f_2$

$(x : \star, y : \star, f_1 : x \to y,$
$f_2 : x \to y, \alpha : f_1 \to f_2) \vdash_{\mathsf{ps}} f_2 : x \to y$

$(x : \star, y : \star, f_1 : x \to y,$
$f_2 : x \to y, \alpha : f_1 \to f_2) \vdash_{\mathsf{ps}} y : \star$

$(x : \star, y : \star, f_1 : x \to y,$
$f_2 : x \to y, \alpha : f_1 \to f_2$
$z : \star, g : y \to z) \vdash_{\mathsf{ps}} g : y \to z$

$(x : \star, y : \star, f_1 : x \to y,$
$f_2 : x \to y, \alpha : f_1 \to f_2$
$z : \star, g : y \to z) \vdash_{\mathsf{ps}} z : \star$

$\Gamma_w \vdash_{\mathsf{ps}}$

(PSS)
(PSE)
(PSE)
(PSD)
(PSD)
(PSE)
(PSD)
(PS)

Ps-contexts are well-formed The following property shows that any ps-context is in fact firstly a well-formed context. Hence our judgment $\Gamma \vdash_{\mathsf{ps}}$ only characterizes a sub class of contexts.

Proposition 31. *Any pre-context that is a ps-context is also a context. In other words, the following rule is admissible*

$$\frac{\Gamma \vdash_{\mathsf{ps}}}{\Gamma \vdash}$$

Proof. We prove by induction on the derivation tree that given a derivation of $\Gamma \vdash_{\mathsf{ps}} x : A$, we can construct a derivation of $\Gamma \vdash x : A$. We give a sketch of the argument here, as we have proved it in the file `CaTT/Ps-contexts.agda` of our Agda formalization[2]

- For a tree that is a single application of the rule (PSS), necessarily, the derived judgment is f the form $x : \star \vdash_{\mathsf{ps}} x : \star$, and we can construct a derivation of the judgment $x : \star \vdash x : \star$.
- For a tree whose last rule is (PSE), the corresponding judgment is necessarily of the form $(\Gamma, y : A, f : x \to y) \vdash_{\mathsf{ps}} f : x \to y$. Then we have a derivation of $\Gamma \vdash_{\mathsf{ps}} x : A$, which by induction lets us construct a derivation of $\Gamma \vdash x : A$. Since moreover, $y, f \notin \mathrm{Var}(\Gamma)$, we successively derive that $(\Gamma, y : A) \vdash$, that $(\Gamma, y : A) \vdash x \to y$ and that $(\Gamma, y : A, f : x \to y)$. Using the rule (VAR), it follows that $(\Gamma, y : A, f : x \to y) \vdash f : x \to y$.
- For a tree whose last rule is (PSD), necessarily the corresponding judgment is of the form $\Gamma \vdash y : A$, and we have derivation of the judgment $\Gamma \vdash_{\mathsf{ps}} f : x \underset{A}{\to} y$, which gives by induction a derivation of $\Gamma \vdash x \underset{A}{\to} y$, from which we can extract a derivation of $\Gamma \vdash y : A$.

We now apply this inductive result to the ps-context $\Gamma \vdash_{\mathsf{ps}}$: there necessarily exists a variable x such that $\Gamma \vdash_{\mathsf{ps}} x : \star$, so we extract a derivation of $\Gamma \vdash x : \star$. Using Proposition 2, we get a derivation of $\Gamma \vdash$. □

Decidability. For all the syntactic contexts Γ, the derivability of the the judgment $\Gamma \vdash_{\mathsf{ps}}$ is a decidable problem. In fact we can show that the derivability of the judgment $\Gamma \vdash_{\mathsf{ps}} x : A$ for any variable x and any type A is a decidable problem. Indeed, if Γ is of the form $(x : \star)$, the only derivable judgment of this form is $x : \star \vdash_{\mathsf{ps}} x : \star$. Otherwise, if it is of the form $(\Gamma, y : A\ , f : x \to y)$, the judgment $\Gamma, y : A, f : x \to y \vdash_{\mathsf{ps}} z : B$ is derivable if and only if $\Gamma \vdash_{\mathsf{ps}} x : A$ is derivable and $(z : B) \in (\Gamma, y : A, f : x \to y)$ is an iterated target of the variable f. Since the second statement is decidable, and by induction the derivability of $\Gamma \vdash_{\mathsf{ps}} x : A$ is also decidable, so is the derivability of $\Gamma, y : A, f : x \to y \vdash_{\mathsf{ps}} z : B$. Since all the contexts are finite, this shows that for every context, the judgment $\Gamma \vdash_{\mathsf{ps}} x : A$ is decidable. Moreover, given a context Γ, the only possible candidate for a variable x such that $\Gamma \vdash_{\mathsf{ps}} x : \star$ is the last variable to appear with type $\star$ in Γ, which shows that the existence of x such that $\Gamma \vdash_{\mathsf{ps}} x : \star$ is decidable, hence the derivability of $\Gamma \vdash_{\mathsf{ps}}$ is decidable.

Uniqueness of derivations. For any context Γ, there is at most one derivation of the judgment $\Gamma \vdash_{\mathsf{ps}}$. Indeed, we can show by induction that there is at most one variable x in each dimension dimension such that $\Gamma \vdash_{\mathsf{ps}} x : A$ is derivable, together with a unique derivation for tree for this judgment. This is again done by induction on Γ.

- For a context of the form $(x : \star)$, there is only one variable y such that $x : \star \vdash_{\mathsf{ps}} y : A$. Indeed, the only possible derivation is an application of the rule (PSS) which implies that $y = x$.

[2] `https://github.com/ThiBen/catt-formalization`

– For a context of the form $(\Gamma, y : A, f : x \to y)$, there is by induction a unique derivation of $\Gamma \vdash_{\mathsf{ps}} x : A$, which yields a unique derivation of $\Gamma, y : A, f : x \to y \vdash_{\mathsf{ps}} f : x \to y$ by application of (PSE). All the other derivations of judgments of the form $\Gamma, y : A, f : x \to y \vdash_{\mathsf{ps}} z : B$ are obtained from this derivation by successive applications of the rule (PSD), which decreases the dimension by 1 every time, hence the uniqueness of the variable, together with the uniqueness of its derivation.

Applying this result for a context Γ such that $\Gamma \vdash_{\mathsf{ps}}$, there is a unique variable x such that $\Gamma \vdash_{\mathsf{ps}} x : \star$, and a unique derivation of $\Gamma \vdash_{\mathsf{ps}} x : \star$, hence the derivation of $\Gamma \vdash_{\mathsf{ps}}$ is unique as well. Although we have not formalized the decidability and the uniqueness of derivation for these judgments, we keep it for future work and expect it to be straightforward.

Ps-contexts are not invariant by isomorphism. Note that the judgment $\Gamma \vdash_{\mathsf{ps}}$ is not derivable for certain contexts that do correspond to pasting schemes, but only for ones that are normalized. Consider the following example

$$\begin{array}{ll} (x : \star, y : \star, f : x \to y, z : \star, g : y \to z) & \text{is a ps-context} \\ (x : \star, y : \star, z : \star, f : x \to y, g : y \to z) & \text{is a not ps-context} \end{array}$$

even though these two contexts are isomorphic in the category $\mathcal{S}_{\mathsf{GSeTT}}$. This is not much of an issue, since we show in the next section that every pasting scheme corresponds exactly to one ps-context, but it is an important point to keep in mind. Moreover, we could have chosen a different normalization for the contexts that correspond to pasting schemes, which would lead to a different notion of ps-context, but this would not change the theory for weak ω-categories. This shows that we are interested in a bijection between the set of ps-contexts and between the set of pasting schemes, and not just an equivalence of categories between these.

2.3.2 Ps-contexts are normalized pasting schemes

The description and explicit example that we have shown for the ps-contexts shows that this new judgment entertain a close connection with the notion of pasting schemes. We now explore this connection and characterize exactly the relation between these two notions.

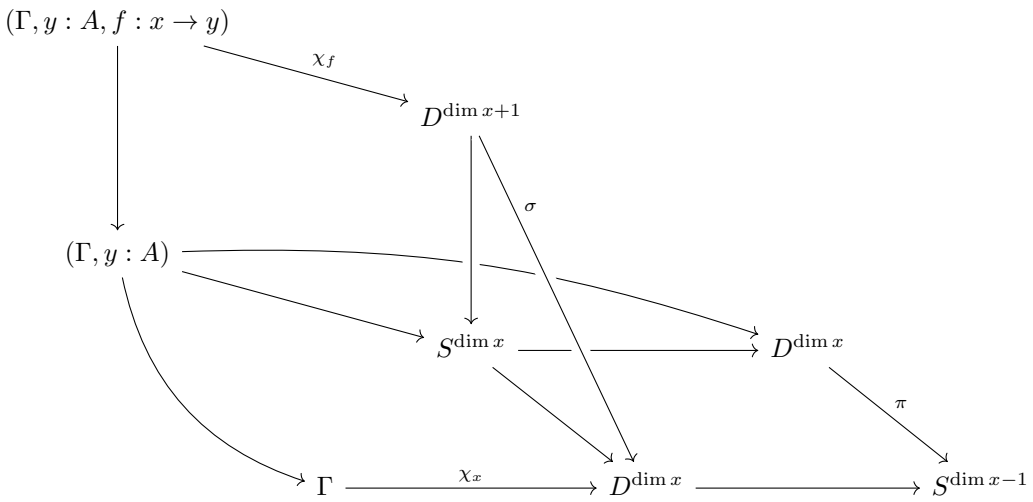
Double context extension. The main difficulty while working with ps-context is that in all the inductions, we perform a succession of two context comprehension operations, which has a rather sketchy interpretation in terms of limits in the category. In order to simplify the proof, we give a much cleaner categorical interpretation of the double context comprehension that we use.

Lemma 32. *If Γ is a context with a well defined variable $\Gamma \vdash x : A$, then $(\Gamma, y : A, f : x \to y)$ realizes the following pullback*

$$\begin{array}{ccc} (\Gamma, y : A, f : x \to y) & \xrightarrow{\chi_f} & D^{\dim x + 1} \\ \downarrow & \lrcorner & \downarrow \sigma \\ \Gamma & \xrightarrow{\chi_x} & D^{\dim x} \end{array}$$

Proof. There are two approaches to show this result. We can prove the dual statement holds for the nerves of the contexts, which can be done by giving an explicit description of the nerves and

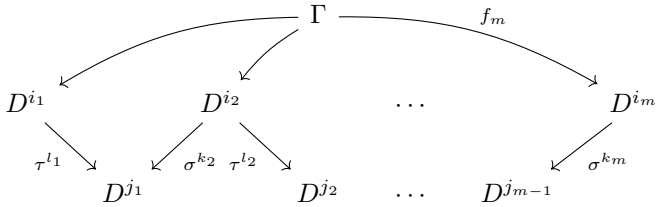
computing colimits in **Set**. We can also show that it holds directly on $\mathcal{S}_{\mathsf{GSeTT}}$, by exhibiting the following diagram



where the two horizontal squares and the vertical square are all pullbacks, and showing by diagram chasing techniques that the expected square, which appears in the figure is a pullback square □

Partial globular product cone. Although proving that ps-contexts are globular products is not conceptually very complicated, it is a bit tricky to formulate precisely the good loop invariant without introducing an ad-hoc categorical notion to describe it. For this reason, we introduce here *partial globular product cones*, whose only purpose is to be a categorical invariant for the induction proof.

Definition 33. A partial globular product cone is a limiting cone over a globular product diagram



together with a map $u : \Gamma \to D^n$ for $n \leq j_m$ such that $\tau^{j_{m-n}} \circ f_m = u$.

This notion aims to encode categorically the information included in the judgment $\Gamma \vdash_{\mathsf{ps}} x : A$, as expressed by the following lemma.

Lemma 34. *A derivation of the judgment $\Gamma \vdash_{\mathsf{ps}}$ yields a partial globular product cone of apex Γ, whose additional specified map is $\chi_x : \Gamma \to D^{\dim x}$*

Proof. We prove this lemma by induction on the structure of the derivation of $\Gamma \vdash_{\mathsf{ps}} x : A$

– If the derivation is a single application of the rule (PSS), then necessarily $A = \star$ and $\Gamma = D^0$, and the identity substitution $\Gamma \vdash \mathrm{id}_\Gamma : \Gamma$ is the classifying substitution $\chi_x = \mathrm{id}_\Gamma$

– If the derivation ends with an application of the rule (PSE) , then necessarily Γ is of the form $(\Gamma', y : B, x : y \to z)$, and $A = y \to z$, and we have a derivation of $\Gamma' \vdash_{\mathsf{ps}} y : B$ obtained by removing the last application of the rule (PSE). By induction, this gives a partial globular product cone of apex Γ and with distinguished map χ_y. Since moreover Γ is obtained as the pullback

$$\begin{array}{ccc} \Gamma & \xrightarrow{\chi_x} & D^{\dim x} \\ \downarrow & & \downarrow \sigma \\ \Gamma' & \xrightarrow{\chi_y} & D^{\dim y} \end{array}$$

The commutation of χ_y in the partial globular product diagram shows that we can glue both diagrams in order to exhibit Γ as a globular product. We then chose the distinguished map χ_x which is the map of cone itself, so satisfy the commutation condition tautologically. We can be a bit more explicit in how we compute this diagram, by looking at the dimension tables, and denoting $n = \dim y$

dimension table of Γ'	dimension table of Γ
	if $n = i_m$:
$\begin{pmatrix} i_1 & & \cdots & & i_m \\ & j_1 & \cdots & j_{m-1} & \end{pmatrix}$	$\begin{pmatrix} i_1 & & \cdots & & i_m + 1 \\ & j_1 & \cdots & j_{m-1} & \end{pmatrix}$
	if $n \leq i_m$:
	$\begin{pmatrix} i_1 & & \cdots & i_m & & n+1 \\ & j_1 & \cdots & j_{m-1} & n & \end{pmatrix}$

– If the derivation of $\Gamma \vdash_{\mathsf{ps}} x : A$ ends with an application of (PSD), then by removing this last rule, we get a derivation of a judgment of the form $\Gamma \vdash_{\mathsf{ps}} f : y \to x$, and the induction hypothesis exhibits Γ gives a partial globular product cone whose distinguished map is $\chi_f : \Gamma \to D^{\dim x+1}$. Since x is the target term of f in Γ, we have the equality $\tau\chi_f = \chi_x$, hence replacing the distinguished map χ_f by χ_x yields another partial globular product cone.

□

Computation of globular product. Given a ps-context Γ, applying the previous lemma to a derivation of $\Gamma \vdash_{\mathsf{ps}}$ yields exhibits straightforwardly Γ as a globular product, thus proving the following result.

Proposition 35. *A ps-context Γ is a globular product in the category $\mathcal{S}_{\mathsf{GSeTT}}$.*

A careful observation of the proof shows the computation that construct the globular product diagram associated to the derivation of $\Gamma \vdash_{\mathsf{ps}}$. Indeed, suppose that the derivation of $\Gamma \vdash_{\mathsf{ps}}$ is of the form

$$\Gamma \vdash_{\mathsf{ps}} = (\mathrm{PSS})(\mathrm{PSE})^{k_1}(\mathrm{PSD})^{l_1}(\mathrm{PSE})^{k_2} \cdots (\mathrm{PSE})^{k_m}(\mathrm{PSD})^{l_m}(\mathrm{PS})$$

then necessarily, for dimension reasons, $\sum_{n=1}^{m} k_n = \sum_{n=1}^{m} l_n$. We denote $i_1, \ldots, i_m$ and $j_1, \ldots, j_m$ the families of numbers solutions of the systems of equations

$$\left\{ \begin{array}{lcl} i_1 & = & k_1 \\ i_{n+1} & = & k_{n+1} + j_n \end{array} \right. \qquad j_n \quad = \quad i_n - l_n$$

then the Proposition 35 exhibits Γ as the globular product whose dimension table is

$$\begin{pmatrix} i_1 & & i_2 & & \cdots & & i_m \\ & j_1 & & j_2 & \cdots & j_{m-1} & \end{pmatrix}$$

Globular products are isomorphic to ps-contexts. We prove a partial converse to the previous result, showing the precise nature of the correspondence between ps-contexts and globular products.

Proposition 36. *Any globular product in the category $\mathcal{S}_{\mathsf{GSeTT}}$ is isomorphic to exactly one ps-context (up to renaming).*

Proof. Given a context Δ which is a globular product with dimension table

$$\begin{pmatrix} i_1 & & i_2 & & \cdots & & i_m \\ & j_1 & & j_2 & \cdots & j_{m-1} & \end{pmatrix}$$

It suffices to inverse the previous system of equations and define the numbers $k_1, \dots, k_m$ and $l_1, \dots, l_m$ as solutions of

$$\begin{cases} k_1 = i_1 \\ k_{n+1} = i_{n+1} - j_{n+1} \end{cases} \qquad \begin{cases} l_n = i_n - j_{n+1} \\ l_m = \sum_{n=1}^{m} k_n - \sum_{n=1}^{m-1} l_n \end{cases}$$

to consider the ps-context Γ, whose derivation $\Gamma \vdash_{\mathsf{ps}}$ is obtained as

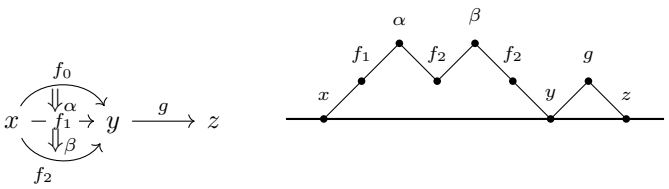
$$\Gamma \vdash_{\mathsf{ps}} = (\text{PSS})(\text{PSE})^{k_1}(\text{PSD})^{l_1}(\text{PSE})^{k_2} \cdots (\text{PSE})^{k_m}(\text{PSD})^{l_m}(\text{PS})$$

which is a valid derivation of a ps-context. By the previous lemma, the context Γ is also a globular product in $\mathcal{S}_{\mathsf{GSeTT}}$ with the same dimension table as Δ, hence Γ and Δ are isomorphic. □

Locally maximal variables. Given a ps-context Γ we define its *locally maximal variables* to be the variables x_n defining the maps $\chi_{x_n} : \Gamma \to D^{i_n}$ in the cone exhibiting Γ as a globular product i.e., the map corresponding to the upper line of the dimension table. The following result follows immediately from the decomposition of Γ as a globular product, but is useful for implementation purposes.

Lemma 37. *Given two substitutions $\Delta \vdash \gamma : \Gamma$ and $\Delta \vdash \gamma' : \Gamma$ where Γ is a ps-context such that for all locally maximal variable x in Γ, $x[\gamma] = x[\gamma']$, necessarily $\gamma = \gamma'$.*

There are various syntactic ways of characterizing locally maximal variables in a ps-context Γ. For instance, they are the variables such that both the variables just before it and just after it are of lower dimension. Equivalently, they are also the variables such that $\Gamma \vdash_{\mathsf{ps}} x : A$ is obtained by an application of (PSE), and followed by an application of (PSD) in the derivation of $\Gamma \vdash_{\mathsf{ps}}$. For instance in the following ps-context



$$(x : \star, y : \star, f_0 : x \to y, f_1 : x \to y, \alpha : f_0 \to f_1, f_2 : x \to y, \beta : f_1 \to f_2, z : \star, g : y \to z)$$

the locally maximal variables are α,β and g, they are the ones at the peaks in the representation as non-decreasing parking functions.

The category $\mathcal{S}_{\mathsf{PS},0}$ We denote $\mathcal{S}_{\mathsf{PS},0}$ the full subcategory of $\mathcal{S}_{\mathsf{GSeTT}}$ whose objects are the contexts Γ such that $\Gamma \vdash_{\mathsf{ps}}$ holds. For most categorical constructions, we could equivalently consider $\mathcal{S}_{\mathsf{PS},0}$ to be the full subcategory whose objects are contexts that are isomorphic to ps-contexts, as by definition these two categories are equivalent, however it is more convenient for our purpose to use only ps-contexts.

Proposition 38. *There is an equivalence of categories*

$$\mathcal{S}_{PS,0} \simeq \Theta_0^{\mathrm{op}}$$

Proof. The nerve functor ν sends limits in $\mathcal{S}_{\mathsf{GSeTT}}$ onto colimits in $\mathbf{FinGSet}^{\mathrm{op}}$, hence it induces a functor $\mathcal{S}_{\mathsf{PS},0} \to \Theta_0^{\mathrm{op}}$ which is fully faithful since ν is fully faithful. Moreover, Proposition 35 shows that this induced functor is essentially surjective. □

In fact $\mathcal{S}_{\mathsf{PS},0}$ is even isomorphic to the category Θ_0, as it is constituted only in globular sums in the category $\mathcal{S}_{\mathsf{GSeTT}}$ that are in normal form.

Subcontexts and epimorphisms. Ps-contexts correspond to pasting schemes, and the latter are equipped with notions of source and target, which are maps σ_X, τ_X in the category Θ_0. We provide a direct characterization of the maps corresponding map in the category $\mathcal{S}_{\mathsf{PS}}$ under the correspondence given by Proposition 38. Instead of providing the substitutions explicitly, we use the fact that σ_X, τ_X are monomorphisms of presheaves and hence epimorphisms in $\mathcal{S}_{\mathsf{GSeTT}}$, and rely on the variable names to present these substitutions in an efficient way. For instance, given the context $\Gamma = (x : \star, y : \star, f : x \to y)$, we have the following correspondence between subcontexts of Γ and epimorphisms in $\mathcal{S}_{\mathsf{GSeTT}}$

subcontext	epimorphism
$(x : \star)$	$\Gamma \vdash \langle a \mapsto x \rangle (a : \star)$
$(y : \star)$	$\Gamma \vdash \langle a \mapsto y \rangle (a : \star)$

In both cases, the subcontext define a substitution from Γ to $(a : \star)$ which identifies $(a : \star)$ as a subcontext of Γ. The substitution is given by the identical variable name in Γ and in its subcontext, in our first example, it is the variable x, and thus the defined substitution is $\langle a \mapsto x \rangle$.

Source and targets. We define for all $i \in \mathbb{N}_{>0}$ the *i-source* of a ps-context Γ induction on the length of Γ, by setting $\partial_i^-(x : \star) = (x : \star)$ and

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

and similarly the *i-target* of Γ is defined by $\partial_i^+(x : \star) = (x : \star)$, and

$$\partial_i^+(\Gamma, y : A, f : x \to y) = \begin{cases} \partial_i^+ \Gamma & \text{if } \dim A \ \geq i \\ \mathrm{drop}(\partial_i^+ \Gamma), y : A & \text{if } \dim A \ = i-1 \\ \partial_i^+ \Gamma, y : A, f : x \to y & \text{otherwise} \end{cases}$$

where $\mathrm{drop}(\Gamma)$ is the context Γ with its last variable removed. One can check by induction on the derivation of the judgment $\Gamma \vdash_{\mathsf{ps}}$ that whenever Γ is a ps-context which is of dimension nonzero, both $\partial_i^- \Gamma$ and $\partial_i^+ \Gamma$ are also ps-contexts. It is straightforward in the case of the i-source, and for the i-target, it relies on the fact that whenever the drop operator is used, immediately afterwards a variable of the same type that the one that was removed is added.

We denote $\partial^-(\Gamma) = \partial^-_{\dim\Gamma-1}\Gamma$ and $\partial^+(\Gamma) = \partial^+_{\dim\Gamma-1}\Gamma$ and call these the *source* and *target* of Γ. Carrying explicitly the computations one can check that whenever a ps-context Γ define the pasting scheme X, the ps-contexts $\partial^-(\Gamma)$ and $\partial^+(\Gamma)$ are isomorphic, and they both define the pasting scheme ∂X. Moreover the "inclusion" substitution that identify a variable in $\partial^-(\Gamma)$ (resp. in $\partial^+(\Gamma)$) with the same variable in Γ corresponds to the map $\sigma_x : \partial X \to X$ (resp. $\tau_x : \partial X \to X$). Note that in the case of the ps-context $(x : \star)$ which is the only ps-context of dimension 0, the source and target are not defined.

2.3.3 The relation $\lhd$

Following Finster and Mimram [33] we introduce a relation $\lhd$ between the variables of a context in GSeTT, by imposing the condition that for all variable f such that $\Gamma \vdash f : x \to y$, we have $x \lhd f \lhd y$. We then consider the transitive closure of this relation.

$\lhd$ in ps-contexts. This relation is particularly well behaved in the case of ps-contexts, as the following lemma shows.

Lemma 39. *The relation $\lhd$ defines a total order on the variables of a ps-context Γ.*

Proof. We prove by induction that whenever the judgment $\Gamma \vdash_{\mathsf{ps}} x : A$ is derivable, then $\lhd$ defines a total order on Γ, and for all y, z such that $x \lhd y \lhd z$, we have $\dim x > \dim y > \dim z$.

- This condition holds for judgment $(x : \star) \vdash_{\mathsf{ps}} x : \star$ obtained by (PSS), as x is then the only variable.
- For a judgment of the form $(\Gamma, y : A, f : x \to y) \vdash_{\mathsf{ps}} f : x \to y$, by induction $\lhd$ defines a total order on the variable of Γ. Note that the variables x, f, y satisfy $x \lhd f \lhd y$, hence for any variable $z \lhd x$, we also have $z \lhd f \lhd y$. Suppose that x is not the greater variable in Γ (otherwise we have already proved the result) and consider the smallest variable z such that $x \lhd z$. Since all the variables greater than x are of dimension greater than x, we necessarily have that $z = \partial^+(x)$. But since x and y have the same type, we also have $z = \partial^+(y)$, and hence $f \lhd y \lhd z$. This proves that $\lhd$ defines a total order on the variables of $(\Gamma, y : A, f : x \to y)$. Moreover, the variables greater than f are y and all the variables greater than x in Γ. For all the variables $x \lhd z \lhd z'$ in Γ, we have $\dim z' < \dim z < \dim y < \dim f$.
- For a judgment $\Gamma \vdash_{\mathsf{ps}} y : A$ obtained by (PSD), the judgment $\Gamma \vdash_{\mathsf{ps}} f : x \underset{A}{\longrightarrow} y$ is necessarily derivable, hence by induction $\lhd$ defines a total order on Γ, and for all variable $y \lhd z \lhd z'$, we have $\dim z' < \dim z < \dim y < \dim f$. the variables

Now considering a derivation of the judgment $\Gamma \vdash$, it necessarily ends with an application of (PS), to a derivation of $\Gamma \vdash_{\mathsf{ps}} x : \star$, hence $\lhd$ defines a total order on the variables of Γ. □

Substitution are order preserving. Consider a substitution $\Delta \vdash \gamma : \Gamma$ in the category $\mathcal{S}_{\mathsf{PS},0}$, then it preserves the typing: For any variable $\Gamma \vdash f : x \to y$, we have $\Delta \vdash f[\gamma] : x[\gamma] \to y[\gamma]$. This shows that for two variables $x \lhd y$ in the context Γ, we have the relation $x[\gamma] \lhd y[\gamma]$ in the context Δ. This proves shows the following

Proposition 40. *For a substitution $\Delta \vdash \gamma : \Gamma$, and two distinct variables x, y in Γ the variables $x[\gamma]$ and $y[\gamma]$ are necessarily distinct in Δ. Moreover, a substitution $\Delta \vdash \gamma : \Gamma$ that uses all the variables of Γ is necessarily an identity.*

Proof. Consider two distinct variables x, y in $\mathrm{Var}(\Gamma)$, then either $x \lhd y$, in which case $x[\gamma] \lhd y[\gamma]$, or $y \lhd x$, in which case $y[\gamma] \lhd x[\gamma]$. In both cases, $x[\gamma]$ and $y[\gamma]$ are distinct. If the substitution $\Delta \vdash \gamma : \Gamma$ uses all the variables of Δ, then necessarily Γ has at least as many variables as Δ, and since there can be no repetition, Δ and Γ have the same number of variables. Then since the substitution γ preserves the typing, its only action is to change the name of the variables of Γ by the names of the variables in Δ. Up to α-equivalence, this is an identity. □

Totally ordered contexts. Finster and Mimram also show [33] this total order on the cell is a characterization of the pasting schemes, and hence every context for which $\lhd$ defines a total order is isomorphic to a ps-context. We state this fact here as it is helps understanding the ps-contexts, but we do not prove it, as we do not use it in any way.

2.4 A type theory for globular weak ω-categories

We now present our type theory to work with weak ω-categories. This theory is originally due to Finster and Mimram [33] and our presentation is very similar. We then study the syntactic category of this theory and its models, an extended version of [16].

2.4.1 Type theory

In order to describe a type theory suitable to work with ω-category, we extend the type theory for globular sets with new term constructors that mirror the lifting that are formally added in the Grothendieck-Maltsiniotis definition of weak ω-categories (see Section 2.1). We add one term constructor for each lift that is added in the construction of Θ_∞. Since there are infinitely many such lifts, we need a way to index these, in order to give a scheme for the induction rules.

First attempt to a description. The most immediate way to obtain this construction, is to introduce a term constructor for each pasting scheme equipped with a pair of admissible arrows. For each pasting scheme θ together with a pair of admissible arrows (f, g) in θ, we introduce a constructor $T_{\theta,f,g}$. The arity of $T_{\theta,f,g}$ is the number of elements of the globular set θ, and the introduction rule is the following

$$\frac{\Gamma \vdash t_1 : A_1 \qquad \cdots \qquad \Gamma \vdash t_n : A_n \qquad \Gamma \vdash B}{\Gamma \vdash T_{\theta,f,g}(t_1, \ldots, t_n) : B}$$

where $t_1, \ldots, t_n$ is a list of terms in Γ whose type "are prescribed by θ", and B is a type is Γ corresponding to the admissible pair of arrows (f, g). This description is intentionally left vague, as this is not our final description, but it gives a good intuition towards it. In fact this approach is very self referencing and a proper definition of it would require to proceed level by level and take some sort of a limiting process. Another problem with this formulation is we do not provide here a computable way of enumerating the term constructors, which prevents for any implementation of this theory.

Encoding this data in a computable way. A first step towards a computable enumeration of the terms is to use the ps-contexts. Indeed we have proved that these are in bijective correspondence with the pasting schemes, hence we can reformulate by introducing a term constructor $T_{\Gamma,f,g}$ for all ps-context Γ, and (f, g) a pair of admissible substitutions in the pasting scheme corresponding to Γ. One can notice that a pair of admissible arrows (f, g) as mentioned previously can be encoded into a type in Γ, and the trick is to give a characterization of those

types in Γ which correspond to pairs of admissible arrows. Recall that there are two situations in which a pair of arrows is admissible: either both are algebraic, or they factor through the source and target of the ps-context into algebraic arrows. Correspondingly, the types characterizing the admissible pairs of arrows are one of the two following form

$$(\mathrm{C}_{\mathsf{op}})\ \ \Gamma \vdash t \underset{A}{\longrightarrow} u \text{ with } \begin{cases} \mathrm{Var}(\partial^-(\Gamma)) = \mathrm{Var}(t) \cup \mathrm{Var}(A) \\ \mathrm{Var}(\partial^+(\Gamma)) = \mathrm{Var}(u) \ \cup \mathrm{Var}(A) \end{cases}$$

$$(\mathrm{C}_{\mathsf{coh}})\ \ \Gamma \vdash t \underset{A}{\longrightarrow} u \text{ with } \begin{cases} \mathrm{Var}(\Gamma) = \mathrm{Var}(t) \cup \mathrm{Var}(A) \\ \mathrm{Var}(\Gamma) = \mathrm{Var}(u) \cup \mathrm{Var}(A) \end{cases}$$

We thus introduce two families of term constructors $\mathsf{op}_{\Gamma,B}$, where Γ is a ps-context and B is a type satisfying $(\mathrm{C}_{\mathsf{op}})$, and $\mathsf{coh}_{\Gamma,B}$ where Γ is a ps-context and B is a type satisfying $(\mathrm{C}_{\mathsf{coh}})$. Moreover a list of terms in a context Δ that are "as prescribed by Γ" can be formulated elegantly as a substitution $\Delta \vdash \gamma : \Gamma$, and the type in Δ corresponding to the admissible pair of substitution and the substitution γ is none other than the type $B[\gamma]$. We show in Section 2.5 that these conditions correspond to the admissibility condition in the Grothendieck-Maltsiniotis definition.

Introduction rules for terms. This lets us reformulate the previous rule, for instance for the family of rules op as

$$\frac{\Delta \vdash \gamma : \Gamma \qquad \Delta \vdash B[\gamma]}{\Delta \vdash \mathsf{op}_{\Gamma,B}[\gamma] : B[\gamma]} \quad \text{for all ps-context } \Gamma \text{ and type } \Gamma \vdash B \text{ satisfying } (\mathrm{C}_{\mathsf{op}})$$

Note that the premise $\Delta \vdash B[\gamma]$ is in fact redundant, as it follows from the fact that substitution transport well-formed types onto well-formed types. Hence, in order to get the rule we actually use, we leave this condition out. Moreover, we chose to write most of the indexing of the term family in the premises of the rules instead of as a side condition, we only keep as a side condition the conditions on variables. This leads to the definitive form of the rule for the family of term constructors op

$$\frac{\Gamma \vdash_{\mathsf{ps}} \qquad \Gamma \vdash A \qquad \Delta \vdash \gamma : \Gamma}{\Delta \vdash \mathsf{op}_{\Gamma,A} : A[\gamma]}(\mathrm{OP}) \qquad \text{where } \Gamma \vdash A \text{ satisfies } (\mathrm{C}_{\mathsf{op}})$$

Written in a similar fashion, the introduction rule for the family of term constructors coh gives

$$\frac{\Gamma \vdash_{\mathsf{ps}} \qquad \Gamma \vdash A \qquad \Delta \vdash \gamma : \Gamma}{\Delta \vdash \mathsf{coh}_{\Gamma,A} : A[\gamma]}(\mathrm{COH}) \qquad \text{where } \Gamma \vdash A \text{ satisfies } (\mathrm{C}_{\mathsf{coh}})$$

We show in Section 3.5 that this rule can be rewritten in a simpler way, by slightly changing the condition $(\mathrm{C}_{\mathsf{coh}})$ in $(\mathrm{C}'_{\mathsf{coh}})$. We say that a type $\Gamma \vdash A$ satisfies $(\mathrm{C}'_{\mathsf{coh}})$ if and only if we have $\mathrm{Var}(\Gamma) = \mathrm{Var}(A)$, and we denote (COH') the introduction rule obtained by changing the condition $(\mathrm{C}_{\mathsf{coh}})$ to $(\mathrm{C}_{\mathsf{coh}})'$ in the rule (COH). We admit in a first time that these two rules give equivalent theories, and postpone the proof to Section 3.5 as it is surprisingly involved and requires some syntactic tools that we have not yet introduced. For now we freely use either of these rules depending on which one is best suited to our needs.

In this formulation, it appears as if Γ and A are arguments of the term constructors op and coh, and from now on, we may treat them as such. But formally they are rather indexes for term constructor families, which allow us to understand this presentation as included in the framework we presented to study a type theory in a cut-free style. We denote CaTT the resulting type theory, which we present completely in Appendix A.2.

2.4.2 Some examples of derivations

We provide some examples of derivations that one may compute in CaTT, using an actual syntax that we have implemented in [15]. A new coherence is introduced with the keyword `coh` and is followed by a name to identify it. Then comes a list of arguments which is the description of a ps-context followed by a column and a type. For instance the following line

```
coh id (x:*) : x -> x
```

defines a coherence called "id", which correspond to the construction $\mathsf{coh}_{(x:\star):x\to x}$. Further references to this coherence in order to produce a term have to include a substitution to the context $(x:\star)$, that is expressed as a list of arguments, for instance one may write `id y`. As we explain in details in Section 3.1, we only specify the term in the substitution that correspond to locally maximal variable of the ps-context, for instance, considering the following coherence

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

one needs to write only `comp f g` instead of `comp x y f z g` when applying this coherence. Other examples of coherences one may define in CaTT include

- left unitality and its inverse

```
coh unitl (x:*)(y:*)(f:x->y) : comp (id x) f -> f

coh unitl- (x:*)(y:*)(f:x->y) : f -> comp (id x) f
```

- right unitality and its inverse

```
coh unitr (x:*)(y:*)(f:x->y) : comp f (id y) -> f

coh unitr- (x:*)(y:*)(f:x->y) : f -> comp f (id y)
```

- associativity and its inverse

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

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

- vertical composition of 2-cells

```
coh vcomp (x:*)(y:*)(f:x->y)(g:x->y)(a:f->g)(h:x->y)(b:g->h) :
f -> h
```

- horizontal composition of 2-cells

```
coh hcomp (x:*)(y:*)(f:x->y)(f':x->y)(a:f->f')
                (z:*)(g:y->z)(g':y->z)(b:g->g') :
comp f g -> comp f' g'
```

– left whiskering

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

– right whiskering

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

We also provide a syntax to define terms in an arbitrary context, and not only coherences. The corresponding keyword is `let` followed with an identifier and a context, and equal and a full definition of the term in terms of previously defined term and coherences. For instance, the following term defines the squaring of an endomorphism

```
let sq (x:*)(f:x->x) = comp f f
```

We refer the reader to Section 3.1 where we provide a more in-depth presentation or our implementation, the corresponding syntax and more examples of expressions that we have implemented and checked.

2.4.3 Syntactic properties

With the tools we have introduced, we can show directly a lot of properties satisfied by the type theory CaTT. Firstly, as for any cut-free type theory, it satisfies Proposition 2, and implements all the structure of a cut-full type theory. Moreover, since every term corresponds only to a unique introduction rule and the theory has no definitional equality, it enjoys the uniqueness of derivations.

Embedding of $\mathcal{S}_{\mathsf{GSeTT}}$. Since all the rules of GSeTT are also valid in CaTT, it follows that any judgment in the theory GSeTT is also derivable in the theory CaTT. On a categorical level, this translates by the existence of a morphism of categories with families $\mathcal{S}_{\mathsf{GSeTT}} \to \mathcal{S}_{\mathsf{CaTT}}$. The uniqueness of derivations then implies that this morphism is faithful. Since $\mathcal{S}_{\mathsf{GSeTT}}$ has a coglobular structured, this morphism transports this structure and endows $\mathcal{S}_{\mathsf{CaTT}}$ with a natural coglobular structure.

Globular products. By definition of the coglobular structure on the category $\mathcal{S}_{\mathsf{CaTT}}$, the embedding $I : \mathcal{S}_{\mathsf{GSeTT}} \to \mathcal{S}_{\mathsf{CaTT}}$ preserves all the finite globular limits, hence it preserves the globular products. In particular, the ps-contexts in the category $\mathcal{S}_{\mathsf{CaTT}}$ are again the globular products. We denote $\mathcal{S}_{\mathsf{PS},\infty}$ the full subcategory of $\mathcal{S}_{\mathsf{CaTT}}$ whose objects are ps-contexts, note that $\mathcal{S}_{\mathsf{PS},\infty}$ has exactly the same objects as $\mathcal{S}_{\mathsf{PS}}$, but its morphisms are not the same. More specifically, there is a strict inclusion $\mathcal{S}_{\mathsf{PS}} \hookrightarrow \mathcal{S}_{\mathsf{PS},\infty}$ which is the identity on the objects.

Familial representability of Ty**.** The embedding $I : \mathcal{S}_{\mathsf{GSeTT}} \to \mathcal{S}_{\mathsf{CaTT}}$ also provides the contexts S^n and D^n in the theory CaTT, which enjoy a similar property that the one they have in the theory GSeTT

Theorem 41. *For every $n \in \mathbb{N}_{-1}$, the canonical map*

$$\begin{array}{rcl} \mathcal{S}_{\mathsf{GSeTT}}(\Gamma, S^n) & \to & \mathrm{Ty}^{\Gamma} \\ \gamma & \mapsto & A_n[\Gamma] \end{array}$$

induces a bijection between the substitutions $\Gamma \to S^n$ and the types in Γ of dimension n.

Corollary 42. *For every $n \in \mathbb{N}$, the canonical map*

$$\begin{array}{rcl} \mathcal{S}_{\mathsf{GSeTT}}(\Gamma, D^n) & \to & \mathrm{Tm}^{\Gamma} \\ \gamma & \mapsto & x_{2n}[\Gamma] \end{array}$$

induces a bijection between the substitutions $\gamma : \Gamma \to D^n$ and the terms of type classified by $\pi \circ \gamma$ in Γ.

Proof. As in the case of GSeTT we can prove these two result by mutual induction, and the proof is essentially the same. Indeed, since the embedding I preserves the globular limits, the sphere contexts enjoy the same inductive definition as successive pullbacks, and the substitution extension $\langle \_, \_ \rangle$ also enjoys the same universal property, as in any category with families. We have also formalized this result in Agda for a more general framework, we refer the reader to Section 4.1 for an introduction to this framework and a discussion of the formal proof. □

Globular contexts, globular substitutions. Note that a context in GSeTT is a fortiori a context in CaTT, let us call *globular context* a context in CaTT which is also a context in GSeTT. Syntactically, a globular context is a context whose types do not contain any cohrences. For instance, any ps-context is a globular context. Similarly, we define a *globular substitution* to be a substitution whose defining terms do not contain any coherences. Note that a globular substitution between two globular contexts is exactly a substitution in GSeTT, but there are also globular substitutions that do not come from substitutions in GSeTT, since their contexts source and target are not globular. For instance, the following substitution is globular

$$\Delta \vdash \langle y \mapsto x, z \mapsto x, g \mapsto f, h \mapsto f, b \mapsto a \rangle : \Gamma$$

$$\text{with } \left\{ \begin{array}{rcl} \Delta & = & (x : \star)(f : x \to x)(a : \mathsf{comp}\ f\ f \to \mathrm{id}_x) \\ \Gamma & = & (y : \star)(z : \star)(g : y \to z)(h : z \to y)(b : \mathsf{comp}\ g\ h \to \mathrm{id}_y) \end{array} \right.$$

The variables of dimension n of a context Γ are exactly the globular substitutions $\Gamma \vdash \gamma : D^n$

Decidability of type checking. In order to use induction on the syntax, we introduce the notion of *depth* of a term, that we define as follows

$$\mathrm{depth}(v) = 0 \qquad \mathrm{depth}(\mathsf{op}_{\Gamma,A}[\gamma]) = 1 + \max_{u \in \gamma} \mathrm{depth}(u)$$
$$\mathrm{depth}(\mathsf{coh}_{\Gamma,A}[\gamma]) = 1 + \max_{u \in \gamma} \mathrm{depth}(u)$$

This notion expresses exactly how many nested coherences are needed to write a given term.

Proposition 43. *The derivability of a judgment is a decidable problem for the type theory CaTT.*

Sketch of the proof. The demonstration is a lot more complicated than it seems, a mere mutual induction does not suffice here as it is not well-formed. To circumvent this problem, we proceed in three steps.

1. We prove by mutual induction that these judgments are valid in a subset of the theory considering only the ps-contexts, that, we prove that for all ps-contexts Γ, Δ, the judgments $\Gamma \vdash A$, $\Gamma \vdash t : A$ and $\Delta \vdash \gamma : \Gamma$ are decidable. The fact that this induction is well-defined is ensured by a careful observation of both the dimension and the depth of the different objects involved, and relies on the fact that any type that uses all the variables in a context is necessarily of dimension at least the dimension of the context.

2. We prove that for all context Δ, the judgments $\Delta \vdash$, $\Delta \vdash A$, $\Delta \vdash t : A$ and $\Delta \vdash \gamma : \Gamma$ are decidable, in the case where Γ is a ps-context. This works by mutual induction, using the previous result, without extra argument.

3. We finally prove that the judgment $\Delta \vdash \gamma : \Gamma$ is decidable without any assumption, using the decidability of the other judgments. This is a simple induction

We have formalized this proof in Agda as well, for the framework we present in Section 4.1 □

2.5 The syntactic category of CaTT

We now study and characterize completely the syntactic category $\mathcal{S}_{\mathsf{CaTT}}$. This is our main contribution to the theoretical understanding of the theory CaTT as well as the key result to characterize its models.

2.5.1 The category of ps-contexts with coherences

The first step towards relating the models of the theory CaTT with the Grothendieck-Maltsiniotis definition of weak ω-categories is to study the category $\mathcal{S}_{\mathsf{PS},\infty}$, which is the full subcategory of $\mathcal{S}_{\mathsf{CaTT}}$ whose objects are exactly the ps-contexts. The precise study of this category is subtle, and requires a good understanding of the syntactic properties satisfied by the expressions that are derivable, along with appropriate tools to study these properties. A full presentation of these tools would take us too far from the goal of understanding the syntactic category and the models of the theory CaTT. For this reason, we state here the result that we are using, and refer the reader to Section 3.5 where we introduce all the required tools and prove the theorem

Theorem 44. *There is an equivalence of categories* $\mathcal{S}_{PS,\infty} \simeq \Theta_\infty^{\mathrm{op}}$

We can still give a general idea of the proof: We start by introducing the notion of *coherence depth* that exhibits the category $\mathcal{S}_{\mathsf{PS},\infty}$ as the colimit of an iterative sequence

$$\mathcal{S}_{\mathsf{PS},\infty} = \mathrm{colim}\,(\mathcal{S}_{\mathsf{PS},0} \to \mathcal{S}_{\mathsf{PS},1} \to \mathcal{S}_{\mathsf{PS},2} \to \dots)$$

and we show that each of the category $\mathcal{S}_{\mathsf{PS},i}$ is equivalent to Θ_i^{op} since they satisfy the dual universal property: The category $\mathcal{S}_{\mathsf{PS},i+1}$ is obtained from the category $\mathcal{S}_{\mathsf{PS},i}$ by formally adding the lifts to the appropriate coadmissible pairs. In order to prove this, we characterize the types $\Gamma \vdash_{\mathsf{ps}} A$ satisfying (C_{op}) or ($\mathrm{C}_{\mathsf{coh}}$) as classifying the coadmissible pairs, and the term introduction rules as providing a lift for the corresponding coadmissible pair.

2.5.2 Kan extensions and density

In the following, we make a heavy use of right Kan extensions and related constructions like density. The Kan extensions that we consider are sometimes called "pointwise", since all our Kan extensions are pointwise, we leave this adjective implicit. Moreover, we only use right Kan extensions and specific results about those, and thus only give a narrow exposition of the tools we need to [53, 57] for a more complete presentation of these notions.

Nerve associated to a functor. Given a functor $F : \mathcal{C} \to \mathcal{D}$, we define its associated *nerve functor* (or associated *Yoneda embedding*)

$$\begin{array}{rccc} \nu_F & : & \mathcal{D}^{\mathrm{op}} & \to & \widehat{\mathcal{C}} \\ & & d & \mapsto & \mathcal{D}(d, F\_) \end{array}$$

To understand the significance of this functor, it is useful to think of the particular case where $\mathcal{D}$ as the opposite of the presheaf category over $\mathcal{C}$, i.e., $\mathcal{D} = \widehat{\mathcal{C}}^{\mathrm{op}}$ and F is the Yoneda embedding. In this case $\mathcal{D}(d, F\_)$ rewrites as $\mathcal{D}^{\mathrm{op}}(F\_, d)$, which by the Yoneda lemma is exactly d. Hence in this particular case, the nerve associated to F is the identity. We can also consider the slightly more complicated case where $\mathcal{D}$ is the opposite of a presheaf category $\widehat{\mathcal{A}}$, and F defines the inclusion of a full subcategory $\mathcal{C}$ into $\widehat{\mathcal{A}}$. Then every object c in $\mathcal{C}$ can interpreted as a presheaf in $\widehat{\mathcal{A}}$ and thus can be decomposed as a colimit of representable objects in $\mathcal{A}$, we denote this colimit $c = \mathrm{colim}_a \mathrm{Y}(a)$. The Yoneda lemma and the continuity of the hom-functor then shows

$$\begin{aligned}\widehat{\mathcal{A}}(\mathrm{colim}_a \mathrm{Y}(a), d) &= \mathrm{colim}_a \widehat{\mathcal{A}}(a, d) \\ &= \mathrm{colim}_a d(a)\end{aligned}$$

hence $\nu_F(d)$ characterizes the colimits of the elements of the presheaf d as prescribed by the category $\mathcal{C}$. Equivalently, it characterizes the limits of the co-elements of d in the category $\mathcal{D}$. In the more general case, the nerve functors $\nu_F(d)$ characterizes the elements in the image of F that sit above d in the category $\mathcal{D}$.

Comma category. For a functor $F : \mathcal{C} \to \mathcal{D}$ as above and an object d of the category $\mathcal{D}$, we define the *comma category* of d over F, denoted $(d \downarrow F)$ to be the category whose objects are the pairs (c, f) where c is an object of $\mathcal{C}$ and $f : d \to Fc$ is an arrow in $\mathcal{D}$. The set of morphisms $(d \downarrow F)((c, f), (c', f'))$ is by definition the set of arrows g in $\mathcal{C}$ such that $(Fg)f = f'$

$$\begin{array}{ccc} d & \xrightarrow{f} & Fc \\ & \searrow^{f'} & \downarrow{\scriptstyle Fg} \\ & & Fc' \end{array}$$

In the case of F is the inclusion of a subcategory, we denote $(d \downarrow \mathcal{C})$ instead of $(d \downarrow F)$, leaving the inclusion functor implicit. There is a canonical forgetful functor $\Pi_d : (d \downarrow F) \to \mathcal{C}$ sending a pair (c, f) onto the object c and a morphism g onto g itself. This induce a *canonical diagram* associated to every object d in $\mathcal{D}$, defined as the composite

$$(d \downarrow F) \xrightarrow{\Pi_d} \mathcal{C} \xrightarrow{F} \mathcal{D}$$

Intuitively, this diagram witnesses the objects of $\mathcal{C}$ that, seen through F sit above d in the category $\mathcal{D}$.

Nerve and comma category. The intuitions behind the nerve functor $\nu_F(d)$ and behind the canonical diagram over d are closely related: Both quantify the objects of $\mathcal{C}$ which seen through F sit above d in $\mathcal{D}$. Indeed, these notions carry similar information and are closely related. We relate them by considering a pair of functor

$$\begin{array}{ccc} \mathcal{D} & & \mathcal{E} \\ {\scriptstyle F}\uparrow & \nearrow_{G} & \\ \mathcal{C} & & \end{array}$$

and constructing the diagram

$$(d \downarrow F) \xrightarrow{\Pi_d} \mathcal{C} \xrightarrow{G} \mathcal{E}$$

Lemma 45. *Under the hypotheses as above, the category of cones of apex e in $\mathcal{E}$ over the diagram $G \circ \Pi_d$ is equivalent to the category of natural transformations $\widehat{\mathcal{C}}(\nu_G(e), \nu_F(d))$*

Proof. The proof of this result can be found in [53, Lemma on p. 245] and [57, Lemma 6.3.8 on p. 202]. □

In the case where $\mathcal{E} = \mathcal{D}$ and $G = F$, and choosing the object d as the apex of the cone, this gives an isomorphism between the cones of apex d over the canonical diagram of d and the natural transformations $\widehat{\mathcal{C}}(\nu_F(d), \nu_F(d))$. The identity natural transformation $\mathrm{id}_{\nu_F(d)}$ then defines a cone of apex d over the canonical diagram of d, that we call the *canonical cone* of d.

Right Kan extensions. The notion of Kan extension is very general and can be characterized in different ways. We here give the definition that is best suited to the situation we want to describe. This is not the usual definition of the Kan extensions, but rather an equivalent characterization (see [53, Corollary 4, p. 245]). Consider two functors $\mathcal{D} \xleftarrow[F]{} \mathcal{C} \xrightarrow[G]{} \mathcal{E}$ as above, and a functor $R : \mathcal{D} \to \mathcal{E}$ together with a natural transformation $\epsilon : RF \Rightarrow G$. Then a morphism $f : e \to Rd$ in $\mathcal{E}$ induces a natural transformation $\nu_F(d) \Rightarrow \nu_G(e)$, defined on any object c of $\mathcal{C}$ as the composite

$$\mathcal{D}(d, Fc) \xrightarrow{R} \mathcal{E}(Rd, RFc) \xrightarrow{\mathcal{E}(f, \epsilon_c)} \mathcal{E}(e, Gc)$$

Definition 46. With the notations as above, (R, ϵ) is the *(pointwise) right Kan extension* of G along F if for all objects d of $\mathcal{D}$ and e of $\mathcal{E}$, the map $\mathcal{E}(e, Rd) \to \widehat{\mathcal{C}}(\nu_F(d), \nu_G(e))$ described above is a bijection. When it is the case, we denote $R = \mathrm{Ran}_F(G)$.

Another characterization of right Kan extension [53, 57] is the following: R is the right Kan extension of G along F, if and only if for every object d of $\mathcal{D}$, we can compute Rd as the colimit

$$Rd = \lim \Big((d \downarrow F) \xrightarrow{\Pi_d} \mathcal{C} \xrightarrow{G} \mathcal{E} \Big)$$

Indeed, Lemma 45 characterizes the natural transformations between the nerves as the cones, the canonical cone is a cone in $\mathcal{D}$ that factors through F, so G defines the image of the canonical cone over d in $\mathcal{E}$. The definition of Kan extension states exactly the universal property for this cone to be a limiting cone. Intuitively, the right Kan extension of an object d is the object of $\mathcal{E}$ that gives the best over-approximation for d in $\mathcal{E}$ using only objects from $\mathcal{C}$.

Codensity. A particularly important case of Kan extension is given when the functor R is the identity functor and the functors G and F are equal, i.e., when $\mathrm{id}_{\mathcal{D}} = \mathrm{Ran}_F(F)$. When this is satisfied, we say that the functor F is *codense*, and the previous characterization of Kan extensions as limits shows that in this case, every object d in the category $\mathcal{D}$ is obtained as the limit

$$d = \lim \Big((d \downarrow F) \xrightarrow{\Pi_d} \mathcal{C} \xrightarrow{F} \mathcal{D} \Big)$$

We then call this limit the *canonical limit* of the object d. The intuition is that the functor F is codense if for every object d of the category $\mathcal{D}$ the best over-approximation of d using only objects for the image of F is d itself. Equivalently, the functor F is codense if every object of $\mathcal{D}$ is canonically a limit of objects in the image of F.

Lemma 47. *For a codense functor $F : \mathcal{C} \to \mathcal{D}$ together with a functor $G : \mathcal{D} \to \mathcal{E}$. Then G preserves the canonical limits if and only if it is the right Kan extension*

$$G = \mathrm{Ran}_F(GF)$$

Proof. This is exactly the second characterization that we have given for the right Kan extension. □

Free completions. Kan extensions behave nicely with fully faithful functors, as shown by the following result (c.f. [57, Corollary 6.3.9, p.203] or [53, Corollary 3, p.239] for a proof).

Proposition 48. *If F is fully faithful and $R = \mathrm{Ran}_F(G)$, then RF is naturally isomorphic to G.*

Using this proposition, we can show that a functor $F : \mathcal{C} \to \mathcal{D}$ that is both codense and fully faithful realizes the free completion of $\mathcal{C}$ by finite limits.

Theorem 49. *For any category $\mathcal{E}$ together with a functor $G : \mathcal{C} \to \mathcal{E}$ such that the G-images of the canonical diagrams of all the objects of $\mathcal{D}$ have a limit in $\mathcal{E}$, there exists an essentially unique functor $\tilde{G} : \mathcal{D} \to \mathcal{E}$ which preserves canonical limits and such that $\tilde{G}F = G$.*

Proof. If such a functor $\tilde{G}$ exists, since it preserves the canonical limits, it has to be defined as the Kan extension $\tilde{G} = \mathrm{Ran}_F(G)$, hence it is essentially unique. Conversely, our assumption about the existence of the limits shows the existence of the Kan extension $\mathrm{Ran}_F(G)$, then Lemma 47 shows that it preserves canonical limits and Proposition 48 shows that it satisfies $\mathrm{Ran}_F(G)F = G$. □

We apply this particular in the particular case where $\mathcal{E}$ is the category **Set**, which is complete. In this case, the condition of the existence of limits is automatically satisfied, and this theorem can be reformulated as follows

Corollary 50. *There is an equivalence of categories between the category of functors $[\mathcal{C}, \mathbf{Set}]$ and the functors $\mathcal{D} \to \mathbf{Set}$ preserving the canonical limits, that we denote $[\mathcal{C}, \mathbf{Set}]_{\mathrm{canlim}}$.*

This result can be seen as a refinement (and dual version) of the fact that the presheaf category is the free cocompletion of a category: Instead of completing with all the limits, we only complete by a sub-class of limits.

Free completions limit-preserving. Suppose that we have a functor $F\mathcal{C} \to \mathcal{D}$ which is codense and fully faithful, and that moreover $\mathcal{C}$ has a class of limits $\mathcal{L}$, and that the functor F preserves those limits. Then a functor $\tilde{G} : \mathcal{D} \to \mathbf{Set}$ preserves the limits $\mathcal{L}$ if and only if $\tilde{G} \circ F$ preserves the limits $\mathcal{L}$. This shows that the preservation of limits in $\mathcal{L}$ can be required in the previous corollary, to get the following result

Corollary 51. *There is an equivalence of categories between the category of functors $[\mathcal{C}, \mathbf{Set}]_{\mathcal{L}}$ preserving the limits in $\mathcal{L}$ and the category of functors $[\mathcal{C}, \mathbf{Set}]_{\mathcal{L},\mathrm{canlim}}$ preserving the limits in $\mathcal{L}$ and the canonical limits.*

2.5.3 A characterization of substitutions

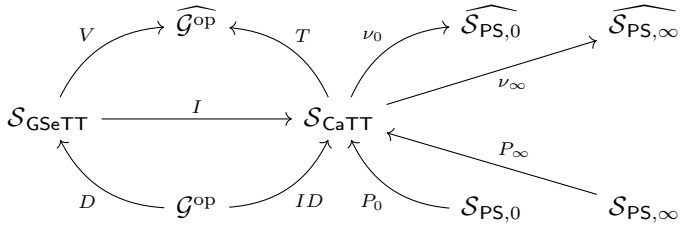
In the theory CaTT as in any contextual category, a substitution is completely determined by its action on variables of a context, our goal is now to study the converse problem: Given a function sending variables of a context to terms of another contexts, does there exist a substitution whose action on variables is given by the specified function? Since substitution have to respect typing, the action on type cannot be completely free, it has to satisfy some conditions. For instance, there is no substitution $(x : \star, f : x \to x) \vdash \gamma : (x : \star)$ that sends the term $x : \star \vdash \mathsf{coh}_{(x:\star, x\to x)}[\langle x \mapsto x\rangle] : x \to x$ onto f, since by definition of the action of a substitution,

we have $(\mathsf{coh}_{(x:\star),x\to x}[\langle x \mapsto x\rangle])[\gamma] = \mathsf{coh}_{(x:\star),x\to x}[\langle x \mapsto x[\gamma]\rangle]$. Our objective is now to describe all the possible ways of associating a term of a context Δ to each term of a context Γ while ensuring that it corresponds to a substitution $\Delta \vdash \gamma : \Gamma$.

The various nerve functors. In the case of the theory CaTT, the categories and functors that we consider and the associated nerves are the following

- The functor $I \ : \mathcal{S}_{\mathsf{GSeTT}} \to \mathcal{S}_{\mathsf{CaTT}}$, which is the inclusion allowing to interpret a context with only variables (i.e., a globular context) as a context in the theory CaTT.
- The functor $D : \mathcal{G}^{\mathrm{op}} \to \mathcal{S}_{Glob}$, which is given by the characterization of the disk contexts in the theory GSeTT. We denote its nerve functor $V = \nu_D$ (this is the functor that we had simply denoted ν in Section 2.2). Corollary 23 shows that an element of $(V\Gamma)_n$ is exactly a term of Γ of dimension n in the theory GSeTT, that is a variable in Γ, thus we call $V\Gamma$ the *presheaf of variables* of Γ.
- The functor $ID : \mathcal{G}^{\mathrm{op}} \to \mathcal{S}_{\mathsf{CaTT}}$ which is given by the disks contexts in the theory CaTT. We denote its associated nerve functor $T = \nu_{ID}$, and Corollary 42 shows that an element of $(T\Gamma)_n$ is exactly a term in Γ of dimension n. We call $T\Gamma$ the *presheaf of terms* of Γ.
- The functor $P_0 : \mathcal{S}_{\mathsf{PS},0} \to \mathsf{CaTT}$, which is the inclusion allowing to consider ps-contexts as a particular contexts of CaTT, and the globular substitutions between those as particular cases of substitutions of CaTT. We denote the nerve associated to this functor $\nu_0 = \nu_{P_0}$.
- The functor $P_\infty : \mathcal{S}_{\mathsf{PS},\infty} \to \mathsf{CaTT}$ which is the inclusion allowing to consider ps-contexts as contexts in CaTT, and substitutions between those as substitutions in CaTT. We denote the nerve associated to this functor $\nu_\infty = \nu_{P_\infty}$.

The situation can be visualized on the following diagram.



In a globular contexts the source and target of a variable is again a variable which is not the case in arbitrary contexts, so V is only definable as globular set on globular contexts, for a general context, we can only define a set of variables. Consider for instance the context $\Gamma =$ `(x : *, a : id x -> id x)`, then the set of variables is $\{\mathtt{x}, \mathtt{a}\}$ but we cannot see it as a globular set: the source of `a` is the term `id x` which is not a variable.

Action on terms. We use these nerve functors to encode an action, which for every term of a context associates a term in another context. Indeed, a natural transformation $T\Gamma \Rightarrow T\Delta$ associates to each term of Γ a term of Δ of the same dimension, while also respecting the source and target. This in particular implies that it preserves the typing, which is required if we want the action to define a substitution. We start by treating the case where we describe actions of substitutions $\Delta \vdash \gamma : \Gamma$ with Γ a globular context, as in this case, it suffices to specify the image of all the variables of Γ, encoded as a natural transformation $V\Gamma \Rightarrow T\Delta$. In the general case

however, we cannot do this, since we do not have access to a presheaf of variables. For this reason, we consider natural transformations $T\Gamma \Rightarrow T\Delta$ instead, but we have to add an extra condition that we call algebraicity, to make sure that those indeed define a substitution. The nerves ν_0 and ν_∞ play a role in defining and understanding this algebraicity condition.

Substitutions to a globular context. We first give a characterization of all the substitutions whose target is a globular context. Since the set of variables of such a context is naturally a globular sets, these substitutions are easier to characterize.

Lemma 52. *Let $\Delta \vdash$ be a context and $\Gamma \vdash$ a globular context. Then there is an isomorphism*

$$\mathcal{S}_{\mathsf{CaTT}}(\Delta, I(\Gamma)) \simeq \widehat{\mathcal{G}^{\mathrm{op}}}(V\Gamma, T\Delta)$$

Proof. To any substitution $\Delta \vdash \gamma : \Gamma$, we associate a natural transformation $\overline{\gamma} : V\Gamma \Rightarrow T\Delta$ defined as follows : For any dimension n, we define

$$\begin{array}{rcccc} \overline{\gamma}_n & : & (V\Gamma)_n & \to & (T\Delta)_n \\ & & x & \mapsto & x[\gamma] \end{array}$$

In order to check that this defines a natural transformation, it suffices to check that it is compatible with the source and target inclusion of disks, i.e., that it respects the types. If x, y, z are variables in Δ such that $\Delta \vdash x : y \to z$, this amounts to saying that $\Gamma \vdash x[\gamma] : y[\gamma] \to z[\gamma]$. This comes from Proposition 3, and hence $\overline{\gamma}$ is a natural transformation. This allows us to define the map

$$\overline{} \quad : \quad \mathcal{S}_{\mathsf{CaTT}}(\Delta, \Gamma) \quad \to \quad \mathrm{Nat}(V\Gamma, T\Delta)$$

Lemma 6 states exactly that this map is injective, and we prove that it is surjective, by constructing for any natural transformation $\eta : V\Gamma \Rightarrow T\Delta$, a substitution γ such that $\overline{\gamma} = \eta$. We construct this substitution by induction on Γ.

- For the empty context $\varnothing$, we pose $\gamma = \langle\rangle$ and since $V\Gamma$ is the empty presheaf, it is initial and hence $\overline{\langle\rangle} = \eta$.
- For a context of the form $\Gamma = (\Gamma', x : A)$, the natural transformation $\eta : V\Gamma \Rightarrow T\Delta$ induces a natural transformation $\eta' : V\Gamma' \Rightarrow T\Delta$. Let $\Delta \vdash \gamma' : \gamma'$ be the preimage substitution of η', define γ to be the substitution $\Delta \vdash \langle \gamma', x \mapsto \eta(x) \rangle : \Gamma$. Since both judgments $\Delta \vdash \gamma' : \Gamma'$ and $(\Gamma', x : A) \vdash$ are derivable, and the naturality of η implies that $\Delta \vdash t : A[\gamma']$ is also derivable, an application of the rule (SE) shows that this definition yields a valid substitution $\Delta \vdash \gamma : \Gamma$. We now check that $\overline{\gamma} = \eta$. For a variable y in Δ, either y is a variable of Δ', and then $y[\sigma] = y[\sigma'] = \eta'(y) = \eta(y)$, or y is the variable x and then $x[\sigma] = \eta(x)$

□

This result shows exactly that I is in fact the pointwise right Kan extension

$$I = \mathrm{Ran}_D(ID)$$

Explicitly, a natural transformation $\eta : V\Gamma \Rightarrow T\Delta$ is the data of, for each variable x of Γ, a term $\eta(x)$ in Δ of the same dimension as x, such that if $\Gamma \vdash x : y \to z$, then $\Delta \vdash \eta(x) : \eta(y) \to \eta(z)$. By Lemma 45 such a natural transformation is equivalent to a cone of apex Δ in the category $\mathcal{S}_{\mathsf{CaTT}}$ over the diagram $(\Gamma \downarrow V) \to \mathcal{G}^{\mathrm{op}} \hookrightarrow \mathcal{S}_{\mathsf{CaTT}}$. Lemma 52 shows that this cone is colimiting, which is an equivalent characterization of the Kan extension.

Algebraic natural transformations $T\Gamma \Rightarrow T\Delta$. We now study the more general case of substitutions whose targets are generic contexts, and not necessarily globular ones. Note that the same exact result does not make sense anymore, since $V\Gamma$ is not defined in general. For instance, for the context

$$\Gamma = \texttt{(x : *) (f : id x -> id x)}$$

we would like to define $(V\Gamma)_2 = \{\texttt{f}\}$, but then its source and target are the term `id x` which is not a variable, and hence not an element of $(V\Gamma)_1$. In order to express the condition, we change the point of view, and consider natural transformations $T\Gamma \Rightarrow T\Delta$ instead of $V\Gamma \Rightarrow T\Delta$, to establish a correspondence between the substitutions $\Delta \vdash \gamma : \Gamma$. However, by switching from the presheaf of variables to the presheaf of terms, we have added too much freedom, and there are now such natural transformations that are ill-defined substitutions; for instance, consider the contexts

$$\Delta = \texttt{(x:*)(f:x->x)} \qquad\qquad \Gamma = \texttt{(x:*)}$$

together with a natural transformation $\eta : T\Gamma \Rightarrow T\Delta$ such that $\eta(\texttt{id}\ x) = \texttt{f}$. This can never be the action of a substitution, since, $(\texttt{id}\ x)[\gamma] = \texttt{id}\ (x[\gamma])$. To account for this, we express a compatibility condition of the natural transformations with the term constructors `op` and `coh`, and show that it is the only obstruction to correspond to a well-defined substitution. We call this compatibility condition *algebraicity*, and is defined inductively on the depth of the substitution we consider.

We first define a function $\_^* : \widehat{\mathcal{G}^{\mathrm{op}}}(T\Gamma, T\Delta) \to \widehat{\mathcal{S}_{\mathsf{PS},0}}(\nu_0\Gamma, \nu_0\Delta)$. Suppose given the natural transformation $\eta : T\Gamma \Rightarrow T\Delta$, and consider a ps-context Ξ, together with a substitution $\Gamma \vdash \xi : \Xi$ (i.e., ξ is an element of $(\nu_0\Gamma)_\Xi$). By Lemma 52, ξ induces a natural transformation $\overline{\xi} : V\Xi \Rightarrow T\Gamma$. By vertically composing this natural transformation with η, we get a natural transformation $\eta \circ \overline{\xi} : V\Xi \Rightarrow T\Delta$. We define the substitution $\Delta \vdash \eta^*(\xi) : \Xi$ to be the substitution associated to this natural transformation by Lemma 52, it is thus characterized by the equation $\overline{\eta^*(\xi)} = \eta \circ \overline{\xi}$. The following lemma ensures that this definition is natural in η.

Lemma 53. *For any natural transformation $\eta \in \widehat{\mathcal{G}^{\mathrm{op}}}(T\Gamma, T\Delta)$, the family of functions η^* defines a natural transformation in $\widehat{\mathcal{S}_{\mathsf{PS},0}}(\nu_0\Gamma, \nu_0\Delta)$*

Proof. Consider a substitution $\Xi \vdash \psi : \Psi$ in the category $\mathcal{S}_{ps,0}$, we want to show that the following square commutes

$$\begin{array}{ccc} (\nu_0\Gamma)_\Xi & \xrightarrow{\eta^*} & (\nu_0\Delta)_\Xi \\ {\scriptstyle \psi\circ\_}\downarrow & & \downarrow{\scriptstyle \psi\circ\_} \\ (\nu_0\Gamma)_\Psi & \xrightarrow[\eta^*]{} & (\nu_0\Delta)_\Psi \end{array}$$

Consider a substitution $\Gamma \vdash \xi : \Xi$ and a variable x in Ψ, on one hand we have

$$\begin{aligned} \overline{\eta^*(\psi \circ \xi)}(x) &= \eta \circ \overline{(\psi \circ \xi)}(x) \\ &= \eta(x[\psi \circ \xi]) \\ &= \eta(x[\psi][\xi]) \\ &= \eta \circ \overline{\xi} \circ \overline{\psi}(x) \end{aligned}$$

where the first line is by definition of η^* and the last line holds since $x[\psi]$ is a variable in Ξ (since

ψ is a map in $\mathcal{S}_{\mathsf{PS},0}$), which is a ps-context hence a globular context. And on the other hand,

$$\begin{aligned}\overline{\psi \circ \eta^*(\xi)}(x) &= x[\psi \circ \eta^*(\xi)] \\ &= x[\psi][\eta^*(\xi)] \\ &= \overline{\eta^*(\xi)} \circ \overline{\psi}(x) \\ &= \eta \circ \overline{\xi} \circ \overline{\psi}(x)\end{aligned}$$

where the third line holds since $x[\psi]$ is a variable of Ξ, and the last line holds by definition of η^*. This proves that for all x in Ψ we have $\overline{\eta^*(\psi \circ \xi)}(x) = \overline{\psi \circ \eta^*(\xi)}(x)$, and hence $\eta^*(\psi \circ \xi) = \psi \circ \eta^*(\xi)$ □

We can understand this result in terms of canonical diagrams: Lemma 45 shows that a natural transformation $\nu_0(\Gamma) \Rightarrow \nu_0(\Delta)$ is equivalent to a cone of apex Γ over the diagram $(\Gamma \downarrow \mathcal{S}_{0,\infty}) \to \mathcal{S}_{0,\infty} \to \mathcal{S}_{\mathsf{CaTT}}$. Lemma 45 also shows that a natural transformation $T\Gamma \Rightarrow T\Delta$ is equivalent to a cone of apex Δ over the diagram $(\Gamma \downarrow \mathcal{G}^{\mathrm{op}}) \to \mathcal{G}^{\mathrm{op}} \to \mathcal{S}_{\mathsf{CaTT}}$. The first diagram has the ps-contexts as objects, whereas the second diagram has only the disks. But the ps-contexts are themselves globular products, that are limits of disks. So Lemma 53 can be understood as a procedure to develop the globular products in the former diagram to obtain the latter.

Definition 54. A natural transformation $\eta : T\Gamma \Rightarrow T\Delta$ is said to be *algebraic* if for every ps-context Ξ together with a term t derivable in Ξ and a substitution $\Gamma \vdash \xi : \Xi$, one has the equality

$$\eta(t[\xi]) = t[\eta^*(\xi)]$$

The following lemma, aside from being technically relevant shows that restricting ourselves to algebraic natural transformation removes the additional unwanted freedom that we added when we switched from the presheaf of variables to the presheaf of terms. Indeed, it gives an equivalent of Lemma 6 for algebraic natural transformations.

Lemma 55. *Two algebraic natural transformations $\eta, \eta' : T\Gamma \Rightarrow T\Delta$ are equal if and only if they coincide on all variables of Γ.*

Proof. Suppose that $\eta, \eta' : T\Gamma \Rightarrow T\Delta$ are two algebraic natural transformations that coincide on all variables of Γ. We prove that for any term t derivable in Γ, $\eta(t) = \eta'(t)$, by induction on the depth of t.

- For a variable x, we have by hypothesis $\eta(x) = \eta'(x)$.
- For a term t of the form $\Gamma \vdash \mathsf{op}_{\Xi,A}[\xi] : A[\xi]$, since η and η' are algebraic we have $\eta(t) = \mathsf{op}_{\Xi,A}[\eta^*(\xi)]$ and $\eta'(t) = \mathsf{op}_{\Xi,A}[\eta'^*(\xi)]$, so it suffices to prove that $\eta^*(\xi) = \eta'^*(\xi)$. Moreover, for every variable x in Ξ, the term $x[\xi]$ is of depth strictly less than t hence by induction we have $\eta(x[\xi]) = \eta'(x[\delta])$, which translates to $\overline{\eta^*(\xi)}(x) = \overline{\eta'^*(\xi)}(x)$. Since this holds for all x, it shows that we have $\eta^*(\xi) = \eta'^*(\xi)$. □

Algebraic natural transformations and nerves. We have defined algebraic natural transformations as morphisms in the category of globular sets, between two presheaves of terms. It turns out that these correspond exactly to natural transformations in the category of presheaves over $\mathcal{S}_{\mathsf{PS},\infty}$ between two nerves.

Lemma 56. *If a natural transformation $\eta \in \widehat{\mathcal{G}^{\mathrm{op}}}(T\Gamma, T\Delta)$ is algebraic, then η^* defines a natural transformation in $\widehat{\mathcal{S}_{\mathsf{PS},\infty}}(\nu\Gamma, \nu\Delta)$.*

Proof. We have already proved in Lemma 53 that η^* defines a natural transformation $\nu_0\Gamma \Rightarrow \nu_0\Delta$ and since for any ps-context Ξ, we have the equality of sets $(\nu_0\Gamma)_\Xi = (\nu\Gamma)_\Xi$, η^*_Ξ defines a function $(\nu\Gamma)_\Xi \to (\nu\Delta)_\Xi$. It suffices to check that this family is a natural transformation, that is for all substitution $\Xi \vdash \psi : \Psi$ between two ps-contexts, the following square commutes

$$\begin{array}{ccc} (\nu\Gamma)_\Xi & \xrightarrow{\eta^*} & (\nu\Delta)_\Xi \\ {\scriptstyle \psi^*}\downarrow & & \downarrow{\scriptstyle \psi^*} \\ (\nu\Gamma)_\Psi & \xrightarrow[\eta^*]{} & (\nu\Delta)_\Psi \end{array}$$

Consider a substitution $\Gamma \vdash \xi : \Xi$ in $\mathcal{S}_{\mathsf{PS},\infty}$, then for any variable $x \in V\Xi$ we have on one hand

$$\begin{aligned} \overline{\psi \circ \eta^*(\xi)}(x) &= x[\psi \circ \eta^*(\xi)] \\ &= x[\psi][\eta^*(\xi)] \end{aligned}$$

and on the other hand

$$\begin{aligned} \overline{\eta^*(\psi \circ \xi)}(x) &= \eta \circ (\overline{\psi \circ \xi})(x) \\ &= \eta(x[\psi \circ \xi]) \\ &= \eta(x[\psi][\xi]) \end{aligned}$$

Since η is algebraic, this shows that $\overline{\psi \circ \eta^*(\xi)}(x) = \overline{\eta^*(\psi \circ \xi)}(x)$ and this being true for all the variables $x \in V\Xi$, it shows the equality $\theta \circ \eta^*(\delta) = \eta^*(\theta \circ \delta)$.

□

This proof sheds a new light on the algebraicity condition: It is exactly the condition required for a natural transformation between the presheaves of terms to lift to a transformation between the nerves which is still natural. Once again this result can be understood in terms of canonical diagrams: By Lemma 45, a natural transformation $T\Gamma \Rightarrow T\Delta$ is equivalent to a cone of apex Δ over the diagram $(\Gamma \downarrow \mathcal{G}^{\mathrm{op}}) \to \mathcal{G}^{\mathrm{op}} \to \mathcal{S}_{\mathsf{CaTT}}$, and a natural transformation $\nu(\Delta) \Rightarrow \nu(\Gamma)$ is equivalent to a cone of apex Δ over the diagram $(\Gamma \downarrow \mathcal{S}_{\mathsf{PS},\infty}) \to \mathcal{S}_{\mathsf{PS},\infty} \to \mathcal{S}_{\mathsf{CaTT}}$. The algebraicity condition provides a condition in which a cone over the disks can be factored as a cone over the ps-contexts.

Lemma 57. *There is a natural isomorphism between the set of algebraic natural transformations $T\Gamma \Rightarrow T\Delta$ and the set of natural transformations $\nu(\Gamma) \Rightarrow \nu(\Delta)$*

Proof. We have already proved in Lemma 56 that an algebraic natural transformation $\eta T\Gamma \Rightarrow T\Delta$ defines a natural transformation $\eta^* : \nu(\Gamma) \Rightarrow \nu(\Delta)$. Conversely, given a natural transformation $\eta : \nu(\Gamma) \Rightarrow \nu(\Delta)$, it restricts along the inclusion functor $\mathcal{G} \hookrightarrow \mathcal{S}_{\mathsf{PS},\infty}$ to a natural transformation $\eta' : T\Gamma \Rightarrow T\Delta$. We first show that η' is an algebraic natural transformation: Given a ps-context Ξ and a term t in Ξ, together with a substitution $\Gamma \vdash \xi : \Xi$, the naturality of η shows the commutation of the following square

$$\begin{array}{ccc} (\nu\Gamma)_\Xi & \xrightarrow{\eta} & (\nu\Delta)_{D^n} \\ {\scriptstyle t[\_]}\downarrow & & \downarrow{\scriptstyle t[\_]} \\ (\nu\Gamma)_\Xi & \xrightarrow[\eta]{} & (\nu\Delta)_{D^n} \end{array}$$

which gives the equation $t[\eta(\xi)] = \eta(t[\xi])$. Since $t[\xi]$ is a term, $\eta'(t[\xi])$ is well defined as $\eta'(t[\xi]) = \eta(t[\xi])$, so in order to show that η' is algebraic, it suffices to show that $t[\eta(\xi)] = t[\eta'^*(\xi)]$. We show that $\eta'^*(\xi) = \eta(\xi)$: For any variable x in Ξ, we have that

$$\begin{aligned}\overline{\eta(\xi)}(x) &= x[\eta(\xi)] \\ &= \eta(x[\xi]) \\ &= \eta'(x[\xi]) \\ &= (\eta' \circ \overline{\xi})(x)\end{aligned}$$

So this proves the equality of natural transformations $\overline{\eta(\xi)} = \eta' \circ \overline{\xi}$. Since this equation characterizes $\eta'^*(\xi)$, it follows that $\eta(\xi) = \eta'^*(\xi)$, and hence the algebraicity of η'.

We now show that the restriction $\eta \mapsto \eta'$ is inverse to the extension $\eta \mapsto \eta^*$. We have already proved that for all transformation $\eta : \nu(\Gamma) \Rightarrow \nu(\Delta)$, and all element ξ of $\nu(\Gamma)_\Xi$, we have $\eta(\xi) = \eta'^*(\xi)$, so this proves that $\eta'^* = \eta$. Conversely, consider an algebraic natural transformation $\eta : T\Gamma \Rightarrow T\Delta$, we show that $(\eta^*)' = \eta$, that is, for all term t of dimension n in Γ, we have $\eta(t) = \eta^*(t)$. It suffices to show that $\overline{\eta(t)} = \overline{\eta^*(t)}$. Consider a variable x in D^n, then we have on one hand

$$\overline{\eta(t)}(x) = x[\eta(t)]$$

and on the other hand,

$$\overline{\eta^*(t)}(x) = \eta \circ \overline{t}(x) = \eta(x[t])$$

The naturality of η gives the equality between these two expressions, proving that $(\eta^*)' = \eta$. □

In terms of canonical diagrams, this can be understood as showing that the algebraicity condition characterizes exactly the diagrams over the disks that can be factored as a cone over the ps-contexts.

Substitutions to arbitrary contexts. These results let us characterize all the morphisms in the category $\mathcal{S}_{\mathsf{CaTT}}$ and exhibit $\mathcal{S}_{\mathsf{PS},\infty}$ as a codense subcategory of this category.

Theorem 58. *The category $\mathcal{S}_{PS,\infty}$ is a codense subcategory of $\mathcal{S}_{CaTT}$, or equivalently, the nerve functor ν is fully faithful*

Proof. We define a natural isomorphism $\mathcal{S}_{\mathsf{CaTT}}(\Delta, \Gamma) \simeq \mathrm{Nat}(\nu(\Gamma), \nu(\Delta))$ for any two contexts Δ and Γ in $\mathcal{S}_{\mathsf{CaTT}}$. By Lemma 57, it suffices to prove that the substitutions $\Delta \vdash \gamma : \Gamma$ are naturally in bijection with the algebraic natural transformations $T\Gamma \Rightarrow T\Delta$.

To any substitution $\Delta \vdash \gamma : \Gamma$, we associate the natural transformation $\overline{\gamma}$ defined for all terms t in $T\Gamma$ by $\overline{\gamma}(t) = t[\gamma]$. This family of functions defines a natural transformation, since the morphisms in $\mathcal{G}$ are the source and target, and the application of γ preserves the typing, i.e., if $\Delta \vdash t : u \to v$, then $\Gamma \vdash t[\gamma] : u[\gamma] \to v[\gamma]$. Moreover we show that this natural transformation is algebraic: Consider a ps-context Ξ together with a term t derivable in Ξ and a substitution $\Gamma \vdash \xi : \Xi$. Then we have $\overline{\gamma}(t[\xi]) = t[\xi][\gamma] = t[\xi \circ \gamma]$. Moreover for all variable x, $\overline{\xi \circ \gamma}(x) = x[\xi \circ \gamma] = (\overline{\gamma} \circ \overline{\xi})(x)$, which is the defining equation for $\overline{\gamma}^*(\xi)$, hence $\xi \circ \gamma = \overline{\gamma}^*(\xi)$, and $\overline{\gamma}(t[\xi]) = t[\overline{\gamma}^*(\xi)]$. This proves that $\overline{\gamma}$ is a algebraic natural transformation.

Conversely, given an algebraic natural transformation $\eta : T\Gamma \Rightarrow T\Delta$, we define a substitution $\Delta \vdash \delta_\eta : \Gamma$ such that $\overline{\delta_\eta} = \eta$ by induction over the length of Γ.

- If Γ is of length 0, then it is the terminal context $\varnothing$, and we define δ_η to be the unique substitution $\Delta \vdash \langle\rangle : \varnothing$.

– If Γ is of length $n+1$, then $\Gamma = (\Gamma', x : A)$, and the projection $\Gamma \vdash \pi : \Gamma'$ induces a "restriction" natural transformation $T\Gamma' \Rightarrow T\Gamma$. By composing this transformation with η, we get $\eta' : T\Gamma' \Rightarrow T\Delta$, and since η' is a restriction of η it is also algebraic. By induction this gives a substitution $\Gamma' \vdash \delta_{\eta'} : \Delta$. We then define $\delta_\eta \langle \delta_{\eta'}, \eta(x) \rangle$, and check that an application of the rule (ES) shows that δ_η is well defined. Since we have by induction $\Delta \vdash \delta_{\eta'} : \Gamma'$, and by hypothesis $\Gamma', x : A \vdash$, it suffices to check that $\Delta \vdash \eta(x) : A[\delta_{\eta'}]$. This is immediate if $A = \star$, and if $A = y \to z$, the naturality of η shows that $\Delta \vdash \eta(x) = \eta(y) \to \eta(z)$, and by definition of $\delta_{\eta'}$, we have that $A[\delta_{\eta'}] = \eta'(y) \to \eta'(z) = \eta(y) \to \eta(z)$. Hence the substitution is well defined.

  It now suffices to check that $\overline{\delta_\eta} = \eta$, and since it is an equality between two algebraic natural transformation, it suffices by Lemma 55 to check that they have the same actions on variables. Let y be a variable in Γ, then either $y = x$, and then by definition of δ_η, we have $x[\delta_\eta] = \eta(x)$, or y is a variable in Γ', and then $y[\delta_\eta] = y[\delta_{\eta'}]$ and by induction, we have that $y[\delta_{\eta'}] = \eta'(y) = \eta(y)$.

We then prove that these two maps are inverse to one another. We have already proved during the induction that $\overline{\delta_\eta} = \eta$ for all algebraic natural transformation, so it suffices to show that for all substitution $\Delta \vdash \gamma : \Gamma$, we have $\delta_{\overline{\gamma}} = \gamma$. By Lemma 6, it suffices to prove that for all variable x in Γ, we have $x[\delta_{\overline{\gamma}}] = x[\gamma]$, which holds by the following argument

$$\begin{aligned} x[\delta_{\overline{\gamma}}] &= \overline{\delta_{\overline{\gamma}}}(x) \\ &= \overline{\gamma}(x) \\ &= x[\gamma] \end{aligned}$$

□

2.5.4 $\mathcal{S}_{\mathsf{CaTT}}$ as a free completion

Theorem 58 shows that $\mathcal{S}_{\mathsf{PS},\infty}$ is a codense full subcategory of $\mathcal{S}_{\mathsf{CaTT}}$, so Corollary 50 shows that the inclusion $\mathcal{S}_{\mathsf{PS},\infty} \to \mathcal{S}_{\mathsf{CaTT}}$ exhibits $\mathcal{S}_{\mathsf{CaTT}}$ as the free completion of $\mathcal{S}_{\mathsf{PS},\infty}$ by the canonical limits.

Canonical limits. Apply the definition of canonical limits to our specific case of the codense subcategory $\mathcal{S}_{\mathsf{PS},\infty}$ shows that every object Γ in the category $\mathcal{S}_{\mathsf{CaTT}}$ is obtained as a limit of the following form, which is the *canonical limit*

$$\Gamma = \lim \left((\Gamma \downarrow \mathcal{S}_{\mathsf{PS},\infty}) \to \mathcal{S}_{\mathsf{PS},\infty} \to \mathcal{S}_{\mathsf{CaTT}} \right)$$

Preservation of globular products. Since $P_\infty : \mathcal{S}_{\mathsf{PS},\infty} \hookrightarrow \mathcal{S}_{\mathsf{CaTT}}$ is fully faithful and $\mathcal{S}_{\mathsf{CaTT}}$ has the globular products, P_∞ preserves the globular products. Corollary 51 apply to show the following

Corollary 59. *The category $\mathcal{S}_{\mathsf{CaTT}}$ is the free completion of the category $\mathcal{S}_{\mathsf{PS},\infty}$ by the canonical limits preserving the globular products. Equivalently for any complete category $\mathcal{C}$, the functor $P_\infty : \mathcal{S}_{\mathsf{PS},\infty} \hookrightarrow \mathcal{S}_{\mathsf{CaTT}}$ induces an equivalence of categories*

$$[\mathcal{S}_{\mathsf{PS},\infty}, \mathcal{C}]_{\text{gprod}} \simeq [\mathcal{S}_{\mathsf{CaTT}}, \mathcal{C}]_{\text{gprod,canlim}}$$

2.5.5 Functors preserving globular products

We now suppose given a category $\mathcal{C}$ equipped with a functor $F : \mathcal{S}_{\mathsf{CaTT}} \to \mathcal{C}$ which preserves the globular products, and we reproduce the previous results for the category $\mathcal{C}$ seeing the objects as generalization of contexts, as allowed by the functor F.

Morphism to the image of ps-contexts.

Lemma 60. *If Γ is a ps-context, and X is an object of $\mathcal{C}$, then there is a bijection*

$$\mathbf{Set}(X, F\Gamma) \simeq \mathrm{Nat}(V\Gamma, \nu_F X)$$

Proof. Consider a ps-context Γ, then it can be written as a globular product. Since F preserves the globular products $F\Gamma$ is also a globular product, and a natural transformation $V\Gamma \Rightarrow \nu_F X$ is exactly a cone of apex X over a diagram which is equivalent to the globular product diagram of $F\Gamma$, hence the equality, by definition of a limit. □

This lemma can also be formulated in terms of Kan extensions. Note that we have the following induced functors

$$\begin{array}{ccc} \mathcal{S}_{\mathsf{PS},0} & \xrightarrow{I_p} & \mathcal{S}_{\mathsf{PS},\infty} \\ {\scriptstyle D_p}\uparrow & \nearrow_{I_p D_p} & \\ \mathcal{G}^{\mathrm{op}} & & \end{array}$$

and the previous lemmas restricted to this functor state that $I_p = \mathrm{Ran}_{D_p}(I_p D_p)$. Lemma 60 states that a functor $F : \mathcal{S}_{\mathsf{CaTT}} \to \mathcal{C}$ preserving the globular products necessarily preserves this right Kan extension, i.e., that we have $FI_p = \mathrm{Ran}\, D_p(FI_p D_p)$.

Algebraic natural transformations $T\Gamma \Rightarrow T_F X$. Given a functor $F : \mathcal{S}_{\mathsf{CaTT}} \to \mathbf{Set}$, preserving the globular products, we denote T_F the nerve functor associated to the composite $\mathcal{G}^{\mathrm{op}} \to \mathcal{S}_{\mathsf{CaTT}} \to \mathbf{Set}$. Similarly to the previous section, a natural transformation $\nu\Gamma \Rightarrow \nu_F X$ is redundant, and can be reduced to a natural transformation $T\Gamma \Rightarrow T_F X$, satisfying a particular algebraicity condition.

Lemma 61. *A natural transformation $\eta : T\Gamma \Rightarrow T_F X$ induces a natural transformation between the nerves $\eta^* : \nu(\Gamma) \Rightarrow \nu_F(X)$.*

Proof. For a natural transformation $\eta : T\Gamma \Rightarrow T_F X$, given a ps-context Ξ together with a substitution $\Gamma \vdash \xi : \Xi$, we have the natural transformation $\overline{\gamma} : V(\Xi) \Rightarrow \nu(\Gamma)$, by vertically composing with η we get the natural transformation $\eta \circ \overline{\gamma} : V(\Xi) \Rightarrow \nu_F(X)$, and we define $\eta^*(\gamma)$ to be the unique map such that $\overline{\eta^*(\gamma)} = \eta \circ \overline{\gamma}$. The fact that this defines a natural transformation is analogous to Lemma 53 □

Definition 62. A natural transformation $\eta : T\Gamma \Rightarrow T_F X$ is *algebraic* if for all ps-context Ξ and for all term t in Ξ, along with a substitution $\Gamma \vdash \gamma : \Xi$, the following equality is satisfied

$$\eta(t[\gamma]) = F(t) \circ (\eta^* \gamma)$$

The term algebraicity comes from an analogy with the monads, and has no direct connection with the algebraicity of a morphism that we have introduced to present Grothendieck-Maltsiniotis definition of weak ω-categories.

Lemma 63. *Two algebraic natural transformation $T\Gamma \Rightarrow T_F X$ are equal if and only if they coincide on all variables.*

Proof. The proof is the same as the one of Lemma 55 □

Lemma 64. *The set of algebraic natural transformations $T\Gamma \Rightarrow T_F X$ is naturally isomorphic to the set of natural transformations $\nu\Gamma \Rightarrow \nu_F X$.*

Proof. Again, the proof is essentially the same as the one of Lemma 57. □

2.6 Models of CaTT

The characterization of the syntactic category that we have given enables us to compute the models of the theory CaTT. In particular we show here that they are equivalent to the Grothendieck-Maltsiniotis definition of weak ω-categories [54] that we have presented in Section 2.1. The idea behind this proof is that the canonical limits together with the globular product define the same class of limits as the pullbacks along the display maps and the terminal object. Thus the models, which are the functors preserving the terminal object and the canonical limits, are equivalently described as the functors preserving the canonical limits and the globular products. The latter are equivalent to the Grothendieck-Maltsiniotis definition, by the characterization of the syntactic category as a free completion that we have presented in the previous section.

Lemma 65. *If $F : \mathcal{S}_{\mathsf{CaTT}} \to \mathbf{Set}$ is a functor that preserves globular products, then F preserves canonical limits if and only if F preserves the pullbacks along display maps and the terminal object.*

Proof. Consider a functor $F : \mathcal{S}_{\mathsf{CaTT}} \to \mathbf{Set}$ that preserves globular products. We first reformulate our goal by applying Lemma 47 together with Lemma 64, and show that F preserves pullbacks along display maps if and only if for every context Γ and every set X, the association $f \mapsto \overline{f}$ induces a bijection between $\mathbf{Set}(X, F\Gamma)$ and the algebraic natural transformations $T\Gamma \Rightarrow T_F X$.

First we assume that F preserves the pullbacks along the display maps and the terminal object, and show that the desired map is an bijection, this by induction on the context Γ

- For the empty context $\varnothing$, it is the terminal object in the category $\mathcal{S}_{\mathsf{CaTT}}$ and F preserves terminal object, hence $F\varnothing$ is the terminal object in **Set**, and hence for all X, $\mathbf{Set}(X, F\varnothing)$ is a singleton. Moreover, by construction $(T\Gamma)_n$ is the set of terms of dimension n in the empty context. Since in the theory CaTT no term is derivable in the empty context, $T\varnothing$ is the empty presheaf which is initial, hence there is a unique natural transformation $T\varnothing \Rightarrow T_F X$, and it is vacuously algebraic. Hence the map $\overline{}$ is a map between two singleton sets, so it is a bijection.

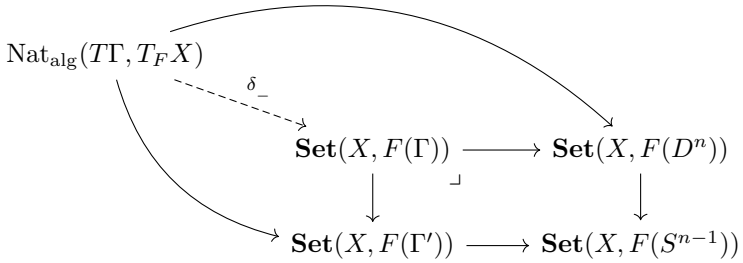
- Consider a context $\Gamma = (\Gamma', x : A)$, and assume the bijection holds for Γ. Then Γ writes as the following pullback (on the left) and since F preserves pullbacks along display maps, taking image by F yields the following pullback square (on the right)

$$\begin{array}{ccc} \Gamma & \xrightarrow{x} & D^n \\ \downarrow & \lrcorner & \downarrow \\ \Gamma' & \xrightarrow[A]{} & S^{n-1} \end{array} \qquad \begin{array}{ccc} F(\Gamma) & \longrightarrow & F(D^n) \\ \downarrow & \lrcorner & \downarrow \\ F(\Gamma') & \longrightarrow & F(S^{n-1}) \end{array}$$

For any set X, the continuity of the hom-functor with respect to its second variable shows that the following square is a pullback

$$\begin{array}{ccc} \mathbf{Set}(X, F(\Gamma)) & \longrightarrow & \mathbf{Set}(X, F(D^n)) \\ \downarrow & \lrcorner & \downarrow \\ \mathbf{Set}(X, F(\Gamma')) & \longrightarrow & \mathbf{Set}(X, F(S^{n-1})) \end{array}$$

An algebraic natural transformation $\eta : T\Gamma \Rightarrow T_F X$ restricts as a natural transformation $\eta' : T\Gamma' \Rightarrow T_F X$ which is again algebraic, and by induction this gives a map $\delta_\eta : X \to F\Gamma'$. Applying the last variable to η also gives a specific element $\eta(x) \in (T_F X)_n = \mathbf{Set}(X, FD^n)$. Moreover, the naturality of η shows that these two constructions fit into the following commutative triangle, which by the property of the pullback gives a unique map $\delta\_$



Where $\mathrm{Nat}_{\mathrm{alg}}(T\Gamma, T_F X)$ is the set of algebraic natural transformations. By precomposing with the map $\overline{} : \mathbf{Set}(X, F\Gamma) \to \mathrm{Nat}_{\mathrm{alg}}(T\Gamma, T_F X)$, we have that for all map $\gamma : X \to F\Gamma$, by induction hypothesis $\delta_{\overline{\gamma}'} = F(\pi)\gamma$, and $\overline{\gamma}(x) = x[\gamma]$, and hence by universal property of the pullback, this implies that $\delta\_ \circ \overline{} = \mathrm{id}_{\mathbf{Set}(X,F\Gamma)}$. Conversely, for all natural algebraic transformation $\eta : T\Gamma \Rightarrow T_F X$ and all variable y in Γ, we have that $\overline{\delta_\eta}(y) = Fy \circ \sigma_\eta$, then either y is a variable Γ' and by induction $Fy \circ \delta_\eta = \eta'(y) = \eta(y)$, or $y = x$ and then $Fy \circ \delta_\eta = \eta(x)$ by definition of δ_η. Hence for all variable y of Γ, $\eta(y) = \overline{\delta_\eta}(y)$, and by Lemma 63 this shows $\eta = \overline{\sigma_\eta}$ Hence $\delta\_$ is an inverse to the map $\overline{}$, and thus the map is a bijection.

Conversely, we suppose that the map $\overline{}$ is a bijection, and show that then it also preserves pullback along display maps and terminal object. First note that for the terminal object $\varnothing$, we have already proved that there is exactly one natural transformation $T\varnothing \Rightarrow T_F X$ for all set X, and the assumed bijection then ensures that $F\varnothing$ is terminal in $\mathbf{Set}$. So we are left proving that F preserves pullbacks along display maps, and for this it suffices to prove that it preserves pullbacks along generating display maps. Consider such a pullback, which is of the following form (on the left), and we consider a commutative on the following form (on the right) for an arbitrary set X.

$$\begin{array}{ccc} (\Gamma, x : A) & \xrightarrow{x} & D^n \\ \downarrow & \lrcorner & \downarrow \\ \Gamma & \xrightarrow[A]{} & S^{n-1} \end{array} \qquad \begin{array}{ccc} X & \xrightarrow{t} & F(D^n) \\ \downarrow & & \downarrow \\ F\Gamma & \xrightarrow{g} & F(S^{n-1}) \end{array}$$

By the assumed bijection, the map $g : X \to F\Gamma$ corresponds exactly to an algebraic natural transformation $\overline{g} : T(\Gamma, x : A) \Rightarrow T_F X$. Under this bijection, the fact that $F(\Gamma, x : A)$ is the

preserved pullback is equivalent to saying that there exists a unique algebraic natural transformation $\eta : T(\Gamma, x : A) \Rightarrow T_F X$ which coincide with $\overline{g}$ for all terms that are definable in Γ, and such that $\eta(x) = t$. By Lemma 63 the uniqueness is clear, since the requirement specifies the values on all variables of $(\Gamma, x : A)$, so it suffices to show that such a natural transformation exists. We define this natural transformation by first setting $\eta(t) = \overline{g}(t)$ for all terms in Γ, and extend it by induction on the depth of the term that are not definable in Γ.

- For a term of depth 0, it is necessarily the variable x, and we set $\eta(x) = t$. The fact that this is natural for x is equivalent to the fact that the initial square we consider commutes.
- Suppose that we have constructed η which is natural for all terms of depth at most d, and consider a term t of depth $d+1$ derivable in $(\Gamma, x : A)$. Then t is necessarily a coherence oh the form $t = \mathsf{coh}_{\Delta,B}[\gamma]$ with γ being a substitution of depth d. We then set $\eta(t) = F(\mathsf{coh}_{\Delta,B}[\mathrm{id}_\Delta]) \circ (\eta^* \gamma)$. We now check that this is natural for t : consider the source substitution $\sigma : D^n \to D^{n-1}$, we have that

$$\begin{aligned} F\sigma \circ \eta(t) &= F\sigma \circ F(\mathsf{coh}_{\Delta,B}[\mathrm{id}_\Delta]) \circ (\eta^* \gamma) \\ &= F(\sigma \circ \mathsf{coh}_{\Delta,B}[\mathrm{id}_\Delta])(\eta^* \gamma) \\ &= \eta(\sigma \circ t) \end{aligned}$$

  and similarly for the target substitution. This proves the naturality of η on the term t we constructed.

This natural transformation is algebraic by construction, hence we have proved the existence of a unique algebraic natural transformation that meets the requirements, and hence F preserves the pullbacks along the generating display maps. This shows that F preserves pullbacks along all display maps. □

Theorem 66. *The models of CaTT are equivalent to the weak ω-categories.*

Proof. The models of CaTT are the functors $\mathcal{S}_{\mathsf{CaTT}} \to \mathbf{Set}$ preserving the terminal object and the pullbacks along display map as stated in Lemma 10. Moreover, those induce a morphism of coglobular structured categories with families from $\mathcal{S}_{\mathsf{CaTT}} \to \mathbf{Set}$, hence by Lemma 30, they also preserve the globular products, so it follows by Lemma 65 that these are exactly the functors $\mathcal{S}_{\mathsf{CaTT}} \to \mathbf{Set}$ preserving the globular products and the canonical limits. Such functors are equivalent to weak ω-categories by Corollary 59 □

Following Ara's result [4], under a mild conjecture [4, Conjecture 4.1.7, p.55] this proves that the models of CaTT are also equivalent to the definition of weak ω-categories due to Batanin [11] and Leinster [48]. In fact we believe that CaTT is a good framework to explore the connection between the Grothendieck-Maltsiniotis definition of weak ω-categories and the Batanin-Leinster definition of weak ω-categories, but we have not pushed research in this direction yet.

Equalizers in the category $\mathcal{S}_{\mathsf{CaTT}}$. We present a second characterization of the models of $\mathcal{S}_{\mathsf{CaTT}}$, based on Corollary 18. Indeed, given that our contextual category comes from an actual syntax, all the most general unifiers that it has are primitive. Moreover, we assume the following

Conjecture 67. *For any two substitutions of the form $\Delta \vdash \langle \gamma, x \mapsto t \rangle : (\Gamma, x : A)$ and $\Delta \vdash \langle \gamma', x \mapsto t' \rangle : (\Gamma, x : A)$ that have an equalizer in $\mathcal{S}_{\mathsf{CaTT}}$, the substitutions $\Delta \vdash \gamma : \Gamma$ and $\Delta \vdash \gamma' : \Gamma$ also have an equalizer in $\mathcal{S}_{\mathsf{CaTT}}$.*

Under this conjecture, Corollary 18 gives another characterization of the models of CaTT.

Corollary 68. *The category of models* $\mathbf{Mod}(\mathcal{S}_{\mathsf{CaTT}})$ *is equivalent to the category of functors* $\mathcal{S}_{\mathsf{CaTT}} \to \mathbf{Set}$ *that preserves all the finite limits that are in* $\mathcal{S}_{\mathsf{CaTT}}$.

This gives an important connection with categorical notion of models of an algebraic theory.

2.6.1 The syntactic category

In the light of Theorem 66, we understand the syntactic category $\mathcal{S}_{\mathsf{CaTT}}$ as describing an appropriate notion of *finite presentation* for weak ω-categories. The traditional notion of presentation for higher structure is the notion of polygraphs [21, 60], which is well-known in the strict and truncated cases. Intuitively, polygraphs allow for presenting the cells that generate a higher category, these cells may have sources and targets which are not themselves generators, but rather composite of generators. The situation is analogous to the example we have presented, of the context $\Gamma =$ `(x : *, a : id x -> id x)`, where the variable `a` has a source `id x`, which is not itself a variable, but a term which represents a certain composite of variables. This observation supports our intuition that the contexts define a notion of polygraph for the weak ω-categories. Moreover, since the contexts necessarily have finitely many variables, we believe that they in fact define *finite polygraphs*. For any context Δ of the theory CaTT, we can construct the functor $\mathcal{S}_{\mathsf{CaTT}}(\Delta, \_) : \mathcal{S}_{\mathsf{CaTT}} \to \mathbf{Set}$. By continuity of the hom-functor, this functor preserves the limits, so in particular it preserves the terminal objects and the pullbacks along the display maps, hence Lemma 10 shows that it is a model of CaTT. The association $\Delta \mapsto \mathcal{S}_{\mathsf{CaTT}}(\Delta, \_)$ thus defines a functor $F : \mathcal{S}^{\mathrm{op}}_{\mathsf{CaTT}} \to \mathbf{Mod}(\mathcal{S}_{\mathsf{CaTT}})$, and since it is a coYoneda embedding this functor is fully faithful. The functor F hence exhibits the syntactic category $\mathcal{S}^{\mathrm{op}}_{\mathsf{CaTT}}$ as a full subcategory of the category of models $\mathbf{Mod}(\mathcal{S}_{\mathsf{CaTT}})$. Our intuition from categorical logic is that this full subcategory is defined by the fact that its objects are exactly the weak ω-categories freely generated by a finite polygraph.

2.6.2 Towards a structure with weak functors

Even though the category of models of CaTT is equivalent to the Grothendieck-Maltsiniotis category of weak ω-categories, we claim that this category is not completely satisfying. The objects in these category correspond indeed to our intuition of weak ω-categories, but the morphisms are not the expected ones: We expect the weak ω-categories to be equipped with a notion of weak functors, that satisfy all the axioms of functors up to invertible higher cells, whereas the morphisms of models are a strict variant satisfying all these axioms on the nose. In fact we could not have obtained our expected notion of functors as morphisms in the categories of models, since those do not compose in a strictly associative way, hence they do not define a category, but rather an $(\infty, 1)$-category. We believe that the category of models of the theory CaTT could equipped with a model structure, or a variation of it, that presents this $(\infty, 1)$-category. In particular we believe that the weak ω-categories should be equipped with an appropriate notion of weak equivalence, and that the properties that make sense for weak ω-categories are the ones that are invariant under this notion of weak equivalence. Moreover, this discussion can be repeated for natural transformations, and all the higher cells, and we believe that the weak ω-categories in fact naturally define a weak ω-category, even though we do not know a way to present or manipulate it.

Chapter 3

Practical use and partial automation

Now that we have presented and analyzed the type theory CaTT from a theoretical standpoint and justified its correctness, we study practical aspects and in particular explain how such a theory can be used to prove results about weak ∞-categories.

3.1 Implementation

Along with the formal description of the type theory, we provide an implementation of a proof-assistant for weak ω-categories that functions as a type checker, for this theory. Our implementation, realized in the programming language OCaml is available at [15], and we discuss the choices that were factored in during the implementation in order to make this proof-assistant into a practical tool, along with its syntax.

3.1.1 Syntax

The type $\star$ is written in CaTT as the constant `*`, and the type $t \underset{A}{\longrightarrow} u$ is written `A | t -> u`, moreover a context is simply a sequence of variables together with their types, denoted with parentheses and colons as follows

```
(x : *) (y : *) (f : * | x -> y)
        (f' : * | x -> y) (a : * | x -> y | f -> f')
        (z : *) (g : * | y -> z)
```

Term constructors. Since our implementation was performed at the early stage of the thesis, it has only one family of type constructor called coh that is meant to encompass both the type constructors op and coh that we have presented. To this end, this type constructor has two introduction rules, one corresponding to the constructor op and one corresponding to the constructor coh. This is just a choice of implementation, and does not change the theory at all since there is no occurrence where both rules can apply simultaneously. Since there are two introduction rules for this type constructors that lead to terms having different syntactic properties, we have chosen since then to separate this into the two term families op and coh, and keep this distinction as much as possible in our discussion, even if our implementation, and hence the syntax that we use hide this distinction.

Type inference. As noticed previously, whenever there is an arrow type $t \underset{A}{\longrightarrow} u$, the argument A can always be omitted since it is recovered as the common type of the terms t and u, and this justifies to denote this type $t \to u$. The same principle is implemented in the proof assistant. This requires the proof assistant to work with type inference and not only type checking. Since there is only one derivation that leads to a well-typed term, type inference is not more difficult than type checking and this strategy is fully implemented.

```
(x : *) (y : *) (f : x -> y)
        (f' : x -> y) (a : f -> f')
        (z : *) (g : y -> z)
```

3.1.2 Declaration

In order to manipulate terms in the theory without having to write them completely explicitly, we define *declarations* introduce intermediate terms, that we can then substitute. So if we have already defined a term t, simply providing a substitution γ lets us access the term $t[\gamma]$ without writing it out explicitly entirely. Using this technique we decompose complicated terms into successive applications of simpler terms that are easier to handle.

Declarations of operations. A particular case of this instance that is very important for us is the introduction of a *operation*: We call an *operation* a term of the form $t = \mathsf{op}_{\Gamma,A}[\mathrm{id}_\Gamma]$, and we think of it as the primitive definition of the operation. Any other instance of a term $\mathsf{op}_{\Gamma,A}[\gamma]$ can then be obtained by simply applying the substitution γ to the term t. In order to introduce a term of this form, we only need to specify the ps-context Γ and the type A, and thus we provide a dedicated syntax to introduce the declaration of an operation.

```
coh name G : T
```

where `name` is a string denoting the name we give to the declaration, `G` is the CaTT translation of the ps-context Γ and `T` is the CaTT translation of the type A. For instance, we can use a declared operation to define the composition of 1-cells in a weak ω-category as follows

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

And we can later on refer to the constructed term using only the name `comp`, followed by a list of term that we call the *arguments*, but that are formally understood as defining a substitution. For instance in a context that defines terms `a`, `b`, `c` of type `*`, a term `ab` of type `a -> b` and a term `bc` of type `b -> c`, one can freely refer to the term

```
comp a b ab c bc
```

which is understood as the composition of `ab` and `bc`, and reduces internally to a term of the form

$$\mathsf{op}_{(x:\star,y:\star,f:x\to y,z:\star,g:y\to z),x\to z}[\langle x \mapsto a, y \mapsto b, f \mapsto ab, z \mapsto c, g \mapsto bc\rangle]$$

Declaration of coherences. Similarly to the declarations of operation, we introduce the *declarations of coherence* to define terms of the form $t = \mathsf{coh}_{\Gamma,A}[\mathrm{id}_\Gamma]$. The syntax is in fact exactly the same, due to our implementation not differentiating between the terms constructors `coh` and `op`.

```
coh name G : T
```

where `name` is a string denoting the name we give to the declaration, `G` is the CaTT translation of the ps-context Γ and `T` is the CaTT translation of the type A. For instance, the identity 1-cell of a 0-cell can be defined as a declared coherence as follows

```
coh id (x : *) : x -> x
```

As in the previous case, in any context defining a term `a` of type `*`, we can refer to the identity of `a` by constructing the term

```
id a
```

which internally reduces to

$$\mathsf{coh}_{(x:\star),x\to x}[\langle x \mapsto a\rangle]$$

General declarations. We also provide a syntax to introduce a term in an arbitrary context, regardless of whether it is a ps-context or not. This can be seen as a general declaration, in the sense that we allow for applying substitutions to these new terms as well. We think of these terms as operations or coherences in a weak ω-categories but that are not primitive and can be decomposed into simpler operations and coherences. Our dedicated syntax to introduce such terms is the following

```
let name G = t
```

where `name` is a string denoting the name we give to the declaration, `G` is an arbitrary context, and `t` is a term in CaTT. For instance, using the coherence `comp` previously defined, one can define the square of an endo-1-cell as follows

```
let sq (x : *) (f : x -> x) = comp x x f x f
```

internally, this term reduces to the expression

$$\mathsf{op}_{(x:\star,y:\star,f:x\to y,z:\star,g:y\to z),x\to z}[\langle x \mapsto x, y \mapsto x, f \mapsto f, z \ \mapsto x, g \mapsto f\rangle]$$

Moreover, we can also use this term as a basis of a declaration, for instance provided a context where `a` is a term of type `*` and `aa` is a term of type `a->a`, we can freely refer the term

```
sq a aa
```

which internally is understood as $\mathsf{sq}[\langle x \mapsto a, f \mapsto aa\rangle]$, which then computes to the following expression

$$\mathsf{op}_{(x:\star,y:\star,f:x\to y,z:\star,g:y\to z),x\to z}[\langle x \mapsto a, y \mapsto a, f \mapsto aa, z \ \mapsto a, g \mapsto aa\rangle]$$

Declarations and cuts. Formally, the fact that we can define declarations can be modeled with cut rules, and using a declaration then becomes an application of cut admissibility. Indeed, for instance for the coherence `id`, the declaration of the coherence

```
coh id (x : *) : x -> x
```

amounts to giving a derivation for the term $(x : \star) \vdash \mathsf{coh}_{(x:\star),x\to x}[\mathrm{id}_{(x:\star)}]$. Using this coherence later on, with for instance

```
let (x : *) (y : *) (f : x -> y) = id y
```

amounts to defining the substitution $(x : \star, y : \star, f : x \to y) \vdash \langle x \mapsto y\rangle : (x : \star)$, and then using the cut admissibility for showing that these two derivations yields a derivation of $(x : \star, y : \star, f : x \to y) \vdash \mathsf{coh}_{(x:\star),x\to x}[\langle x \mapsto y\rangle] : y \to y$.

Kernel and meta-operations. In our OCaml implementation of CaTT, we have chosen to restrict as much as possible the meta-theoretic properties we rely on, and thus we have implemented the bare bone theory without any additional feature in what we call the kernel of the program. Every term that is fed to the kernel of the program is then checked to be valid. The cut admissibility is a meta-theoretic statement, and as such is not a feature of the kernel, but an external operation that computes on the syntax. This is a choice we have made to stay as close as possible to the theoretical description of CaTT, but it has consequences: when applying a declaration, the program starts by computing the entire term, and then checks that it is well-formed. For complicated terms, this can become computationally heavy. An implementation more focused on heavy use should incorporate the cut admissibility as a feature of the kernel, in order to stop on an already checked declarations, and simply check the validity of the substitution, instead of checking only completely reduced terms.

3.1.3 Implicit arguments

The syntax that we have introduced until now for the type theory is very heavy, and although it is theoretically usable, the length of terms we need to write, together with a heavy redundancy makes this implementation not usable in practice. So we add to the system *implicit arguments*, which are by far the most useful feature to have in order to make CaTT usable in practice. We can already see the need for having implicit arguments in the following example, which computes the succession of two binary composition to make a ternary composition associated on the left

```
let comp3-left (x : *) (y : *) (f : x -> y)
                       (z : *) (g : y -> z)
                       (w : *) (h : z -> w) =
   comp x z (comp x y f z g) w h
```

Not only this term is complicated to parse for a human reader, it also does not match the usual mathematical notation $(f \cdot g) \cdot h$ in diagrammatic order (or more commonly $h \circ (g \circ f)$), where only the arrows f, g, h are denoted. The implicit arguments will let us write this term instead as

```
let comp3-left (x : *) (y : *) (f : x -> y)
                       (z : *) (g : y -> z)
                       (w : *) (h : z -> w) =
   comp (comp f g) h
```

which matches the usual mathematical practice. On this simple example it does not seem to make that big a difference, but example quickly become much more involved, and each of the term that we could omit may be, in more complicated scenarios, not only variables, but complicated terms themselves, and being able to omit them ends up having a very important impact on the usability of CaTT.

Unification algorithm. In order to implement the implicit arguments, we need a unification algorithm, which matches an implicit term with a known list of constraints in order to recover the unknown term. We keep this algorithm to its simplest form, as it is implemented outside of the kernel, and produce terms that are later on checked by the kernel. This algorithm simply guesses what an unknown term can be, and stops as soon as it finds an answer. In particular it does not check that the various constraints are compatible, incompatibility errors are caught later on by the kernel, as they produce invalid terms.

Implicit arguments for declarations operation and coherence. In order to make the syntax as light as possible, the software performs an inference to determine which arguments should be left implicit and which are the ones that need to be entered by the user. In the special case of the declaration of an operation or a coherence, we have a very efficient algorithm to sort out the variables. Only the locally maximal variables should be left explicit, and all the other variables should be understood as implicit. This can be understood categorically: the ps-contexts are globular products in the category $\mathcal{S}_{\mathtt{CaTT}}$, and given a ps-context Γ and a substitution $\Delta \vdash \gamma : \Gamma$, the image of the locally maximal variables of γ define the maps $\Delta \to D^n$ of the cone induced by Γ to the top row of the diagram of globular product. This shows that two substitutions sending the locally maximal variables onto the same image define the same cone, and thus are equal. For instance, when declaring the composition

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

the software computes the locally maximal variables to be `f` and `g`, and hence only these variables should be left explicit, and one can then write, for the definition of `comp3-left`, the term

```
comp (comp f g) h
```

which is refined outside of the kernel as

```
comp x z (comp x y f z g) w h
```

and then fed to the kernel to be checked. If the user makes a mistake and write for instance

```
comp (comp f g) f
```

the term would still be refined as the first possible guess that the refinement algorithm finds. In this example, such a guess could be

```
comp x z (comp x y f z g) y f
```

But when sent to the kernel, this term does not typecheck since the variable `f` does not have type `z -> y`.

Implicit arguments for generic declarations. For generic declarations, we again have a way to decide whether an argument should be left implicit or not, but since the context is not necessarily a ps-context, the algorithm is a bit more complicated. It consists in, for each variable, looking up if this variable appears in the type of the other variables in the context. If it does, it can be kept implicit, and its associated term can be recovered by computing the type of the term associated to the variable in whose type our variable appears. For instance when defining the square term

```
let sq (x : *) (f : x -> x) = comp f f
```

the software determines that the variables `x` does appear in the type of the variable `f`, and hence can be left implicit, whereas the variable `f` does not appear in any type, and should be kept explicit. Thus the user can define

```
let sq-id (x : *) = sq (id x)
```

which is refined as `sq x (id x)` and computed to be then checked by the kernel. Again, if the user makes a mistake and tries to define

```
let bad-sq (x : *) (y : *) (f : x -> y) = sq f
```

the term would still be refined, for example to `sq x f`, but when sent to the kernel, it would not type-check since `f` is not of type `x -> x`.

Towards more implicit arguments. The cases we have identified reduce a lot the redundancy in the arguments that are provided when defining a term in CaTT, and they are the only ones that are implemented as of now. However, they do not cover all the cases. Consider for instance the following definition

```
let idf (x : *) (y : *) (f : x -> y) = comp (id x) f
```

One can notice that following our description, we need to specify the argument `x`, whereas in this specific term, it could have been inferred. One could imagine, in order to cover this kind of cases, to introduce an Agda-like syntax to tell the system that a particular argument can be inferred and should not be specified. With this syntax, one could write

```
let idf (x : *) (y : *) (f : x -> y) = comp (id _) f
```

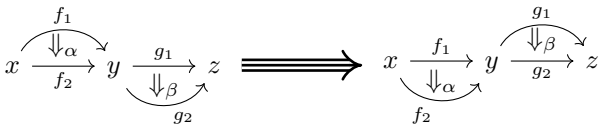
We have not implemented this feature yet, and in fact in practice this case does not come up too often.

Towards a syntax with holes. The unification algorithm that we have provided is really weak as it just makes an educated guess on what a term should be, and lazily stops as long as it gets an answer without checking against the other constraints. In mathematical terms, it provides a necessary condition, which may not be sufficient. Having a more powerful unification algorithm could allow us to add holes in the syntax, like one could do in Agda. This is a feature that would be useful to add for practical purposes, but that our current implementation is pretty far from as of now.

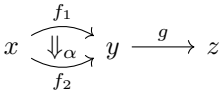
3.1.4 An extensive example: the Eckmann-Hilton morphism

Our running example of a result that we prove for weak ω-categories is the *Eckmann-Hilton morphism*. We start by explaining the result before explaining how we proceed to formalize it. The Eckmann-Hilton morphism can be described in a ω-category that has one 0-cell x, and two 2-cells a and b whose source and target are the 1-cell id_x. In this situation, one can compose vertically the 2-cells a and b in two ways, and construct $a \cdot b$ and $b \cdot a$. The Eckmann-Hilton morphism $\mathrm{eh}(a, b)$ is a 3-cell that performs a braiding and relates these two 2-cell (i.e., $\partial^-(\mathrm{eh}(a, b)) = a \cdot b$ and $\partial^+(\mathrm{eh}(a, b)) = b \cdot a$).

The exchange rule. The main idea behind the Eckmann-Hilton morphism is the existence of an *exchange rule* for ω-categories. This exchange rule is a primitive operation, and can be described pictorially as follows



In this diagrammatic representation, a part of the diagram of the following shape symbolizes the *right whiskering*



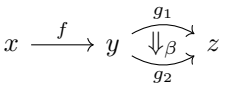
an operation which, given a 2-cell α and a 1-cell g produces a 2-cell $\mathrm{rw}(\alpha, f)$ such that

$$\partial^-(\mathrm{rw}(\alpha, f)) = \partial^-(\alpha) \cdot f$$
$$\partial^+(\mathrm{rw}(\alpha, f)) = \partial^+(\alpha) \cdot f$$

Formally, the right whiskering can be defined in CaTT as

```
coh rw (x : *) (y : *) (f1 : x -> y)
                       (f2 : x -> y) (a : f1 -> f2)
               (z : *) (g : y -> z)
    : comp f1 g -> comp f2 g
```

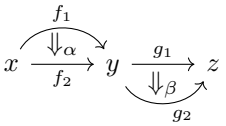
Symmetrically, a part of the diagram of the following form represents a *left whiskering*



which can be formally defined in CaTT as the following

```
coh lw (x : *) (y : *) (f : x -> y)
               (z : *) (g1 : y -> z)
                       (g2 : y -> z) (b : g1 -> g2)
   : comp f g1 -> comp f g2
```

The following diagram, obtained by superposition of the two previous diagrams, is meant to denote the vertical composition of their result.



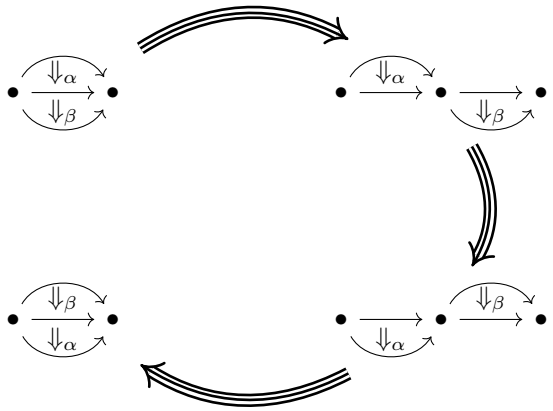
and similarly, the target diagram of the exchange represents the vertical composition of the results of the two whiskerings the other way around. We can formally define the vertical composition of 2-cells in CaTT as follows

```
coh vcomp (x : *) (y : *) (f1 : x -> y)
                          (f2 : x -> y) (a : f1 -> f2)
                  (z : *) (g1 : y -> z)
                          (g2 : y -> z) (b : g1 -> g2)
    : comp  f1 g1 -> comp f2 g2
```

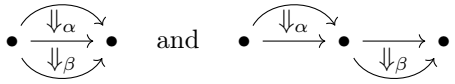
which lets us give a completely formal definition of the exchange rule

```
coh exch (x : *) (y : *) (f1 : x -> y)
                         (f2 : x -> y) (a : f1 -> f2)
                 (z : *) (g1 : y -> z)
                         (g2 : y -> z) (b : g1 -> g2)
    : vcomp (rw a g1) (lw f2 b) -> vcomp (lw f1 b) (rw a g2)
```

A pictorial proof. We now give a pictorial definition for the Eckmann-Hilton morphism, that we then explain how to formalize inside our implementation of CaTT. In this diagram, all the objects are denoted $\bullet$ and denote the same 0-cell x, and all the 1-dimensional arrows simply denoted $\to$ are the identity on this object

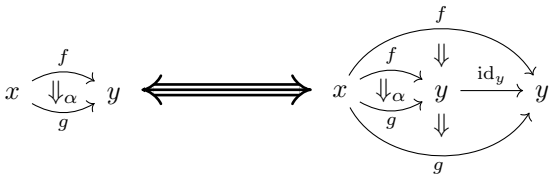


Although this picture conveys the correct intuition, it represents operations that are actually ill-defined in weak ω-categories. Instead, it is the correct proof for the Eckmann-Hilton in *strict* ω-categories. Indeed, for instance the two diagrams



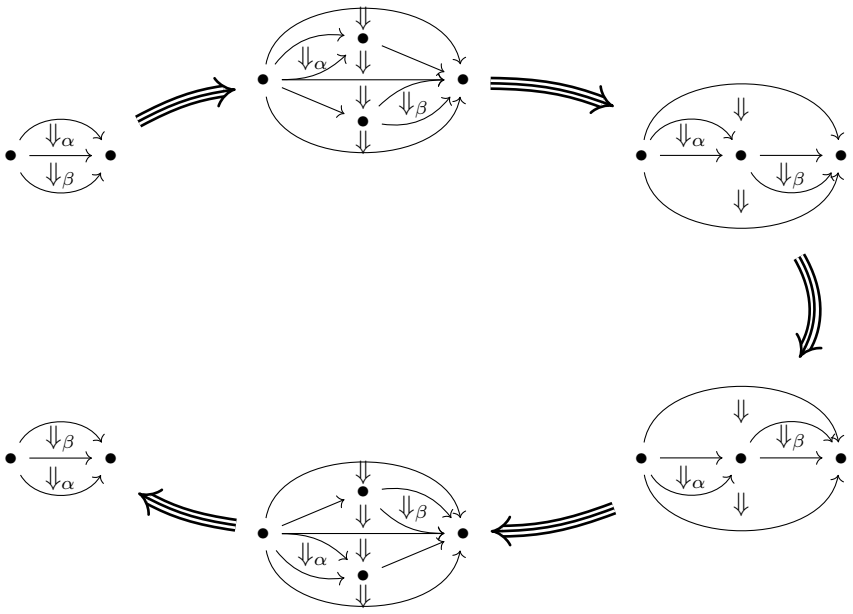
represent two 2-cells whose respective source and target are id_x and $\mathrm{id}_x \cdot \mathrm{id}_x$. These two 2-cells are thus not parallel, and hence there cannot exist a 3-cell between them.

Pictorial proof for the weak case. We can correct the previous picture, in order to obtain a definition of the Eckmann-Hilton morphism in weak ω-categories. To achieve this, we compose our cells with a correction term, that cancels $\mathrm{id}_x \cdot \mathrm{id}_x$ into id_x and conversely. We use two definable 3-cells defined as in the following picture



Using these arrows and their analogue for the left whiskering, together with cancellation of the corrective 3-cells that we have introduced, we construct the following graphical proof for the

Eckmann-Hilton morphism in weak ω-categories



Formal definition in CaTT. We have formalized the definition suggested by this diagram in CaTT, and we present here some of the steps towards it. We start be defining the 3-cells witnessing that a 2-cell whiskered with an identity 1-cell is equivalent to the original 2-cell (again, we need to compose with corrective terms to ensure the cells we mention are parallel)

```
coh rw-unit (x : *) (y : *) (f : x -> y) (g : x -> y) (a : f -> g)
    : a -> 3vcomp (unitr- f) (rw a (id y)) (unitr f)
```

where `3vcomp` is the ternary vertical composition on 2-cells and `unitr-` is the inverse of the unitor: Given a cell it relates it to its composition with the identity. Similarly, we denote `rw-unit-` the "opposite coherence" obtained by switching the source and target of $rw - unit$, and also `lw-unit` and `lw-unit-` the analogue coherences for the operation `lw` instead of `rw`. Assuming that we also have defined a horizontal composition for 3-cells denoted `hcomp3`, we can express the first step of our diagram as

```
let 1 (x : *) (a : id x -> id x) (b : id x -> id x)
    = hcomp3 (rw-unit a) (lw-unit b)
```

as well as the last step of the diagram

```
let 5 (x : *) (a : id x -> id x) (b : id x -> id x)
    = hcomp3 (lw-unit- b) (rw-unit- a)
```

Using the following coherences stating that left unitality of identity and right unitality of the identity 1-cell are inverse to each other

```
coh unit(rl-) (x : *) :
 vcomp (unitr (id x)) (unitl- (id x)) -> id2 (comp (id x) (id x))
coh unit(lr-)- (x : *) :
 id2 (comp (id x) (id x)) -> vcomp (unitl (id x)) (unitr- (id x))
```

where `id2` is the identity 2-cell of a 1-cell, we can define the second and fourth steps of the diagram as follows

```
let 2 (x : *) (a : id x -> id x) (b : id x -> id x) =
    5hcomp3 (id2 (unitr- (id x)))
            (id2 (rw a (id x)))
            (unit(rl-) x)
            (id2 (lw (id x) b))
            (id2 (unitl (id x)))
let 4 (x : *) (a : id x -> id x) (b : id x -> id x) =
    5hcomp3 (id2 (unitl- (id x)))
            (id2 (lw b (id x)))
            (unit(lr-)- x)
            (id2 (rw (id x) a))
            (id2 (unitr (id x)))
```

where `5hcomp3` is the 5-ary horizontal composition of 3-cells. Finally, using the previously defined `exch`, as well as the following coherence which let us interchange left and right unitality for the identity

```
coh unit(l->r) (x : *) : unitl (id x) -> unitr (id x)
coh unit-(r->l) (x : *) : unitr- (id x) -> unitl- (id x)
```

we can define the third step of our graphical proof as

```
let 3 (x : *) (a : id x -> id x) (b : id x -> id x) =
    3hcomp 3 (unit-(r->l) x)
             (exch a b)
             (unit(l->r) x)
```

where `3hcomp3` is the ternary horizontal composition on 3-cells. We have now defined all the steps, of the proof, and we just need to assemble them into one big term. However, they do not assemble on the nose, because of associativity and identity cancellation issues. So we need to associate and cancel the identities that appear in between each steps. We do not explain in details how we do that here, as it is merely a technical difficulty. The fully formalized definition is accessible online[1], and it uses a total of 93 different definitions, spread over 531 lines of code.

3.2 Suspension

Using the proof assistant CaTT to prove any non-immediate result about weak ω-categories tends to be extremely cumbersome and time consuming, for various reasons. A good reason for this observation is simply that the theory of weak ω-categories is complicated, a lot of intuitive reasoning that one performs implicitly while working in informal mathematics has to be made explicit, and induces coherence problems that tend to grow with the dimension. We view this as a good reason for CaTT to be complicated to use, as it factors all the verification that were very hard to carry by hand into computer checked ones, and justifies the importance of having a computer aided implementation of the theory such as CaTT. But there is also a bad reason why the usage of CaTT tends to be complicated, which is the fact that a lot of developments have to be repeated many times, with very slight changes, to express things that intuitively feel the

[1] `https://github.com/ThiBen/catt/blob/master/examples/eckmann-hilton-versions/eh-no-susp-no-func.catt`

same, but that are described in different ways in the theory. In order to address this problem, we propose two ways to partially automate proofs in CaTT, that we call the *suspension* and the *functorialization*. These operation have been defined in [17], and we give here an updated version.

3.2.1 Example of suspensions

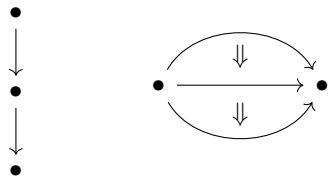
We start by presenting the suspension and how it can be used to reunite terms that intuitively describe "the same coherence in different dimensions", but whose formal descriptions as terms in the theory are not related in an obvious way. We chose to present the suspension first as it is simpler to describe, more general and more powerful than the functorialization. In order to motivate our definition, we start by an informal discussion where the phenomenon of suspension appears naturally.

Identity cells. The first natural occurrence of the suspension is in the definition of the identity cells. For every cell x of dimension n is an ω-category $\mathcal{C}$, there is a cell id_x of dimension $n+1$ such that $\partial^-(\mathrm{id}_x) = \partial^+(\mathrm{id}_x) = x$. We can define these operations formally in CaTT using the disk contexts: A cell of dimension n defines a term, which can be interpreted as a substitution $\Gamma \vdash \chi_t : D^n$ the identity for this cell t is obtained as the term

$$\Gamma \vdash \mathsf{coh}_{D^n, x_{2n} \to x_{2n}}[\chi_t] : t \to t$$

On this example, we see that even though the identities for the terms of the same dimensions are expressed by the same coherence, applied to a different substitution, the identities for terms of different dimensions are not very related, except from the relation between the disk contexts. The fact that we call all these cells identities, and thus perceive them as "the same thing in different dimensions" is not at all reflected by the theory.

Composition of 1-cells and vertical composition. Another instance of two definitions that intuitively are very close, but whose connection is not immediate from their definition in CaTT is the composition of the 1-cell and the vertical composition of the 2-cells. Graphically these are given by the two following diagrams.



Intuitively, both of these operations correspond to the same idea of gluing the end of a cell to the beginning of another one, but in different dimensions. The terms defining these are

```
coh comp (x:*)(y:*)(f:x->y)(z:*)(g:y->z):x->z
coh vcomp (x:*)(y:*)(f:x->y)(g:x->y)(a:f->g)(h:x->y)(b:g->h):f->h
```

The intuition of these terms defining the same idea is lost without the diagrams. On the diagrams, we see that the diagram describing the vertical composition of the 2-cells is obtained from the diagram describing the composition of the 1-cells by adding two new 0-cells, and then replacing all the previous 0-cells by 1-cells between those, and all the previous 1-cells by 2-cells. This operation reminds us of the topological notion of suspension, our goal is to describe formally this operation for the entire syntax, we call it suspension by analogy.

Coherences. This remark does not only apply to these operations: It also applies to many other operations, but also to their coherences. For instance there is an associativity witness for the composition of 1-cells and an associativity witness for the vertical composition of 2-cells, and both express the same idea about two operations that we had noticed to be similar. Similarly, there are unitality witnesses for the identity 1-cell with respect to the composition of 1-cells as well as for the identity 2-cells with respect to the vertical composition. Both of these also express similar ideas. Iterating this argument shows that a lot of terms that we can define for the 2-cells are in fact just the same of terms that we defined for 1-cells, but in another dimension. Without any automation, one has to define all of these both for 1-cells and for 2-cells, and it becomes worse while working on higher dimensional cells, as the same ideas have to be developed in each dimension. Our goal is to propose a framework to avoid this issue, by defining the terms once and for all in a dimension, and allowing to transport from one dimension to another.

Interpretation in terms of enrichment. A way to understand what it means for two operation to "be the same in different dimensions" is to look at it from an enrichment point of view. Intuitively, ω-categories can be defined coinductively as categories enriched in ω-categories. Although this is true, it yields to a notion of strict ω-category, and describing weak ω-categories would require a notion of weak enrichment that is extremely hard to define, so we keep this discussion on an informal level, and use the word enriched assuming a weak enough notion of enrichment, and building up on intuition from the strict case. Taking as a starting point that an ω-category is a category enriched in ω-categories shows that the operations and coherences of dimension $n+1$ are either operations and coherences in dimensions n that were transferred by enrichment, or a coherence coming from the composition in dimension $n+1$, or characterizing an interaction between these types. For two operations or coherence, to "be the same in different dimension" in fact means that one can be deduced from the other by a series of enrichment.

3.2.2 Suspension of ps-contexts

Our objective is now to make formal the notion of sameness that we have described, in the type theory CaTT. More precisely, we will describe the notion of suspension, that corresponds to deducing an operation or a coherence in dimension $n+1$ from its analogous of dimension n by enrichment.

Definition. In order to describe the suspension, we need to first chose two variable names that are not used in the theory, that we denote $\bullet_-$ and $\bullet_+$. We can simply do this by adding two new elements to the set of variable names. If we work with de Bruijn levels, we can introduce the levels -1 and -2 as an auxiliary step for computation and then renormalize so that all the contexts start at 0. With these two variables, we define the suspension of a context Γ, denoted ΣA by induction on the context, together with the suspension of a type A in GSeTT, denoted ΣA

$$\Sigma\varnothing = (\bullet_- : \star, \bullet_+ : \star) \qquad \Sigma(\Gamma, x : A) = (\Sigma\Gamma, x : \Sigma A)$$

$$\Sigma\star = \bullet_- \underset{\star}{\to} \bullet_+ \qquad \Sigma(x \underset{A}{\to} y) = x \underset{\Sigma A}{\longrightarrow} y$$

Correctness. We have given an algorithm that, given a certain context generates a new context, by induction. This is for now a purely syntactic description, so in order to check that this operation can be used safely in practice, we need to prove a correctness result, showing that it only yields well-formed contexts when the original context is well-formed.

Lemma 69. *The suspension of a ps-context is again a ps-context, the following rule is admissible*

$$\frac{\Gamma \vdash_{\mathsf{ps}}}{\Sigma\Gamma \vdash_{\mathsf{ps}}}$$

Proof. We prove by induction on the derivation that for all variable x such that $\Gamma \vdash_{\mathsf{ps}} x : A$, we also have $\Sigma\Gamma \vdash_{\mathsf{ps}} x : \Sigma A$.

- If the derivation of $\Gamma \vdash_{\mathsf{ps}} x : A$ is a single application of the rule (PSS), then necessarily $\Gamma = (x : \star)$ and $A = \star$. In that case, we can compute explicitly the suspensions $\Sigma\Gamma = (\bullet_- : \star, \bullet_+ : \star, x : \bullet_- \to \bullet_+)$ and $\Sigma A = \bullet_- \to \bullet_+$, and a successive application of the rules (PSS) and (PSE) yields a derivation of the judgment $\Sigma\Gamma \vdash_{\mathsf{ps}} x : \Sigma A$.
- If the derivation of $\Gamma \vdash_{\mathsf{ps}} x : A$ ends with an application of the rule (PSE), then necessarily, Γ is of the form $(\Gamma', z : B, x : y \to z)$, and $A = y \to z$, and we have a derivation of $\Gamma' \vdash_{\mathsf{ps}} y : B$. In that case, $\Sigma\Gamma = (\Sigma\Gamma', z : \Sigma B, x : y \to z)$, and $\Sigma A = y \to z$, then by induction we get a derivation of $\Sigma\Gamma' \vdash_{\mathsf{ps}} y : \Sigma B$, and applying the rule (PSE) to this derivation yields a derivation of $\Sigma\Gamma \vdash_{\mathsf{ps}} x : \Sigma A$.
- If the derivation of $\Gamma \vdash_{\mathsf{ps}} x : A$ ends with an application of the rule (PSD), then we have a derivation of the form $\Gamma \vdash_{\mathsf{ps}} f : y \underset{A}{\to} x$. Then by induction, we get a derivation of $\Sigma\Gamma \vdash_{\mathsf{ps}} y \underset{\Sigma A}{\longrightarrow} x$, and by applying (PSD), we get a derivation of $\Gamma \vdash_{\mathsf{ps}} x : \Sigma A$.

Now consider a derivation of $\Gamma \vdash_{\mathsf{ps}}$, it necessarily comes from a derivation of $\Gamma \vdash_{\mathsf{ps}} x : \star$, and applying the result we just proved yields a derivation of $\Sigma\Gamma \vdash x : \bullet_- \underset{\star}{\to} \bullet_+$. Applying successively the rules (PSD) and (PS) gives then a derivation of $\Sigma\Gamma \vdash_{\mathsf{ps}}$ □

Example. We give a few examples of computing the suspension for some ps-contexts, and in particular we pay close attention to the action of this operation on our combinatorial description of the ps-contexts. An immediate induction shows that for the disk context D^n, the suspension computes to $\Sigma D^n = D^{n+1}$. If we consider for instance the following ps-context

$$\Gamma = (x : \star, y : \star, f : x \to y, z : \star, g : y \to z)$$

its suspension computes to the following ps-context

$$\Sigma\Gamma = (\bullet_- : \star, \bullet_+ : \star, x : \bullet_- \to \bullet_+, y : \bullet_- \to \bullet_+, f : x \to y, z : \bullet_- \to \bullet_+, g : y \to z)$$

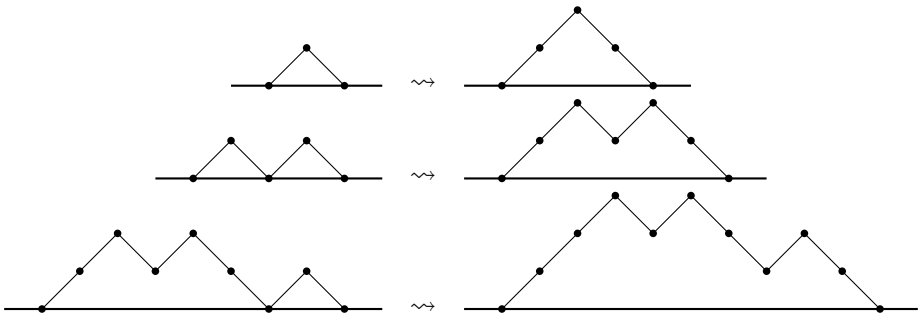
As a globular set, the new context $\Sigma\Gamma$ is obtained from Γ by adding two new 0-cells $\bullet_-$ and $\bullet_+$, and shifting all the other variable by one dimension: the variable x of dimension 0 in Γ becomes of dimension 1 in $\Sigma\Gamma$ (and of type $\bullet_- \to \bullet_+$), the variable f of dimension 1 in Γ becomes of dimension 2 in $\Sigma\Gamma$. In general, following our proof of correctness shows that for a ps-context $\Gamma \vdash_{\mathsf{ps}}$, such that the derivation of the context $\Gamma \vdash_{\mathsf{ps}}$ is obtained as

$$\Gamma \vdash_{\mathsf{ps}} = (\text{PSS})(\text{PSE})^{k_1}(\text{PSD})^{l_1} \dots (\text{PSE})^{k_m}(\text{PSD})^{l_m}(\text{PS})$$

then the derivation of the judgment $\Sigma\Gamma \vdash_{\mathsf{ps}}$ is obtained by the derivation

$$\Sigma\Gamma \vdash_{\mathsf{ps}} = (\text{PSS})(\text{PSE})^{k_1+1}(\text{PSD})^{l_1} \dots (\text{PSE})^{k_m}(\text{PSD})^{l_m+1}(\text{PS})$$

Note that the equality $\sum k_n = \sum l_n$ is preserved by this transformation. Visually, this transformation of the derivation can be represented on our graphical representation on the combinatorial structure that underlies the ps-contexts.



Pictorially, this operation can be understood as "lifting" a ps-context by one dimension. It has also a very simple description using Dyck words: representing the structure of a ps-context Γ by a Dyck word w, the structure of $\Sigma\Gamma$ is represented by the word (w).

Source and target of suspension. We can compute the source and the target of a suspended ps-context, and show that the suspension is compatible with these notions.

Lemma 70. *Given a ps-context $\Gamma \vdash_{\mathsf{ps}}$, we have the equalities*

$$\partial^-(\Sigma\Gamma) = \Sigma(\partial^-(\Gamma))$$
$$\partial^+(\Sigma\Gamma) = \Sigma(\partial^+(\Gamma))$$

Proof. We show by induction that for all $i \in \mathbb{N}$, we have $\partial^-_{i+1}(\Sigma\Gamma) = \Sigma(\partial^-_i(\Gamma))$.

– For the context $\Gamma = (x : \star)$, we have $\partial^-_i(x : \star) = (x : \star)$, hence

$$\Sigma(\partial^-_i(x : \star)) = (\bullet_- : \star, \bullet_+ : \star, x : \bullet_- \to \bullet_+)$$

On the other hand $\Sigma\Gamma = (\bullet_- : \star, \bullet_+ : \star, x : \bullet_- \to \bullet_+)$ and since $i + 1 \geq 1$, we have

$$\partial^-_i(\Sigma\Gamma) = (\bullet_- : \star, \bullet_+ : \star, x : \bullet_- \to \bullet_+)$$

– For a context of the form $\Gamma = (\Gamma', y : A, f : x \to y)$ with $\dim A \geq i - 1$, we have $\partial^-_i(\Gamma) = \partial^-_i(\Gamma')$, so

$$\Sigma(\partial^-_i(\Gamma)) = \Sigma(\partial^-_i(\Gamma)')$$

Moreover $\Sigma\Gamma = (\Sigma\Gamma', y : \Sigma A, f : x \to y)$, and since $\dim(\Sigma A) = \dim A + 1$, we also have $\dim(\Sigma A) \geq i$, hence

$$\partial^-_{i+1}(\Sigma\Gamma) = \partial^-_{i+1}(\Sigma(\Gamma'))$$

We thus have the equality by induction.

– For a context of the form $\Gamma = (\Gamma', y : A, f : x \to y)$ with $\dim A < i - 1$, we have $\partial^-_i(\Gamma) = (\partial^-_i(\Gamma'), y : A, f : x \to y)$, so

$$\Sigma(\partial^-_i(\Gamma)) = (\Sigma(\partial^-_i(\Gamma)'), y : \Sigma A, f : x \to y)$$

Moreover $\Sigma\Gamma = (\Sigma\Gamma', y : \Sigma A, f : x \to y)$, and since $\dim(\Sigma A) = \dim A + 1$, we also have $\dim(\Sigma A) < i$, hence

$$\partial^-_{i+1}(\Sigma\Gamma) = (\partial^-_{i+1}(\Sigma(\Gamma')), y : \Sigma A, f : x \to y)$$

The induction hypothesis then gives the equality.

Applying this inductive result to the case $i = \dim\Gamma - 1$, we get for all ps-contexts the equality $\Sigma(\partial^-(\Gamma)) = \partial^-_{\dim\Gamma}(\Sigma\Gamma)$, and since $\dim\Sigma\Gamma = \dim\Gamma + 1$, this rewrites as $\Sigma(\partial^-(\Gamma)) = \partial^-(\Sigma\Gamma)$. The case of the target of a ps-context is analogous. □

Note that in the case of the ps-context $(x : \star)$, the source and targets are not defined, and we can check explicitly that $\partial^-(\Sigma(x : \star)) = (\bullet_- : \star)$ and $\partial^+(\Sigma(x : \star)) = (\bullet_+ : \star)$.

3.2.3 Suspension for the theory CaTT

Using the suspension for ps-contexts, we can extend this operation to all the terms in the theory, and express formally the intuition that two terms are "the same operation in different dimensions".

Definition. Even though we are interested in the suspension of terms of CaTT, we have to define this operation also for contexts, types and substitutions, in order to use mutual induction. We again denote $\bullet_-$ and $\bullet_+$ two fresh variables.

$$\begin{aligned}
\Sigma\varnothing &= (\bullet_- : \star, \bullet_+ : \star) & \Sigma(\Gamma, x : A) &= (\Sigma\Gamma, x : \Sigma A)\\
\Sigma\star &= \bullet_- \underset{\star}{\to} \bullet_+ & \Sigma(t \underset{A}{\to} u) &= \Sigma t \underset{\Sigma A}{\longrightarrow} \Sigma u\\
\Sigma x &= x & \Sigma(\mathsf{op}_{\Gamma,A}[\gamma]) &= \mathsf{op}_{\Sigma\Gamma,\Sigma A}[\Sigma\gamma]\\
& & \Sigma(\mathsf{coh}_{\Gamma,A}[\gamma]) &= \mathsf{coh}_{\Sigma\Gamma,\Sigma A}[\Sigma\gamma]\\
\Sigma\langle\rangle &= \langle \bullet_- \mapsto \bullet_-, \bullet_+ \mapsto \bullet_+ \rangle & \Sigma(\gamma, x \mapsto t) &= \langle \Sigma\gamma, x \mapsto \Sigma t \rangle
\end{aligned}$$

Note that for a ps-context Γ, this operation coincide with the suspension that we have previously defined, hence there is no ambiguity in denoting $\Sigma\Gamma$. It is immediate from the definition that we have for all type A, $\mathrm{Var}(\Sigma A) = \mathrm{Var}(A) \cup \{\bullet_-, \bullet_+\}$, and for all term t, $\mathrm{Var}(\Sigma t) = \mathrm{Var}(t)$ if t is a variable, or $\mathrm{Var}(\Sigma t) = \mathrm{Var}(t) \cup \{\bullet_-, \bullet_+\}$ otherwise.

Application of substitution and suspension. In order to show the correctness of our general version of the suspension, we need a result about the computational interaction of this suspension with the application of the substitution.

Lemma 71. *Suppose given a substitution $\Delta \vdash \gamma : \Gamma$, then for any type $\Gamma \vdash A$, we have $\Sigma(A[\gamma]) = \Sigma A\ [\Sigma\gamma]$, for any term $\Gamma \vdash t : A$, we have $\Sigma(t[\gamma]) = \Sigma t[\Sigma\gamma]$ and for any substitution $\Gamma \vdash \xi : \Xi$, we have $\Sigma(\xi \circ \gamma) = \Sigma\xi \circ \Sigma\gamma$.*

Proof. We prove these results by mutual inductions on the structure of the type, term or substitution.

Induction for types:

– For the type $\star$, we have $\star[\gamma] = \star$, hence $\Sigma(\star[\gamma]) = \bullet_- \underset{\star}{\to} \bullet_+$. On the other hand, $\Sigma\star[\Sigma\gamma] = \bullet_-[\Sigma\gamma] \underset{\star}{\to} \bullet_+[\Sigma\gamma]$, and since $\bullet_- \mapsto \bullet_- \in \Sigma\gamma$, we have $\bullet_-[\gamma] = \bullet_-$, and similarly, $\bullet_+[\gamma] = \bullet_+$. This proves that $\Sigma \star [\Sigma\gamma] = \bullet_- \underset{\star}{\to} \bullet_+$, and hence $\Sigma(\star[\gamma]) = \Sigma \star [\Sigma\gamma]$.

– For a type of the form $\Delta \vdash t \underset{A}{\to} u$, we have the equalities

$$\Sigma((t \underset{A}{\to} u)[\gamma]) = \Sigma(t[\gamma]) \xrightarrow[\Sigma(A[\gamma])]{} \Sigma(u[\gamma])$$
$$\Sigma(t \underset{A}{\to} u)[\Sigma\gamma] = \Sigma t[\Sigma\gamma] \xrightarrow[\Sigma A[\Sigma\gamma]]{} \Sigma u[\Sigma\gamma]$$

The induction for types and terms then shows $\Sigma((t \underset{A}{\to} u)[\gamma]) = \Sigma(t \underset{A}{\to} u)[\gamma]$.

Induction for terms:

– For a variable term, on the one hand we have $\Sigma x = x$, hence $\Sigma x[\Sigma\gamma] = x[\Sigma\gamma]$

– For a term of the form $\mathsf{op}_{\Xi,A}[\xi]$, we have

$$\Sigma(\mathsf{op}_{\Xi,A}[\xi][\gamma]) = \mathsf{op}_{\Sigma\Xi,\Sigma A}[\Sigma(\xi \circ \gamma)]$$
$$\Sigma(\mathsf{op}_{\Xi,A}[\xi])[\Sigma\gamma] = \mathsf{op}_{\Sigma\Xi,\Sigma A}[\Sigma\xi \circ \Sigma\gamma]$$

and then induction case for substitutions gives the result.

– Similarly, for a term of the form $\mathsf{coh}_{\Gamma,A}[\gamma]$,

$$\Sigma(\mathsf{coh}_{\Xi,A}[\xi][\gamma]) = \mathsf{coh}_{\Sigma\Xi,\Sigma A}[\Sigma(\xi \circ \gamma)]$$
$$\Sigma(\mathsf{coh}_{\Xi,A}[\xi])[\Sigma\gamma] = \mathsf{coh}_{\Sigma\Xi,\Sigma A}[\Sigma\xi \circ \Sigma\gamma]$$

and the result comes from the induction case for substitutions.

Induction for substitutions:

– For the empty substitution $\langle\rangle$, we have

$$\Sigma(\langle\rangle \circ \gamma) = \langle \bullet_- \mapsto \bullet_-, \bullet_+ \mapsto \bullet_+ \rangle$$
$$\Sigma(\langle\rangle) \circ \Sigma\gamma = \langle \bullet_- \mapsto \bullet_-[\Sigma\gamma], \bullet_+ \mapsto \bullet_+[\Sigma\gamma] \rangle$$

but since we have the associations $\bullet_- \mapsto \bullet_-$ and $\bullet_+ \mapsto \bullet_+$ in $\Sigma\gamma$, this shows that $\bullet_-[\Sigma\gamma] = \bullet_-$ and $\bullet_+[\Sigma\gamma] = \bullet_+$.

– For a substitution of the form $\langle \xi, x \mapsto t \rangle$, we have

$$\Sigma(\langle \xi, x \mapsto t \rangle \circ \gamma) = \langle \Sigma(\xi \circ \gamma), x \mapsto \Sigma(t[\gamma]) \rangle$$
$$\Sigma(\langle \xi, x \mapsto t \rangle) \circ \Sigma\gamma = \langle \Sigma\xi \circ \Sigma\gamma, x \mapsto \Sigma t[\Sigma\gamma] \rangle$$

The induction cases for substitutions and for terms then gives $\Sigma(\xi \circ \gamma) = \Sigma\xi \circ \Sigma\gamma$ and $\Sigma(t[\gamma]) = \Sigma t[\Sigma\gamma]$.

□

Correctness. Again, we have defined this operation as a purely syntactic operation, so we need to provide a guarantee that it only yields to well-defined objects of the theory, in order to use it freely.

Lemma 72. *The following rules are admissible*

$$\frac{\Gamma \vdash}{\Sigma\Gamma \vdash} \qquad \frac{\Gamma \vdash A}{\Sigma\Gamma \vdash \Sigma A} \qquad \frac{\Gamma \vdash t : A}{\Sigma\Gamma \vdash \Sigma t : \Sigma A} \qquad \frac{\Delta \vdash \gamma : \Gamma}{\Sigma\Delta \vdash \Sigma\gamma : \Sigma\Gamma}$$

Proof. We prove all these properties by mutual induction on the derivation trees.

Induction for contexts:

- For the empty context $\varnothing$, we have $\Sigma\varnothing = (\bullet_- : \star, \bullet_+ : \star)$, and we can construct explicitly a derivation tree for the judgment $\Sigma\varnothing$ as follows

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\varnothing \vdash}(\text{EC})}{\varnothing \vdash \star}(\star\text{-INTRO})}{\bullet_- : \star \vdash}(\text{CE})}{\bullet_- : \star \vdash \star}(\star\text{-INTRO})}{\bullet_- : \star, \bullet_+ : \star \vdash}(\text{CE})$$

- For a context of the form $(\Gamma, x : A)$, a derivation of $(\Gamma, x : A) \vdash$ induces a derivation of $\Gamma \vdash A$. By the inductive case for types, this gives a derivation of $\Sigma\Gamma \vdash \Sigma A$, and applying the rule (CE) gives a derivation of $(\Sigma\Gamma, x : \Sigma A) \vdash$.

Induction for types:

- For the type $\star$, a derivation of $\Gamma \vdash \star$ induces a derivation of $\Gamma \vdash$, which by induction gives a derivation of $\Sigma\Gamma \vdash$. Since $\Sigma\star = \bullet_- \underset{\star}{\to} \bullet_+$, and $(\bullet_- : \star) \in \Sigma\Gamma$ and $(\bullet_+ : \star) \in \Sigma\Gamma$, this gives the following derivation for the judgment $\Sigma\Gamma \vdash \Sigma\star$

$$\dfrac{\dfrac{\Sigma\Gamma \vdash \quad (\bullet_- : \star) \in \Sigma\Gamma}{\Sigma\Gamma \vdash \bullet_- : \star}(\text{VAR}) \qquad \dfrac{\Sigma\Gamma \vdash \quad (\bullet_+ : \star) \in \Sigma\Gamma}{\Sigma\Gamma \vdash \bullet_+ : \star}(\text{VAR})}{\Sigma\Gamma \vdash \bullet_- \underset{\star}{\to} \bullet_+}(\to\text{-INTRO})$$

- For a type of the form $t \underset{A}{\longrightarrow} u$, a derivation of the judgment $\Gamma \vdash t \underset{A}{\longrightarrow} u$ induces a derivation of $\Gamma \vdash A$, of $\Gamma \vdash t : A$ and of $\Gamma \vdash u : A$. By the induction cases for types and for terms, these give derivations for $\Sigma\Gamma \vdash \Sigma A$, for $\Sigma\Gamma \vdash \Sigma t : \Sigma A$ and for $\Sigma\Gamma \vdash \Sigma t : \Sigma A$, which by an application of the rule ($\to$-INTRO) gives a derivation of the judgment $\Sigma\Gamma \vdash \Sigma t \underset{\Sigma A}{\longrightarrow} \Sigma u$.

Induction for terms:

- For a variable term x, a derivation of the form $\Gamma \vdash x : A$ is necessarily obtained from the rule (VAR), and thus induces a derivation of $\Gamma \vdash$, and we necessarily have $(x : A) \in \Gamma$. Then, we also necessarily have $(x : \Sigma A) \in \Sigma\Gamma$, and by induction on the case for contexts, we get a derivation of $\Sigma\Gamma \vdash$. Hence, applying the rule (VAR) yields a derivation of $\Sigma\Gamma \vdash x : \Sigma A$.
- For a term of the form $\Delta \vdash \mathsf{coh}_{\Gamma, t \underset{A}{\longrightarrow} u}[\gamma] : (t \underset{A}{\to} u)[\gamma]$, we have a derivation of $\Gamma \vdash_{\mathsf{ps}}$, which by Lemma 69 gives a derivation of $\Sigma\Gamma \vdash_{\mathsf{ps}}$. Moreover, the condition (C_{op}) imply that Γ is not the ps-context $(x : \star)$ and hence Lemma 70 along with the variable for suspension and the conditions (C_{op}), shows

$$\begin{aligned}\mathrm{Var}(\partial^-(\Sigma\Gamma)) &= \mathrm{Var}(\partial^-(\Gamma)) \cup \{\bullet_-, \bullet_+\} \\ \mathrm{Var}(\partial^+(\Sigma\Gamma)) &= \mathrm{Var}(\partial^+(\Gamma)) \cup \{\bullet_-, \bullet_+\}\end{aligned}$$

hence we have the equalities

$$\mathrm{Var}(\Sigma t) \cup \mathrm{Var}(\Sigma A) = \mathrm{Var}(\partial^-(\Sigma\Gamma))$$
$$\mathrm{Var}(\Sigma u) \cup \mathrm{Var}(\Sigma A) = \mathrm{Var}(\partial^+(\Sigma\Gamma))$$

so the condition (C_{op}) are satisfied by the suspended term. Finally by the induction hypotheses for types and for terms, we have that $\Sigma\Gamma \vdash \Sigma(t \underset{A}{\longrightarrow} u)$ and $\Sigma\Delta \vdash \Sigma\gamma : \Sigma\Gamma$. Thus an application of the rule (OP) gives a derivation of the judgment

$$\Sigma\Delta \vdash \mathsf{op}_{\Sigma\Gamma,\Sigma(t \underset{A}{\longrightarrow} u)}[\Sigma\gamma] : \Sigma t[\Sigma\gamma] \xrightarrow[\Sigma A\ [\Sigma\gamma]]{} \Sigma u[\Sigma\gamma]$$

By Lemma 71, this shows that we have obtained a derivation of

$$\Sigma\Delta \vdash \Sigma(\mathsf{op}_{\Gamma,t \underset{A}{\longrightarrow} u}[\gamma]) : \Sigma((t \underset{A}{\longrightarrow} u)[\gamma])$$

– Similarly, for a term of the form $\Delta \vdash \mathsf{coh}_{\Gamma,A}[\gamma] : A[\gamma]$ we have a derivation of $\Gamma \vdash_{\mathsf{ps}}$, a derivation of $\Gamma \vdash A$ and a derivation of $\Delta \vdash \gamma : \Gamma$. Lemma 69 gives a derivation of $\Gamma \vdash_{\mathsf{ps}}$, and the induction cases for types and substitutions give a derivation of $\Sigma\Gamma \vdash \Sigma A$ and $\Sigma\Delta \vdash \Sigma\gamma : \Sigma\Gamma$. Moreover, the condition ($\mathrm{C}'_{\mathsf{coh}}$) with the variables of a suspension show that

$$\mathrm{Var}(\Sigma A) = \mathrm{Var}(A) \cup \{\bullet_-, \bullet_+\} = \mathrm{Var}(\Sigma\Gamma)$$

and hence $\Sigma\Gamma$ satisfies ($\mathrm{C}'_{\mathsf{coh}}$). Applying (COH'), we get a derivation of

$$\Sigma\Delta \vdash \mathsf{coh}_{\Sigma\Gamma,\Sigma A}[\Sigma\gamma] : (\Sigma A)[\Sigma\gamma]$$

Lemma 71 shows that we have a derivation of

$$\Sigma\Delta \vdash \Sigma(\mathsf{coh}_{\Gamma,A}[\gamma]) : \Sigma(A[\gamma])$$

Induction for substitutions:

– For the empty substitution $\Delta \vdash \langle\rangle : \varnothing$, we necessarily have a derivation of the judgment $\Delta \vdash$, which by the induction case for contexts gives a derivation of $\Sigma\Delta \vdash$. Since moreover $\Sigma\varnothing = (\bullet_- : \star, \bullet_+ : \star)$, and $\Sigma\langle\rangle = \langle \bullet_- \mapsto \bullet_-, \bullet_+ \mapsto \bullet_+ \rangle$, we can construct first construct derivation of $\Sigma\Delta \vdash \langle \bullet_- \mapsto \bullet_- \rangle : (\bullet_- : \star)$ as follows

$$\dfrac{\dfrac{\Sigma\Delta \vdash}{\Sigma\Delta \vdash \langle\rangle : \varnothing}(\text{ES}) \qquad \dfrac{\varnothing \vdash}{\varnothing \vdash \star}(\star\text{-INTRO}) \qquad \dfrac{\Sigma\Delta \vdash \qquad \bullet_- : \star \in \Sigma\Delta}{\Sigma\Delta \vdash \bullet_- : \star}(\text{VAR})}{\Sigma\Delta \vdash \langle \bullet_- \mapsto \bullet_- \rangle : (\bullet_- : \star)}(\text{SE})$$

and using this derivation, we can construct a derivation of the judgment

$$\Sigma\Delta \vdash \langle \bullet_- \mapsto \bullet_-, \bullet_+ \mapsto \bullet_+ \rangle : (\bullet_- : \star, \bullet_+ : \star)$$

as follows

$$\dfrac{\Sigma\Delta \vdash \langle \bullet_- \mapsto \bullet_- \rangle : (\bullet_- : \star) \qquad \varnothing \vdash \star \qquad \dfrac{\Sigma\Delta \vdash \qquad \bullet_+ : \star \in \Sigma\Delta}{\Sigma\Delta \vdash \bullet_+ : \star}(\text{VAR})}{\Sigma\Delta \vdash \langle \bullet_- \mapsto \bullet_-, \bullet_+ \mapsto \bullet_+ \rangle : (\bullet_- : \star, \bullet_+ : \star)}(\text{SE})$$

– For a substitution of the form $\Delta \vdash \langle \gamma, x \mapsto t \rangle : (\Gamma, x : A)$, we have a derivation of $\Delta \vdash \gamma : \Gamma$, a derivation of $\Gamma \vdash A$ and a derivation of $\Delta \vdash t : A[\gamma]$. By the induction cases for substitutions, types and terms, these give a derivation for the judgment $\Sigma\Delta \vdash \Sigma\gamma : \Sigma\Gamma$, the judgment $\Sigma\Gamma \vdash \Sigma A$ and the judgment $\Sigma\Delta \vdash \Sigma t : \Sigma(A[\gamma])$. By Lemma 71, this last judgment is equal to $\Sigma\Delta \vdash \Sigma t : (\Sigma A)[\Sigma\gamma]$, hence we can apply the rule (SE) and get a derivation of $\Sigma\Delta \vdash \langle \Sigma\gamma, x \mapsto \Sigma t \rangle : (\Sigma\Gamma, x : \Sigma A)$.

Note that in order to prove that this induction is well-founded, we need to perform factor both the dimension and the depth of the terms in the proof, similar to an argument that we explain in 43. We have formalized a version of this argument for proving the decidability of type checking of a theory in Section 4.1. □

3.2.4 Implementation of the suspension

Since we have defined the suspension operation for the terms of the type theory, we can add this operation as a feature of our OCaml implementation of CaTT.

First examples. In order to respect the philosophy of introducing coherences and then applying them to get terms, we chose to add the possibility of computing suspensions for coherences and for terms. For instance, if we define the identity of a 0-cell

```
coh id (x:*) : x -> x
```

and later on want to access to the identity of a one cell, we can simply apply the suspension of `id`, as shown in the following example, where we denote `S id` the suspension of `id` (The notation `S id` is not the actual syntax, as we explain in the next paragraph).

```
let id1 (x:*) (y:*) (f:x->y) = (S id) f
```

Note that we now define this term with the keyword `let`, indicating that it can be constructed from the coherence previously defined, and in this case, it is constructed from the suspension of the coherence `id`. Similarly, when we define a term, for instance the square of an endomorphism

```
let sq (x:*) (f:x->x) = comp  f f
```

we can later on refer to the square of an endo-2-cell, that is vertically composing this 2-cell with itself, by using the suspension of the term `sq` as follows

```
let sq2 (x:*) (y:*) (f:x->y) (a:f->f) = (S sq) a
```

Implicit suspensions. For our first examples, we have introduced the notations `S id` and `S sq` in order to indicate that we need to use the suspension of a previously defined term. This is still a little heavy syntax, so for the actual implementation, we opted for a syntactically lighter method. Indeed, if we define a coherence $c = \mathsf{coh}_{\Gamma,A}$, and apply it to a substitution γ, the dimensions of the various terms in the substitution γ are determined by the dimensions of the variables in Γ. Since now $\Sigma c = \mathsf{coh}_{\Sigma\Gamma,\Sigma A}$, when we apply the coherence Σc to a substitution γ', the dimensions of the terms in γ' are determined by the dimensions of the variables in $\Sigma\Gamma$, which are all one more than the dimensions of variables in Γ. A consequence is that we can leave implicit the suspension, and simply guess if a coherence or term needs to be suspended and if so how many times simply by looking at the dimension of its arguments. In practice, this means that we can define apply the declared identity to any cell, or the declared composition to any pair of composable cells of the same dimension, and the appropriate suspension is automatically generated by the system. For instance, with our two previous examples, the actual terms in our implementation of CaTT are

```
let id1 (x:*) (y:*) (f:x->y) = id f
let sq2 (x:*) (y:*) (f:x->y) (a:f->f) = sq a
```

From now on, all the expression that we write in CaTT are assumed to have implicit suspension.

Kernel syntax and meta-theory. The kernel of our implementation of bare type theory CaTT corresponds in the strictest sense to the theory we have described in Section 2.4, and thus does not recognize the suspension operation. Instead we have chosen to implement it as a meta-operation, that takes a term in our syntax and refines it to produce a syntactic expression in CaTT, that is then fed to the kernel to be checked. The suspension happens at this refinement, and a consequence of this is that every time a new suspension is generated it is checked formally by the kernel as if it was a new term. Together with our proof of admissibility of suspensions, this check is redundant, and hence this choice is not computationally efficient, but at the time of the implementation, we did not have a formal proof for correctness of the suspension. Moreover, we introduce also another operation, the functorialization, whose correctness is more case specific, and this choice helped us realize it. A future implementation of CaTT designed with more extensive usage in mind should incorporate the suspension in the kernel of the theory, for maximal computational efficiency.

Eckmann-Hilton morphism revisited We can use our implementation of the suspension to simplify the formalization of the Eckmann-Hilton morphism that we have presented in Section 3.1. Out of the 93 definitions that were introduced in the version without suspension, 58 of them can be simplified by using a suspension. This does not necessarily mean that all 58 definition could be removed, but all of them rely of a definition that could be removed and made implicit. We have formalized the same result using the suspension, and the resulting file[2] contains 85 definition, and is 476 lines long so 10% shorter. Another measure of the impact is given by the size of the files, corresponding to the number of characters needed, and without suspension, the file weighs 29k, whereas with the suspension, it weighs 22k, which corresponds to a reduction of 24%. This shows how implementing the suspension goes a long towards having extensive developments like the definition of Eckmann-Hilton morphism more accessible in practice. It also spare the user from needing to find different names for suspended morphisms, which forces to adopt awkward convention, like the name `id2` and `5hcomp3` that we have introduced before. Instead, the user can now define the 5-ary horizontal composition on 2-cells, as `5hcomp`, and the identity 1-cells `id`, and use these simpler names for the suspended coherence.

Interpretation. The fact that the suspension is well defined on the syntax of the theory gives additional structure to its syntactic category: It endows it with an endofunctor Σ, which associates to each context $\Gamma \vdash$ its substitution $\Sigma\Gamma \vdash$ and to each substitution $\Delta \vdash \gamma : \Gamma$ the substitution $\Sigma\Delta \vdash \Sigma\gamma : \Sigma\Gamma$. Moreover, the fact that we can also suspend types and terms, and that the suspension respects the typing and the application of substitution shows that the endofunctor Σ is an endomorphism of categories with families. Under our assumed correspondence between the syntactic category of CaTT and a adequate notion of finite polygraphs for weak ω-categories, the definition of the suspension gives an endofunctor for these polygraphs.

[2] `https://github.com/ThiBen/catt/blob/master/examples/eckmann-hilton-versions/eh-susp-no-func.catt`

3.3 Degree of a term

Before introducing our other meta-operation similar to the suspension and that we call the functorialization, we introduce the notion of *degree of a term* in out theory, which gives the necessary keys to understand the functorialization. The notion of degree is closely related to the one of *invertibility*, that we define coinductively as follows

Definition 73. A term $\Gamma \vdash a : t \to u$ is invertible if there exists a term $\Gamma \vdash b : u \to t$ together with two invertible terms

$$\Gamma \vdash \mathrm{can}(a, b) \ : \mathtt{comp}\ a\ b \to \mathtt{id}(\mathtt{id}\ t)$$
$$\Gamma \vdash \mathrm{can}(b, a) : \mathtt{comp}\ b\ a \to \mathtt{id}(\mathtt{id}\ u)$$

As we show with the notion of degree, the invertibility of a term is completely conditioned by its syntactic properties.

3.3.1 Definition of the degree

The notion of degree of a term emerges naturally while carrying extensive developments in the type theory CaTT. Intuitively, this notion characterizes how much invertibility information the term carries. Terms of degree 0 are non-invertible, whereas terms of strictly positive degree are invertible. Moreover, a term of degree $k+1$ has sources and target of degree k, thus the degree expresses how many layers of invertible cells are stacked.

Definition 74. We define the *internal dimension* $\dim_{\mathrm{i}}(t)$ of a term t in the context Γ as follows

$$\begin{aligned}
&\text{On variables :} && \dim_{\mathrm{i}}(x) = \dim(x)\\
&\text{On operations :} && \dim_{\mathrm{i}}(\mathsf{op}_{\Gamma,A}[\gamma]) = \max_{x \in \mathrm{Var}(\Gamma)} \{\dim_{\mathrm{i}}(x[\gamma])\}\\
&\text{On coherences :} && \dim_{\mathrm{i}}(\mathsf{coh}_{\Gamma,A}[\gamma]) = \max_{x \in \mathrm{Var}(\Gamma)} \{\dim_{\mathrm{i}}(x[\gamma])\}
\end{aligned}$$

and the *degree* of a term t to be

$$\deg(t) = \dim(t) - \dim_{\mathrm{i}}(t)$$

The internal dimension of a term t is simply the maximal dimension among all the dimensions of the variables that appear in t, and hence the degree of t is the difference between its dimension and the variable of maximal dimension in t. Note that since the term constructors op and coh only produce terms of type constructed with $\to$, the only terms of type $\star$ are variables, which are necessarily of degree 0.

Internal degree of a variable. We can also understand the dimension more globally, by considering the entire term and looking the contribution of each variable. Each of the contribution is called the *internal degree* of the variable x in the term t, denoted $\deg_t(x)$. Formally, the internal degree of a variable is defined as

$$\deg_t(x) = \dim t - \dim x$$

The degree of the term t is then the minimal among all the internal degrees of the variables appearing in t. Intuitively, there are two phenomena that contribute to the dimension of a term, one is the dimension of the variables in the terms, and the other one is the term constructor coh. The internal degree quantifies how much each variable contributes to the dimension of the term, and the degree of the term removes the contribution of the variable and quantifies the contribution of the term constructor coh. When there is no confusion possible, we sometimes refer to the internal degree more simply as the degree of a variable in a term.

Example of degree. In order to get a better grasp of the notion, we introduce some example of terms, along with their degrees. Our first recurring example is the composition of two morphisms

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

We compute the type of this term to be of dimension 1, since its type is $x \to z$ which is of dimension 0, and its variables of maximal dimension are f and g which are of dimension 1 as well. Hence the composition is a term of degree 0. Considering the identity

```
coh id (x : *) : x -> x
```

the dimension of the result is 1, and its only variable is of dimension 0, hence the resulting term is of degree 1. Similarly, the associativity

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

defines a term of degree 1, since the term itself is of dimension 2, and its variables of maximal dimension are f, g and h that are of dimension 1. Finally the cancellation of associativity with its inverse

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

defines a term of degree 2. Up to this point it might seem that the degree quantifies how many term constructor `coh` are stacked on top of each other to construct the term, however, it is more subtle than this. Indeed, consider the term

```
let comp-id (x : *) = comp (id x) (id x)
```

Even though this term is constructed by the term constructor `op`, its degree is still 1, as both arguments are themselves of degree 1. Moreover, considering the two following terms

```
let id-rw (x : *) (y : *) (f : x -> y)
                  (z : *) (g : y -> z)
    = rw (id f) g
let rw-id (x : *) (y : *) (f1 : x -> y)
                          (f2 : x -> y) (a : f1 -> f2)
   = rw a (id y)
```

Even though both these terms are constructed with one the constructor `op` and have one argument of degree 0 and one argument of degree 1, the degree of `id-rw` is 1 and the degree of `rw-id` is 0. In fact the degree captures an interaction between the degree of the arguments and their dimension.

3.3.2 Terms of degree 0

The notion of degree of a variable interacts very strongly with the typing rules, to give very useful conditions relating the variables of a term with the variables of its type.

Variables of degree 0**.** First note that self evidently, if we have a type $\Delta \vdash t : A$ together with a variable x of degree 0 in t, then x is of the same dimension as t, hence x is of dimension strictly greater than the dimension of A. This implies that $x \notin \mathrm{Var}(A)$. The converse also holds, and gives a characterization of the variables of degree 0 in a term. It is however harder to prove, and is given by the two following result, that we prove by mutual induction.

Lemma 75. *Consider a term $\Delta \vdash t : A$ with a variable $x \in \mathrm{Var}(t)$ which is not in $\mathrm{Var}(A)$, then x is of degree 0 in t.*

Proof. We prove this result by mutual induction on the coherence depth with Lemma 76, each step of the induction is itself an induction on the depth of the terms.

- For a term of coherence depth 0 or of depth 0, it is a variable, and then its only variable is itself, which is of internal degree 0.
- For a term of the form $\Delta \vdash \mathsf{op}_{\Gamma,B}[\gamma] : B[\gamma]$, since we have a variable $x \in \mathrm{Var}(t)$, there is a $v \in \mathrm{Var}(\Gamma)$ such that $x \in \mathrm{Var}(v[\gamma])$, but because $x \notin \mathrm{Var}(B[\gamma])$, we necessarily have that $v \notin \mathrm{Var}(B)$. The condition $(\mathrm{C}_{\mathsf{op}})$ implies that $\mathrm{Var}(B) = \mathrm{Var}(\partial^-(\Gamma)) \cup \mathrm{Var}(\partial^+(\Gamma))$, and a variable in Γ that is not in $\mathrm{Var}(B)$ is then a variable of maximal dimension in Γ, hence $\dim v = \dim \Gamma$. Moreover, all the variables in the type of v appear in B, hence all the variables in the type of $v[\gamma]$ appear in A, and since x does not appear in A by hypothesis, x does not appear in the type of $v[\gamma]$, hence by induction x is of degree 0 in $v[\gamma]$. This implies $\dim x \leq \dim v[\gamma] = \dim v = \dim \Gamma$. Moreover, Lemma 76 shows that $\dim B = \dim \Gamma - 1$, and hence $\dim t = \dim \Gamma$. So in the end, we have proved that $\dim x = \dim t$, hence x is of degree 0 in Γ.
- For a term $\Delta \vdash \mathsf{coh}_{\Gamma,B}[\gamma]$, the condition $(\mathrm{C}_{\mathsf{coh}})$ implies that all variables of γ are also in $B[\gamma]$, and hence there is no variable to check.

□

Lemma 76. *Suppose that the type $\Gamma \vdash A$ is derivable and satisfies $(\mathrm{C}_{\mathsf{op}})$, then $\dim A = \dim \Gamma - 1$.*

Proof. From the condition $(\mathrm{C}_{\mathsf{op}})$, we necessarily have

$$A = t \underset{B}{\longrightarrow} u \quad \text{with} \begin{cases} \mathrm{Var}(t) \cup \mathrm{Var}(B) = \mathrm{Var}(\partial^-(\Gamma)) \\ \mathrm{Var}(u) \cup \mathrm{Var}(B) = \mathrm{Var}(\partial^+(\Gamma)) \end{cases}$$

Then there necessarily exists a variable x of maximal dimension in $\partial^-(\Gamma)$ that does not appear in $\partial^+(\Gamma)$, and since $\partial^+(\Gamma) \vdash u : B$ is also derivable, x cannot appear in B. Hence by mutual induction, Lemma 75 shows that x is of degree 0 in t, and hence $\dim t = \dim x = \dim \partial^-(\Gamma)$, or in other words, $\dim A = \dim \Gamma - 1$. □

We thus have proved the following result

Proposition 77. *Given a term $\Delta \vdash t : A$, a variable $x \in \mathrm{Var}(t)$ is of degree 0 in t if and only if it appears in $\mathrm{Var}(A)$*

Terms of degree 0**.** With the help of this characterization of the variables of degree 0 in a term, we can give an inductive description of the class of terms that are of degree 0 in our theory as follows

Proposition 78. *A term $\Delta \vdash t : A$ is of degree 0 if and only if it is a variable or it is of the form $\mathsf{op}_{\Gamma,A}[\gamma]$, where γ is a substitution which has at least one argument of maximal dimension of degree 0*

Proof. First note that a variable is always of degree 0, since its only variable is itself, and it is of internal degree 0, and for a term of the form $\Delta \vdash \mathsf{coh}_{\Gamma,A}[\gamma] : A[\gamma]$, for every variable x in this term, there is a variable v in Γ such that $x \in \mathrm{Var}(v[\Gamma])$. The condition ($\mathrm{C}_{\mathsf{coh}}$) then implies that $v \in \mathrm{Var}(A)$, and hence $x \in \mathrm{Var}(A[\gamma])$. Hence x appears both in the term and in its type, so it is of internal degree non-zero, and since it is the case for all the variables, the term $\mathsf{coh}_{\Gamma,A}[\gamma]$ is necessarily of internal degree non-zero. Consider a term t of the form $\Delta \vdash \mathsf{op}_{\Gamma,A}[\gamma] : A[\gamma]$, we necessarily have, by Lemma 76 that $\dim t = \dim \Gamma$. This term is of degree 0 if and only if it contains a variable x of dimension $\dim \Gamma$. Such a variable can only appear in $\mathrm{Var}(v[\gamma])$ for v a maximal variable of γ, and the term $v[\gamma]$ is of dimension 0 if and only if it contains such a variable, hence t is of dimension 0 if and only if at least one argument of γ is of degree 0. □

3.3.3 Degree and invertibility

We now explore the connection between the the degree of a term and its invertibility. As we have not implemented this as a feature in CaTT, we keep this part less formal and only give sketches of proofs. We keep the formalization and the proof of this construction for future work, as it is very involved and does not work well with the inductive structure of the theory.

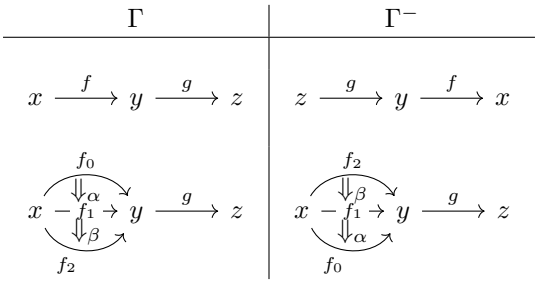
Inversion of a ps-context. In order to prove associativity, we construct a term associated to any given term, that we then prove to be an inverse. In order to construct this term, we define given a ps-context Γ its *inversion* that we denote Γ^-. We can describe this operation using the ordering on ps-contexts as follows, and it corresponds intuitively to reversing the direction of the cells of maximal dimension. For each sequence of consecutive variables

$$y_0 \lhd f_1 \lhd y_1 \lhd f_2 \lhd \ldots \lhd f_n \lhd y_n$$

with $\dim y_i = \dim \Gamma - 1$ and $\dim f_i = \dim \Gamma$ that is of maximal length, we replace this sequence by its reverse in Γ^-

$$y_n \lhd f_n \lhd y_{n-1} \lhd f_{n-1} \lhd \ldots \lhd f_1 \lhd y_0$$

We give a few instance of computation of inversion of ps-contexts, using diagrammatic notations



By definition, the inversion of a ps-context is constructed from a linear order on its variables, hence it is a ps-context. Moreover, one can check that $\partial^-(\Gamma^-) = \partial^+(\Gamma)$ and $\partial^+(\Gamma^-) = \partial^-(\Gamma)$.

Inversion of a term. We start by associating to each term t of strictly positive degree a term that we denote t^-, and that we define by induction

– For a term of the form $t = \mathsf{op}_{\Gamma,A}[\gamma]$, suppose the maximal arguments of γ are $(u_0, \dots, u_n)$, then they are all of degree strictly positive, then we pose $t^- = \mathsf{op}_{\Gamma^-,A^-}[\gamma^-]$, where γ^- is obtained from γ by posing for each variable x in Γ, $x[\gamma^-] = x[\gamma]^-$ if x is of dimension maximal in Γ and $x[\gamma^-] = x[\gamma]$ otherwise.

– For a term of the form $t = \mathsf{coh}_{\Gamma,A}[\gamma]$, we pose $t^- = \mathsf{coh}_{\Gamma,A^-}[\gamma]$.

– For a type of the form $A = t \underset{B}{\longrightarrow} u$, we define A^- to be the type $u \underset{B}{\longrightarrow} t$

We illustrate this construction in particular the first case with an example, as the description we have given of it is not completely formal: Consider the term

$$(x : \star) \vdash \mathtt{id}\ x : x \to x$$

its inverse is given by itself. consider the an associativity witness

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

then its inverse is given by the term

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

In a generic context, if we consider for instance the following term

```
let ex (x : *) (f : x -> x)
    = comp (assoc (id x) f f ) (id (comp (id x) (comp f f)))
```

its inverse is given by

```
let ex- (x : *) (f : x -> x)
    = comp (id (comp (id x) (comp f f))) (assoc- (id x) f f )
```

Lemma 79. *For any term $\Gamma \vdash t : A$ derivable in CaTT, the term $\Gamma \vdash t^- : A^-$ is also derivable.*

Proof. We prove this by induction on the term of degree non-zero t.

– For a term $\Delta \vdash \mathsf{coh}_{\Gamma,A}[\gamma] : A[\gamma]$, by the rule (COH), we have $A = t \underset{B}{\longrightarrow} u$, and $\Gamma \vdash A$ with $\mathrm{Var}(t) \cup \mathrm{Var}(B) = \mathrm{Var}(\Gamma)$ and $\mathrm{Var}(u) \cup \mathrm{Var}(B) = \mathrm{Var}(\Gamma)$. Hence the type $\Gamma \vdash A^-$ is also derivable and also satisfies ($\mathrm{C}_{\mathsf{coh}}$), so the type $\Delta \vdash \mathsf{coh}_{\Gamma,A^-}[\gamma] : A[\gamma]^-$ is derivable.

– For a term of the form $\Delta \vdash \mathsf{op}_{\Gamma,t \underset{A}{\longrightarrow} u}[\gamma]$, first note that the condition (C_{op}) imply than we have $\partial^-(\Gamma^-) \vdash t : A$ and $\partial^+(\Gamma^-) \vdash u : A$ with the condition of using all the variables. Since Γ^- has inverted sources and target with respect to Γ, we also have $\Gamma^- \vdash u \underset{A}{\longrightarrow} t$ satisfying (C_{op}). Moreover, for all variable in $\Gamma^- \vdash x : B$, we also have $\Gamma \vdash x : B$, unless x is of maximal dimension, in which case $\Gamma \vdash x : B^-$. These match exactly the type of $x[\gamma^-]$, and hence we have $\Delta \vdash \gamma^- : \Gamma^-$. Moreover, since neither t or u contain variables of maximal dimensions, we have $t[\gamma] = t[\gamma^-]$ and $u[\gamma] = u[\gamma^-]$, hence $(u \underset{A}{\longrightarrow} t)[\gamma^-] = (t \underset{A}{\longrightarrow} u[\gamma])^-$. This shows that the term $\Delta \vdash \mathsf{op}_{\Gamma^-,u \underset{A}{\longrightarrow} t}[\gamma^-] : ((t \underset{A}{\longrightarrow} u)[\gamma])^-$.

□

One can also check that operation of taking the inverse is an involution, that is for all term of strictly positive degree, we have $(t^-)^- = t$.

Cancellation witnesses. We claim that given a term t of degree non-zero, we can define a *cancellation witness* associated to t, that we denote $c(t)$. This term is easy to define for a term of the form $\Delta \vdash \mathsf{coh}_{\Gamma,A}[\gamma]$, by posing $c(t)$ to be the following term

$$\mathsf{coh}_{\Gamma,\mathtt{comp}\ (\mathsf{coh}_{\Gamma,A}[\mathrm{id}_\Gamma])\ (\mathsf{coh}_{\Gamma,A^-}[\mathrm{id}_\Gamma])\to\mathtt{id}\ (\partial^-(A))}[\gamma]$$

we then have a derivation of $\Delta \vdash c(t) : \mathtt{comp}\ t\ t^- \to \mathrm{id}_{\partial^-(t)}$. We conjecture that such a term $c(t)$ is also definable by induction for a term t of strictly positive degree , although explicit example even in simple cases hint that this requires heavier automation and meta-operations that we have as of now. This is a problem that we reserve for future work, as defining this automation and proving its correctness may have drastic consequences on the practical use of CaTT. Our complete conjecture is the following:

Conjecture 80. *For any term $\Gamma \vdash t : A$ of strictly positive degree, we can define a term $c(t)$ which has the same variables as t, and such that $\Gamma \vdash c(t) : \mathtt{comp}\ t\ t^- \to \mathtt{id}\ (\partial^-(t))$.*

Under this conjecture, we have a reinterpretation of the degree of a term, as a way to quantify the invertibility of a term.

Proposition 81. *A term $\Gamma \vdash t : A$ of degree strictly positive is invertible.*

Proof. We prove this result by using coinduction, indeed, we have proven that for any term $\Gamma \vdash t : A$, there exists a term $\Gamma \vdash t^- : A^-$, along with a term $\Gamma \vdash c(t) : \mathtt{comp}\ t\ t^- \to \partial^-(A)$, and this term is of degree strictly positive, hence by coinduction, it is invertible. Moreover, t^- is also of degree strictly positive, hence it defines a cancellation witness $\Gamma \vdash c(t^-) : \mathtt{comp}\ t^-\ (t^-)^- \to \partial^-(A^-)$ which is invertible. By simplification, we can simplify the type of this cancellation witness as follows $\Gamma \vdash c(t^-) : \mathtt{comp}\ t^-\ t \to \partial^+(A)$, which then proves that the term is invertible. □

Non-invertible terms. We claim that the terms of degree zero are necessarily non-invertible. Even though we do not provide a full proof of this fact, we give the general sketch of one. First consider the case of a variable $\Gamma \vdash x : t \to u$, there is no derivation that allows us to derive a term of type $u \to t$ from x, but there might be a variable y in our context which is of the type $u \to t$, or which allows for a derivation of a term of this type. If there is such a variable, there is still no derivation of the cancellation witness, unless the context contains yet another variable that allows to derive it. By iterating this argument, in order to make x invertible, the context would need to contain infinitely many variables, which is impossible, hence x cannot be invertible. The intuition is similar for other terms of degree 0, and making such a term invertible would require infinitely many variables in the context. This proof does not formalize well, due to the arbitrary nature of the variables that can be in a context. However, we do not make use of this result, and only present it to help building up our intuition, so we limit ourselves to this informal proof.

Source and target.

Lemma 82. *Consider a term $\Delta \vdash t : A$ of degree strictly positive, then we necessarily have $\deg(\partial^-(t)) = \deg(\partial^+(t)) = \deg(t) - 1$.*

Proof. Consider the variable of dimension maximal x in t, since it is of degree strictly positive, it has to appear also in $\partial^-(t)$ and $\partial^+(t)$, in which it is necessarily maximal. Since moreover $\dim \partial^-(t) = \dim \partial^+(t) = \dim t - 1$, we conclude that $\deg(\partial^-(t)) = \deg(\partial^+(t)) = \deg(t) - 1$. □

This gives a clearer picture on the nature of the terms in the theory CaTT. A term of degree n has necessarily its source and target of degree $n-1$, which themselves necessarily have their source and target of degree $n-2$ and so on, until hitting the degree 0. Necessarily, all these intermediate degrees are invertible terms, and moreover an invertible term (i.e., of degree strictly positive) is always between two terms of the same degree. Note that we do not provide any restrictions on the source and the target of a term of degree 0, since one can have variables of arbitrary types. For instance, we may consider the context

$$\Gamma = (x : \star, f : x \to x, a : \texttt{id}\ x \to f)$$

In this context, the term a is of degree 0 but its source is of degree 1 and its target is of degree 0. We can nevertheless give a restriction if we only consider contexts with only variables whose source and target are of degree 0, then a term of degree 0 in such a context has its source and target of degree 0.

3.3.4 Degree for automation

We describe here a feature that we wish to add to CaTT for in a foreseeable future, but that we have not implemented yet, using the notion of degree. We expect to add a feature to compute the degree of a term. As it is not very hard for the user to read the degree of the expression of the term, this computation could be completely internal, or it could be made accessible by the user via a syntax like

```
deg G |- t
```

where `G` is the context in which the term is defined and `t` is the expression defining the term. If our Conjecture 80 holds, we could implement our algorithm that given a term of degree strictly positive, computes its inverse and all the higher cells witnessing that the term is invertible. Of course since there are infinitely many such cells, the algorithm would be able to compute any given one of them, but not all at once. A good syntactic convention to manage such a feature is also not completely clear at the moment, our first instinct is that it should have new keywords to introduce the inverse such as for instance

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

in order to define the inverse term of the term `assoc`. The issue with such an approach is that we then need to provide a syntax for the invertible terms of type

```
comp (assoc f g h) (assoc- f g h) -> id (comp (comp f g) h)
```

which is part of the invertibility condition. Let us assume that we introduce also a dedicated syntax for these witnesses, for instance

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

Then the invertible term of type

```
comp (assoc- f g h) (assoc f g h) -> id (comp f (comp g h))
```

part of the invertibility condition could be obtained as `can (inv (assoc f g h))` Then for extensive developments, one would typically need to access the inverses of these terms, using for instance `inv (can (inv (assoc f g h)))`, and their cancellation witnesses, generating expressions of the form `can (can (inv (assoc a b c)))`, that would be very difficult to parse for a human reader. Another option is to introduce a meta-theoretic tactic, that generates the right cell when it is called. For instance, one could imagine writing an expression like

```
let example (x : *) (y : *) (f : x -> y)
                            (f- : y -> x) (a : comp f- f -> id y)
                    (z : *) (g : y -> z)
  : comp f- (comp f g) -> g
  = comp3 (invertibility (assoc f- f g))
          (rw a g)
          (unitl g)
```

where `comp3` is the ternary composition. In this example, the first of 2-cell that we compose is expressed as `invertibility (assoc f- f g)`, which would be understood by the system as a term automatically generated from the fact that `assoc f- f g` is invertible. Then using the environment, and in particular in our example, the type of the term that has to be `comp f- (comp f g) -> comp (comp f- f) g`, the system would generated the appropriate term - here `assoc- f- f g`. This however raises other questions, that we have not addressed, and leave for future improvements of our implementation of CaTT. It is unclear for instance, how to guarantee enough information for the system to be able to determine uniquely a cell from an expression of the form `invertibility t`.

3.4 Functorialization

We now introduce the second meta-operation that we have implemented in the theory CaTT. This meta-operation is much more constrained than the suspension, and only yields valid terms under some conditions, that are well-expressed using the notion of internal degree of a variable. This has been introduced briefly in [17], but we extend and complete the study of this operation.

Intuition and example. The intuition behind this operation is that all the operations that are introduced in the theory of weak ω-category act on the cells similarly to how functors act on a category. A functor not only acts on the objects of a category, but on the morphisms as well; in our case each operation not only acts on the cells it is defined, but on higher cells as well. Consider for instance the composition

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

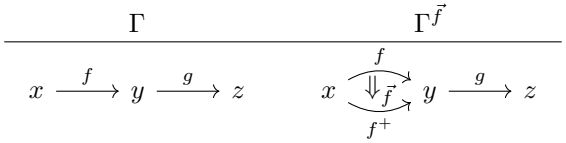
this operation is functorial with respect to both of its arguments `f` and `g`. What we mean by this statement is that for instance fixing `g`, and replacing `f` by a two cell `a : f -> f'`, we can define an operation that sends `a` onto a 2-cell of type `comp f g -> comp f' g`. We have already met this operation, and called it the right whiskering `rw`.

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

The functorialization thus provides us with a new way to understand the right whiskering as the action of the composition with a fixed morphism on the right on a 2-cell by functoriality. We describe formally the operation that generates the term `rw` from the term `comp`. The idea is to duplicate the variable `f` into `f` and `f'`, and add an higher cell that relates the original variable `f` to its duplicate `f'`. The return type is computed by applying the original term (in our example `comp`), once to the original variable `f`, and once to the duplicate `f'`, while keeping the other arguments the same. This idea generalizes to duplicating several variables at once which is in fact the operation we formally describe. For a variable x, we always denote x^+ its duplicate and $\vec{x}$ the higher dimensional cell that relates x and x^+.

3.4.1 Functorialization of contexts

We start by defining an operation on contexts, which given a context Γ and set of variables $X \subset \mathrm{Var}(\Gamma)$, produces a new context $\Gamma^{\vec{X}}$ that we call the *functorialization* of the context Γ with respect to X. Intuitively, this operation duplicates all the variables x that are in X into a variable x^+, and add a higher cell $\vec{x}$ of type $x \to x^+$ to the context. When $X = \{x\}$ is a singleton, we only denote $\Gamma^{\vec{x}}$ the functorialized context. We give a visual example, where we only represent a context by the finite globular set it corresponds to, that emphasizes our example of composition and right whiskering.



At this point we have defined the functorialization of a context with respect to any of its variables. In the particular case of the ps-context that we study later, we add the condition that the variables are of dimension maximal for reasons that have to do with the typing of the term constructors.

Definition. We proceed with defining this operation by induction on the context. For the rest of this section, we assume that we have fixed two variables, that we call x' and f, and that are completely fresh. This can be done by extending the alphabet. If we work in de Bruijn indices, we just use x' and f to denote two integers, and renormalize the context to ensure that the variables conditions are still met.

$$\varnothing^{\vec{X}} = \varnothing$$

$$(\Gamma, x : A)^{\vec{X}} = \begin{cases} (\Gamma^{\vec{X}}, x : A, x^+ : A, \vec{x} : x \to x') & \text{if } x \in X \\ (\Gamma^{\vec{X}}, x : A) & \text{otherwise} \end{cases}$$

In addition, we define a substitution that we denote $\iota_X(\Gamma)$, or simply ι which has the same syntactic expression as id_Γ, i.e., $\iota_X(\Gamma) = \langle y \mapsto y \rangle_{y \in \mathrm{Var}(\Gamma)}$. We pick a different name than id_Γ in order to suggest that $\iota_X(\Gamma)$ is not destined to be typed as going from Γ to Γ. Since the action of a substitution on types and terms is purely syntactical, and $\iota_X(\Gamma)$ has the same expression as id_Γ, it follows, that it acts the same way on types and terms, and thus for all type A, we have $A[\iota_X(\Gamma)] = A$.

Lemma 83. *Functorialization can be performed successively in any order. For any partition $X = X_0 \sqcup X_1$, we have $\Gamma^{\vec{X_1}\,\vec{X_0}} = \Gamma^{\vec{X}}$*

Proof. We can prove this by induction on the context Γ

- For the empty context $\varnothing$, we have $\varnothing^{\vec{X_1}\,\vec{X_0}} = \varnothing$ and $\varnothing^{\vec{X}} = \varnothing$.
- For a context of the form $(\Gamma, x : A)$ where $x \notin X$, we have $(\Gamma, x : A)^{\vec{X_1}\,\vec{X_0}} = (\Gamma^{\vec{X_1}\,\vec{X_0}}, x : A)$ and $(\Gamma, x : A)^{\vec{X}} = (\Gamma^{\vec{X}}, x : A)$. By induction, the equality $\Gamma^{\vec{X_1}\,\vec{X_0}} = \Gamma^{\vec{X}}$ gives the result.
- For a context of the form $(\Gamma, x : A)$ where $x \in X$, then either $x \in X_0$ or $x \in X_1$. If $x \in X_0$ then since X_0 and X_1 are disjoint, $x \notin X_1$ and hence $(\Gamma, x : A)^{\vec{X_1}} = (\Gamma^{\vec{X_1}}, x : A)$. This lets us compute
$$(\Gamma, x : A)^{\vec{X_1}\,\vec{X_0}} = (\Gamma^{\vec{X_1}\,\vec{X_0}}, x : A, x^+ : A, \vec{x} : x \to x^+)$$
And the equality $\Gamma^{\vec{X_1}\,\vec{X_0}} = \Gamma^{\vec{X}}$ obtained by induction allow us to conclude. If $x \in X_1$, then $(\Gamma, x : A)^{\vec{X_1}} = (\Gamma^{\vec{X_1}}, x : A, x^+ : A, \vec{x} : x \to x^+)$. Since neither of x, x^+ of $\vec{x}$ are in X, this lets us again compute that
$$(\Gamma, x : A)^{\vec{X_1}\,\vec{X_0}} = (\Gamma^{\vec{X_1}\,\vec{X_0}}, x : A, x^+ : A, \vec{x} : x \to x^+)$$
which gives the expected result.

□

Correctness. We have defined the operation of functorialization of a context purely syntactically and we need to ensure that it only ever produces valid contexts.

Lemma 84. *The functorialized of a well-defined context with respect to any of its variable is a well-defined contexts. The following rule is admissible*
$$\frac{\Gamma \vdash}{\Gamma^{\vec{X}} \vdash}(X \subset \mathrm{Var}(\Gamma))$$
Moreover, in this case, the substitution $\iota_X(\Gamma)$ is a well defined substitution $\Gamma^{\vec{X}} \vdash \iota_X(\Gamma) : \Gamma$

Proof. We prove these two results mutually, by induction on the context, by first assuming that $X = \{x\}$ is a singleton

- For a context of the form $(\Gamma, x : A)$, an explicit computation shows
$$(\Gamma, x : A)^{\vec{x}} = (\Gamma, x : A, x' : A, f : x \underset{A}{\longrightarrow} x')$$
Denote $\Gamma_{x'} = (\Gamma, x : A, x'A)$, then the derivation of $\Gamma, x : A \vdash$ gives a derivation of $\Gamma \vdash A$, which by weakening, gives a derivation of $\Gamma, x : A \vdash A$. By applying the rule (CE) to

this derivation, we construct a derivation of $\Gamma_{x'} \vdash$. Using this derivation, we can give a derivation of $\Gamma^{\vec{x}} \vdash$

$$\dfrac{\dfrac{\dfrac{\Gamma_{x'} \vdash \qquad x : A \in \Gamma_{x'}}{\Gamma_{x'} \vdash x : A}(\text{VAR}) \qquad \dfrac{\Gamma_{x'} \vdash \qquad x' : A \in \Gamma_{x'}}{\Gamma_{x'} \vdash x' : A}(\text{VAR})}{\Gamma_{x'} \vdash x \underset{A}{\longrightarrow} x'}(\text{HOM})}{\Gamma^{\vec{x}} \vdash}(\text{CE})$$

Moreover, the substitution $\Gamma, x : A \vdash \mathrm{id}_{(\Gamma,x:A)} : \Gamma, x : A$ can be weakened on the left twice, yielding the valid substitution $\Gamma^{\vec{x}} \vdash \iota_x(\Gamma) : \Gamma$.

– For a context of the form $(\Gamma, y : A)$ where $y \neq x$, we have $(\Gamma, y : A)^{\vec{x}} = (\Gamma^{\vec{x}}, y : A)$. By the induction cases, we have derivations for the judgments, $\Gamma^{\vec{x}} \vdash$ and $\Gamma^{\vec{x}} \vdash \iota_x(\Gamma) : \Gamma$. From the derivation of $\Gamma, y : A \vdash$, we extract a derivation of $\Gamma \vdash A$, and since $A[\iota_x(\Gamma)] = A$, applying the substitution $\iota_x(\Gamma)$ gives a derivation of $\Gamma^{\vec{x}} \vdash A$. A single application of the rule (CE) then gives a derivation for $\Gamma, y : A^{\vec{x}} \vdash$ as follows

$$\dfrac{\Gamma^{\vec{x}} \vdash A}{\Gamma^{\vec{x}}, y : A \vdash}(\text{CE})$$

Moreover, by weakening of the derivation of the substitution, we get a derivation of $(\Gamma, y : A)^{\vec{x}} \vdash \iota_x(\Gamma) : \Gamma$, which lets us get a derivation of $(\Gamma, y : A)^{\vec{x}} \vdash \iota_x(\Gamma, y : A) : \Gamma, y : A$ as follows

$$\dfrac{(\Gamma, y : A)^{\vec{x}} \vdash \iota_x : \Gamma \qquad \Gamma, y : A \vdash \qquad (\Gamma, y : A)^{\vec{x}} \vdash y : A}{(\Gamma, y : A)^{\vec{x}} \ \vdash \langle \iota_x(\Gamma), y \mapsto y \rangle : \Gamma, y : A}(\text{SE})$$

By Lemma 83, it suffices to apply the singleton case successively for all elements of X to prove the result for an arbitrary finite set of variable. We only consider finite sets, since by assumption $X \subset \mathrm{Var}(\Gamma)$ which is finite. □

Target context of the functorialized. Additionally, we define a second substitution associated to a functorialization of a context, that we denote $\wr_X(\Gamma)$, or simply $\wr$, and that fixes every variable but the ones in X, each variable $x \in X$ being sent to x^+. We can give an inductive definition for this substitution as follows

$$\wr_X(\varnothing) = \langle\rangle$$
$$\wr_X(\Gamma, x : A) = \begin{cases} \langle \wr_X(\Gamma), x \mapsto x^+ \rangle & \text{if } x \in X \\ \langle \wr_X(\Gamma), x \mapsto x \rangle & \text{otherwise} \end{cases}$$

This definition also yields a well-defined substitution of the same type as $\iota_X(\Gamma)$.

Lemma 85. *For all context $\Gamma \vdash$ and all set of variables $X \subset \mathrm{Var}(\Gamma)$, the judgment $\Gamma^{\vec{X}} \vdash \wr_X(\Gamma) : \Gamma$ is derivable.*

Proof. We proceed by induction over the context Γ, following the definition of $\wr_X(\Gamma)$.

– For the empty context $\varnothing$, we have $\wr_X(\varnothing) = \langle\rangle$, and since $\varnothing^{\vec{X}} = \varnothing$, the rule (EC) give $\varnothing^{\vec{X}} \vdash$, and the rule (ES) then gives $\varnothing^{\vec{X}} \vdash \wr_X(\varnothing) : \varnothing$.

– For a context of the form $(\Gamma, x : A)$ where $x \in X$, by induction we have a derivation of $\Gamma^{\vec{X}} \vdash \wr_X(\Gamma) : \Gamma$. Lemma 84 ensures that $(\Gamma, x : A)^{\vec{X}} \vdash$, thus by weakening this substitution (c.f. Proposition 2) we get a derivation of $(\Gamma, x : A)^{\vec{X}} \vdash \wr_X(\Gamma) : \Gamma$. We can then apply the rule (SE) in order to get the following derivation

$$\dfrac{(\Gamma, x : A)^{\vec{X}} \vdash \wr_X(\Gamma) : \Gamma \qquad (\Gamma, x : A) \vdash \qquad \dfrac{(\Gamma, x : A)^{\vec{X}} \vdash \qquad x^+ : A \in (\Gamma, x : A)^{\vec{X}}}{(\Gamma, x : A)^{\vec{X}} \vdash x^+ : A}\text{(VAR)}}{(\Gamma, x : A)^{\vec{X}} \vdash \langle \wr_X(\Gamma), x \mapsto x^+ \rangle : (\Gamma, x : A)}\text{(SE)}$$

– For a context of the form $(\Gamma, x : A)$ with $x \notin X$, we have by induction a derivation of $\Gamma^{\vec{X}} \vdash \wr_X(\Gamma) : \Gamma$, which gives by weakening a derivation of $\Gamma, y : A^{\vec{X}} \vdash \wr_X(\Gamma) : \Gamma$. This lets us construct the following derivation

$$\dfrac{(\Gamma, x : A)^{\vec{X}} \vdash \wr_X(\Gamma) : \Gamma \qquad (\Gamma, x : A) \vdash \qquad \dfrac{(\Gamma, x : A)^{\vec{X}} \vdash \qquad \times : A \in (\Gamma, x : A)^{\vec{X}}}{(\Gamma, x : A)^{\vec{X}} \vdash \times : A}\text{(VAR)}}{(\Gamma, x : A)^{\vec{X}} \vdash \langle \wr_X(\Gamma), x \mapsto \times \rangle : (\Gamma, x : A)}\text{(SE)}$$

□

3.4.2 Functorialization of a maximal variable in a ps-context

In order to define the functorialization of an operation, we are particularly interested in the case of the functorialization of a ps-context with respect to one of its variables of maximal dimension.

Functorialization preserves ps-contexts. The process of functorializing a context is very reminiscent of the algorithm checking that a given context is a ps-context, and in particular of the rule (PSE), in that it introduces a new variable which is a copy of a known variable, and a variable from the known variable to the new copy. This analogy can be exploited to show the following lemma

Lemma 86. *Let $\Gamma \vdash_{\mathsf{ps}}$ be a ps-context and $x \in \mathrm{Var}(\Gamma)$ be a locally maximal variable in Γ, then $\Gamma^{\vec{x}} \vdash_{\mathsf{ps}}$ is again a ps-context.*

Proof. We first treat separately the case of the context D^0, whose functorialization with respect to its unique variable is D^1, which is a ps-context. So from now on, we may assume that Γ is not the ps-context D^0, and hence all its locally maximal variables are of dimension at least 1. In this case, the derivation of $\Gamma \vdash_{\mathsf{ps}}$ necessarily is of the form

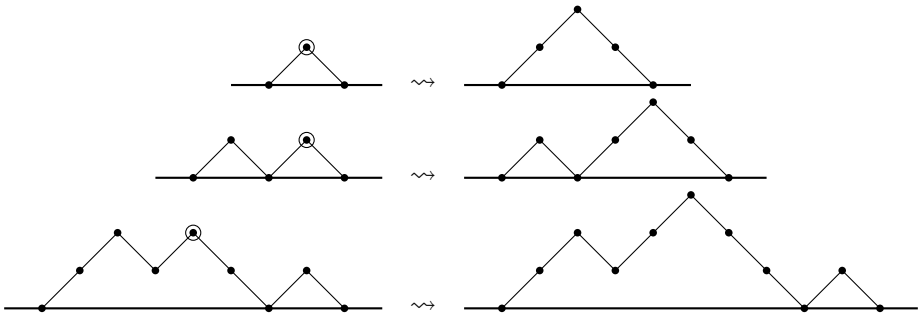
$$\dfrac{\dfrac{\dfrac{\mathcal{D}_{\Gamma'}}{\Gamma' \vdash_{\mathsf{ps}} y : A}}{\Gamma', z : A, x : y \underset{A}{\longrightarrow} z \vdash_{\mathsf{ps}} x : y \underset{A}{\longrightarrow} z}\text{(PSE)}}{\Gamma', z : A, x : y \underset{A}{\longrightarrow} z \vdash_{\mathsf{ps}} z : A}\text{(PSD)}}{\mathcal{D}}$$

where $\mathcal{D}_{\Gamma'}$ is a derivation of the judgment $\Gamma' \vdash_{\mathsf{ps}} y : A$, and $\mathcal{D}$ is a derivation tree that finishes, from a derivation of $\Gamma', z : A, x : y \to z \vdash_{\mathsf{ps}} z : A$ to a derivation of $\Gamma \vdash_{\mathsf{ps}}$. We can then obtain a derivation of $\Gamma', z : A, x : y \to z^{\vec{x}} \vdash_{\mathsf{ps}} z : A$ as follows

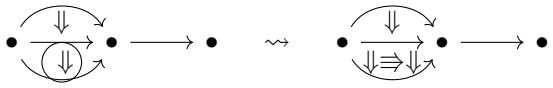
$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\mathcal{D}_{\Gamma'}}{\Gamma' \vdash_{\mathsf{ps}} y : A}}{\Gamma', z : A, x : y \underset{A}{\to} z \vdash_{\mathsf{ps}} x : y \underset{A}{\to} z}(\text{PSE})}{\Gamma', z : A, x : y \underset{A}{\to} z^{\vec{x}} \vdash f : x \to x'}(\text{PSE})}{\Gamma', z : A, x : y \to z^{\vec{x}} \vdash_{\mathsf{ps}} x' : y \underset{A}{\to} z}(\text{PSD})}{\Gamma', z : A, x : y \to z^{\vec{x}} \vdash_{\mathsf{ps}} z : A}(\text{PSD})$$

We can then finish this derivation into a derivation of $\Gamma^{\vec{X}} \vdash_{\mathsf{ps}}$ by following the structure of derivation tree $\mathcal{D}$. □

Combinatorial interpretation. As we had done in the case of the suspension, we can provide an interpretation on the combinatorial structure of the ps-contexts. Ge give a few examples of ps-contexts graphically represented by their structure, and since we always functorialize with respect to a given variable of dimension locally maximal, we specify this variable by circling a position on our diagram.



Graphically, this operation can be understood as selecting one peek and pulling this peek alone one dimension higher. In terms of the underlying globular sets, we give another visualization for the last example



Functorialization with respect to a set of variables. We can generalize our result to functorialize ps-contexts with respect to a set of variables, instead of a single variable. Indeed, consider a set of variables X such that all variables in X are locally maximal, and consider a variable $x \in X$, by Lemma 86, we know that $\Gamma^{\vec{x}}$ is again a ps-context. But $\Gamma^{\vec{X}} = \Gamma^{\vec{x}^{X-\vec{\{x\}}}}$, and now all the variables in $X - \{x\}$ are locally maximal in $\Gamma^{\vec{x}}$. We can thus iterate this construction, which proves the following result

Lemma 87. *For all ps-contexts $\Gamma \vdash_{\mathsf{ps}}$ and all set X of locally maximal variables in Γ, the context $\Gamma^{\vec{X}}$ is again a ps-context.*

Source and target of the functorialized. In the particular case of functorializing with respect to a set of maximal variables, we can compute explicitly the source and target of the functorialized context.

Lemma 88. *If $\Gamma \vdash_{\mathsf{ps}}$ is a ps-context and $X \subset \mathrm{Var}(\Gamma)$ is a set of variables of dimension maximal in Γ, then $\partial^-(\Gamma^{\vec{X}}) = \Gamma$, and the substitution $\Gamma^{\vec{X}} \vdash \iota_X(\Gamma) : \Gamma$ is the source substitution ∂^-. Moreover, the substitution $\imath_X(\Gamma)$ defines an isomorphism between Γ and $\partial^+(\Gamma^{\vec{X}})$, which commutes with the substitution ∂^+*

$$\begin{array}{ccc} \Gamma^{\vec{X}} & \xrightarrow{\partial^+} & \partial^+(\Gamma^{\vec{X}}) \\ & \searrow^{\imath_X(\Gamma)} & \wr \downarrow \imath_X(\Gamma) \\ & & \Gamma \end{array}$$

Proof. We handle separately the case $\Gamma = (x : \star)$, where $\Gamma^{\vec{x}} = (x : \star, x^+ : \star, \vec{x} : x \to x^+)$, and we can compute explicitly that $\partial^-(\Gamma^{\vec{x}}) = \Gamma$, $\partial^+(\Gamma^{\vec{x}}) = (x^+ : \star)$, and $\imath = \langle x \mapsto x^+ \rangle$ defines an isomorphism $\langle x \mapsto x^+ \rangle$. In the other cases, we follow the inductive definitions:

- If $\Gamma = (\Gamma', z : A, x : y \to z)$ with $x \in X$, denote $X' = X - \{x\}$. Since x is of maximal dimension in Γ, it follows that $\vec{x}$ is of maximal dimension in $\Gamma^{\vec{X}}$, and hence the rule for computing source gives $\partial^-(\Gamma^{\vec{X}}) = (\partial^-({\Gamma'}^{\vec{X'}}), z : A, x : y \to z)$. By induction $\partial^-({\Gamma'}^{\vec{X'}}) = \Gamma'$, which shows the equality. Similarly, the target is computed by $\partial^+(\Gamma^{\vec{X}}) = (\partial^-({\Gamma'}^{\vec{X'}}), z : A, x^+ : y \to z)$, and by induction $\partial^+([]{\Gamma'}^{\vec{X'}}] \vdash \imath_{X'}(\Gamma') : \Gamma'$ induces an isomorphism. By completing this, we get the isomorphism

$$\partial^+(\Gamma^{\vec{X}}) \vdash \langle \imath_{X'}(\Gamma'), z \mapsto z, x \mapsto x^+ \rangle : \Gamma$$

- If $\Gamma = (\Gamma', z : A; x : y \to z)$ with $x \notin X$, since $\dim \Gamma^{\vec{X}} = \dim \Gamma + 1$, x is not of dimension maximal in $\Gamma^{\vec{X}}$, hence, $\partial^-(\Gamma^{\vec{X}}) = (\partial^-({\Gamma'}^{\vec{X}}), z : A, x : y \to z)$, and the induction gives $\partial^-(\Gamma^{\vec{X}}) = \Gamma$. Similarly for the target, we have $\partial^+(\Gamma^{\vec{X}}) = (\partial^+({\Gamma'}^{\vec{X}}), z : A, x : y \to z)$, and completing the isomorphism $\imath_X(\Gamma')$ yields an isomorphism

$$\partial^+(\Gamma^{\vec{X}}) \vdash \langle \imath_X(\Gamma'), z \mapsto z, x \mapsto x \rangle : \Gamma$$

□

3.4.3 Functorialization of a term

We can use our definition of the functorialization of a ps-context in order to define the functorialization of any term, and thus generate new terms from previously defined ones, as we have already presented for the suspension. This operation is a bit more complicated to define, so we split its definition into different cases, depending on the term we are computing with. In all the cases, we always functorialize a term with respect to a list of variables of internal degree 0 in this term.

Functorialization of a variable. The definition of the functorialization of a variable is straightforward, we simply require that $x^{\vec{x}} = \vec{x}$. Note that this is the only possible case, since we require the variable with respect to which we functorialize to appear in the term.

Lemma 89. *Given a variable x in a context such that $\Gamma \vdash x : A$, we also have $\Gamma^{\vec{x}} \vdash x^{\vec{x}} : x \underset{A}{\longrightarrow} x^+$.*

Proof. Since we have proved that we always have a derivation of $\Gamma^{\vec{x}} \vdash$, and that moreover $f : x \underset{A}{\longrightarrow} x' \in \Gamma^{\vec{x}}$ by definition, we can apply the rule (VAR) in order to get a derivation of $\Gamma^{\vec{x}} \vdash \vec{x} : x \underset{A}{\longrightarrow} x^+$. □

Functorialization of the declaration of operation. In order to define the functorialization for a general term, we first study the case of the declaration of an operation.

Lemma 90. *Suppose that we have a ps-context $\Gamma \vdash_{\mathsf{ps}}$, together with a type $\Gamma \vdash A$ such that Γ, A declares a valid operation, then for all set X of maximal variables in Γ, the functorialized pair $\Gamma^{\vec{X}}, \mathsf{op}_{\Gamma,A}[\mathrm{id}_\Gamma] \to \mathsf{op}_{\Gamma,A}[\mathfrak{s}]$ also declares a valid operation.*

Proof. We have already proved by Lemma 86 that the context $\Gamma^{\vec{X}}$ is a ps-context, so we have $\Gamma^{\vec{X}} \vdash_{\mathsf{ps}}$. We denote t the term $t = \mathsf{op}_{\Gamma,A}[\mathrm{id}_\Gamma]$, since the pair (Γ, A) declares a valid operation, we have a derivation of the judgment $\Gamma \vdash t : A$. Moreover, the variables of t are specified by the identity morphism of Γ: It contains all the variables of Γ, hence $\mathrm{Var}(t) = \mathrm{Var}(\Gamma)$. Since $\Gamma = \partial^-(\Gamma^{\vec{X}})$, this proves that we have $\partial^-(\Gamma^{\vec{X}}) \vdash t : A$, with $\mathrm{Var}(t) \cup \mathrm{Var}(A) = \mathrm{Var}(\Gamma)$. Since we have a substitution $\partial^+(\Gamma^{\vec{X}}) \vdash \mathfrak{s} : \Gamma$, we can as well build the term $\partial^+(\Gamma) \vdash t[\mathfrak{s}] : A[\mathfrak{s}]$. Note that (Γ, A) satisfying the conditions for declaring an operation operation forces A not to contain any variable of maximal dimension of Γ, which are the only variables that can be in X. Since A does not contain any variable in X, we have $A[\mathfrak{s}] = A$, hence the derivation $\partial^+(\Gamma^{\vec{X}}) \vdash t[\mathfrak{s}] : A$. Finally, $\mathrm{Var}(t[\mathfrak{s}]) = \mathrm{Var}(\mathfrak{s})$, and since $\mathfrak{s}$ is an isomoprhism, it contains all the variables of $\partial^+(\Gamma^{\vec{X}})$, which proves the equality $\mathrm{Var}(t[\mathfrak{s}]) \cup \mathrm{Var}(A) = \mathrm{Var}(\partial^+(\Gamma^{\vec{X}}))$. We thus have verified all the conditions shows that the pair $(\Gamma^{\vec{X}}, \mathsf{op}_{\Gamma,A}[\mathrm{id}_\Gamma] \underset{A}{\longrightarrow} \mathsf{op}_{\Gamma,A}[\mathfrak{s}])$ declares a valid operation. □

In order to simplify the notations, given the declaration of an operation defined by the pair $\Gamma \vdash_{\mathsf{ps}}$ and $\Gamma \vdash A$ satisfying the side condition, we denote $(\Gamma, A)^{\vec{X}}$ the declaration of the operation defined in Lemma 90.

Functorialization of a general term. We now define the functorialization of a general term, with respect to a set X of variables of internal degree 0 in this term. Note that having variables of internal degree 0 forces the term to be of degree 0, and hence this eliminates terms constructed with the term constructor coh which are all of degree strictly positive. We define by induction the functorialization for terms as follows

$$\mathsf{op}_{\Gamma,A}[\gamma]^{\vec{X}} = \mathsf{op}_{(\Gamma,A)^{\vec{X}_\gamma}}[\gamma^{\vec{X}}]$$

where X_γ is the set of variables y in Γ such that $\mathrm{Var}(y[\gamma]) \cap X \neq \emptyset$, and the functorialization is defined on substitution by induction with $\langle\rangle^{\vec{X}} = \langle\rangle$ and we pose $\langle \gamma, x \mapsto t \rangle^{\vec{X}}$ to be

$$\begin{cases} \langle \gamma^{\vec{X}}, x \mapsto t \rangle & \text{if } x \in X_\gamma \\ \langle \gamma^{\vec{X}}, x \mapsto t, x' \mapsto t[\mathfrak{s}_{\mathrm{Var}(t) \cap X}(\Gamma)], f \mapsto t^{\mathrm{Var}(\vec{t}) \cap X} \rangle & \text{otherwise} \end{cases}$$

Lemma 91. *Given a substitution $\Delta \vdash \gamma : \Gamma$ and an arbitrary set of variables $X \subset \mathrm{Var}(\Delta)$, for all variable x in $\mathrm{Var}(\Gamma)$, we have $x[\gamma] = x[X^{\vec{X}} \gamma]$*

Proof. This result is immediate from the definition of $X^{\vec{X}}\gamma$: it only associates to a variable x the term $x[\gamma]$, if x already has an image by γ. We can prove this by induction.

- For the substitution $\Delta \vdash \langle\rangle : \varnothing$, the result is vacuously true since there is no variable in $\varnothing$.
- For a substitution of the form $\Delta \vdash \langle \gamma, x \mapsto t\rangle : (\Gamma, x : A)$ where $x \notin X_\gamma$, we have for the variable x, $x[\langle \gamma, x \mapsto t\rangle] = t$, and since $\langle \gamma, x \mapsto t\rangle^{\vec{X}} = \langle \gamma^{\vec{X}}, x \mapsto t\rangle$, we also have $x[\langle \gamma, x \mapsto t\rangle^{\vec{X}}] = t$. For any other variable variable y in $(\Gamma, x : A)$, y is actually a variable in Γ and $y[\langle \gamma, x \mapsto t\rangle^{\vec{X}}] = y[\gamma^{\vec{X}}]$, which by induction is equal to $y[\gamma]$.
- For a substitution of the form $\Delta \vdash \langle \gamma, x \mapsto t\rangle : (\Gamma, x : A)$ where $x \in X_\gamma$, the proof is very similar: we have for the variable x, $x[\langle \gamma, x \mapsto t\rangle] = t$, and since

  $$\langle \gamma, x \mapsto t\rangle^{\vec{X}} = \langle \gamma^{\vec{X}}, x \mapsto t, x^+ \mapsto t[\mathfrak{s}], \vec{x} : x \to x^+\rangle$$

  we also have $x[\langle \gamma, x \mapsto t\rangle^{\vec{X}}] = t$. For any other variable variable y in $(\Gamma, x : A)$, y is actually a variable in Γ and $y[\langle \gamma, x \mapsto t\rangle^{\vec{X}}] = y[\gamma^{\vec{X}}]$, which by induction is equal to $y[\gamma]$.

□

Correctness. Under the assumption that we functorialize a term with respect to variables of internal degree 0, the definition of functorialization only produces valid terms.

Theorem 92. *Given a term $\Delta \vdash t : B$ and a set of variables X of internal degree 0 in t, the following judgment is derivable*

$$\Delta^{\vec{X}} \vdash t^{\vec{X}} : t \underset{B}{\longrightarrow} t[\mathfrak{s}]$$

We prove this result by induction, but it is a bit technical to perform, so in order to simplify it, we state and prove the following lemma, on which relies each of the induction steps

Lemma 93. *Assuming that Theorem 92 holds for all terms of depth at most t, for any substitution $\Delta \vdash \gamma : \Gamma$ and any set of variables X the judgment $\Gamma^{\vec{X}} \vdash \gamma^{\vec{X}} : \Gamma^{\vec{X}_\gamma}$ is derivable.*

Proof. We prove this result by induction on the derivation of the substitution $\Delta \vdash \gamma : \Gamma$.

- For a substitution of the form $\Delta \vdash \langle\rangle : \varnothing$, we have both $\langle\rangle^{\vec{X}} = \langle\rangle$ and $\varnothing^{\vec{X}_\Gamma} = \varnothing$, giving a derivation of the judgment $\Delta^{\vec{X}} \vdash \langle\rangle : \varnothing$.
- For a substitution of the form $\Delta \vdash \langle \gamma, x \mapsto t\rangle : (\Gamma, x : A)$ with $x \notin X_{\langle \gamma, x \mapsto t\rangle}$, we have by induction that $\Delta^{\vec{X}} \vdash \gamma^{\vec{X}} : \Gamma^{\vec{X}_\gamma}$, and since $x \notin X_{\langle \gamma, x \mapsto t\rangle}$, we have the equality $X_{\langle \gamma, x \mapsto t\rangle} = X_\gamma$. Moreover, from the derivation of $\Delta \vdash t : A[\gamma]$, we extract by weakening a derivation of $\Delta^{\vec{X}} \vdash t : A[\gamma]$. Moreover, since all the variables of A are in Γ, we have $A[\gamma] = A[\gamma^{\vec{X}}]$. This lets us apply the rule (SE) in order to obtain a derivation as follows

  $$\frac{\Delta^{\vec{X}} \vdash \gamma^{\vec{X}} : \Gamma^{\vec{X}_\gamma} \qquad \Gamma^{\vec{X}_\gamma}, x : A \vdash \qquad \Delta^{\vec{X}} \vdash t : A[\gamma^{\vec{X}}]}{\Delta^{\vec{X}} \vdash \langle \gamma^{\vec{X}}, x \mapsto t\rangle : \Gamma^{\vec{X}_\gamma}, x : A}(\text{SE})$$

  The judgment proved by this derivation can in fact be rewritten as

  $$\Delta^{\vec{X}} \vdash \gamma^{\vec{X}} : (\Gamma, x : A)^{\vec{X}_{\langle \gamma, x \mapsto t\rangle}}$$

– For a substitution of the form $\Delta \vdash \langle \gamma, x \mapsto t \rangle : (\Gamma, x : A)$ with $x \in X_\gamma$, then by induction, we have a derivation of $\Delta^{\vec{X}} \vdash \gamma^{\vec{X}} : \Gamma^{\vec{X}_\gamma}$. Moreover, we have a derivation of $\Delta \vdash t : A[\gamma]$, and since A only uses variables in Γ, the equality $A[\gamma] = A[X^{\vec{X}}\gamma]$ lets us use the rule (SE) as follows

$$\frac{\Delta^{\vec{X}} \vdash \gamma^{\vec{X}} : \Gamma^{\vec{X}_\gamma} \qquad \Gamma^{\vec{X}_\gamma}, x : A \vdash \qquad \Delta^{\vec{X}} \vdash t : A[\gamma^{\vec{X}}]}{\Delta^{\vec{X}} \vdash \langle \gamma^{\vec{X}}, x \mapsto t \rangle : \Gamma^{\vec{X}_\gamma}, x : A}(\text{SE})$$

Moreover, Lemma 84 ensures that $(\Gamma, x : A)^{\vec{X}_{\langle \gamma, x \mapsto t \rangle}} \vdash$ is derivable, and since the variable x is not in Γ, we have that $\Gamma^{\vec{X}_\gamma} = \Gamma^{\vec{X}_{\langle \gamma, x \mapsto t \rangle}}$, which gives the equality

$$(\Gamma, x : A)^{\vec{X}_{\langle \gamma, x \mapsto t \rangle}} = (\Gamma^{\vec{X}_\gamma}, x : A, x^+ : A, \vec{x} : x \to x^+)$$

Moreover, Theorem 92 for the term t of depth at most d ensures that $\Delta^{\vec{X}} \vdash t^{\vec{X}} : t \xrightarrow[A[\gamma]]{} t[\wr]$. With the help of these two judgment and our previous derivation, we can construct the derivation tree indicated in Figure 3.1. All the leaves of this tree being derivable, this gives a derivation for the judgment $\Delta^{\vec{X}} \vdash \langle \gamma, x \mapsto t \rangle^{\vec{X}} : (\Gamma, x : A)^{\vec{X}_{\langle \gamma, x \mapsto t \rangle}}$

$$\dfrac{\dfrac{\Delta^{\vec{X}} \vdash \langle \gamma^{\vec{X}}, x \mapsto t \rangle : \Gamma^{\vec{X}_\gamma}, x : A \qquad \Gamma^{\vec{X}_\gamma}, x : A, x^+ : A \vdash \qquad \Delta^{\vec{X}} \vdash t[\wr] : A[\gamma^{\vec{X}}]}{\Delta^{\vec{X}} \vdash \langle \gamma^{\vec{X}}, x \mapsto t, x^+ \mapsto t[\wr] \rangle : \Gamma^{\vec{X}_\gamma}, x : A, x^+ : A}\text{(SE)} \qquad (\Gamma, x : A)^{\vec{X}_{\langle \gamma, x \mapsto t \rangle}} \vdash \qquad \Delta^{\vec{X}} \vdash t^{\vec{X}} : t \xrightarrow[A[\gamma^{\vec{X}}]]{} t[\wr]}{\Delta^{\vec{X}} \vdash \langle \gamma, x \mapsto t \rangle^{\vec{X}} : (\Gamma, x : A)^{\vec{X}_{\langle \gamma, x \mapsto t \rangle}}}\text{(SE)}$$

Figure 3.1: Derivation tree for the judgment $\Delta^{\vec{X}} \vdash \langle \gamma, x \mapsto t \rangle^{\vec{X}} : (\Gamma, x : A)^{\vec{X}_{\langle \gamma, x \mapsto t \rangle}}$

□

With this lemma, we are equipped to prove our Theorem 92

Proof of Theorem 92. We prove this result by induction on the depth of the term t. Lemma 84 already proves it for the terms of depth 0 (i.e., the variables), so it suffices to assume the result for all terms of depth at most d, and prove it for a term of depth $d+1$. Consider a term $\Delta \vdash t : B$ of degree $d+1$, together with a (non-empty) set X of variables of internal degree 0 in t. Then as discussed previously, necessarily t is of the form $t = \mathsf{op}_{\Gamma,A}[\gamma]$.First note that any variable in X_Γ is necessarily of maximal dimension in Γ. Indeed, suppose that we have a variable $y \in X_\Gamma$, then there exists $x \in \mathrm{Var}(y[\gamma]) \cap X$, and being in X implies that x is of internal degree 0, hence $\dim x = \dim \Gamma$. On the other hand since $x \in \mathrm{Var}(y[\gamma])$, we have that $\dim x \leq \dim y[\gamma] = \dim y$. This implies that $\dim y = \dim \Gamma$. By Lemma 86, this proves that $(\Gamma, A)^{\vec{X}_\gamma}$ is declares a valid operation. We then apply Lemma 93, which we can apply thanks to the induction hypothesis, and shows that $\Delta^{\vec{X}} \vdash \gamma^{\vec{X}} : \Gamma^{\vec{X}_\gamma}$. This lets us use the introduction rule for op to conclude. □

3.4.4 Implementation and restrictions

We now discuss the practical applications of our definition of functorialization and show how it can be used to reduce proofs, and then focus, using the notations that we introduce, on the technical condition that lead us define only functorializations with respect to lists of variables of internal degree 0, as opposed to any variable.

Implementation as a meta-operation. We have implemented in our software for CaTT the algorithm that we have presented to compute functorializations of general terms. As for the suspension, we chose to implement it as a meta-operation, outside of the kernel, in such a way that any computation of a functorialization produces an expression, that is then checked to be a valid expression in the bare theory CaTT. In particular, our meta operation computes an expression without bothering to check the condition on the internal degree of the term, as whenever the condition fails, the expression will simply fail to type check in the kernel. Although this is a safety guard against our proof that this operation is valid, it again costs a lot computationally as it forces the system to check terms that are known for meta-theoretical reasons to type check. A future more application focused implementation of CaTT should then use the functorialization as a primitive operation to create new terms directly in the kernel, without having to check them first. This replace the problem of type checking an entire term by simply verifying out condition that a list of variables is of internal degree 0, which is computationally much lighter.

Notations. In the case of the suspensions, we had chosen to make it completely implicit as it can be easily guessed by the system by looking at the dimension of the arguments. As a consequence, we cannot use the same trick for the functorialization, indeed writing for instance

```
let example (x : *) (a : id x -> id x) (b : id x -> id x)
   = comp a b
```

would then yield a ambiguous expression: There would be no way of telling if it designates the suspension of `comp` applied to the arguments `a` and `b` which defines the vertical composition of `a` and `b`, or the functorialization of `comp` applied to the arguments `a` and `b`, which is their horizontal composition. Thus, we keep an explicit syntax to indicate when a term should be functorialized, and specifically, we chose to write between square brackets `[]` the places where a functorialization happens. So in our previous ambiguous example, the way it was written

without brackets indicates that the operation to perform is a suspension, and for defining the version where `comp` is functorialized with respect to both its variables, one would write instead

```
let example (x : *) (a : id x -> id x) (b : id x -> ud x)
    = comp [a] [b]
```

Similarly, one can define for instance the right whiskering by functorializing `comp` as follows

```
let rw (x : *) (y : *) (f : x -> y) (f' : x -> y) (a : f -> f')
               (z : *) (g : y -> z)
    = comp [a] g
```

but one could also define directly the right whiskering of an identity 2-cell as follows

```
let id-rw (x : *) (y : *) (f : x -> y)
               (z : *) (g : y -> z)
    = comp [(id f)] g
```

In practice, we do not even need to define the right whiskering anymore, as we can simply write `comp [a] g` in place of `rw a g` every time we use a right whiskering, thus saving extra definitions.

Examples of functorializations. We provide now a few examples of definitions that can be avoided using functorializations. First, note that the left whiskering is obtained by functorializing the term `comp` with respect to the second variable, and the horizontal composition is obtained by functorializing the term `comp` with respect to both the variables as follows

```
let lw (x : *) (y : *) (f : x -> y)
               (z : *) (g : y -> z) (g' : y -> z) (b : g -> g')
    = comp f [b]
let hcomp (x : *) (y : *) (f : x -> y) (f' : y -> z) (a : f -> f')
                  (z : *) (g : y -> z) (g' : y -> z) (b : g -> g')
   = comp [a] [b]
```

We also suggest adding the possibility of performing two functorializations successively, even though we have not implemented it yet, allowing to define for instance the "right whiskering" of a 3-cell composed with a 1-cell along a 0-cell

```
let comp3,1,0 (x : *) (y : *) (f : x -> y)
                              (f' : x -> y) (a : f -> f')
                                            (a' : f -> f') (A : a -> a')
                      (z : *) (g : y -> z)
    = comp [[A]] g
```

Note that the right whiskering of a 3-cell composed with a 2-cell along 0-cell can also be defined using two successive functorializations as follows

```
let comp3,2,0 (x : *) (y : *) (f : x -> y)
                              (f' : x -> y) (a : f -> f')
                                            (a' : f -> f') (A : a -> a')
                      (z : *) (g : y -> z)
                              (g' : y -> z) (b : g -> g')
    = comp [[A]] [b]
```

In this case, we functorialize all the inner brackets at once first, then all the outer brackets and so on. We can also define the composition of two 3-cells along a 0-cell

```
let comp3,3,0 (x : *) (y : *) (f : x -> y)
                              (f' : x -> y) (a : f -> f')
                                            (a' : f -> f') (A : a -> a')
                      (z : *) (g : y -> z)
                              (g' : y -> z) (b : g -> g')
                                            (b' : g -> g') (B : b -> b')
    = comp [[A]] [[B]]
```

Combining the suspension and the functorialization, we can also access the right whiskering of a three cell with a 3-cell along a 1-cell as follows

```
let comp3,2,1 (x : *) (y : *) (f : x -> y)
                              (g : x -> y) (a : f -> g)
                                           (a' : f -> g) (A : a -> a')
                              (h : x -> y) (b : g -> h)
    = comp [A] b
```

In this case, the functorialization is computed first, and then our dimension analysis shows how many times the term has to be suspended. We can access as well the composition of two 3-cells along a 1-cell

```
let comp3,3,1 (x : *) (y : *) (f : x -> y)
                              (g : x -> y) (a : f -> g)
                                           (a' : f -> g) (A : a -> a')
                              (h : x -> y) (b : g -> h)
                                           (b' : g -> h) (B : b -> b')
    = comp [A] [B]
```

and finally, using only the suspension, we can define the composition of two 3-cells along a 2-cell

```
let comp3,3,1 (x : *) (y : *) (f : x -> y)
                              (g : x -> y) (a : f -> g)
                                           (b : f -> g) (A : a -> b)
                                           (c : f -> g) (B : b -> c)
    = comp A B
```

All these examples show how by combining suspensions and functorializations we can define all the six possible compositions of a 3-cell with another cell, using only the operation `comp`.

Our final version of the Eckmann-Hilton morphism. We have formalized the Eckmann-Hilton morphism, that we have presented in Section 3.1 using functorializations as well as suspensions (but without the possibility to add successive functorializations). This is the third implementation of this term, with various degrees of automation, as we have also defined it using the suspension only in Section 3.2. The resulting file[3] contains 76 definitions, compared to the 85 of the file using only the suspension, and the number of lines reduced from 476 to 422, corresponding to a gain of 11%. The size of the file however dropped from $22k$ to $20k$, which corresponds also to 10%, indicating that even though it let us avoid some definitions, it did not contribute a lot to simplify the other definitions, contrarily to the suspension.

[3] `https://github.com/ThiBen/catt/blob/master/examples/eckmann-hilton-versions/eh-susp-no-func.catt`

Towards a more general functorialization. It is a little frustrating to only define the functorialization for variables of internal degree 0, as it implies for instance that even though we can functorialize the composition with respect to both its variables to define the horizontal composition of 2-cells, we cannot functorialize the coherences associated to the composition like the associator, to higher coherences like the associator for the horizontal composition of 2-cells. Indeed, the associator is defined by

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

which is of dimension 2 and the variable of highest dimension are `f`, `g` and `h`, which are of dimension 1. Hence we still have to define the associator for the horizontal composition manually. It also does not match our intuition very well. We believe that it is possible to generalize the functorialization to arbitrary variables, or at least to the variables such that no other variable in the context depends on them (i.e., the variables that we have chosen to be explicit), but we now explain why this problem has to be hard. We consider the simplest example of a term with a variable of internal degree 1, the identity 1-cell.

```
coh id (x : *) : x -> x
```

If we give a meaning to the functorialization of this term, we should then be able to write

```
let id[] (x : *) (y : *) (f : x -> y) = id [f]
```

and we expect the type of this term to be `id x -> id y`. But this type is ill-formed, since `id x` and `id y` are not parallel. Instead, the situation can be described as in the following square diagram

$$\begin{array}{ccc} \texttt{x} & \xrightarrow{\texttt{id x}} & \texttt{x} \\ {\scriptstyle\texttt{f}}\downarrow & & \downarrow{\scriptstyle\texttt{f}} \\ \texttt{f} & \xrightarrow[\texttt{id y}]{} & \texttt{y} \end{array}$$

So the only way to define the functorialized version of `id` is to encode this square diagram in a globular setting and define the functorialization such that `id[]` reduces to the term

$$\mathsf{coh}_{(x:\star,y:\star,f:x\to y),\texttt{comp (id } x\texttt{) } f\to\texttt{comp } f\texttt{ (id } y\texttt{)}}[\langle x\mapsto x, y\mapsto y, f\mapsto f\rangle]$$

The situation gets even more complicated if we define the composition of two identities

```
let compid (x : *) = comp (id x) (id x)
```

Now functorializing this simple term yields the situation described by the following diagram

$$\begin{array}{ccccc} \texttt{x} & \xrightarrow{\texttt{id x}} & \texttt{x} & \xrightarrow{\texttt{id x}} & \texttt{x} \\ {\scriptstyle\texttt{f}}\downarrow & & \downarrow{\scriptstyle\texttt{f}} & & \downarrow{\scriptstyle\texttt{f}} \\ \texttt{f} & \xrightarrow[\texttt{id y}]{} & \texttt{y} & \xrightarrow[\texttt{id y}]{} & \texttt{y} \end{array}$$

and defining this functorialization requires encoding this diagram in a globular setting, and in particular it requires encoding the composition of two squares in a globular setting. Already on these two very simple examples, with only argument of dimension 0, the functorialization

is difficult to define, and it goes increasingly complicated as the dimension and the depth of the terms increases, requiring to encode harder and harder cubical composition in a globular setting. We present in Section 6.2 a type theory for working with a cubical variant of the weak ω-categories; we believe that an approach close to the one would be better suited to compute functorializations.

3.5 The category $\mathcal{S}_{\mathsf{PS},\infty}$

Our goal is to prove Theorem 44 that we have claimed in Section 2.5. For this we introduce tools to understand subtle syntactic conditions relating the variables of a term and the variables of its type. The main idea is to exhibit $\mathcal{S}_{\mathsf{PS},\infty}$ as a colimit of the form

$$\mathcal{S}_{\mathsf{PS},\infty} = \operatorname{colim}\left(\mathcal{S}_{\mathsf{PS},0} \to \mathcal{S}_{\mathsf{PS},1} \to \mathcal{S}_{\mathsf{PS},2} \to \dots\right)$$

that reproduces the iterative construction of Θ_n in the Grothendieck-Maltsiniotis definition of weak ω-categories.

Coherence depth. We introduce the notion of *coherence depth* of a term, type or substitution in order to relate our definition of the type theory CaTT with the Grothendieck-Maltsiniotis definition of weak ω-groupoids. Note that this definition is distinct from the one of depth, we use for syntactic reasoning.

$$\begin{aligned}
\operatorname{cd}(v) &= 0 & \operatorname{cd}(\mathsf{op}_{\Gamma,A}[\gamma]) &= \max(\operatorname{cd}(A)+1, \operatorname{cd}(\gamma)) \\
& & \operatorname{cd}(\mathsf{op}_{\Gamma,A}[\gamma]) &= \max(\operatorname{cd}(A)+1, \operatorname{cd}(\gamma)) \\
\operatorname{cd}(\star) &= 0 & \operatorname{cd}(t \underset{A}{\to} u) &= \max(\operatorname{cd}(A), \operatorname{cd}(t), \operatorname{cd}(u)) \\
\operatorname{cd}(\langle\rangle) &= 0 & \operatorname{cd}(\langle \gamma, x \mapsto t\rangle) &= \max(\operatorname{cd}(\gamma), \operatorname{cd}(t))
\end{aligned}$$

We consider the subcategory $\mathcal{S}_{\mathsf{CaTT},n}$, whose objects are the objects are the same as $\mathcal{S}_{\mathsf{CaTT}}$, but whose morphisms are substitutions whose all terms are of depth at most n, and the full subcategory $\mathcal{S}_{\mathsf{PS},n}$ of $\mathcal{S}_{\mathsf{CaTT},n}$ whose objects are the contexts Γ such that $\Gamma \vdash_{\mathsf{ps}}$ holds. To emphasize this construction, we sometimes denote $\mathcal{S}_{\mathsf{CaTT},\infty}$ the syntactic category of CaTT. We then have the following inclusions.

$$\begin{array}{ccccccccc}
\mathcal{S}_{\mathsf{GSeTT}} = \mathcal{S}_{\mathsf{CaTT},0} & \hookrightarrow & \mathcal{S}_{\mathsf{CaTT},1} & \hookrightarrow & \mathcal{S}_{\mathsf{CaTT},2} & \hookrightarrow & \cdots & \hookrightarrow & \mathcal{S}_{\mathsf{CaTT},\infty} \\
\uparrow & & \uparrow & & \uparrow & & & & \uparrow \\
\mathcal{S}_{\mathsf{PS}} = \mathcal{S}_{\mathsf{PS},0} & \hookrightarrow & \mathcal{S}_{\mathsf{PS},1} & \hookrightarrow & \mathcal{S}_{\mathsf{PS},2} & \hookrightarrow & \cdots & \hookrightarrow & \mathcal{S}_{\mathsf{PS},\infty} \\
\uparrow & & & & & & & & \\
\mathcal{G}^{\mathrm{op}} & & & & & & & &
\end{array}$$

The rules (COH) and (COH') We now want to prove that each of the category $\mathcal{S}_{\mathsf{PS},n}$ is equivalent to Θ_n^{op}, and to do so, we characterize the term introduction rules as a formalism to add lifts for some coadmissible pairs of morphisms in the category $\mathcal{S}_{\mathsf{PS},n}$. For the rules (OP) and (COH') this turns out to be fairly straightforward, however, for the rule (COH), this correspondence is more involved. Recall that when introducing the theory CaTT, we have defined two versions (COH) and (COH') for the same rule

$$\frac{\Gamma \vdash_{\mathsf{ps}} \qquad \Gamma \vdash A \qquad \Delta \vdash \gamma : \Gamma}{\Delta \vdash \mathsf{coh}_{\Gamma,A}[\gamma] : A[\gamma]}$$

The difference between these two is that it does not apply under the same conditions: We require the rule (COH) to apply only under the condition

$$(\mathrm{C}_{\mathsf{coh}})\ \ A \text{ is of the form } t \underset{B}{\longrightarrow} u \text{ with } \begin{cases} \mathrm{Var}(t) \cup \mathrm{Var}(B) = \mathrm{Var}(\Gamma) \\ \mathrm{Var}(u) \cup \mathrm{Var}(B) = \mathrm{Var}(\Gamma) \end{cases}$$

whereas for applying the rule (COH') we require the condition

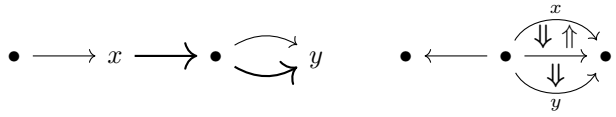
$$(\mathrm{C}'_{\mathsf{coh}})\ \ \mathrm{Var}(A) \ = \mathrm{Var}(\Gamma).$$

We have claimed these two rules to be equivalent, and thus we reason mostly with the rule (COH), whereas in the implementation, we use the rule (COH'). We now show that these two rules are indeed equivalent, to ensure that the theory CaTT that we study theoretically is indeed the same as the one we have implemented.

3.5.1 Equivalence between the rules (COH) and (COH')

Our goal is now to show that the rules (COH) and (COH') can be used interchangeably, so that we can prove Theorem 44 using the rule (COH'). It is immediate from the rule that any application of (COH) is a particular case of application of (COH'), thus changing the rule (COH) for the rule (COH') may only create new terms. Our goal is to show the converse: Any application of the rule (COH') can in fact be proved to satisfy $(\mathrm{C}'_{\mathsf{coh}})$. We first remark that a ps-context Γ is never empty, hence the condition $\mathrm{Var}(A) = \mathrm{Var}(\Gamma)$ implies that A cannot be the type $\star$, so it has to be of the form $t \underset{B}{\longrightarrow} u$. The rest of this section is dedicated to showing that in this case, we have both the equalities $\mathrm{Var}(t) \cup \mathrm{Var}(B) = \mathrm{Var}(\Gamma)$ and $\mathrm{Var}(u) \cup \mathrm{Var}(B) = \mathrm{Var}(\Gamma)$, and to achieve this, we introduce tools that let us analyze precisely how the variables in a term and in its type are related. Precisely, we show that whenever the type of a term contains two parallel variables, the term has to contain a path between them, and in the case of locally maximal variables, their is only one path. Hence whenever the common type of two terms contain two parallel variables delimiting a path of maximal variables, both of the terms have to contain the same path, which forces them to contain the same variables.

Path of variables. Given a context Γ in the theory GSeTT and two variables x, y of same type in Γ, we call a *path from x to y* in Γ a family of variables of the form $f_1, \dots, f_n$ such that we have a family x_i, with $x_0 = x$ and $x_n = y$, satisfying $\Gamma \vdash f_i : x_{i-1} \to x_i$. We give an visual interpretation of various contexts with two marked variables and a path between them, in the form of finite globular sets.



To simplify our vocabulary, we say that $(f_1, \dots, f_n)$ is a path between x and y in order to say that it is either a path from x to y or a path from y to x. We also denote a path $x \rightsquigarrow y$ to express that it is a path from x to y. By convention, the empty path defines a path $x \rightsquigarrow x$ for every variable x, and we say that a term t contains the empty path from x to x if the variable x appears both in the source and in the target of t.

Paths in ps-contexts. In a ps-context, if there is a non-empty path $(f_1, \dots, f_n)$ from x to y, then we have by definition of $\lhd$ that $x \lhd f_1 \lhd x_1 \lhd f_2 \lhd \dots \lhd f_n \lhd y$, and hence by transitivity, we necessarily have $x \lhd y$.

Lemma 94. *For any two variables x, y of the same type in a ps-context, with $x \lhd y$, there exists a path $x \rightsquigarrow y$.*

Proof. We prove this result by induction over the number of variables f such that $x \lhd f \lhd y$. Note that since x and y have the same type, they cannot be of locally maximal dimension, hence there exists a variable f whose source is x, and this is necessarily the smallest variable greater than x, thus it is between x and y.

- If there is only one variable f between x and y, then we necessarily have $\Gamma \vdash f : x \to y$, and hence (f) defines a path $x \rightsquigarrow y$.
- If there are $n > 1$ variables between x and y, then consider the smallest variable f such that $x \lhd f$, then we necessarily have $\Gamma \vdash f : x \to z$, with $z \lhd y$, and there are strictly less than n variables between z and y. Hence by induction we have a path $(f_1, \dots, f_n)$ from z to y, and by appending f, we construct the path $(f, f_1, \dots, f_n)$ from x to z.

□

Paths of locally maximal variables in a ps-context. In a ps-context Γ there can be at most one path $x \rightsquigarrow y$ if $\dim x = \dim y = \dim \Gamma - 1$. Indeed, consider two paths $(f_0, \dots, f_n)$ and $(g_0, \dots, g_m)$, since f_0 and g_0 are of dimension maximal and share the same source, they are necessarily equal, and we can rewrite the second path as $(f_0, g_1, \dots, g_m)$. Doing this successively on all variables of the second path, we show that it necessarily use the variables f_i. Moreover, since the variables all increase along a path, a same path cannot reach twice the variable y, hence we have $m = n$, and the two paths are equal. This proves the following result

Lemma 95. *In a ps-context Γ, given two variables x and y with the same type and such that $\dim x = \dim y = \dim \Gamma - 1$, there exists exactly one path $x \rightsquigarrow y$ in Γ.*

Variables of degree 0 in the type of a term. The variables of degree 0 in the source of a term are particularly important in our study, as they constraint the variables that the terms must have. Namely, they must span a path from it inside the term.

Lemma 96. *For a context Δ of the theory GSeTT, consider a term $\Delta \vdash t : A$ in the theory CaTT, together with a variable $x \in \mathrm{Var}(\partial^-(t))$ which is of degree 0 in $\partial^-(t)$, then either $x \in \mathrm{Var}(\partial^+(t))$ or there exists a variable $y \in \mathrm{Var}(\partial^+(t))$ together with a path of variables from x to y in Δ.*

Proof. We prove this result by induction

- For a variable term $\Delta \vdash z : t \underset{A}{\longrightarrow} u$. Since Δ is a context of GSeTT, t and u are necessarily variables, and we thus have $t = x$, posing $y = u$, the variable z defines a path $x \rightsquigarrow y$.
- For a term of the form $\Delta \vdash \mathsf{op}_{\Gamma, t \underset{A}{\longrightarrow} u}[\gamma] : t[\gamma] \underset{A[\gamma]}{\longrightarrow} u[\gamma]$. Consider a variable $x \in \mathrm{Var}(t[\gamma])$ of degree 0. Then there exists a variable v in $\partial^-(\Gamma)$ such that $x \in v[\gamma]$, hence we have $\dim v \geq \dim x$ and since x is of degree 0 in $\partial^-(t)$ we have, $\dim x = \dim t[\gamma] \geq \dim \Gamma - 1$. Since $v \in \mathrm{Var}(\partial^-(\Gamma))$, this implies $\dim v = \dim \Gamma - 1$. If v is locally maximal in Γ, then we also have $v \in \partial^+(\Gamma)$, in which case $\mathrm{Var}(v[\gamma]) \subset \mathrm{Var}(u[\gamma])$, and in particular $x \in \mathrm{Var}(u[\gamma])$.

If v is not locally maximal in Γ, then there exists a variable w in $\mathrm{Var}(\partial^+(\Gamma))$ of the same type as v, and by Lemma 94, there exists a path $(f_1, \dots, f_n)$ of variables of dimension maximal in Γ, from v to w. We can now apply the induction to each of the terms $f_0[\gamma], \dots, f_n[\gamma]$. There is a variable $x_1 \in \partial^+(f_1[\gamma])$, together with a path from x to x_1 in $f_1[\gamma]$, then there exists a variable x_2 in $\mathrm{Var}(\partial^+(f_2[\gamma]))$ such that $f_2[\gamma]$ contains a path from x_1 to x_2, and so on. In the end we get a variable $x_n \in \mathrm{Var}(w[\gamma])$, together with a family of paths. By concatenating these paths, we create a path from x to x_n. Moreover, since $w \in \partial^+(\Gamma)$, we have $w \in \mathrm{Var}(u)$, and hence $x_n \in \mathrm{Var}(u[\gamma])$.

– For a term of the for $\Delta \vdash \mathsf{coh}_{\Gamma, t \xrightarrow[A]{} u}[\gamma] : t[\gamma] \longrightarrow_{A[\gamma]} u[\gamma]$, and consider a variable x of degree 0 in $t[\gamma]$. This implies that we have a variable v in $\mathrm{Var}(t)$ such that $x \in \mathrm{Var}(v[\gamma])$. Moreover, since x is of degree 0 in $t[\gamma]$, the term $t[\gamma]$ itself is of degree 0, hence the term $\mathsf{coh}_{\Gamma, t \to u}[\gamma]$ is of degree 1 (since it is necessarily of degree strictly positive) so it is of dimension $\dim \Gamma + 1$, and x is of degree 1 in it. Thus $\dim x = \dim \Gamma$ and v is of dimension maximal in Γ, so this shows that $v \notin \mathrm{Var}(A)$. By the rule (COH'), we necessarily have $v \in \mathrm{Var}(u) \cup \mathrm{Var}(A)$, which implies $v \in \mathrm{Var}(u)$, and hence $x \in \mathrm{Var}(u[\gamma])$. Hence the empty path defines a path $x \rightsquigarrow x$ in out term.

□

We have a symmetric result concerning the variables that are of degree 0 in the target of a term. We do not repeat the proof as it is completely symmetric.

Lemma 97. *For a context Δ of the theory GSeTT, consider a term $\Delta \vdash t : A$ in the theory CaTT, together with a variable $y \in \mathrm{Var}(\partial^+(t))$ of degree 0 in $\partial^+(t)$, then either $y \in \mathrm{Var}(\partial^-(t))$ or there exists a variable $x \in \mathrm{Var}(\partial^-(t))$ together with a path of variables from x to y in Γ.*

Variables of degree 0. We present a converse to the previous remark: every variable of degree 0 in a term must be part of a path from a variable of its source to a variable of its target.

Lemma 98. *For a context Δ in the theory GSeTT, consider a term $\Delta \vdash t : A$ of dimension non-zero, in the theory CaTT, together with a variable $x \in \mathrm{Var}(t)$ of degree 0. Then there exist two variables $x^- \in \mathrm{Var}(\partial^-(t))$ and $x^+ \in \mathrm{Var}(\partial^+(t))$ such that t contains a path from x^- to x^+ that goes through x.*

Proof. We prove this result by induction on the depth of the term.

– For a variable term $\Delta \vdash x : t \to u$, since the context Δ is derivable in the theory GSeTT, necessarily t and u are variables, and we chose $x^- = t$, $x^+ = u$.

– For a term of the form $\Delta \vdash \mathsf{op}_{\gamma, t \xrightarrow[A]{} u}[\gamma] : t[\gamma] \longrightarrow_{A[\gamma]} u[\gamma]$, consider a variable x of degree 0 in this term, then there exists v such that $x \in \mathrm{Var}(v[\gamma])$, and since $\dim v \geq \dim x = \dim \Gamma$, v is necessarily of dimension maximal in Γ. Hence there exists two variables $v^- \in \mathrm{Var}(\partial^-(\Gamma))$ and $v^+ \in \mathrm{Var}(\partial^+(\Gamma))$ such that there is a path from v^- to v^+ which goes through v in Γ. Applying the induction on the term $v[\gamma]$, we get two variables x_0^- and x_0^+ such that $v[\gamma]$ contains the path from x_0^- to x_0^+ that goes through x. Then applying the lemma 97 to all the variables before v in the path from v^- to v^+ and the lemma 96 to all the variables after v in the same path, we get families of paths

$$x^- \rightsquigarrow x_{n-1}^-, x_{n-1}^- \rightsquigarrow x_{n-2}^-, \dots, x_1^- \rightsquigarrow x_0^-$$
$$x^+ \rightsquigarrow x_{n-1}^+, x_{n-1}^+ \rightsquigarrow x_{n-2}^+, \dots, x_1^+ \rightsquigarrow x_0^+$$

concatenating all these paths with the path $x_0^- \rightsquigarrow x_0^+$, we get a path $x^- \rightsquigarrow x^+$ containing x. Moreover, Since $v^- \in \partial^-(\Gamma)$, we have $v^- \in \mathrm{Var}(t)$, and hence $x^- \in \mathrm{Var}(t[\gamma])$, and similarly $v^+ \in \partial^+(\Gamma)$ hence $x^+ \in \mathrm{Var}(u[\gamma])$.

– For a term of the form $\Delta \vdash \mathsf{coh}_{\Gamma,A}[\gamma]$, then it is of degree strictly positive, and do not contain any variable of degree 0, thus the result is vacuously true.

□

Uniqueness. We now prove the key lemma to show that the rules (COH) and (COH') are equivalent. The structure of the proof is a little involved, since it requires an induction on the depth of the terms, whose induction case uses a lemma which depends on the induction hypothesis and that is itself proved by induction on the dimension. In order to present it in a clearer way, we present these as two separate lemmas that we prove by mutual induction.

Lemma 99. *Consider a term $\Delta \vdash t : A$ in a ps-context Γ, and two variables $x, y \in \mathrm{Var}(t)$ of degree 0 and of the same type. Then $x = y$.*

Proof. We prove this result by mutual induction with Lemma 100. This part is by induction on the depth of the term.

– For a variable term $\Gamma \vdash x : A$, the only variable is can ever contain is x itself.

– For a term of the form $\Delta \vdash \mathsf{op}_{\Gamma,t\to u}[\gamma] : t[\gamma] \to u[\gamma]$, we necessarily have two variables $v, w \in \mathrm{Var}(\Gamma)$ with $x \in \mathrm{Var}(v[\gamma])$ and $y \in \mathrm{Var}(w[\gamma])$. Lemma 100 then applies by mutual induction and shows that v and w necessarily have the same type. Since x and y are of degree 0, necessarily v and w are of dimension maximal in Γ. Since they are also in the same type, this implies that $v = w$. We thus have exhibited the term $v[\gamma]$ such that $x, y \in \mathrm{Var}(v[\gamma])$, so by induction, this proves $x = y$.

– A term $\Delta \vdash \mathsf{coh}_{\Gamma,A}[\gamma] : A[\gamma]$ is of degree strictly positive, hence do not contain variables of degree 0, so the result is vacuously true.

□

Lemma 100. *Given a substitution $\Delta \vdash \gamma : \Gamma$ between two ps-contexts Γ and Δ, and two variables $v, w \in \mathrm{Var}(\Gamma)$. Suppose we have $x \in \mathrm{Var}(v[\gamma])$ and $y \in \mathrm{Var}(w[\gamma])$ of degree 0 and of the same type, then v and w are of the same type.*

Proof. We prove this lemma by induction on the dimension of the variables v, w Note that the condition that x and y are of the same type and both of degree 0 imply that $\dim v[\gamma] = \dim w[\gamma]$, and hence $\dim v = \dim w$.

– If v, w are of dimension 0, then they are necessarily of the same type $\star$.

– If v, w are of dimension $n+1$, then by applying Lemma 98 we get a path $x^- \rightsquigarrow x^+$ in $v[\gamma]$ which goes through x with $x^- \in \mathrm{Var}(\partial^-(v)[\gamma])$ and $x^+ \in \mathrm{Var}(\partial^+(v)[\gamma])$, and a path $y^- \rightsquigarrow y^+$ in $w[\gamma]$ which goes through y with $y^- \in \mathrm{Var}(\partial^-(w)[\gamma])$ and $y^+ \in \mathrm{Var}(\partial^+(w)[\gamma])$ We then have the following inequalities, with x^-, x^+, y^- and y^+ all sharing the same type that we denote A.

$$\begin{array}{ccccccc} x^- & \trianglelefteq & \partial^-(x) & \triangleleft & \partial^+(x) & \trianglelefteq & x^+ \\ & & \| & & \| & & \\ y^- & \trianglelefteq & \partial^-(y) & \triangleleft & \partial^+(y) & \trianglelefteq & y^+ \end{array}$$

Note that for all variable $\partial^+(v) \lhd v^+$ of the same type as $\partial^+(v)$, we necessarily have by Lemma 94 a path $\partial^+(v) \rightsquigarrow v^+$ in Γ, hence Lemma 96 gives a variable $z \in \mathrm{Var}(v^+[\gamma])$ of type A and such that $x^+ \trianglelefteq z$. Moreover, Lemma 99 by mutual induction shows that, it is the only variable type A in v^+. Consider now the variable $\partial^-(w)$: since $\partial^-(w)[\gamma]$ contains the variable y^- of type A and of degree 0 and $\partial^+(v)[\gamma]$ contains the variable x^+ of type A and of degree 0, we have by induction that $\partial^-(w)$ and $\partial^+(v)$ have the same type. Since $y^- \in \mathrm{Var}(\partial^-(w)[\gamma])$ and $y^- \lhd x^+$, we necessarily have $\partial^-(w) \lhd \partial^+(v)$, hence $\partial^-(w) \trianglelefteq \partial^-(v)$. A symmetric argument shows that $\partial^-(v) \trianglelefteq \partial^-(w)$, hence $\partial^-(v) = \partial^-(w)$. Moreover, the same argument on the targets shows also that $\partial^+(v) = \partial^+(w)$, and hence v and w necessarily have the same type.

□

Paths in terms. The terms that are derivable in a ps-context satisfy a very strong property regarding the variables they must contain. This condition can be expressed using the notion of path of variables.

Lemma 101. *Consider a term $\Delta \vdash t : A$ derivable in a ps-context Δ, together with two variables $x, y \in \mathrm{Var}(A)$ that are of the same type in Δ, then there exists a path $f_1, \dots, f_n$ between x and y in $\mathrm{Var}(t) \cup \mathrm{Var}(A)$.*

Proof. We prove this result by induction on the terms. In order to ensure that the induction is well-formed, we keep track of both the depth and the dimension of the terms we manipulate.

- For a term of depth 0, it is necessarily a variable term $\Delta \vdash f : A$, and we proceed by induction on its dimension, note that the term cannot be of dimension 0, or its type would not contain variables. If it is of dimension 1, then necessarily $A = z \underset{\star}{\to} w$ with $\{z, w\} = \{x, y\}$, and $\{f\}$ defines a path form z to w. For a term of dimension at least 2, since Δ is derivable in GSeTT, A necessarily contains only variables, so it is of the form $z \underset{B}{\longrightarrow} w$. If $\{z, w\} = \{x, y\}$, then f is a path from z to w. Otherwise, the dimension shows that we necessarily have $\{x, y\} \cap \{z, x\} = \emptyset$, and hence $x, y \in \mathrm{Var}(B)$. Then by induction $\mathrm{Var}(B) \cup z$ contains a path from x to y or from y to x, hence so does $\mathrm{Var}(A)$.

- For a term of the form $\Delta \vdash \mathsf{op}_{\Gamma, t \underset{A}{\longrightarrow} u}[\gamma] : t[\gamma] \underset{A[\gamma]}{\longrightarrow} u[\gamma]$ or $\Delta \vdash \mathsf{coh}_{\Gamma, t \underset{A}{\longrightarrow} u}[\gamma] : t[\gamma] \underset{A[\gamma]}{\longrightarrow} u[\gamma]$, consider two variables $x, y \in \mathrm{Var}(u[\gamma]) \cup \mathrm{Var}(t[\gamma]) \cup \mathrm{Var}(A[\gamma])$. If $x \in \mathrm{Var}(A[\gamma])$, then $\dim x < \dim t$, and hence $\dim y < \dim t$. Hence if y appears in $t[\gamma]$ or in $u[\gamma]$, it is of degree strictly positive, and also appears in $A[\gamma]$. In this case, we have $x, y \in \mathrm{Var}(A[\gamma])$, hence by induction $t[\gamma]$ contains a path between x and y. So we suppose that $x \notin \mathrm{Var}(A[\gamma])$, which implies that $\dim x = \dim t$. In this case since we also have $\dim y = \dim t$, we also have $y \notin \mathrm{Var}(A[\gamma])$, hence $x, y \in \mathrm{Var}(t[\gamma]) \cup \mathrm{Var}(u[\gamma])$. Moreover, by Lemma 99, both of these variables have to be in different sets. We assume that $x \in \mathrm{Var}(t[\gamma])$ and $y \in \mathrm{Var}(u[\gamma])$, then we have $v, w \in \mathrm{Var}(\Gamma)$ such that $x \in \mathrm{Var}(v[\gamma])$ and $y \in \mathrm{Var}(w[\gamma])$. Moreover, x and y are of degree 0 in $v[\gamma]$ and $w[\gamma]$, which implies by Lemma 100 that v and w are of the same type. Lemma 94 then applies to give a path $(g_1, \dots, g_n)$ from v to w in Γ. Applying Lemma 96 successively to $g_1[\gamma], \dots, g_n[\gamma]$ gives a family of paths $x \rightsquigarrow x_1, x_1 \rightsquigarrow x_2, \dots, x_{n-1} \rightsquigarrow x_n$ with $x_n \in \mathrm{Var}(w[\gamma])$, and by concatenation, we get a path $x \rightsquigarrow x_n$ in $\mathrm{Var}(\gamma)$. By Lemma 99, we necessarily have $x_n = y$, so we have indeed defined a path $x \rightsquigarrow y$.

□

The equivalence. Our careful analysis of the interaction of the variables of degree 0 in a term with the existence of a total order in the ps-contexts lets us prove the equivalence between the two versions of the side condition that we have given for the rule (COH).

Proposition 102. *Given a type $\Gamma \vdash A$ in a ps-context satisfying $(\mathrm{C}'_{\mathsf{coh}})$, then it also satisfies $(\mathrm{C}_{\mathsf{coh}})$.*

Proof. Consider a variable $x \in \mathrm{Var}(t) \cup \mathrm{Var}(u)$, which is not in $\mathrm{Var}(A)$ (if no such variable exists, the result is trivial). This implies in particular that x is of dimension maximal in Γ, and of degree 0 in t or u. Suppose that $x \in \mathrm{Var}(t)$, then Lemma 98 shows that there exists a path $x^- \rightsquigarrow x^+$ in t which goes through x, with $x^- \in \partial^-(t)$ and $x^+ \in \partial^+(t)$. Since t and u have the same type A, we also have $x^- \in \partial^-(u)$ an $x^+ \in \partial^+(u)$, and hence by Lemma 101, u must contain a path $x^- \rightsquigarrow x^+$, but the condition on dimension imply that this path is of maximal dimension, hence it is unique. But since we already have a path $x^- \rightsquigarrow x^+$ in t which goes through x, the path in u has to be the same, and hence it also goes through u. Hence $x \in \mathrm{Var}(u)$. If we suppose $x \in \mathrm{Var}(u)$, a symmetric argument shows that $x \in \mathrm{Var}(t)$. □

Corollary 103. *The rules* (COH) *and* (COH') *are equivalent.*

Proof. Although this might seem immediate from our previous proposition, there is a subtlety to take into account. We have proved that, in the theory using the rule (COH), every type $\Gamma \vdash A$ satisfying $(\mathrm{C}'_{\mathsf{coh}})$ also satisfies $(\mathrm{C}_{\mathsf{coh}})$, but we have not proved that it is the case in the theory using the rule (COH'). In order to solve this, we replace successively the rule (COH) by the rule (COH') for each depth of term, and prove the result by induction.

- For a term of depth 0, the rules (COH) and (COH') are equivalent, hence we can freely replace (COH) by (COH') to derive terms of depth 1.
- Suppose that the rules (COH) and (COH') are equivalent for all terms of depth at most d, and consider a type $\Gamma \vdash A$ of depth d derivable in the theory using the rule (COH') and satisfying $(\mathrm{C}'_{\mathsf{coh}})$, then by induction hypothesis, the type $\Gamma \vdash A$ is also derivable in the theory using the rule (COH), and since it satisfies $(\mathrm{C}'_{\mathsf{coh}})$ in this theory, by Proposition 102, it also satisfies $(\mathrm{C}_{\mathsf{coh}})$, hence the rules (COH) and (COH') are also equivalent for all terms of depth $d+1$.

□

Since these rules are equivalent, we use either one of them, depending on our purpose. For the implementation, we prefer the rule (COH'), which gives a computationally lighter side condition to check, whereas for reasoning we tend to prefer (COH) since it provides better guarantees on the terms.

3.5.2 The category $\mathcal{S}_{\mathsf{PS},\infty}$ and the cat-coherator

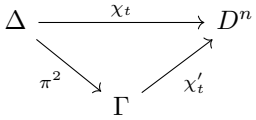
We now prove the equivalence between the subcategory $\mathcal{S}_{\mathsf{PS},\infty}$ of the syntactic category and the opposite of the cat-coherator Θ_∞. We proceed in an iterative way, showing that there is an equivalence in each step, and for this, we start by characterizing the coadmissible pairs of substitutions in the category $\mathcal{S}_{\mathsf{PS},n}$. Using Corollary 103, we chose to reason with the rule (COH), as it fits better with the definition of coadmissible pair.

Coadmissible pairs substitutions. We work in the category $\mathcal{S}_{\mathsf{PS},n}$, for $n \in \mathbb{N} \cup \{\infty\}$, which is a coglobular theory, and we consider a morphism $\xi : \Delta \to D^n$ in this category. Such a morphism is a substitution to a disk context, so by Corollary 42, it can be written as $\xi = \chi_t$ for a term t. Using the notations that we introduced in Section 2.2 for defining the disk and sphere contexts, the term t in Δ can be recovered as $t = x_{2n}[\xi]$ and the type of t in Δ can be computed to be $A_n[\xi]$. Then note that A_n contains all the variables of D^n, except for the variable x_{2n}, hence $\mathrm{Var}(x_{2n}) \cup \{A_n\} = \mathrm{Var}(D^n)$. This equality shows

$$\begin{aligned}\mathrm{Var}(\xi) &= \mathrm{Var}(x_{2n}[\xi]) \cup \mathrm{Var}(A_n[\xi]) \\ &= \mathrm{Var}(t) \cup \mathrm{Var}(A)\end{aligned}$$

Lemma 104. *A term $\Delta \vdash t : A$ defines a coalgebraic morphism χ_t in $\mathcal{S}_{\mathsf{PS},n}$ if and only if $\mathrm{Var}(t) \cup \mathrm{Var}(A) = \mathrm{Var}(\Delta)$*

Proof. First suppose that $\mathrm{Var}(t) \cup \mathrm{Var}(A) = \mathrm{Var}(\Delta)$ and consider a factorization of the form $\chi_t = \xi' \circ \gamma$, for a globular substitution $\Delta \vdash \gamma\Gamma$. Since $\mathrm{Var}(\xi) = \mathrm{Var}(\Delta)$, we also necessarily have $\mathrm{Var}(\gamma) = \mathrm{Var}(\Delta)$. Since γ is a globular substitution, Proposition 40 shows that γ is an identity, hence χ_t is coalgebraic. Conversely, suppose that there exists a variable f of dimension locally maximal in Δ such that f does not appear in $\mathrm{Var}(t) \cup \mathrm{Var}(A)$. Then there is necessarily at least one among the source and the target of f in Δ is not in $\mathrm{Var}(t) \cup \mathrm{Var}(A)$. Indeed, by contraposition: if both the source and the target of f appear in $\mathrm{Var}(t) \cup \mathrm{Var}(A)$, then by Lemma 101, $\mathrm{Var}(t) \cup \mathrm{Var}(A)$ contains a path between these, and since f is of dimension locally maximal, this path is necessarily (f). Suppose that the target $\partial^+(f)$ of f does not appear in $\mathrm{Var}(t) \cup \mathrm{Var}(A)$, then consider the context Γ obtained by removing the variables f and $\partial^+(f)$ from Δ: This context is again a ps-context (this can be checked by induction on the structure of ps-contexts and is very similar to checking the correctness of the source and target of a ps-context). There is a substitution $\Delta \vdash \pi^2 : \Gamma$ obtained by associating to each variable in Γ the same variable seen in Δ, and since Γ and Δ are not isomorphic, this map is globular but is not an identity. Moreover, the judgment $\Gamma \vdash t : A$ is necessarily derivable (in fact it is exactly the same derivation than the derivation of $\Delta \vdash t : A$ since neither t nor A uses the variables f and $\partial^+(f)$) and we denote χ'_t the substitution $\Gamma \vdash \chi'_t : D^n$ that classifies this term seen in the context Γ. We then have the equality $t = t[\pi^2]$ since π^2 acts trivially on the variables, which provides the non trivial factorization



This shows that in this condition the substitution χ_t is not coalgebraic, and the same proof also holds if the target of f appears in $\mathrm{Var}(t) \cup \mathrm{Var}(A)$, it which case its source cannot appear. In general, given any variable x of Δ which does not appear in $\mathrm{Var}(t) \cup \mathrm{Var}(A)$, there is a variable f of dimension maximal in Δ whose type contains x, thus f cannot appear in $\mathrm{Var}(t) \cup \mathrm{Var}(A)$, and the previous example shows that χ_t is not coalgebraic. □

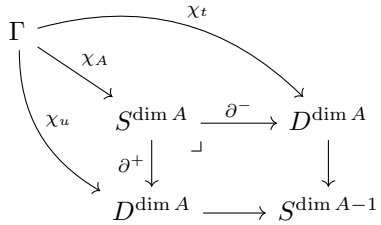
Lemma 105. *The pairs of coadmissible morphisms in Γ are classified by the types $\Gamma \vdash A$ satisfying either $(\mathrm{C}_{\mathsf{op}})$ or $(\mathrm{C}_{\mathsf{coh}})$. For such a type $\Gamma \vdash A$, the terms $\Gamma \vdash t : A$ classify exactly the lifts of the corresponding coadmissible pair.*

Proof. Note that the types $\Gamma \vdash A$ of dimension nonzero of the form $t \underset{B}{\longrightarrow} u$ classify the pairs of terms (t, u) of same type B, which are exactly the pairs of parallel maps (χ_t, χ_u). Moreover, these pair is coadmissible whenever:

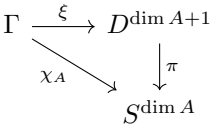
– Either both χ_t and χ_u are coalgebraic, which by Lemma 104 translates to the conditions $\mathrm{Var}(t) \cup \mathrm{Var}(B) = \mathrm{Var}(\Gamma)$ and $\mathrm{Var}(u) \cup \mathrm{Var}(B) = \mathrm{Var}(\Gamma)$: This is exactly the condition ($\mathrm{C}_{\mathsf{coh}}$).

– Or χ_t factors through the source inclusion of Γ as a coalgebraic morphism and χ_u factors through the target as a coalgebraic morphism. Again by Lemma 104, these conditions translate into $\partial^-(\Gamma) \vdash t : B$ with $\mathrm{Var}(t) \cup \mathrm{Var}(B) = \mathrm{Var}(\partial^-(\Gamma))$ and $\partial^+(\Gamma) \vdash u : B$ with $\mathrm{Var}(u) \cup \mathrm{Var}(B) = \mathrm{Var}(\partial^+(\Gamma))$: This is the condition (C_{op}).

A lift for such a coadmissible is a map $\xi : \Gamma \to D^{\dim A+1}$, such that we have both $\partial^-(\xi) = \chi_t$ and $\partial^+(\xi) = \chi_u$. In the category $\mathcal{S}_{\mathsf{CaTT},n}$ we can encode this data into a substitution towards the sphere context



A lift thus becomes equivalent to a morphism $\xi : \Gamma \to D^{\dim A+1}$ in $\mathcal{S}_{\mathsf{CaTT},n}$ which makes the following triangle commute

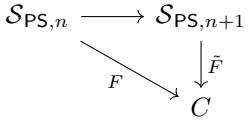


By Corollary 42, these are classified by the terms $\Gamma \vdash t : A$ in the theory CaTT_n □

Tower of definition. We define the set E_n to be the set of all types $\Gamma \vdash t \underset{A}{\longrightarrow} u$ of coherence depth n in a ps-context Γ, satisfying (C_{op}) or ($\mathrm{C}_{\mathsf{coh}}$). With our previous discussion, the family E_n can be defined inductively as the set of all pair of coadmissible maps in $\mathcal{S}_{\mathsf{PS},n}$ that do not belong to any $E_{n'}$ for $n < n'$.

Lemma 106. *The inclusion $\mathcal{S}_{\mathsf{PS},n} \to \mathcal{S}_{\mathsf{PS},n+1}$ exhibits $\mathcal{S}_{\mathsf{PS},n+1}$ as the universal coglobular extension of $\mathcal{S}_{\mathsf{PS},n}$ which has a lift for all pair of morphisms in E_n.*

Proof. As we have already proved, this map commutes with the coglobular structure, and both these categories have all the globular products, hence it defines a coglobular extension. Moreover consider a coadmissible pair $(f, g) : \Gamma \to D^n$ in E_n, then by Lemma 105, corresponds to a type $\Gamma \vdash A$ in the ps-context Γ, which satisfies (C_{op}) or ($\mathrm{C}_{\mathsf{coh}}$) and which is of depth n. Hence we can derive a term t by $\Gamma \vdash \mathsf{op}_{\Gamma,A}[\mathrm{id}_\Gamma] : A$ if A satisfies (C_{op}), or $\Gamma \vdash \mathsf{coh}_{\Gamma,A}[\mathrm{id}_\Gamma] : A$ if A satisfies ($\mathrm{C}_{\mathsf{coh}}$), which is of depth $n + 1$. This term defines a map χ_t in the category $\mathcal{S}_{\mathsf{PS},n+1}$, which by Lemma 105 is a lift for the coadmissible pair (f, g). Hence $\mathcal{S}_{\mathsf{PS},n+1}$ is a coglobular extension which contains a lift for all pairs in E_n. We now show that this extension is universal: consider another extension $F : \mathcal{S}_{\mathsf{PS},n} \to C$ that defines a lift for all the pairs in E_n, we show that there exists a unique $\tilde{F}$ that preserves the chosen lifts which makes the following diagram commute



Indeed, the map $\tilde{F}$ is already defined on all objects of $\mathcal{S}_{\mathsf{PS},n+1}$, and all maps of coherence depth less than n, to coincide with F, so it suffices that there is a unique extension to the maps of coherence depth $n+1$. Since all the object in $\mathcal{S}_{\mathsf{PS},n+1}$ are globular products, it suffices to show it for the maps of the form $\Gamma \to D^n$. We can thus reformulate by saying that it suffices to show that there is a unique map $\tilde{F}$ on terms, with the condition that $\tilde{F}(t[\gamma]) = \tilde{F}t \circ \tilde{F}\gamma$. We proceed by induction on the depth, noticing that a term of coherence depth $n+1$ cannot be a variable, hence we have already defined a unique value for $\tilde{F}$ on terms of depth 0, by our previous condition, and thus the induction is already initialized

- For a term $\Delta \vdash \mathsf{op}_{\Gamma,A}[\gamma] : A[\gamma]$ of depth $d+1$, the value of F is uniquely determined by $\tilde{F}(\mathsf{op}_{\Gamma,A}[\gamma]) = \tilde{F}(\mathsf{op}_{\Gamma,A}[\mathrm{id}_\Gamma])\tilde{F}\gamma$, and since γ is of depth d, by induction $\tilde{F}(\gamma)$ is defined, and $\tilde{F}(\mathsf{op}_{\Gamma,A}[\mathrm{id}_\Gamma])$ is uniquely defined by the condition of preserving the lifts for the pairs in E_n
- Similarly for a term $\Delta \vdash \mathsf{coh}_{\Gamma,A}[\gamma] : A[\gamma]$ of depth $d+1$, the value of F is uniquely determined by $\tilde{F}(\mathsf{op}_{\Gamma,A}[\gamma]) = \tilde{F}(\mathsf{op}_{\Gamma,A}[\mathrm{id}_\Gamma])\tilde{F}\gamma$, and since γ is of depth d, by induction $\tilde{F}(\gamma)$ is defined, and $\tilde{F}(\mathsf{coh}_{\Gamma,A}[\mathrm{id}_\Gamma])$ is uniquely defined by the condition of preserving the lifts for the pairs in E_n

This proves that there exists a unique $\tilde{F}$ satisfying the condition, and hence $\mathcal{S}_{\mathsf{PS},n+1}$ is the universal coglobular extension obtained by adding a lift for all arrows in E_n to $\mathcal{S}_{\mathsf{PS},n}$ □

This relates very strongly the categories $\mathcal{S}_{\mathsf{PS},n}$ to the categories Θ_n and lets us prove the following theorem.

Theorem 44. *There is an equivalence of categories $\mathcal{S}_{\mathsf{PS},\infty} \simeq \Theta_\infty^{\mathrm{op}}$*

Proof. By construction $\mathcal{S}_{\mathsf{PS},\infty}$ is obtained as the colimit of the inclusions of categories

$$\mathcal{G}^{\mathrm{op}} \to \mathcal{S}_{\mathsf{PS},0} \to \mathcal{S}_{\mathsf{PS},1} \to \cdots \to \mathcal{S}_{\mathsf{PS},n} \to \cdots \to \mathcal{S}_{\mathsf{PS},\infty} = \mathrm{colim}_n\, \mathcal{S}_{\mathsf{PS},n}$$

so it suffices to prove that $\mathcal{S}_{ps,n}$ is equivalent to Θ_n^{op}, which we do by induction.

- We have already proved the $\mathcal{S}_{\mathsf{PS},0}$ is equivalent to Θ_0^{op}: this is Proposition 38.
- Suppose that $\mathcal{S}_{\mathsf{PS},k}$ is equivalent to Θ_k^{op} for all k until n. Note that Lemma 106 shows that $\mathcal{S}_{\mathsf{PS},n+1}$ is the universal coglobular extension that adds a lift for each pair in the set E_n. Moreover, the set F_n coincide with the set E_n and by definition, Θ_{n+1} is the exact dual. Hence $\mathcal{S}_{\mathsf{PS},n+1}$ and $\Theta_{n+1}^{\mathrm{op}}$ satisfy the same universal property, hence they are equivalent.

□

Chapter 4

A type theoretic framework for monads with arities

4.1 A framework for globular type theories

The type theory CaTT that we have presented is obtained from the type theory GSeTT by adding some term constructors, and in this sense, one can think about it as a *globular type theory*. We now propose a general framework for introducing and studying other globular type theories, of which CaTT is a particular case. We later on generalize this even further to encompass type theories with various shapes, and not necessarily just globular, but the process is similar, and building up this framework in the globular case helps gaining intuition about the general case. Additionally, we provide a fully formalized account for globular type theories implemented in Agda [12].

4.1.1 Presentation of the framework

Our framework for globular type theories is meant to encompass a class of type theories, in a cut-free style, whose type constructors are the same as those of GSeTT and CaTT. We thus assume a cut-free theory with type constructors $\star$ and $\to$ satisfying the same introduction as CaTT.

Indexes for term constructors. Such a theory is completely defined by its term constructors and term introduction rules. In order to keep track of these term constructors, we suppose that they are *indexed* by a set J. We denote for every $j \in J$, T_j the corresponding term constructor. To handle the arities in a uniform way, we chose to use the same trick as in the case of CaTT, and represent a family of terms in a given configuration as a substitution to a specific globular context. Thus we define a term to be either a variable, of an expression of the form $T_i[\gamma]$ where γ is a substitution.

Introduction rules. Each term constructor has to come with a specific introduction rule, as explained earlier one of the data we need to express the rules for constructors, are a specific globular contexts that encode the arities and their dependency of each term constructor. We thus assume that for each $j \in J$ we have a given globular context Γ_j. Moreover, produces a term, which needs to be of a given type, we thus assume that for each $j \in J$, we have a specified

type A_j. With this data, we can express the term construction rule for each term constructor as

$$\frac{\Gamma_i \vdash A_i \qquad \Delta \vdash \gamma : \Gamma_i}{\Delta \vdash T_i[\gamma] : A_i[\gamma]} \quad \text{for } i \in J$$

It is important in order for the theory to be well-founded to require that Γ_i is always a globular context, hence the theory GSeTT needs to be defined and studied separately, so that globular type theories can rely on it. Additionally, we require that whenever $\Gamma_i \vdash A_i$ holds, we have $\dim A_i \geq \dim \Gamma_i - 1$: It is a technical condition to ensure that the type checking is decidable in all these theories. Intuitively it means that any information about a dimension can only be encoded into higher dimensions.

GSeTT and CaTT. By definition, choosing $J_{\mathsf{GSeTT}} = \emptyset$ exhibits GSeTT as a specific case of a globular type theory. We show how CaTT is also obtained as a globular type theory. We now chose J_{CaTT} to be the set of pairs (Γ, A), where $\Gamma \vdash_{\mathsf{ps}}$ is a ps-context, and A is a type satisfying either (C_{op}) or ($\mathrm{C}_{\mathsf{coh}}$). We define the context associated to the pair (Γ, A) to be Γ, and the type associated to be A, and we recover exactly the theory CaTT in this way. In Section 5.2 we introduce another type theory MCaTT which has the same indexing set J as the theory CaTT, but different introduction rules, since we chose the specified globular contexts and types to be different from those of CaTT.

Restriction. Our framework for globular type theory is more restricted than it may appear: it does not allow for instance, rules of the form

$$\frac{\Gamma \vdash A \qquad \Gamma, x : A \vdash u : B}{\Gamma \vdash T(u) : C}$$

Such rules are common in type theory in general, but are not required for our purposes and restricting ourselves to the framework we introduce allow for studying both in-depth and with enough generality the theories that we study.

Formalization. In order to study this general framework for globular type theories along with their properties, we have formalized it in Agda [12], using de Bruijn levels for variables. This construction is found in the directory `Globular-TT/` of the project and follows the same structure as the one of the folder `GSeTT` that we have presented in Section 2.2. The folder `CaTT` is dedicated to the work in progress of formalizing the type theory CaTT as a particular case of a globular type theory (which is slightly more difficult in a constructive and proof relevant setup, that what we have presented here). We define in particular the syntax of a globular type theory in the file `Syntax.agda` as follows, and define the action of substitutions on types and terms as well as their composition

```
module Globular-TT.Syntax {l} (index : Set l) where

  data Pre-Ty : Set (lsuc l)
  data Pre-Tm : Set (lsuc l)
  data Pre-Sub : Set (lsuc l)
  data Pre-Ctx : Set (lsuc l)

  data Pre-Ty where
    ∗ : Pre-Ty
```

```
    ⇒ : Pre-Ty → Pre-Tm → Pre-Tm → Pre-Ty

  data Pre-Tm where
    Var : ℕ → Pre-Tm
    Tm-constructor : ∀ (i : index) → Pre-Sub → Pre-Tm

  data Pre-Sub where
    <> : Pre-Sub
    <_,_↦_> : Pre-Sub → ℕ → Pre-Tm → Pre-Sub

  data Pre-Ctx where
    ∅ : Pre-Ctx
    _·_#_ : Pre-Ctx → ℕ → Pre-Ty → Pre-Ctx

  _[_]Pre-Ty : Pre-Ty → Pre-Sub → Pre-Ty
  _[_]Pre-Tm : Pre-Tm → Pre-Sub → Pre-Tm
  _∘_ : Pre-Sub → Pre-Sub → Pre-Sub
```

We then define the judgments as inductive inductive types whose generators are the inference rules of the theory, they are indexed over typed contexts in the theory GSeTT, which we include as an argument of the module.

```
module Globular-TT.Rules {l} (index : Set l)
  (rule : index → GSeTT.Typed-Syntax.Ctx × (Globular-TT.Syntax.Pre-Ty index))
where
  open import Globular-TT.Syntax index

  {- Notational shortcuts : the context corresponding to an index -}
  Ci : index → Pre-Ctx
  Ci i = GPre-Ctx (fst (fst (rule i)))

  Ti : index → Pre-Ty
  Ti i = snd (rule i)

  data _⊢C : Pre-Ctx → Set (lsuc l)
  data _⊢T_ : Pre-Ctx → Pre-Ty → Set (lsuc l)
  data _⊢t_#_ : Pre-Ctx → Pre-Tm → Pre-Ty → Set (lsuc l)
  data _⊢S_>_ : Pre-Ctx → Pre-Sub → Pre-Ctx → Set (lsuc l)

  data _⊢C where
    ec : ∅ ⊢C
    cc : ∀ {Γ A} → Γ ⊢C → Γ ⊢T A → (Γ · (C-length Γ) # A) ⊢C

  data _⊢T_ where
    ob : ∀ {Γ} → Γ ⊢C → Γ ⊢T ∗
    ar : ∀ {Γ A t u} → Γ ⊢T A → Γ ⊢t t # A → Γ ⊢t u # A → Γ ⊢T ⇒ A t u

  data _⊢t_#_ where
    var : ∀ {Γ x A} → Γ ⊢C → x # A ∈ Γ → Γ ⊢t (Var x) # A
    tm : ∀ {Δ γ} → (i : index)
```

```
                → Ci i ⊢T Ti i
                → Δ ⊢S γ > Ci i
                → Δ ⊢t Tm-constructor i γ # (Ti i [ γ ]Pre-Ty)

  data _⊢S_>_ where
    es : ∀ {Δ} → Δ ⊢C → Δ ⊢S <> > ∅
    sc : ∀ {Δ Γ γ x A t} → Δ ⊢S γ > Γ
                         → (Γ · x # A) ⊢C
                         → (Δ ⊢t t # (A [ γ ]Pre-Ty))
                         → Δ ⊢S < γ , x ↦ t > > (Γ · x # A)
```

4.1.2 Properties

Most of the properties that we have stated or proved for the theory CaTT are in fact non-specific, and the vast majority of results can be verified for any globular type theory, the exact same way we have verified them for CaTT. We present here all of these results, along with, when we have one it, its formalization.

Syntactic properties. We can first check by mutual induction that all the properties of Proposition 2 hold in any globular type theory. Our formalization shows all these properties in the file `Rules.agda`. Note that in this framework, term constructors depend mutually inductively on substitution, which makes some of these properties harder to prove - If we are not careful, some of the inductions become ill-formed. For this reason, some of the properties stated here are proved in the file `CwF-Structure.agda`, although the corresponding properties for GSeTT are in the file `Rules.agda`. The statements corresponding to these properties are similar to the ones we have presented for the theory GSeTT in Section 2.2.

Structure of category with families. We have defined all the structure of a cut-full category as we have presented in Section 1.1, and proved that all the defining equations of a cut-full type theory hold for globular type theories. The corresponding proofs are in the file `CwF-Structure.agda`. The fact that term constructors depend on substitution also makes this fact a little bit harder to prove, and most of the results are mutually inductive together. This makes the proof very hard to check by hand, and illustrates the use of having a proof-assistant as Agda to manipulate type theories in our case.

Decidability of type checking. The existence of a derivation for any judgment in a globular type theory is decidable, and we have proved this by induction. For this proof, we follow the structure described in Proposition 43, and it requires a subtle argument keeping track both of the depth and the dimension of the terms to ensure that the induction is well-formed. In particular, it is this property that motivates us to assume that $\dim A_i \geq \dim \Gamma_i - 1$. We have formalized this argument in the file `Dec-Type-Checking.agda`.

```
module Globular-TT.Dec-Type-Checking {l} (index : Set l)
   (rule : index → GSeTT.Typed-Syntax.Ctx × (Globular-TT.Syntax.Pre-Ty index))
   (assumption : Globular-TT.Rules.well-founded index rule)
   (eqdec-index : eqdec index)
where

  dec-G⊢T : ∀ (Γ : GSeTT.Typed-Syntax.Ctx) n A
```

```
          → dim A ≤ n
          →  dec (GPre-Ctx (fst Γ) ⊢T A)
  dec-G⊢t : ∀ (Γ : GSeTT.Typed-Syntax.Ctx) n d A t
          → dim A ≤ n → depth t ≤ d
          →  dec (GPre-Ctx (fst Γ) ⊢t t # A)
  dec-G⊢S : ∀ (Δ Γ : GSeTT.Typed-Syntax.Ctx) n d γ
          → dimC (GPre-Ctx (fst Γ)) ≤ n
          → depthS γ ≤ d
          → dec (GPre-Ctx (fst Δ) ⊢S γ > GPre-Ctx (fst Γ))

  dec-⊢C : ∀ Γ → dec (Γ ⊢C)
  dec-⊢T : ∀ Γ A →  dec (Γ ⊢T A)
  dec-⊢t : ∀ Γ A t → dec (Γ ⊢t t # A)
  dec-⊢S:G : ∀ Δ (Γ : GSeTT.Typed-Syntax.Ctx) γ
           → dec (Δ ⊢S γ > GPre-Ctx (fst Γ))

  dec-⊢S : ∀ Δ Γ γ → dec (Δ ⊢S γ > Γ)
```

The statements separate in three groups, corresponding to the three steps of the proof that we have sketch for CaTT (Proposition 43). We first prove the decidability for derivability of judgments in a context that is in the theory GSeTT. We then prove the derivability for the regular judgments, restricting the substitution to those whose target is in the theory GSeTT. We then can prove it for every substitution. The first part of this proof is a bit subtle, as it requires keeping track both of the dimension and the depth of the objects we manipulate, and relies on the assumption (which is denoted by the argument `assumption` of the module) that for all $j \in J$, we have the inequality $\dim \Gamma_j \leq \dim A_j + 1$, we have formalized this by the type well-founded defined as follows

```
  well-founded : Set (lsuc l)
  well-founded = ∀ (i : index) → Ci i ⊢T Ti i → dimC (Ci i) ≤ dim (Ti i)
```

We also rely on the fact that our indexing set for the term constructors have decidable equality.

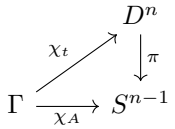
Uniqueness of derivation. Every judgment is derivable in at most one way in a globular type theory: This is a straightforward mutual induction, and can be seen by the fact that the syntactic expression of a context, type, term or substitution completely encodes its possible derivation. We have proved this property in the file `Uniqueness-Derivations.agda` of our formalization. The formulation is similar to the one presented in Section 2.2

```
module Globular-TT.Uniqueness-Derivations {l} (index : Set l)
  (rule : index → GSeTT.Typed-Syntax.Ctx × (Globular-TT.Syntax.Pre-Ty index))
where

  is-prop-⊢C : ∀ Γ → is-prop (Γ ⊢C)
  is-prop-⊢T : ∀ Γ A → is-prop (Γ ⊢T A)
  is-prop-⊢t : ∀ Γ A t → is-prop (Γ ⊢t t # A)
  is-prop-⊢S : ∀ Δ Γ γ → is-prop (Δ ⊢S γ > Γ)
```

Familial representability of Ty**.** The sphere contexts S^n and the disk contexts D^n that we have defined in GSeTT are also valid contexts in any globular type theory, and they satisfy

the same universal properties: The substitution towards the context S^n classify the types of dimension n. As a consequence, the terms of type A in a context Γ are classified by the triangles



This can be proved in the exact same way we have proved it in the case of GSeTT (Theorem 22 and Corollary 23) or of CaTT (Theorem 41 and Corollary 42), we have formalized a reformulation without diagrams of this properties for all globular type theories in the file `Disks.agda`. Again the formalization is very close to the case of GSeTT presented in Section 2.2.

```
𝕊 : ℕ → Pre-Ctx
𝔻 : ℕ → Pre-Ctx
n⇒ : ℕ → Pre-Ty

𝕊⊢ : ∀ n → 𝕊 n ⊢C
𝔻⊢ : ∀ n → 𝔻 n ⊢C
𝕊⊢⇒ : ∀ n → 𝕊 n ⊢T n⇒ n

Ty-n : ∀ {Γ} → Σ (ℕ × Pre-Sub) (λ {(n , γ) →  Γ ⊢S γ > 𝕊 n})
             → Σ Pre-Ty (λ A → (Γ ⊢T A))
Ty-classifier : ∀ Γ → is-equiv (Ty-n {Γ})
```

Characterization of the syntactic category. Given a globular type theory GTT, we define its *coherator* to be the category Θ_{GTT} obtained as the opposite of the full subcategory of $\mathcal{S}_{\mathsf{GTT}}$ whose objects are the contexts Γ_j, for $j \in J$. Since all the contexts Γ_i are globular contexts, they can be written as canonical limits of disks in the category $\mathcal{S}_{\mathsf{GSeTT}}$, and since the embedding $\mathcal{S}_{\mathsf{GSeTT}} \to \mathcal{S}_{\mathsf{GTT}}$ induces a coglobular structure on $\mathcal{S}_{\mathsf{GTT}}$ for which it is a morphism of coglobular structured categories with families, it preserves these limits. Hence every context Γ_i is canonically a limit of disks contexts in $\mathcal{S}_{\mathsf{GTT}}$ and in Θ_{GTT} since it is a full subcategory. We call these limits the *arity limits* of the theory.

Theorem 107. *The inclusion $\Theta_{GTT} \to \mathcal{S}_{GTT}$ defines a free completion of Θ_{GTT} preserving the arity limits, or equivalently, Θ_{GTT} is a dense and full subcategory of $\mathcal{S}_{GTT}$*

exhibiting all objects of $\mathcal{S}_{\mathsf{GTT}}$ as *canonical limits* of objects of Θ_{GTT}. Although we have not formalized this result, it can be proved by following the exact same construction we have presented for CaTT in Section 2.5. In fact our proof of this result for CaTT never uses the specific term constructors or the notion of ps-contexts, but only the fact that they are of a specific form.

Models of a globular type theory. Following once again our proof for CaTT presented in Section 2.6, we have a complete characterization of the models of any globular theory

Theorem 108. *The models of the theory GTT are equivalent to the presheaves $\Theta_{GTT} \to$ **Set** preserving the arity limits and the canonical limits.*

This construction gives a new understanding of our study of CaTT: Most of the properties are in reality properties of a globular type theory, and only our computation of the arity limits as globular products and our characterization of the coherator as the cat-coherator Θ_∞ are specific to CaTT.

4.1.3 Non-free globular type theories

In our examples, all the theory we consider are *free*, that is, they do not have congruence for terms. One may also add congruence for terms, but it may break the properties listed above. In particular decidability of type checking amounts to the decidability of the word problem. Since this problem is known to be undecidable in general, it has to be treated individually for each system of congruence. Since this is somewhat tangential to the appplications we are interested in, where all the type theories are free, we do not study these here.

4.2 Free category with families over a direct category

We now generalize the framework of globular type theories to other kinds of dependent type theories. This construction is based on ideas due to Leena Subramaniam and LeFanu Lumsdaine [46], and adapt it slightly in order to give a more syntactic presentation. The intuition is to replace the category $\mathcal{G}$ by another category thus changing the globes by other shapes. We restrict our study to the case where we replace $\mathcal{G}$ by another direct category I, the intuition being that I encodes the dependencies of type constructors, and requiring it to be direct eliminates any circular dependency.

Definition 109. A *direct category* is a category $\mathcal{C}$ in which every object c is equipped with a *dimension* $\dim c \in \mathbb{N}$, and such that for all morphism $f : c \to d$ in $\mathcal{C}$, f is either an identity of $\dim c < \dim d$

We also assume that I is finitely branching, i.e., for any object i of I, the presheaf $I(\_, i)$ is finite, to encode that any type constructor may only depend on a finite amount of data.

4.2.1 Type theory associated to a direct category

We construct a type theory T_I whose only terms are variable to the direct category I. To present such a type theory, it suffices to define its type constructors along with the introduction rules. To each object i of I, we associate a type constructor C_i, whose arity are the element of the presheaf $I(\_, i)$: If the presheaf $I(\_, i)$ has n elements, and $t_1, \dots, t_n$ are n term expressions in the theory, then $C_i(t_1, \dots, t_n)$ defines a type expression. In particular, we chose an enumeration of all the elements of the presheaves $I(\_, i)$, for all i, such that the level of the target of the element is always increasing. These type constructors are subject to the following introduction rules, that are well defined by induction on the level of the object i

- For an object i of level 0, the constructor C_i has the introduction rule

$$\frac{\Gamma \vdash}{\Gamma \vdash C_i}(C_i\text{-INTRO})$$

- For an object i of level n, writing the given enumeration of the elements of the presheaf $I(\_, i)$ as $f_1 : j_1 \to i, \dots, f_n : j_n \to i$ (with $j_1, \dots, j_n$ ordered by increasing levels) the constructor C_i has for introduction rule

$$\frac{\Gamma \vdash t_1 : A_1 \qquad \dots \qquad \Gamma \vdash t_n : A_n}{\Gamma \vdash C_i(t_1, \dots, t_n)}(C_i\text{-INTRO})$$

  where the type A_k is constructed with the type constructor C_k, and we require also the following condition : for all functions $g : j_k \to j_l$ such that $gf_k = f_l$ such that g is the

m-th element in the chosen enumeration of the presheaf $I(\_, j_l)$, then the m-th argument of the constructor C_l in the type A_l is the term t_k. Note that since necessarily the level of j_k is strictly less that the level of j_l (as witnessed by the existence of the map g), this implies that the premise $\Gamma \vdash t_k : A_k$ comes before the premise $\Gamma \vdash t_l : A_l$ in the rule, thus proving that the type A_l is well-formed.

The type theory GSeTT. The theory GSeTT is a particular case of this definition, taking as index category the category $\mathcal{G}$. Indeed, following this definition, there is a type $\star$ along with a type $\to_n$ for each n in $\mathbb{N}_{\geq 1}$, with the following introduction rules

$$\frac{\Gamma \vdash}{\Gamma \vdash \star}$$

$$\frac{\Gamma \vdash t : \star \qquad \Gamma \vdash u : \star}{\Gamma \vdash t \to_1 u}$$

$$\frac{\Gamma \vdash t_0 : \star \qquad \Gamma \vdash u_0 : \star \qquad \Gamma \vdash t_1 : t_0 \to_1 u_0 \qquad \Gamma \vdash u_1 : t_0 \to_1 u_0}{\Gamma \vdash t_1 \to_2 u_1}$$

$$\vdots$$

The rule for the constructor $\to_2$ can then reformulated: Since the derivability of $\Gamma \vdash t_1 : t_0 \to_1 u_0$ implies the derivability of $\Gamma \vdash t_0 : \star$ and of $\Gamma \vdash u_0 : \star$, these premises are superfluous, and the rule rewrites simply as

$$\frac{\Gamma \vdash t_1 : t_0 \to_1 u_0 \qquad \Gamma \vdash u_1 : t_0 \to_1 u_0}{\Gamma \vdash t_1 \to_2 u_1}$$

Performing the same kind of simplification for all the other rules yields the theory that we have proved to be equivalent to GSeTT in Section 2.2

4.2.2 The syntactic category $\mathcal{S}_{T_I}$

We study the properties that the syntactic category $\mathcal{S}_{T_I}$ enjoys. Since T_I is a generalization of GSeTT, the properties are very similar to the ones of the theory GSeTT, and we follow a presentation that is parallel to the one of Section 2.2.

Limits in $\mathcal{S}_{T_I}$. Since all the terms in T_I are variables, all the maps in $\mathcal{S}_{T_I}$ are display maps, and hence $\mathcal{S}_{T_I}$ has all pullbacks. Given that, as a category with families it also has a terminal object, it follows that $\mathcal{S}_{T_I}$ has all finite limits, this is similar to Lemma 26.

Familial representability of Ty**.** We construct two families of contexts that we denote $\mathrm{Y}(i)$ and $\partial\,\mathrm{Y}(i)$, and that we call respectively the *representable context* and the *boundary context* associated to an object i of I. These contexts play the role respectively of the disks and sphere context of the theory GSeTT. We define them by induction on the dimension of i, proving along the way that they satisfy the following property

> In any context Γ, the types definable in Γ built from the constructor C_i are naturally isomorphic to the substitutions in $\mathcal{S}_{T_I}(\Gamma, \partial\,\mathrm{Y}(i))$, the terms in Γ whose type is built from the constructor C_i are naturally isomorphic to the substitutions in $\mathcal{S}_{T_I}(\Gamma, \mathrm{Y}(i))$, and their type corresponds to the substitution obtained by composing with π.

The contexts $\partial\,\mathrm{Y}(i)$ and $\mathrm{Y}(i)$ are built by induction on the level of i.

- For i of dimension 0, we chose $\partial\,\mathrm{Y}(i) = \varnothing$ to be the empty context, and $\mathrm{Y}(i)$ to be the context $(x : C_i)$.
- For i of dimension $n > 0$, we consider the introduction rule (C_i-INTRO). This rule provides with a family of terms $\Gamma \vdash t_k : A_k$, which by the above correspondence defines a family of substitutions $\mathcal{S}_{T_I}(\Gamma, \mathrm{Y}(j_k))$. This family of substitution defines a cone over a certain diagram. We chose the context $\partial\,\mathrm{Y}(i)$ to be the limit of this diagram. The property for the type is then tautological. Moreover, we chose $\mathrm{Y}(i)$ to be the context $(\partial\,\mathrm{Y}(i), A)$, where A is the type in $\partial\,\mathrm{Y}(i)$ defined by the identity substitution $\mathrm{id}_{\partial\,\mathrm{Y}(i)} \in \mathcal{S}_{T_I}(\partial\,\mathrm{Y}(i), \partial\,\mathrm{Y}(i))$. The condition for the terms is again tautologically respected.

Note that the context $\partial\,\mathrm{Y}(i)$ being defined as a limit, and we can express its diagram as

$$\partial\,\mathrm{Y}(i) = \lim(I \downarrow i \to I \xrightarrow{\mathrm{Y}} \mathcal{S}_{T_I}) = \lim_{j \to i} \mathrm{Y}(j)$$

it naturally comes equipped with a morphism $a_f : \partial\,\mathrm{Y}(i) \to \mathrm{Y}(j_k)$ for each morphism $f : j_k \to i$ in the category I.

Proposition 110. *The functor* $\mathrm{Ty} : \mathcal{S}_{T_i} \to \mathbf{Set}$ *is familially represented by the family of objects* $\partial\,\mathrm{Y}(i)$.

Proof. This is the property proved by induction, in the definition of $\partial\,\mathrm{Y}(i)$. Note however that the same context may be the classifier of different type constructors. This is the case for instance if there are two object i_0 and i'_0 of dimension 0 in I, in this case, the types in Γ constructed with C_{i_0} and the types in Γ constructed with $C_{i'_0}$ are both classified by a substitution $\mathcal{S}_{T_I}(\Gamma, \varnothing)$ (i.e., there is only one of them). But the unique substitution $\Gamma \vdash \langle\rangle : \varnothing$ constructs the type C_{i_0} when viewed as a classifying the constructor C_{i_0}, and the type $C_{i'_0}$ when viewed as classifying the constructor $C_{i'_0}$ □

To each type $\Gamma \vdash A$ constructed by C_i, we associate the substitution $\chi_A : \Gamma \to \mathrm{Y}(i)$, the terms $\Gamma \vdash t : A$ are then classified by the commutative triangles

$$\begin{array}{ccc} \Gamma & \xrightarrow{\chi_t} & \mathrm{Y}(i) \\ & \searrow^{\chi_A} & \downarrow \pi \\ & & \partial\,\mathrm{Y}(i) \end{array}$$

From here onward, the study of this category generalizes effortlessly from the study of $\mathcal{S}_{\mathsf{GSeTT}}$. This is because we have studied most properties of $\mathcal{S}_{\mathsf{GSeTT}}$ via categorical arguments, that do not rely on the exact syntax of the theory, and that are again applicable to our case.

Yoneda embedding. The assignment $\mathrm{Y} : i \to \mathrm{Y}(i)$ defines a contravariant functor $I \to \mathcal{S}_{T_I}$. Indeed, given a morphism $f : i \to j$ in the category I, one has the substitution $a_f : \partial\,\mathrm{Y}(j) \to \mathrm{Y}(i)$ obtained canonically and we define $\mathrm{Y}(f)$ to be the composite of a_f with the display map π

$$\mathrm{Y}(f) : \mathrm{Y}(j) \xrightarrow{\pi} \partial\,\mathrm{Y}(j) \xrightarrow{a_f} \mathrm{Y}(i)$$

Note that this definition is not valid in the case of the identity morphism $\mathrm{id}_i : i \to i$, in which case we define $\mathrm{Y}(\mathrm{id}_i)$. to be the identity substitution of $\mathrm{Y}(i)$. We now check that this assignment is functorial, indeed, consider two maps $f : i \to j$ and $g : j \to k$ that are not identities, then we have $\mathrm{Y}(gf) = a_{gf}\pi$. Note that πa_g defines a map $\partial\,\mathrm{Y}(k) \to \partial\,\mathrm{Y}(j)$, hence it commutes with the legs of the cone defining $\partial\,\mathrm{Y}(j)$, which in particular contain the map a_f, so we have $a_{gf} = a_f \pi a_g$, and hence $\mathrm{Y}(gf) = \mathrm{Y}(f)\,\mathrm{Y}(g)$. Hence Y defines a contravariant functor $I \to \mathcal{S}_{T_I}$.

Nerve functor. We introduce the a nerve functor similar $\nu : \mathcal{S}_{T_I} \to \widehat{I}^{\mathrm{op}}$ to the one we have studied for GSeTT, by posing $\nu(\Gamma)_i = \mathcal{S}_{T_I}(\Gamma, \mathrm{Y}(i))$. In fact, since I has finite branchings, the finite presheaves on I are exactly those which have a finite number of elements, we denote $\mathrm{Fin}(I)$ the category of finite presheaves over I. The elements of $\nu(\Gamma)$ are exactly the terms in Γ, which are the variables in Γ, so there are only finitely many of them. Hence ν corestricts as a functor $\mathcal{S}_{T_I} \to \mathrm{Fin}(I)^{\mathrm{op}}$.

Lemma 111. *For all context Δ, and all $n \in \mathbb{N}$, there are two bijections*

$$\mathcal{S}_{T_I}(\Delta, \mathrm{Y}(i)) \simeq \widehat{I}(\nu(\mathrm{Y}(i)), \nu(\Delta)) \quad \textit{and} \quad \mathcal{S}_{T_I}(\Delta, \partial\,\mathrm{Y}(i)) \simeq \widehat{I}(\nu(\partial\,\mathrm{Y}(i)), \nu(\Delta))$$

Proof. First note that we have $\nu(\mathrm{Y}(i)) = \mathrm{Y}(i)$, hence for any context Δ, we have, by the Yoneda lemma

$$\widehat{I}(\nu(\mathrm{Y}(i)), \nu(\Delta)) \simeq \nu(\Delta)_i = \mathcal{S}_{T_I}(\Delta, \mathrm{Y}(i))$$

Moreover, since the context $\partial\,\mathrm{Y}(i)$ is defined as the limit $\partial\,\mathrm{Y}(i) = \lim_{j \to i} \mathrm{Y}(j_k)$. Hence by continuity of the nerve and hom functors, we have

$$\begin{aligned}
\widehat{I}(\nu(\partial\,\mathrm{Y}(i)), \nu(\Delta)) &\simeq \widehat{I}(\mathrm{colim}_{j \to i}(\nu(\mathrm{Y}(j))), \nu(\Delta)) \\
&\simeq \mathrm{colim}_{i \to j}\, \widehat{I}(\nu(\mathrm{Y}(j)), \nu(\Delta)) \\
&\simeq \mathrm{colim}_{i \to j}\, \mathcal{S}_{T_I}(\Delta, \mathrm{Y}(j)) \\
&\simeq \mathcal{S}_{T_I}(\Delta, \lim_{j \to i} \mathrm{Y}(j)) \\
&\simeq \mathcal{S}_{T_I}(\Delta, \partial\,\mathrm{Y}(i))
\end{aligned}$$

□

Lemma 112. *The inclusion functor $\mathrm{Y} : I \to \mathcal{S}_{T_I}$ is codense, or equivalently the associated nerve ν is fully faithful.*

Proof. Consider two contexts Δ and Γ, we prove by induction on the context Γ the bijection

$$\mathcal{S}_{T_I}(\Delta, \Gamma) \simeq \widehat{I}(\nu(\Gamma), \nu(\Delta))$$

If Γ is is the empty context $\varnothing$ which is terminal, since no term is derivable in $\varnothing$, $\nu(\varnothing)$ is the empty globular set, which is initial, this proves the bijection. Suppose $\Gamma = (\Gamma', x : A)$ with the bijection holding for Γ', then by construction Γ is obtained as the following pullback, whose image by ν yields the following pushout in $\widehat{I}$

$$\begin{array}{ccc} \Gamma & \longrightarrow & \mathrm{Y}(i) \\ \downarrow & \lrcorner & \downarrow \\ \Gamma' & \xrightarrow[\chi_A]{} & \partial\,\mathrm{Y}(i) \end{array} \qquad \begin{array}{ccc} \nu(\partial\,\mathrm{Y}(i)) & \longrightarrow & \nu(\Gamma') \\ \downarrow & \ulcorner & \downarrow \\ \mathrm{Y}(n) & \longrightarrow & \nu(\Gamma) \end{array}$$

These two diagrams yield, for any context Δ, the following pullbacks in **Set**

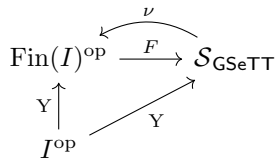
$$\begin{array}{ccc} \mathcal{S}_{T_I}(\Delta, \Gamma) & \longrightarrow & \mathcal{S}_{T_I}(\Delta, \mathrm{Y}(i)) \\ \downarrow & \lrcorner & \downarrow \\ \mathcal{S}_{T_I}(\Delta, \Gamma') & \longrightarrow & \mathcal{S}_{T_I}(\Delta, \partial\,\mathrm{Y}(i)) \end{array} \qquad \begin{array}{ccc} \widehat{I}(\nu(\Gamma), \nu(\Delta)) & \longrightarrow & \widehat{I}(\nu(\Delta)_n) \\ \downarrow & \lrcorner & \downarrow \\ \widehat{I}(\nu(\Gamma'), \nu(\Delta)) & \longrightarrow & \widehat{I}(\nu(S^{n-1}), \nu(\Delta)) \end{array}$$

By induction and Lemma 111, $\mathcal{S}_{T_I}(\Delta, \Gamma)$ and $\widehat{I}(\nu(\Gamma), \nu(\Delta))$ are thus pullbacks over the same span, hence they are isomorphic. □

$\mathcal{S}_{T_I}$ **and finite presheaves.** We now characterize the syntactic category $\mathcal{S}_{T_I}$ in the same way we have characterized the category $\mathcal{S}_{\mathsf{GSeTT}}$, that is as the opposite of the category of finite presheaves over the category that encodes the dependencies. Note that since T_I has only variables as terms, all the maps of $\mathcal{S}_{T_I}$ are display maps, and hence $\mathcal{S}_{T_I}$ has all finite limits, then the functor $\mathrm{Y} : I^{\mathrm{op}} \to \mathcal{S}_{T_I}$ extends in an essentially unique functor $F : \mathrm{Fin}(I)^{\mathrm{op}} \to \mathcal{S}_{T_I}$ preserving the finite limits.

Theorem 113. *The functors ν and F define an equivalence of categories between $\mathcal{S}_{T_I}$ and $\mathrm{Fin}(I)^{\mathrm{op}}$*

Proof. Since Lemma 112 proves that ν is fully faithful, it suffices to show that $\nu \circ F \simeq \mathrm{id}$. The functors that we have defined fit into the following diagram



Note that $\nu F \, \mathrm{Y} = \nu \, \mathrm{Y} = \mathrm{Y}$, and since both ν and F preserve finite limits, so does νF. By essentially uniqueness in the free completion by finite limits, this proves that $\nu F \simeq \mathrm{id}$. □

This allows us to understand this construction of the theory T_I as the syntactic theory describing a structure of contextual category on a category that is equivalent to $\mathrm{Fin}(I)^{\mathrm{op}}$. Since the notion of contextual category is not invariant by equivalence of categories, there may well not exist a structure of contextual category on $\mathrm{Fin}(I)^{\mathrm{op}}$ itself, but this is as close as one can get to such a structure.

Models of $\mathcal{S}_{T_I}$. We can now use our characterization of the syntactic category $\mathcal{S}_{T_I}$ to compute the models of the theory T_I, similarly to what we have presented for GSeTT.

Theorem 114. *The category of models of $\mathcal{S}_{T_I}$ is equivalent to the presheaf category $\widehat{I}$.*

Proof. By Lemma 10, the models of T_I are equivalent to the functors $\mathcal{S}_{T_I} \to \mathbf{Set}$ that preserve finite limits. Under the equivalence of Theorem 113 these are equivalent to the functors $\mathbf{FinGSet}^{\mathrm{op}} \to \mathbf{Set}$ that preserve finite limits. Since $\mathbf{FinGSet}^{\mathrm{op}}$ is the free completion of $\mathcal{G}^{\mathrm{op}}$ by finite limits, these are equivalent to the functors $\mathcal{G}^{\mathrm{op}} \to \mathbf{Set}$, which are exactly the globular sets. □

This terminates our study of the type theory T_I and shows that in all aspects it plays the exact same role than the theory GSeTT in the case of a base category $\mathcal{G}$.

4.2.3 Examples

This construction gives a syntactic theory associated to the presheaf category over any direct category I. There are a lot of examples of such presheaves that appear naturally in various areas in mathematics, so we present only few of them that are of particular interest to us, aside from our motivating example of the type theory GSeTT.

Sets. When the direct category is the terminal category $\mathbf{1}$ (the one-point category), then the associated type theory $T_{\mathbf{1}}$ is the theory obtained by adding one type constructor $\star$, together with the introduction rule

$$\frac{\Gamma \vdash}{\Gamma \vdash \star}$$

The syntactic category associated to this theory is equivalent to $\mathbf{FinSet}^{\mathrm{op}}$, and the models are equivalent to the category **Set**. This is a toy example, in which we retrieve naturally the usual one-sorted algebraic theories (Lawvere theories).

Product of sets. When the direct category $\mathbf{k}$ is the discrete category on k objects, then the associated type theory $T_{\mathbf{k}}$ is the type theory obtained by adding k type constructors $\star_1, \ldots, \star_k$, subject to all the same introduction rule

$$\frac{\Gamma \vdash}{\Gamma \vdash \star_i} \quad (1 \leq i \leq k)$$

The syntactic category is equivalent to $(\mathbf{FinSet}^k)^{\mathrm{op}}$ and the models are equivalent to Set^k. This is again a toy example as the types do not include any dependency, but it allows for retrieving the notion of multi-sorted Lawvere theory.

Graphs. If the direct category is the category $\mathcal{G}_0$ freely generated by the graph

$$0 \underset{\tau}{\overset{\sigma}{\rightrightarrows}} 1$$

Then the associated theory $T_{\mathcal{G}_0}$ is the type theory with two type constructors $\star$ and $\to$ subject to the following introduction rules

$$\frac{\Gamma \vdash}{\Gamma \vdash \star} \qquad \frac{\Gamma \vdash t : \star \qquad \Gamma \vdash u : \star}{\Gamma \vdash t \to u}$$

The syntactic category associated to this theory is equivalent to the opposite of the category of finite graphs, and its models are equivalent to the category of graphs.

Semi-simplicial sets. We denote $\mathbb{\Delta}_+$ the *semi-simplicial category*, whose objects are the natural numbers and whose morphisms are generated by

$$0 \underset{\delta_1}{\overset{\delta_0}{\rightrightarrows}} 1 \;\overset{\delta_0}{\underset{\delta_2}{\overset{\delta_1}{\rightrightarrows}}}\; 2 \;\overset{\delta_0}{\underset{\delta_3}{\overset{\delta_1}{\underset{\delta_2}{\rightrightarrows}}}}\; \cdots$$

satisfying the relations, for all $i < j$,

$$\delta_j \circ \delta_i = \delta_i \circ \delta_{j-1}$$

It is a direct finitely branching category, whose presheaves are called *semi-simplicial sets*. We can describe the theory $T_{\mathbb{\Delta}_+}$, using the same sort of simplification that we have presented for GSeTT. We denote $\triangle_i$ its type constructor for all $i \in \mathbb{N}$, in such a way that $\triangle_i$ is of arity $i + 1$,

for $i \neq 0$, and $\triangle_0$ is of arity 0. We give a description of the first few introduction rules of the theory T_{Δ_+}.

$$\frac{\Gamma \vdash}{\Gamma \vdash \triangle_0}$$

$$\frac{\Gamma \vdash t : \triangle_0 \qquad \Gamma \vdash u : \triangle_0}{\Gamma \vdash \triangle_1(t,u)}$$

$$\frac{\Gamma \vdash t : \triangle_1(x,y) \qquad \Gamma \vdash u : \triangle_1(x,z) \qquad \Gamma \vdash v : \triangle_1(y,z)}{\Gamma \vdash \triangle_2(t,u,v)}$$

$$\vdots$$

We do not provide a full description of this theory here, but this framework shows that it defines a type theory whose syntactic category is the opposite of the finite semi-simplicial sets, and whose models are the semi-simplicial sets.

4.3 I-type theories

We generalize our framework for globular type theories to I type theories, by changing the basic theory GSeTT for the theory T_I. This yields our final framework, which may encompass different theories based on different shapes. Although it is important from a theoretical standpoint, as it establishes a connection with existing notions of algebraic theories, our main focus relies on our application of this framework, through an example of a type theory for cubical weak ω-categories that we give in Section 6.2. As in the case of globular theory, we start by adding term constructors.

Indexes for term constructors. As for globular type theories, I-type theories are cut-free type theories, whose type constructors and type introduction are always the same: They are the ones given by the type theory T_I. In order to keep track of the term constructors, we start by assuming an *indexing set* J, and a term constructor T_j for every $j \in J$, and we define a term to by either a variable or of the form $T_j[\gamma]$, where $j \in J$ and γ is a substitution.

Introduction rules We encode the introduction rules as a pair of a context Γ_j in T_I and a type A_j for each $j \in J$, and we assume an introduction rule of the form

$$\frac{\Gamma_j \vdash A_j \qquad \Delta \vdash \gamma : \Gamma_j}{\Delta \vdash T_j[\gamma] : A_j[\gamma]} \quad \text{for } j \in J$$

We also need to assume (for the type checking to be decidable), that whenever $\Gamma_j \vdash A_j$ holds, we have that $\dim A_j \geq \dim \Gamma_j$. The dimension of a type is defined using the dimension in the direct category I: If a type A is constructed by C_i, then we pose $\dim A = \dim i$. We denote $T_{I,J}$ the I-type theory obtained this way.

4.3.1 Properties

All the properties that we have proved for globular type theories still hold when changing the constructors. Here we assume all the following properties, as their proof is no different than the proof for globular type theories. Our formalization still focuses on globular type theories, and does not include generic I-type theories for practical reasons: It lets us avoid manipulating

the category I. However, there is no fundamental obstacle in doing so, and the proofs can be transferred, to any I-type theory.

Syntax and cut-full type theory. All the properties listed in Proposition 2 still hold for any I-type theory, and an I-type theory encompasses the structure of a cut-full type theory satisfying all its defining equations. Hence the syntactic category of a cut-full type theory is a category with families.

Decidability of type checking. The existence of a derivation for any judgment in a I-type theory is decidable. This we can again prove by induction with a subtlety to prove the termination, by manipulating both the dimension and the depth. Note that the dimension of a type is induced by the dimension of the objects in the direct category I.

Uniqueness of derivation. Every judgment is derivable in at most one way in a I-type theory

Familial representability of Ty**.** The border contexts $\partial\,\mathrm{Y}(i)$ and the representable contexts $\mathrm{Y}(i)$ that we have defined in T_I are also valid contexts in any I-type theory, and they satisfy the same universal properties: The substitution towards the context $\partial\,\mathrm{Y}(i)$ classify the types constructed by C_i. As a consequence, the terms of type A in a context Γ are classified by the triangles

$$\begin{array}{ccc} & & \mathrm{Y}(i) \\ & \overset{\chi_t}{\nearrow} & \downarrow{\scriptstyle \pi} \\ \Gamma & \xrightarrow[\chi_A]{} & \partial\,\mathrm{Y}\,i \end{array}$$

In fact the proof of this fact given for the theory T_I does not depend in any way on the term constructors, and applies exactly to the theory $T_{I,J}$ as well.

Characterization of the syntactic category. Denote $\Theta_{I,J}$ the opposite of the full subcategory $\mathcal{S}_{T_{I,J}}$ whose objects are the contexts Γ_j for $j \in J$, we call this category the *coherator* of the theory. Then all the objects of $\Theta_{I,J}$ are canonically limits of the objects $\mathrm{Y}(i)$, we call these limits the *arity limits*.

Theorem 115. *The inclusion $\Theta_{I,J} \to \mathcal{S}_{T_{I,J}}$ defines a free completion of $\Theta_{I,J}$ preserving the arity limits, or equivalently, $\Theta_{I,J}$ is a dense and full subcategory of $\mathcal{S}_{I,J}$*

As the consequence, every object of $\mathcal{S}_{T_{I,J}}$ is a *canonical limit* of objects of $\Theta_{I,J}$.

Models of a globular type theory. This gives a full characterization of the models of the theory as follows.

Theorem 116. *The models of the theory $T_{I,J}$ are equivalent to the presheaves $\Theta_{T_{I,J}} \to$ **Set** preserving the arity limits and the canonical limits.*

These results that we assume are non-trivial and important results, however, they can be proved following the exact same method that we have introduced in the case of CaTT, and for globular type theories.

Non-free I-type theories As before, we have introduced only *free* theory. We do not consider non-free ones, that is theory where we explicitly add congruences between terms, as they might break the properties we have shown and our applications of interest are free.

4.4 I-contextual categories

We now introduce a construction due to Leena Subramaniam and LeFanu Lumsdaine [46] that sheds a new light on our construction.

4.4.1 I-contextual categories

Following the construction due to Leena Subramaniam and LeFanu Lumsdaine, we introduce the notion of an *I-contextual category*. Intuitively, it is the syntactic category of a theory that has only the types as described in I. In order to express this notion we introduce the *contextual completion* of a category.

Contextual completion. Consider a category C, we call the *contextual completion* of C a contextual category D equipped with a functor $F : C \to D$ which is universal. Explicitly, it satisfies the following universal property: For every contextual category D' together with a functor $G : C \to D'$ there is a unique factorization $G = G' \circ F$, with G' a morphism of contextual categories.

$$\begin{array}{ccc} D & \xrightarrow{G'} & D' \\ {\scriptstyle F}\uparrow & \nearrow_{G} & \\ C & & \end{array}$$

In the theory CaTT, the inclusion functor $\mathcal{S}_{\mathsf{PS},\infty} \to \mathcal{S}_{\mathsf{CaTT}}$ is an example of such a completion.

I-contextual categories. A I-contextual category is a contextual category C equipped with a morphism of contextual categories $F : \mathcal{S}_{T_I} \to C$ which factors as

$$\mathcal{S}_{T_I} \xrightarrow{F_1} C_I \xrightarrow{F_2} C \qquad (F : \mathcal{S}_{T_I} \to C)$$

with F_1 an identity on objects functor and F_2 a fully faithful functor which exhibits C as the contextual completion of C_I.

I-type theories. We claim that the notion of I-type theories that we have presented gives an important example of I-contextual categories, and that in particular

Conjecture 117. *The syntactic category of an I-type theory is an I-contextual category.*

Note though that the I-type theories and the I-contextual categories are not equivalent: two different type theories may have the same syntactic category; in this case, the type theories are often said to be *Morita equivalence*. So a type theory is more of a *presentation* of a contextual category than an actual contextual category. Moreover, since we have only presented free I-type theories, our framework may not be able to present any monad with arities. We leave for future work a general description of I-type theory that includes rewriting rules while ensuring that the expected properties of a type theory are still satisfied, and we expect such a framework to present a wider range of I-contextual categories.

4.4.2 Monads with arities

We introduce the notion of monad with arities, that has been primarily studied by Berger, Mellies and Weber [19]. This notion, along with the theorem 116 due to Leena Subramaniam and LeFanu Lumsdaine [46] gives an interpretation of our framework for I-categories in terms of categorical logic.

Definition. Consider a category C equipped with a subcategory A, denote $\nu_A : C \to [A^{\mathrm{op}}, \mathbf{Set}]$ the nerve functor associated to the inclusion of A in C. For every object X of C, we define its *canonical A-cocone*, to be the cocone over the forgetful functor $(A \downarrow X) \to C$. Then a monad T in C is said to have *arities* A if $T \circ \nu_A$ sends the canonical A-cocones to colimitng cocones. This definition is very reminiscent of our study of the syntactic category $\mathcal{S}_{\mathsf{CaTT}}$, and more generally of the category $\mathcal{S}_{T_{I,J}}$. Our aim is now to give a sketch on how they are related.

The nerve theorem. In the case where the subcategory A is a dense subcategory of C, we have the following *nerve theorem*. We refer the reader to [19] for a proof

Theorem 118. *Consider a monad T with arities A in a category C, with A dense, then the following holds*

- *The free T-algebras on the objects of A define a dense subcategory Θ_T of the category C^T of algebras of T*
- *A presheaf $P : \Theta_T^{\mathrm{op}} \to \mathbf{Set}$ is in the essential image of ν_Θ if and only if $P \circ j : A^{\mathrm{op}} \to \mathbf{Set}$ is in the essential image of ν_A, where $j : A \to \Theta_T$ is the free algebra functor.*

The first part of this theorem shows that ν_Θ is fully faithful, and hence it allows us to embed C^T into the category of presheaves $\widehat{\Theta_T}$. The second statement gives a complete characterization of C^T in this category. This proves in particular that the algebras for the monad T are equivalent to the presheaves over Θ_T whose restriction to A is in ν_A.

4.4.3 Equivalence

In an ongoing work, Leena Subramaniam and LeFanu Lumsdaine have proved that the notion of I-contextual categories coincide exactly with the existing notion of monads with arities. As of the time of writing this document, this work has not yet been published, and thus the proof is not available, but we refer the reader to the support of a presentation given by one of the authors for reference [46], in which the following theorem is claimed.

Theorem 119. *The category of I-contextual categories is equivalent to the category of monads with arities on the category $\widehat{I}$.*

Even though this is still unpublished work, we only rely on this result to build up our intuition and establish a satisfying interpretation of our construction in the world of categorical logic. None of our result do depend on this theorem, and the reader should not depend on the proof of this result to tackle our work. However, this is very important theorem in that it gives a really powerful interpretation for our type theory CaTT, and unifies it strongly with the other type theories we introduce in Section 5.2 and Section 6.2.

The monad with arities associated to CaTT. We can illustrate this equivalence in the case of the theory CaTT. Recall our discussion in Section 2.5, in which we have exhibited a fully faithful functor $F : \mathcal{S}^{\mathrm{op}}_{\mathsf{CaTT}} \to \mathbf{Mod}(\mathcal{S}_{\mathsf{CaTT}})$ that we believe to characterize the syntactic category of CaTT as the opposite of the full subcategory of the category of $\mathbf{Mod}(\mathcal{S}_{\mathsf{CaTT}})$ whose objects are the ω-category freely generated by finite polygraphs. Composing this functor with the inclusion $I : \mathcal{S}_{\mathsf{GSeTT}} \to \mathcal{S}_{\mathsf{CaTT}}$ yields the functor $FI : op\mathcal{S}_{\mathsf{GSeTT}} \to \mathbf{Mod}(\mathcal{S}_{\mathsf{CaTT}})$. Our intuition is that this functor sends an object of $\mathcal{S}^{\mathrm{op}}_{\mathsf{GSeTT}}$, that is a finite globular set onto the weak ω-category freely generated by it. By precomposing a model of CaTT with the functor $ID : G^{\mathrm{op}} \to \mathcal{S}_{\mathsf{CaTT}}$ gives a globular set, that we call its underlying globular set, so we can make this functor act on the previous functor, to construct

$$(ID)^* FI : \mathcal{S}^{\mathrm{op}}_{\mathsf{GSeTT}} \to \widehat{\mathcal{G}}$$

Intuitively, given a finite globular set X, this functor gives the underlying globular set of the free weak ω-category on X. We can now generalize this functor to all the globular sets using a Kan extension: There is a canonical functor $\iota : \mathcal{S}^{\mathrm{op}}_{\mathsf{GSeTT}} \to \widehat{\mathcal{G}}$: since $\mathcal{S}^{\mathrm{op}}_{\mathsf{GSeTT}}$ is equivalent to the opposite of the finite globular set, this is the inclusion of the finite globular sets in the globular sets. We can then consider, since $\widehat{\mathcal{G}}$ is cocomplete, the (pointwise) left Kan extension

$$\begin{array}{ccc} \widehat{\mathcal{G}} & \xrightarrow{\mathrm{Lan}_\iota((ID)^* FI)} & \widehat{G} \\ {\scriptstyle\iota}\uparrow & \nearrow_{(ID)^* FI} & \\ \mathcal{S}_{\mathsf{GSeTT}} & & \end{array}$$

We denote T_{CaTT} the functor $\mathrm{Lan}_\iota((ID)^* FI)$, intuitively, this functor produces, given a globular set X, the underlying globular set of the free weak ω-categories generated by X. The universal property of the left Kan extension shows that this functor is a monad. We conjecture the following result, which we believe is the key point of Theorem 119.

Conjecture 120. *The monad* $\mathrm{Lan}_\iota((ID)^* FI)$ *is a monad with arities the category* $\mathcal{S}^{\mathrm{op}}_{\mathsf{PS},0}$.

We believe that research in this direction is a good approach to understand the connection between CaTT and the definition of weak ω-category due to Batanin [11] and Leinster [48], and that in particular this monad entertains a deep connection with the initial globular operad with contraction defined by Leinster [48]. This program would be complementary to the result proved by Ara [4] stating that the Grothendieck-Maltsiniotis definition of weak ω-categories, is equivalent to the Batanin-Leinster definition.

The monad with arities associated to any I-type theory. This construction that we have presented is not specific to CaTT, and we could have performed it following the same process for any I-type theory. We conjecture that this construction associates a monad with arities to any I-type theory. We start by defining the functor $F : \mathcal{S}^{\mathrm{op}}_{T_{I,J}} \to \mathbf{Mod}(\mathcal{S}_{T_{I,J}})$ by imposing for every context Γ, $F\Gamma = \mathcal{S}_{T_{I,J}}(\Gamma, \_)$. By composing this functor with the inclusion $\mathcal{S}_{T_I} \to \mathcal{S}_{T_{I,J}}$ and pulling it back along the functor $I \to \mathcal{S}_{T_{I,J}}$, this defines a functor $T^{\mathrm{op}}_I \to \widehat{I}$. By computing the left Kan extension of this functor along the inclusion $T^{\mathrm{op}}_I \to \widehat{I}$ given by the inclusion of the finite presheaves in the presheaves, this defines an endofunctor T on the category $\widehat{I}$, which by the universal property of the Kan extension can be shown to be a monad. We claim that this is a monad with arities, whose arities are exactly given by the opposite of arity limits of the theory in the category $\mathcal{S}^{\mathrm{op}}_{T_I}$:

Conjecture 121. *The monad* T *defined this way has as arities the opposite of the full subcategory of* $\mathcal{S}_{T_I}$ *whose objects are the contexts* Γ_j *for all* $j \in J$

I-contextual categories. The theorem claims an equivalence between the monads with arities and the I-contextual categories, we have only illustrated one side of the equivalence in the special case of the I-contextual category is given by an I-type theory. We do not illustrate the generic case of an I-contextual category nor do we show do we explain the other side of the equivalence, as we are mostly interested in I-type theories, which provide an actual syntax to work with.

4.4.4 Interpretation of our work

Conjecture 117, together with Theorem 119 gives a new interpretation of I-type theory. As we have explained with Conjecture 117, we can think of an I-type theory as a presentation of certain I-contextual category, even though not all the I-contextual category may be presented in this way. Theorem 119 then shows that one can think of an I-type theory as a presentation of a certain monad with arities on the category $\widehat{I}$, even though not all monads with arities may be presented in such a way.

Nerve theorem. We can also interpret Theorem 116 which characterizes the models of an I-type theory as an analogous to the nerve theorem (Theorem 118) in the light of the correspondence given by Theorem 119. We illustrate this in the case of the theory CaTT, but it is similar for a general I-type theory. In the case of CaTT, Theorem 66 shows that the models of CaTT are equivalent to the presheaves over the category Θ_∞ that preserves the globular sums. We have showed that $\Theta_\infty^{\mathrm{op}}$ is equivalent to $\mathcal{S}_{\mathsf{PS},\infty}$, which through the functor $F : \mathcal{S}^{\mathrm{op}}_{\mathsf{CaTT}} \to \mathbf{Mod}(\mathcal{S}_{\mathsf{CaTT}})$ can be seen as a full subcategory of the category of weak ω-categories. Our intuition is that it is the full subcategory whose objects the weak ω-categories freely generated by the pasting schemes. This intuition motivates the following conjecture

Conjecture 122. *We conjecture that the category $\mathcal{S}_{\mathsf{PS},\infty}$ is equivalent to the opposite of the category $\Theta_{T_{\mathsf{CaTT}}}$, where T_{CaTT} is the monad with arities on $\widehat{\mathcal{G}}$ associated to CaTT as above.*

Following this intuition in the general case, we also state the following conjecture

Conjecture 123. *We conjecture that the category $Theta_T$, where T is the monad with arities associated to a theory $T_{I,J}$ as above is equivalent to the opposite of the full subcategory of $\mathcal{S}_{T_{I,J}}$ whose objects are the contexts Γ_i for all $j \in J$.*

The nerve Theorem 118, together with Conjectures 120 and 122 in the case of CaTT, or together with Conjectures 121 and 123 in the case of an I-type theory show the following

Corollary 124. *The models of the category CaTT are equivalent to the algebras of the monad T_{CaTT}, the models of the theory $T_{I,J}$ are equivalent to the algebras of the monad T, where T is the monad with arities on $\widehat{I}$ associated to $T_{I,J}$ by Theorem 119.*

We have presented various conjectures in this section as it contains the later developments of the thesis. However, we believe most of these conjectures to be fairly accessible with the tools we have developed and that developing this approach further is fruitful from a theoretical point of view.

Chapter 5

Monoidal weak ω-categories

5.1 Monoidal and multiply monoidal weak ω-categories

We now present different variations of the type theory CaTT in order to work with coherent monoidal categories, and we start with a brief overview of the notion of monoidal weak ω-category. In category theory, one usually defines a monoidal category to be a category $\mathcal{C}$ equipped with a bifunctor $\_ \otimes \_ : \mathcal{C} \times \mathcal{C} \to \mathcal{C}$ together with natural transformations called left and right unitors, and associator, satisfying various coherence axioms that can be chosen more or less strict [53]. Such an approach generalizes poorly to higher categories, since the number of coherence axioms one has to introduce becomes very quickly too big to handle, and is even infinite for any notion of ω-category.

5.1.1 Delooping of a monoidal category

We present a key observation for understanding monoidal structure on higher categories, starting from decategorified structures, and building up towards higher structures by categorifying successively the result. This is folklore result in category theory.

Monoids. The definition of category is closely related to that of a monoid

Definition 125. A monoid is a set equipped with a binary product which is associative and unital

For any category $\mathcal{C}$ together with an object c, the axiom of category ensure that the set $\mathcal{C}(c, c)$ is equipped with a composition, defining an associative and unital product, hence it is a monoid. A category with a single object thus amounts exactly to a monoid: The hom-set of the unique object. Conversely, a monoid M defines uniquely a category with a single object that we call the *delooping* of the monoid, denoted BM. Denoting $\bullet$ the unique object of BM, we have $BM(\bullet, \bullet) = M$, with the composition in $BM(\bullet, \bullet)$ is given by the product in M.

Proposition 126. *The delooping is a functor between the category of monoids and the category of categories with a single object. It defines an isomorphism of categories.*

Proof. By definition the monoid $BM(\bullet, \bullet)$ is equal to M, so it suffices to check that for any category with a single object $\mathcal{C}$, the category $B(\mathcal{C}(\bullet, \bullet))$ is isomorphic to $\mathcal{C}$. The functor sending the unique object of $B(\mathcal{C}(\bullet, \bullet))$ onto the unique object of $\mathcal{C}$ and defined as the identity on $\mathcal{C}(\bullet, \bullet)$ realizes this isomorphism. □

Monoidal categories. Categorifying our previous remark yields a more interesting result: The monoidal categories are equivalent to the 2-categories with a single object. This statement holds for various strictness for the axioms of monoidal category and 2-categories, and we illustrate it in the weakest possible case, so our monoidal categories are here implicitly lax. Explicitly, this equivalence is induced by a functor B from the category of monoidal categories to the category of bicategories called the *delooping*. For a monoidal category $\mathcal{C}$ we describe the bicategory $B\mathcal{C}$ as follows

- $B\mathcal{C}$ has a single object $\bullet$.
- The morphisms in $B\mathcal{C}$ from $\bullet$ to $\bullet$ are the objects of the category $\mathcal{C}$.
- The 2-cells in $B\mathcal{C}$ between two morphisms f, g are the morphisms in $\mathcal{C}$ between the objects f and g.
- The composition of morphisms $f \cdot g$ in $B\mathcal{C}$ is given by the monoidal product $f \otimes g$ in the objects of $\mathcal{C}$.
- The vertical composition on the 2 cells $\alpha *_1 \beta$ in $B\mathcal{C}$ is given by the composition of the morphisms $\alpha \cdot \beta$ in $\mathcal{C}$.
- The horizontal composition $\alpha *_0 \beta$ in $B\mathcal{C}$ is given by functoriality of the monoidal product in $\mathcal{C}$ with respect to both its variables as $\alpha \otimes \beta$.

As illustrated by this example, the delooping induces a change in the dimension, sending every n-cell to an $n+1$-cells, and the axioms of monoidal category translates exactly to the fact that this data assembles into a bicategory, thus the axioms for monoidal categories are subsumed by the ones of bicategories.

Later categorifications. These results can again be categorified, and gives an equivalence between the monoidal bicategories and the tricategories with a single object, and then an equivalence between monoidal categories and quadricategories, and so on. We quickly reach a point where no definition of monoidal weak n-categories has been proposed, except by taking this equivalence as primitive and declaring a monoidal weak n-category to be a weak $n+1$ category with a single object.

Monoidal weak ω-categories. We can now define the notion of monoidal weak ω-categories that we describe: They are the weak ω-categories with a single object. With our definition of ω-categories as models of CaTT, this gives the following characterization: For a model F in $\mathbf{Mod}(\mathcal{S}_{\mathsf{CaTT}})$, its set of 0-cells is the image of the 0-th dimensional disk $F(D^0)$, and we thus denote $\mathbf{Mod}_\bullet(\mathcal{S}_{\mathsf{CaTT}})$ the full subcategory of $\mathbf{Mod}(\mathcal{S}_{\mathsf{CaTT}})$ whose objects are exactly the models F of CaTT, such that $F(D^0) = \{\bullet\}$. The category $\mathbf{Mod}_\bullet(\mathcal{S}_{\mathsf{CaTT}})$ is thus the category of monoidal weak ω-categories that we are trying to describe. In what follows, we reserve the term monoidal weak ω-categories for the model of the new type theory MCaTT that we introduce, and call the objects of $\mathbf{Mod}_\bullet(\mathcal{S}_{\mathsf{CaTT}})$ the models of CaTT with a single object before proving that they are equivalent. After proving the equivalence we use the same term for these notions interchangeably.

5.1.2 k-tuply monoidal categories

The construction we have presented here can be iterated, and instead of considering ω-categories with a single object, one can consider a monoidal category with a single object, which corresponds

to the intuition of adding a second monoidal product to a monoidal category. These are known under the name of *twice monoidal categories*, and an Eckmann-Hilton argument shows that corresponds to adding a braiding $a \otimes b \to b \otimes a$ to the monoidal product. Iterating this construction we get *k-tuply monoidal categories* as $(k-1)$-tuply monoidal categories with a single object, and they correspond to categories with a monoidal product which is more and more coherently symmetric. The structure that we get as a limit by iterating this construction infinitely many times is called *symmetric monoidal categories*. These notions are tangential to our work, so we briefly mention them and refer the reader to [9] where they are introduced.

5.2 Type theory for monoidal weak ω-category

We now focus on monoidal categories and define a theory that we call MCaTT. The idea here is to adapt the type theory for weak ω-categories, in order to enforce the constraint that our categories should always have exactly one 0-cell. To this end, we simulate the existence of "a virtual object of dimension -1" in our monoidal categories. There is no formal way in the theory for considering this object, but all the rules act as if it existed. For presenting this theory, we follow the work we have presented in [14] but with a slight modification: We first introduce a globular type theory, that we prove to be correct, and then present a second theory, which is the one presented in the aforementioned article, and that we prove to be equivalent to our globular type theory.

The subcategory of context with one objects. The first candidate for describing the weak ω-categories with only one objects is the full subcategory of $\mathcal{S}_{\mathsf{CaTT}}$ whose objects are the context that define only one object, we denote it $\mathcal{S}_{\mathsf{CaTT},\bullet}$. It is not clear however that there is a structure of categories with families on this category, nor how to find a type theory the present it. The type theory that we introduce now can be understood as a solution to these questions.

5.2.1 The theory MCaTT

We present a globular type theory whose models are monoidal weak ω-categories as an application of our framework from globular type theories. We start by considering our index set J_{MCaTT} to be the same as for CaTT, i.e., it is the set of pairs (Γ, A), where Γ is a ps-context, and A is a type from CaTT satisfying (C_{op}) or ($\mathrm{C}_{\mathsf{coh}}$). The difference with the type theory CaTT relies in the way we associate a context and a type to such a pair.

The desuspension operation. We start by describing an operation in the theory GSeTT only, which, given a non-empty context Γ, associates a context $\downarrow\Gamma$. We call this the *desuspension* of Γ and it is defined together with the corresponding operation on types of GSeTT, by induction

$$\downarrow\varnothing = \varnothing \qquad \downarrow(\Gamma, x : A) = \begin{cases} \downarrow\Gamma & \text{if } A = \star \\ \downarrow\Gamma, x : \downarrow A & \text{otherwise} \end{cases}$$

$$\downarrow\star = \star \qquad x \underset{A}{\longrightarrow} y = \begin{cases} \star & \text{if } A = \star \\ x \longrightarrow y & \text{otherwise} \\ \quad\downarrow_A & \end{cases}$$

Proposition 127. *The desuspension respects the judgments in GSeTT, more exactly:*

- *For any context $\Gamma \vdash$ the judgment $\downarrow\Gamma \vdash$ is derivable*

– *For any type* $\Gamma \vdash A$ *the judgment* $\downarrow\Gamma \vdash \downarrow A$ *is derivable*

– *For any term* $\Gamma \vdash x : A$ *of dimension non-zero, the judgment* $\downarrow\Gamma \vdash x : \downarrow A$ *is derivable.*

Proof. We prove this result by mutual induction

Induction case for contexts:

– For the context $\varnothing \vdash$, the rule (EC) gives a derivation of $\downarrow\varnothing$

– For a context of the form $(\Gamma, x : \star)$, we have $\downarrow(\Gamma, x : \star) = \downarrow\Gamma$ and the induction case for contexts gives directly that $\downarrow(\Gamma, x : A) \vdash$

– For a context of the form $(\Gamma, x : A)$, and A distinct from $\star$, by induction case on types we have $\downarrow\Gamma \vdash \downarrow A$, and applying the rule (CE) yields a derivation for $\downarrow\Gamma, x : \downarrow A \vdash$.

Induction for types:

– For the type of the form $\Gamma \vdash \star$, we necessarily have $\Gamma \vdash$, and by the induction case for contexts, it implies $\downarrow\Gamma \vdash$, hence we can apply the rule ($\star$-INTRO) to get a derivation for $\downarrow\Gamma \vdash \star$.

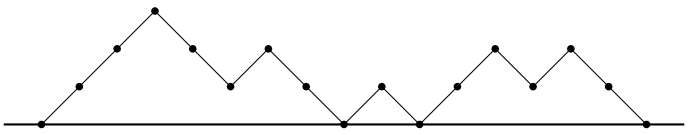
– For a type of the form $\Gamma \vdash t \underset{\star}{\to} u$, we necessarily have $\Gamma \vdash$ by Proposition 2, and the rule ($\to$-INTO) applies to give a derivation of $\downarrow\Gamma \vdash \downarrow(t \underset{\star}{\to} u)$.

– For a type of the form $\Gamma \vdash t \underset{A}{\to} u$ with $A \neq \star$, by the induction cases for types and variables, we have derivations for the judgments $\downarrow\Gamma \vdash$, $\downarrow\Gamma \vdash t : \downarrow A$ and $\downarrow\Gamma \vdash u : \downarrow A$. This lets us apply the rule ($\to$-INTRO) in order to get a derivation for the judgment $\downarrow\Gamma \vdash t \underset{\downarrow A}{\longrightarrow} u$

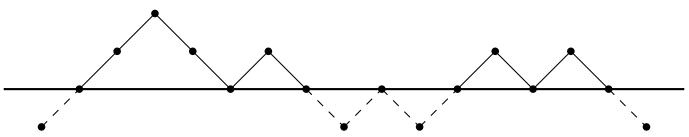
Induction for variables: For any variable $\Gamma \vdash x : A$, of dimension non-zero, A is distinct from $\star$, necessarily we have $\Gamma \vdash$, which by induction for contexts implies $\downarrow\Gamma \vdash$, and $(x : A) \in \Gamma$, which implies $(x : \downarrow A) \in \downarrow\Gamma$. Thus applying the rule (VAR) lets us construct a derivation for $\downarrow\Gamma \vdash x : \downarrow A$.

□

Interpretation of the desuspension. Intuitively, the desuspension operation on ps-contexts corresponds to a shift in dimensions, by decreasing all the dimensions by 1, and forgetting the information that lies in the dimension 0. Visually considering a ps-context with the following combinatorial structure



performing the desuspension yields the following structure, that describes a context that is not a ps-context anymore. In order to visualize, we have kept the dimension 0 as dashed in the diagram, even though there is no such information in the syntax of the theory.



The general desuspension operation. In the theory we are presenting, we denote mop and mcoh the families of term constructors corresponding to op and coh, in such a way that the terms of the theory are either variables or of the form $\mathsf{mop}_{\Gamma,A}[\gamma]$ with Γ a ps-context, A a type satisfying $(\mathrm{C}_{\mathsf{op}})$ and γ a substitution, or $\mathsf{mcoh}_{\Gamma,A}[\gamma]$ with Γ a ps-context, A a type satisfying $(\mathrm{C}_{\mathsf{coh}})$ and γ a substitution. The conditions $(\mathrm{C}_{\mathsf{op}})$ and $(\mathrm{C}_{\mathsf{coh}})$ were introduced in Section 2.4, we recall the definition here

$$(\mathrm{C}_{\mathsf{op}}) \quad A \text{ is of the form } t \underset{B}{\longrightarrow} u \text{ with } \begin{cases} \mathrm{Var}(t) \cup \mathrm{Var}(B) = \mathrm{Var}(\partial^-(\Gamma)) \\ \mathrm{Var}(u) \cup \mathrm{Var}(B) = \mathrm{Var}(\partial^+(\Gamma)) \end{cases}$$

$$(\mathrm{C}_{\mathsf{coh}}) \quad \begin{array}{l} \mathrm{Var}(A) = \mathrm{Var}(\Gamma) \text{ or equivalently} \\ A \text{ is of the form } t \underset{B}{\longrightarrow} u \text{ with } \begin{cases} \mathrm{Var}(t) \cup \mathrm{Var}(B) = \mathrm{Var}(\Gamma) \\ \mathrm{Var}(u) \cup \mathrm{Var}(B) = \mathrm{Var}(\Gamma) \end{cases} \end{array}$$

We generalize the desuspension operation to well-defined contexts, types, terms and substitutions of the theory CaTT, with the following definition

$$\text{For the context } \varnothing \vdash \qquad \text{For the context } (\Gamma, x : A) \vdash$$
$$\downarrow\varnothing = \varnothing \qquad \downarrow(\Gamma, x : A) = \begin{cases} \downarrow\Gamma & \text{if } A = \star \\ \downarrow\Gamma, x : \downarrow A & \text{otherwise} \end{cases}$$

$$\text{For the type } \Gamma \vdash \star \qquad \text{For the type } \Gamma \vdash t \underset{A}{\longrightarrow} u$$
$$\downarrow\star = \star \qquad \downarrow(t \underset{A}{\longrightarrow} u) = \begin{cases} \star & \text{if } A = \star \\ \downarrow t \underset{\downarrow A}{\longrightarrow} \downarrow u & \text{otherwise} \end{cases}$$

$$\text{For a variable } \Gamma \vdash x : A \qquad \text{For the term } \Delta \vdash \mathsf{op}_{\Gamma,A}[\gamma] : A[\gamma]$$
$$\downarrow x = x \qquad \downarrow\mathsf{op}_{\Gamma,A}[\gamma] = \mathsf{mop}_{\Gamma,A}[\downarrow\gamma]$$

$$\text{For the term } \Delta \vdash \mathsf{coh}_{\Gamma,A}[\gamma] : A[\gamma]$$
$$\downarrow\mathsf{coh}_{\Gamma,A}[\gamma] = \mathsf{mcoh}_{\Gamma,A}[\downarrow\gamma]$$

$$\text{For } \Delta \vdash \langle\rangle : \varnothing \qquad \text{For } \Delta \vdash \langle\gamma, x \mapsto t\rangle : (\Gamma, x : A)$$
$$\downarrow\langle\rangle = \langle\rangle \qquad \downarrow\langle\gamma, x \mapsto t\rangle = \begin{cases} \downarrow\gamma & \text{if } A = \star \\ \langle\downarrow\gamma, x \mapsto \downarrow t\rangle & \text{otherwise} \end{cases}$$

The theory MCaTT. We consider the type theory MCaTT obtained by associating to all couple (Γ, A) in J_{CaTT} the context $\downarrow\Gamma$ and the type $\downarrow A$. Note that if (Γ, A) is a couple in J_{CaTT}, then A cannot be the type $\star$. The introduction rules of the theory are thus the following

$$\frac{\Gamma \vdash_{\mathsf{ps}} \qquad \Gamma \vdash A \qquad \Delta \vdash \gamma : \downarrow\Gamma}{\Delta \vdash \mathsf{mop}_{\Gamma,A} : (\downarrow A)[\gamma]}$$

whenever $\Gamma \vdash A$ satisfies $(\mathrm{C}_{\mathsf{op}})$

$$\frac{\Gamma \vdash_{\mathsf{ps}} \qquad \Gamma \vdash A \qquad \Delta \vdash \gamma : \downarrow\Gamma}{\Delta \vdash \mathsf{mcoh}_{\Gamma,A} : (\downarrow A)[\gamma]}$$

whenever $\Gamma \vdash A$ satisfies $(\mathrm{C}_{\mathsf{coh}})$. Note that in order to strictly respect the framework of globular type theory, we should replace the premise $\Gamma \vdash A$ by $\downarrow\Gamma \vdash \downarrow A$. We justify with Proposition 129

that these two premises are equivalent. Importantly, the premise $\Gamma \vdash A$ is a judgment of CaTT, even though we use it to introduce terms in MCaTT, and we thus define a type theory indexed over CaTT. We have compiled and presented all the rules defining MCaTT in Appendix A.3

Properties of this type theory. Since MCaTT is a globular type theory, it enjoys all the properties that we have proved for this framework in Section 4.1, in particular, it has the disks and sphere contexts, which characterize types and terms respectively. Moreover, denoting Θ_{mon} the coherator for this theory, we admit that we can compute the arity limits to be obtained as products of globular products. Theorem 108 then characterizes the models of MCaTT to be the presheaves over Θ_{mon} sending the globular sums to globular products and the sums to products.

Examples. Even though we do not provide an implementation for this theory, we can check by hand a few examples of derivations that are possible

- Monoidal product: We consider the ps-context for the composition of 1-cells,

  $$\Gamma = (x : \star, y : \star, f : x \to y, z : \star, g : y \to z)$$

  and denote

  $$\mathtt{prod} := \mathsf{mop}_{\Gamma, x \to z}$$

  We have $\lfloor \Gamma = (f : \star, g : \star)$, thus for any context Δ in the theory, a substitution $\Delta \vdash \gamma : \lfloor \Gamma$ is just a pair of terms t, u of type $\star$. Suppose that $\Delta \vdash t : \star$ and $\Delta \vdash u : \star$, then the rule (MOP) gives a derivation of the product of t and u:

  $$\Delta \vdash \mathtt{prod}\ t\ u : \star$$

- Associativity of monoidal product: similarly, we consider the ps-context

  $$\Gamma = (x : \star, y : \star, f : x \to y, z : \star, g : y \to z, w : \star, h : z \to w)$$

  and denote

  $$\mathtt{assoc} := \mathsf{mcoh}_{\Gamma, \mathtt{comp}\ (\mathtt{comp}\ f\ g)\ h \to \mathtt{comp}\ f\ (\mathtt{comp}\ g\ h)}$$

  A substitution $\Delta \vdash \gamma : \lfloor \Gamma$ is now a triple of terms t, u, v of type $\star$ in Δ, given such a triple, the rule (MCOH) gives a derivation of

  $$\Gamma \vdash \mathtt{assoc}\ t\ u\ v : \mathtt{prod}\ t\ (\mathtt{prod}\ u\ v) \underset{\star}{\to} \mathtt{prod}\ (\mathtt{prod}\ t\ u)\ v$$

- Neutral element: Consider the ps-context $\Gamma = (x : \star)$, and pose

  $$\mathsf{e} := \mathsf{mcoh}_{\Gamma, x \to x}$$

  A substitution $\Gamma \vdash \gamma : \lfloor \Gamma$ is necessarily the empty substitution, which lets us define by the rule (MCOH) the term

  $$\Delta \vdash \mathsf{e}\langle\rangle : \star$$

We refer the reader to Section 5.3 for a presentation of the theory MCaTT with a lighter syntax, and more examples of derivations along these lines, expressed in that syntax.

5.2.2 Properties of the desuspension

While defining the theory MCaTT we have introduced an operation that we called the desuspension, and that we freely used. We now study this operation more precisely and show some of its properties. Understanding this operation in details is the key step to relating the theories MCaTT and CaTT, in a similar way that monoidal categories and categories are related.

Lemma 128. *Given a substitution $\Delta \vdash \gamma : \Gamma$, for any type $\Gamma \vdash A$, we have $\downarrow(A[\gamma]) = \downarrow A[\downarrow\gamma]$, for any term $\Gamma \vdash t : A$ of dimension non-zero, we have $\downarrow(t[\gamma]) = \downarrow t[\downarrow\gamma]$ and for any substitution $\Gamma \vdash \xi : \Xi$, we have $\downarrow(\xi \circ \gamma) = \downarrow\xi \circ \downarrow\gamma$.*

Proof. We prove this by mutual induction

Induction for types:

- In the case of the type $\star$, we have $\downarrow(\star[\gamma]) = \star$ and $\downarrow\star[\downarrow\gamma] = \star$.
- In the case of a type of the form $A = t \underset{\star}{\to} u$, we have $A[\gamma] = t[\gamma] \underset{\star}{\to} u[\gamma]$, hence $\downarrow(A[\gamma]) = \star$. But also $\downarrow A = \star$, hence $\downarrow A[\downarrow\gamma] = \star$.
- In the case of a type of the form $t \underset{A}{\to} u$ with A distinct from $\star$, we have the following equalities

$$\downarrow((t \underset{A}{\to} u)[\gamma]) = \downarrow(t[\gamma]) \xrightarrow[\downarrow(A[\gamma])]{} \downarrow(u[\gamma])$$

$$\downarrow(t \underset{A}{\to} u)[\downarrow\gamma] = \downarrow t[\downarrow\gamma] \xrightarrow[\downarrow A[\downarrow\gamma]]{} \downarrow u[\downarrow\gamma]$$

  The induction case for types shows that $\downarrow(A[\gamma]) = \downarrow A[\downarrow\gamma]$. Moreover, since the type $t \underset{A}{\to} u$ and the type A is not $\star$, necessarily t and u are of dimension non-zero in Γ, hence the induction case for terms applies, showing the equalities $\downarrow(t[\gamma]) = \downarrow t[\downarrow\gamma]$ and $\downarrow(u[\gamma]) = \downarrow u[\downarrow\gamma]$. These prove the equality between the two above expressions.

Induction for terms of dimension non-zero:

- In the case of a variable $\Gamma \vdash x : A$, denote $t = x[\gamma]$. Since x is of dimension non-zero, A is not the type $\star$, and hence the mapping $x \mapsto \downarrow t$ appears in $\downarrow\gamma$. Hence $x[\downarrow\gamma] = \downarrow(x[\gamma])$.
- In the case of a term of the form $\mathsf{op}_{\Gamma,A}[\delta]$, the following equalities hold

$$\downarrow(\mathsf{op}_{\Gamma,A}[\delta][\gamma]) = \mathsf{mop}_{\Gamma,A}[\downarrow(\delta \circ \gamma)]$$
$$\downarrow\mathsf{op}_{\Gamma,A}[\delta][\downarrow\gamma] = \mathsf{mop}_{\Gamma,A}[\downarrow\delta \circ \downarrow\gamma]$$

  The induction case for substitution then provides the equality between these two expressions.
- Similarly in the case of a term of the form $\mathsf{coh}_{\Gamma,A}[\delta]$, we have the equalities

$$\downarrow(\mathsf{coh}_{\Gamma,A}[\delta][\gamma]) = \mathsf{mcoh}_{\Gamma,A}[\downarrow(\delta \circ \gamma)]$$
$$\downarrow\mathsf{coh}_{\Gamma,A}[\delta][\downarrow\gamma] = \mathsf{mcoh}_{\Gamma,A}[\downarrow\delta \circ \downarrow\gamma]$$

  The induction case for substitution then provides the equality between these two expressions.

Induction for substitutions:

- In the case of the empty substitution $\langle\rangle$, we have $\langle\rangle \circ \gamma = \langle\rangle$, and hence $\downarrow(\langle\rangle \circ \gamma) = \langle\rangle$. But we also have $\downarrow\langle\rangle \circ \gamma = \langle\rangle$.
- In the case of a substitution of the form $\langle \xi, x \mapsto t\rangle$ with x of dimension 0 in Ξ then we have the following equalities

$$\downarrow\langle \xi, x \mapsto t\rangle \circ \gamma = \downarrow(\xi \circ \gamma)$$
$$\downarrow\langle \xi, x \mapsto t\rangle \circ \downarrow\gamma = \downarrow\xi \circ \downarrow\gamma$$

  The induction case for substitution then proves the equality between these two expressions.
- In the case of a substitution of the form $\langle \xi, x \mapsto t\rangle$, with x of dimension non-zero, we have the following equalities

$$\downarrow\langle \xi, x \mapsto t\rangle \circ \gamma = \langle \downarrow(\xi \circ \gamma), x \mapsto \downarrow(t[\gamma])\rangle$$
$$\downarrow\langle \xi, x \mapsto t\rangle \circ \downarrow\gamma = \langle \downarrow\xi \circ \downarrow\gamma, x \mapsto \downarrow t[\downarrow\gamma]\rangle$$

  Then the induction case for substitutions proves that $\downarrow(\xi \circ \gamma) = \downarrow\xi \circ \downarrow\gamma$, and moreover since the term t is of dimension non-zero, the induction case for terms applies and shows that $\downarrow(t[\gamma]) = \downarrow t[\downarrow\gamma]$. This proves the equality between the two above expressions.

□

Correctness of the desuspension. We have defined the desuspension as a syntactic operation between the theories CaTT and MCaTT. This operation actually preserves most of the structure of the theory, and in particular it preserves the derivability of most of the judgments.

Proposition 129. *The following statements hold*

- *For every context $\Gamma \vdash$ in CaTT, the judgment $\downarrow\Gamma \vdash$ is derivable in MCaTT.*
- *For every type $\Gamma \vdash A$ in CaTT, the judgment $\downarrow\Gamma \vdash \downarrow A$ is derivable in MCaTT.*
- *For every term, $\Gamma \vdash t : A$ of dimension non-zero, the judgment $\downarrow\Gamma \vdash \downarrow t : \downarrow A$ is derivable in MCaTT*
- *For every substitution $\Delta \vdash \gamma : \Gamma$ in CaTT, the judgment $\downarrow\Delta \vdash \downarrow\gamma : \downarrow\Gamma$ is derivable in MCaTT.*

Proof. We prove this result by mutual induction

Induction for contexts:

- For the empty context $\varnothing$, we have $\downarrow\varnothing = \varnothing$ and the rule (EC) gives a derivation of $\downarrow\varnothing \vdash$
- For a context of the form $(\Gamma, x : \star) \vdash$, we have necessarily $\Gamma \vdash$ which gives by induction case for the contexts $\downarrow\Gamma \vdash$. Moreover, $\downarrow(\Gamma, x : \star) = \downarrow\Gamma$, hence $\downarrow(\Gamma, x : \star) \vdash$.
- For a context of the form $(\Gamma, x : A) \vdash$ where A is distinct from $\star$, we necessarily have $\Gamma \vdash A$, so the induction case for types shows that $\downarrow\Gamma \vdash \downarrow A$, and hence the rule (CE) applies to give a derivation for $(\downarrow\Gamma, x : \downarrow A) \vdash$. Since we also have $\downarrow(\Gamma, x : A) = (\downarrow\Gamma, x : \downarrow A)$, this lets us conclude that $\downarrow(\Gamma, x : A) \vdash$.

Induction for types:

- For the type $\Gamma \vdash \star$, we necessarily have $\Gamma \vdash$, hence the induction case for contexts shows that $\lfloor\Gamma \vdash$, and applying the rule ($\star$-INTRO) gives a derivation for $\lfloor\Gamma \vdash \lfloor\star$.
- For a type of the form $\Gamma \vdash t \underset{\star}{\to} u$, by Proposition 2, we necessarily have $\Gamma \vdash$, which by the induction case for contexts gives a derivation for $\lfloor\Gamma \vdash$. This lets us apply the rule ($\star$-INTRO) to obtain a derivation of $\lfloor\Gamma \vdash \lfloor(t \underset{\star}{\to} u)$.
- For a type of the form $\Gamma \vdash t \underset{A}{\to} u$ where A is distinct from $\star$, we necessarily have a derivation of $\Gamma \vdash A$, which gives by the induction case for the types a derivation of $\lfloor\Gamma \vdash \lfloor A$. Moreover, we necessarily have two derivations $\Gamma \vdash t : A$ and $\Gamma \vdash u : A$, and since A is not $\star$, both terms t and u have dimension non-zero, hence the induction case for terms gives derivations of $\lfloor\Gamma \vdash \lfloor t : \lfloor A$ and $\lfloor\Gamma \vdash \lfloor u : \lfloor A$. These derivations assemble with the rule ($\to$-INTRO) to provide a derivation of $\lfloor\Gamma \vdash \lfloor t \underset{\lfloor A}{\longrightarrow} \lfloor u$.

Induction for terms:

- For a variable $\Gamma \vdash x : A$ of dimension non-zero, we necessarily have $\Gamma \vdash$ which by the induction case for contexts provides a derivation of $\lfloor\Gamma \vdash$. Moreover, we have the condition $(x : A) \in \Gamma$, and since x is of dimension non-zero, A is not the type $\star$, hence $(x : \lfloor A) \in \lfloor\Gamma$. This lets us apply the rule (VAR) in order to prove $\lfloor\Gamma \vdash x : \lfloor A$.
- For a term of the form $\Delta \vdash \mathsf{op}_{\Gamma,A}[\gamma] : A[\gamma]$, necessarily (Γ, A) defines a valid operation cut, and we have a derivation for $\Delta \vdash \gamma : \Gamma$ which by the induction case for substitutions provides a derivation for $\lfloor\Delta \vdash \lfloor\gamma : \lfloor\Gamma$. This lets us apply the rule (MOP) to construct the term $\lfloor\Delta \vdash \mathsf{mop}_{\Gamma,A}[\lfloor\gamma] : \lfloor A[\lfloor\gamma]$. Lemma 128 then shows that it is a derivation of $\lfloor\Delta \vdash \lfloor(\mathsf{op}_{\Gamma,A}[\lfloor\gamma]) : \lfloor(A[\gamma])$.
- Similarly, for a term of the form $\Delta \vdash \mathsf{coh}_{\Gamma,A}[\gamma] : A[\gamma]$, necessarily (Γ, A) defines a valid coherence cut, and we have a derivation for $\Delta \vdash \gamma : \Gamma$ which by the induction case for substitutions provides a derivation for $\lfloor\Delta \vdash \lfloor\gamma : \lfloor\Gamma$. This lets us apply the rule (MCOH) to construct the term $\lfloor\Delta \vdash \mathsf{mcoh}_{\Gamma,A}[\lfloor\gamma]$. Again Lemma 128 shows that this is a derivation of $\lfloor\Delta \vdash \lfloor(\mathsf{coh}_{\Gamma,A}[\gamma]) : \lfloor(A[\gamma])$.

Induction for substitutions:

- For the empty substitution $\Delta \vdash \langle\rangle : \varnothing$, we necessarily have a derivation of $\Delta \vdash$, and hence by induction case for the context, this produces a derivation of $\lfloor\Delta \vdash$. Applying the rule (ES), we get a derivation of $\lfloor\Delta \vdash \langle\rangle : \varnothing$.
- For a substitution of the form $\Delta \vdash \langle\gamma, x \mapsto t\rangle : (\Gamma, x : \star)$, we necessarily have a derivation of $\Delta \vdash \gamma : \Gamma$, which by the induction case for the substitutions gives a derivation of $\lfloor\Delta \vdash \lfloor\gamma : \lfloor\Gamma$. Since $\lfloor\langle\gamma, x :\mapsto t\rangle = \lfloor\gamma$ and $\lfloor(\Gamma, x : A) = \lfloor\Gamma$, this is in fact a derivation of $\lfloor\Delta \vdash \lfloor\langle\gamma, x \mapsto t\rangle : \lfloor(\Gamma, x : A)$.
- For a substitution of the form $\Delta \vdash \langle\gamma, x \mapsto t\rangle : (\Gamma, x : A)$ with A distinct from $\star$, we necessarily have a derivation of $\Delta \vdash \gamma : \Gamma$, which by the induction case for the substitutions gives a derivation of $\lfloor\Delta \vdash \lfloor\gamma : \lfloor\Gamma$. Moreover, we have a derivation of $(\Gamma, x : A) \vdash$ which provides, by the induction rule for contexts a derivation of $\lfloor(\Gamma, x : A) \vdash$, which reduces to $(\lfloor\Gamma, x : \lfloor A)$ since A is distinct from $\star$. Finally, the substitution also gives derivation of

$\Delta \vdash t : A[\gamma]$, and since A is not $\star$, this term is of dimension non-zero and the induction case for terms applies to give a derivation of $\downarrow\Delta \vdash \downarrow t : \downarrow(A[\gamma])$. Since the term t is of dimension non-zero, Lemma 128 applies and this judgment rewrites as $\downarrow\Delta \vdash \downarrow t : \downarrow A[\downarrow\gamma]$. We can then apply the rule (SE) in order to get a derivation of $\downarrow\Delta \vdash \downarrow\langle\gamma, x \mapsto t\rangle : (\Gamma, x : A)$ as follows

$$\frac{\downarrow\Delta \vdash \downarrow\gamma : \downarrow\Gamma \qquad \downarrow\Gamma, x : \downarrow A \vdash \qquad \downarrow\Delta \vdash \downarrow t : \downarrow A[\downarrow\gamma]}{\downarrow\Delta \vdash \langle\downarrow\gamma, x \mapsto \downarrow t\rangle : (\downarrow\Gamma, x : \downarrow A)}(\text{SE})$$

□

This proposition allows to replace the premise $\downarrow\Gamma \vdash \downarrow A$ in the rules (MOP) and (MCOH), by the premise $\Gamma \vdash A$. This premise is simpler, but it takes place in the theory CaTT, thus making MCaTT indexed over CaTT.

The desuspension as a functor. We have proved in Proposition 129 that the desuspension operation sends a well-defined context in the theory CaTT onto a well-defined context in the theory MCaTT, and sends a well-defined substitution in CaTT onto a well-defined substitution in MCaTT. Moreover Lemma 128 shows that the desuspension respects the composition of substitutions, and we can verify by induction that it also respect identity substitutions. This shows the desuspension defines a functor $\mathcal{S}_{\mathsf{CaTT}} \to \mathcal{S}_{\mathsf{MCaTT}}$. This categorical reformulation concerns only some of the results stated as Lemmas 128 and Proposition 129, and the other stated results which adds additional structure to the desuspension functor. For starters, Proposition 129 shows that the desuspension acts on types and associate to every well-defined type in a context, a well defined type in the its desuspension, and similarly for terms of type distinct from $\star$, while also respecting the typing and Lemma 128 ensures that this association is functorial whenever it is defined. This shows that in fact the desuspension almost lifts as a morphism in **Cat**/**Fam**, where we see $\mathcal{S}_{\mathsf{CaTT}}$ and $\mathcal{S}_{\mathsf{MCaTT}}$ as elements of **Cat**/**Fam** with the functor induced by the structure of categories with families. It does not quite define such a morphism, as it fails to give an image for the terms of type $\star$, but it gives an image for the types and all the other terms. Moreover, we have $\downarrow\varnothing = \varnothing$ so $\downarrow$ preserves the terminal object, and for all type A distinct from $\star$, we have $\downarrow(\Gamma, x : A) = (\downarrow\Gamma, x : \downarrow A)$ so it also preserves these context extensions on the nose. This shows that $\downarrow$ almost defines a morphism of categories with families, the only obstruction is that it fails to provides an image for the terms of type $\star$

5.2.3 Reduced suspension

We define another translation related to the desuspension that goes in the other direction: From an expression in the theory MCaTT, it associates an expression in the theory CaTT. We call this translation the *reduced suspension* and the intuition is that it exhibits MCaTT as the theory of the judgments in CaTT concerned with context that only have one object. It is closely related to the operation of suspension, that we have introduced in Section 3.2.

The reduced suspension operation. We assume that there is a variable $\bullet_0$ that is free (say we extend the set of variable of MCaTT by adding the variable $\bullet_0$), and define the reduced suspension $\uparrow$ by induction on the syntax. Note that contrarily to the desuspension, the reduced suspension is defined purely syntactically and does not require any judgment to be derivable in

the type theory MCaTT.

$$\uparrow\varnothing = (\bullet_0 : \star) \qquad \uparrow(\Gamma, x : A) = (\uparrow\Gamma, x : \uparrow A)$$

$$\uparrow\star = \bullet_0 \underset{\star}{\to} \bullet_0 \qquad \uparrow\Big(t \underset{A}{\to} u\Big) = \uparrow t \underset{\uparrow A}{\longrightarrow} \uparrow u$$

$$\uparrow x = x \qquad \uparrow \mathsf{mop}_{\Gamma,A}[\gamma] = \mathsf{op}_{\Gamma,A}[\bullet_\Gamma \circ \uparrow\gamma]$$

$$\uparrow \mathsf{mcoh}_{\Gamma,A}[\gamma] = \mathsf{coh}_{\Gamma,A}[\bullet_\Gamma \circ \uparrow\gamma]$$

$$\uparrow\langle\rangle = \langle \bullet_0 \mapsto \bullet_0 \rangle \qquad \uparrow\langle \gamma, x \mapsto t\rangle = \langle \uparrow\gamma, x \mapsto \uparrow t\rangle$$

Where the substitution $\bullet_\Gamma$ is defined by induction on the context Γ as follows

$$\bullet_\varnothing = \langle\rangle \qquad \bullet_{(\Gamma, x:A)} = \begin{cases} \langle \bullet_\Gamma, x \mapsto \bullet_0 \rangle & \text{if } A = \star \\ \langle \bullet_\Gamma, x \mapsto x \rangle & \text{Otherwise} \end{cases}$$

Before showing that this operation respects the derivability of judgments, we first show some useful syntactic properties.

Lemma 130. *For all substitution γ, we always have $\bullet_0[\uparrow\gamma] = \bullet_0$.*

Proof. By definition of $\uparrow\gamma$, the mapping $\bullet_0 \mapsto \bullet_0$ appears in $\uparrow\gamma$. Moreover, by assumption $\bullet_0$ is a fresh variable, so it cannot appear in γ, and hence it cannot appear anywhere else in $\uparrow\gamma$, hence we necessarily have $\bullet_0[\uparrow\gamma] = \bullet_0$. Note that we prove this result for all substitutions, even the potentially ill-formed ones, this is why we need to check that $\bullet_0$ can only be mapped on one term. □

Lemma 131. *Given a substitution γ in the theory MCaTT, For any type A we have the equality $\uparrow(A[\gamma]) = \uparrow A[\uparrow\gamma]$, for any term t, we have the equality $\uparrow(t[\gamma]) = \uparrow t[\uparrow\gamma]$ and for any substitution δ, we have the equality $\uparrow(\delta \circ \gamma) = \uparrow\delta \circ \uparrow\gamma$.*

Proof. We suppose given a substitution γ and prove these three results by mutual induction

Induction for types:

- For the type $\star$, we have $\star[\gamma] = \star$, hence $\uparrow(\star[\gamma]) = \bullet_0 \underset{\star}{\to} \bullet_0$. Since we have the equality $\uparrow\star[\uparrow\gamma] = \bullet_0[\uparrow\gamma] \underset{\star}{\to} \bullet_0[\uparrow\gamma]$, and using Lemma 130, this implies $\uparrow A\ [\uparrow\gamma] = \bullet_0 \underset{\star}{\to} \bullet_0$.
- For the type $t \underset{A}{\to} u$, we have the two following equalities

$$\uparrow((t \underset{A}{\to} u)[\gamma]) = \uparrow(t[\gamma]) \underset{\uparrow(A[\gamma])}{\longrightarrow} \uparrow(u[\gamma])$$

$$(\uparrow(t \underset{A}{\to} u))[\uparrow\gamma] = (\uparrow t)[\uparrow\gamma] \underset{(\uparrow A)[\uparrow\gamma]}{\longrightarrow} (\uparrow u)[\uparrow\gamma]$$

  The induction case for type shows that $\uparrow(A[\gamma]) = (\uparrow A)[\uparrow\gamma]$ and the induction case for terms proves the equalities $\uparrow(t[\gamma]) = (\uparrow t)[\uparrow\gamma]$ and $\uparrow(u[\gamma]) = (\uparrow u)[\uparrow\gamma]$. These three equalities show the equality between the two previous expressions.

Induction for terms:

– For a variable x, such that the mapping $x \mapsto t$ appears in γ, we have $x[\gamma] = t$, and hence $\uparrow(x[\gamma]) = \uparrow t$. Moreover the mapping $x \mapsto \uparrow t$ appears in $\uparrow\gamma$, and hence $x[\uparrow\gamma] = \uparrow t$.

– For a term of the form $\mathsf{mop}_{\Gamma,A}[\delta]$ we have the two following equalities

$$\uparrow(\mathsf{mop}_{\Gamma,A}[\delta][\gamma]) = \mathsf{op}_{\Gamma,A}[\bullet_\Gamma \circ \uparrow(\delta \circ \gamma)]$$
$$(\uparrow\mathsf{mop}_{\Gamma,A}[\delta])[\uparrow\gamma] = \mathsf{op}_{\Gamma,A}[(\bullet_\Gamma \circ \uparrow\delta) \circ \uparrow\gamma]$$

and the induction case for substitutions together with the associativity of composition for substitution show that these two terms are equal.

– Similarly for a term of the form $\mathsf{mcoh}_{\Gamma,A}[\delta]$ we have the two following equalities

$$\uparrow(\mathsf{mcoh}_{\Gamma,A}[\delta][\gamma]) = \mathsf{coh}_{\Gamma,A}[\bullet_\Gamma \circ \uparrow(\delta \circ \gamma)]$$
$$(\uparrow\mathsf{mcoh}_{\Gamma,A}[\delta])[\uparrow\gamma] = \mathsf{coh}_{\Gamma,A}[(\bullet_\Gamma \circ \uparrow\delta) \circ \uparrow\gamma]$$

and by induction and associativity these two expressions are equal.

Induction for contexts:

– In the case of the empty substitutions, we have $\langle\rangle \circ \gamma = \langle\rangle$, and hence $\uparrow(\langle\rangle \circ \gamma) = \langle \bullet_0 \mapsto \bullet_0 \rangle$. Moreover, $\uparrow\langle\rangle \circ \uparrow\gamma = \langle \bullet_0 \mapsto \bullet_0[\uparrow\gamma] \rangle$. By Lemma 130 this shows that $\uparrow\langle\rangle \circ \uparrow\gamma = \langle \bullet_0 \mapsto \bullet_0 \rangle$.

– In the case of a substitution of the form $\langle \delta, x \mapsto t \rangle$, we have the following equalities

$$\uparrow(\langle \delta, x \mapsto t \rangle \circ \gamma) = \langle \uparrow(\delta \circ \gamma), x \mapsto \uparrow(t[\gamma]) \rangle$$
$$\uparrow\langle \delta, x \mapsto t \rangle \circ \uparrow\gamma = \langle \uparrow\delta \circ \uparrow\gamma, x \mapsto (\uparrow t)[\uparrow\gamma] \rangle$$

and the induction cases for substitutions and terms provide the result.

□

Reduced suspension on the theory GSeTT. Our definition of the reduced suspension can be restricted to the theory GSeTT, since this theory is included in MCaTT. Since the term constructors op and coh are only used as images for term constructors, computing the reduced suspension of an expression in the theory GSeTT yields again an expression which is in the theory GSeTT and hence the reduced suspension can be understood as an operation from the theory GSeTT to itself. Understanding this operation first is key in studying the general reduced suspension between MCaTT and CaTT, since general terms are introduced by substitutions to a ps-contexts, that are special cases of contexts in GSeTT.

Lemma 132. *In the theory GSeTT, the following properties are satisfied*

– *For any well-defined context* $\Gamma \vdash$*, the judgment* $\uparrow\Gamma$ *is derivable.*

– *For any type* $\Gamma \vdash A$*, the judgment* $\uparrow\Gamma \vdash \uparrow A$ *is derivable.*

– *For any term* $\Gamma \vdash t : A$*, the judgment* $\uparrow\Gamma \vdash \uparrow t : \uparrow A$ *is derivable.*

– *For any substitution* $\Delta \vdash \gamma : \Gamma$*, the judgment* $\uparrow\Delta \vdash \uparrow\gamma : \uparrow\Gamma$ *is derivable.*

Proof. We prove this result by mutual induction on contexts, types, terms and substitutions

Induction for contexts:

– In the case of the empty context $\varnothing$, we have $\uparrow\varnothing = (\bullet_0 : \star)$, and the following derivation shows that $\uparrow\varnothing \vdash$

$$\dfrac{\dfrac{\dfrac{}{\varnothing \vdash}(\text{EC})}{\varnothing \vdash \star}(\star\text{-INTRO})}{\bullet_0 : \star \vdash}(\text{CE})$$

– In the case of a context of the form $(\Gamma, x : A) \vdash$, we can extract a derivation of $\Gamma \vdash A$ and by the induction case for types, it gives a derivation for $\uparrow\Gamma \vdash \uparrow A$. by applying the rule (CE), we get a derivation for $(\uparrow\Gamma, x : \uparrow A) \vdash$

Induction for types:

– In the case of the type $\Gamma \vdash \star$, we can extract a derivation of $\Gamma \vdash$, which by the induction rule for contexts shows $\uparrow\Gamma \vdash$. Since moreover $(\bullet_0 : \star) \in \uparrow\Gamma$, we can construct a derivation of $\uparrow\Gamma \vdash \uparrow\star$ as follows

$$\dfrac{\dfrac{\uparrow\Gamma \vdash}{\uparrow\Gamma \vdash \star}(\star\text{-INTRO}) \qquad \dfrac{\uparrow\Gamma \vdash \qquad (\bullet_0 : \star) \in \uparrow\Gamma}{\uparrow\Gamma \vdash \bullet_0 : \star \qquad \uparrow\Gamma \vdash \bullet_0 : \star}(\text{VAR})}{\uparrow\Gamma \vdash \bullet_0 \underset{\star}{\to} \bullet_0}(\to\text{-INTRO})$$

– In the case of a type of the form $\Gamma \vdash t \underset{A}{\longrightarrow} u$, we have a derivation for $\Gamma \vdash A$, which by the induction case for types gives a derivation for $\uparrow\Gamma \vdash \uparrow A$ and two derivations for $\Gamma \vdash t : A$ and $\Gamma \vdash u : A$, which by the induction case for terms gives $\uparrow\Gamma \vdash \uparrow t : \uparrow A$ and $\uparrow\Gamma \vdash \uparrow u : \uparrow A$. These three derivations assemble with the rule ($\to$-INTRO) in order to produce a derivation for the judgment $\uparrow\Gamma \vdash \uparrow t \underset{\uparrow A}{\longrightarrow} \uparrow u$.

Induction for terms: A term is necessarily a variable $\Gamma \vdash x : A$. Since we have $\Gamma \vdash$, by the induction case for contexts we also get $\uparrow\Gamma \vdash$. Moreover, the condition $(x : A) \in \Gamma$ is also necessarily satisfied, which implies that $(x : \uparrow A) \in \uparrow\Gamma$. Hence, applying the rule (VAR) yields a derivation for $\uparrow\Gamma \vdash x : \uparrow A$.

Induction for substitutions:

– In the case of the empty substitution $\Delta \vdash \langle\rangle : \varnothing$, we necessarily have a derivation of $\Delta \vdash$, which by the induction case for contexts gives a derivation of $\uparrow\Delta \vdash$. Moreover, we also have by definition that $(\bullet_0 : \star) \in \uparrow\Delta$. This lets us construct a derivation for $\uparrow\Delta \vdash \uparrow\langle\rangle : \uparrow\varnothing$ as follows

$$\dfrac{\dfrac{\uparrow\Delta \vdash}{\uparrow\Delta \vdash \langle\rangle : \varnothing}(\text{ES}) \qquad \bullet_0 : \star \vdash \qquad \dfrac{\uparrow\Delta \vdash \qquad (\bullet_0 : \star) \in \uparrow\Delta}{\uparrow\Delta \vdash \bullet_0 : \star}(\text{VAR})}{\uparrow\Delta \vdash \langle \bullet_0 \mapsto \bullet_0 \rangle : (\bullet_0 : \star)}(\text{SE})$$

– In the case of a substitution of the form $\Delta \vdash \langle \gamma, x \mapsto t \rangle : (\Gamma, x : A)$, we necessarily have $\Delta \vdash \gamma : \Gamma$, which gives by the induction case for substitutions, $\uparrow\Delta \vdash \uparrow\gamma : \uparrow\Gamma$. Moreover, we have a derivation of $(\Gamma, x : A) \vdash$, which gives by the induction case for contexts a derivation of $\uparrow\Gamma, x : \uparrow A \vdash$, and a derivation of $\Delta \vdash t : A[\gamma]$, which gives by the induction case for

terms a derivation of $\uparrow\Delta \vdash \uparrow t : \uparrow A[\gamma]$. Lemma 131 allows to rewrite this into a derivation for $\uparrow\Delta \vdash \uparrow t : \uparrow A[\uparrow\gamma]$. Thus these derivations can be assembled using the rule (SE) into a derivation of $\uparrow\Delta \vdash \uparrow\langle\gamma, x \mapsto t\rangle : \uparrow(\Gamma, x : A)$ as follows

$$\frac{\uparrow\Delta \vdash \uparrow\gamma : \uparrow\Gamma \qquad \uparrow\Gamma, x : \uparrow A \vdash \qquad \uparrow\Delta \vdash \uparrow t : \uparrow A[\uparrow\gamma]}{\uparrow\Delta \vdash \langle\uparrow\gamma, x \mapsto \uparrow t\rangle : (\uparrow\Gamma, x : \uparrow A)}(\text{SE})$$

□

Properties of the substitution $\bullet_\Delta$. In order to study the reduced suspension operation, we also need to give the properties of the substitution $\bullet_\Delta$ for a ps-context Δ, on which the reduced suspension relies.

Lemma 133. *We have the following result for the action of the substitution $\bullet_\Delta$ on terms, types and substitutions.*

- *For any type $\Delta \vdash A$ distinct from $\star$, $A[\bullet_\Delta] = \uparrow\downarrow A$*
- *For any term $\Delta \vdash t : A$ with A distinct from $\star$, $t[\bullet_\Delta] = \uparrow\downarrow t$.*
- *For any substitution $\Delta \vdash \gamma : \Gamma$, $\gamma \circ \bullet_\Delta = \bullet_\Gamma \circ \uparrow\downarrow\gamma$.*

Proof. We prove these equalities by mutual induction

Induction on types:

- For a type of the form $A = t \underset{\star}{\to} u$, we have $\uparrow\downarrow A = \bullet_0 \underset{\star}{\to} \bullet_0$. Moreover, we necessarily have $\Delta \vdash t : \star$ and $\Delta \vdash u : \star$ and the only possible terms of type $\star$ in CaTT are the variables, hence t and u are variables of type $\star$ in Γ. This implies that the associations $t \mapsto \bullet_0$ and $u \mapsto \bullet_0$ appear in $\bullet_\Delta$, and hence $t[\bullet_\Delta] = \bullet_0$ and $u[\bullet_\Delta] = \bullet_0$. Since we also have $\star[\bullet_\Delta] = \star$, this proves the equality.
- For a type of the form $t \underset{A}{\to} u$ with A distinct from $\star$, we have the following equalities

$$(t \underset{A}{\to} u)[\bullet_\Delta] = t[\bullet_\Delta] \underset{A[\bullet_\Delta]}{\longrightarrow} u[\bullet_\Delta]$$

$$\uparrow\downarrow(t \underset{A}{\to} u) = \uparrow\downarrow t \underset{\uparrow\downarrow A}{\longrightarrow} \uparrow\downarrow u$$

  By the induction case on types $A[\bullet_\Delta] = \uparrow\downarrow A$, and by the induction case on terms $t[\bullet_\Delta] = \uparrow\downarrow t$ and $u[\bullet_\Delta] = \uparrow\downarrow u$. This proves the equality $(t \underset{A}{\to} u)[\bullet_\Delta] = \uparrow\downarrow(t \underset{A}{\to} u)$.

Induction on terms:

- In the case of a variable $\Delta \vdash x : A$, with A distinct from $\star$, by definition of the substitution $\bullet_\Delta$, the association $x \mapsto x$ appears in $\bullet_\Delta$, hence $x[\bullet_\Delta] = x$. Moreover, by definition, $\uparrow\downarrow x = x$, hence the equality.
- In the case of a term of the form $\Delta \vdash \mathsf{op}_{\Gamma,A}[\gamma] : A[\gamma]$, we have the following equations

$$\mathsf{op}_{\Gamma,A}[\gamma][\bullet_\Delta] = \mathsf{op}_{\Gamma,A}[\gamma \circ \bullet_\Delta]$$
$$\uparrow\downarrow\mathsf{op}_{\Gamma,A}[\gamma] = \mathsf{op}_{\Gamma,A}[\bullet_\Gamma \circ \uparrow\downarrow\gamma]$$

  and the induction case for substitutions then gives the equality.

– Similarly, in the case of a term of the form $\Delta \vdash \mathsf{coh}_{\Gamma,A}[\gamma]$ we have

$$\mathsf{coh}_{\Gamma,A}[\gamma][\bullet_\Delta] = \mathsf{coh}_{\Gamma,A}[\gamma \circ \bullet_\Delta]$$
$$\uparrow\downarrow\mathsf{coh}_{\Gamma,A}[\gamma] = \mathsf{coh}_{\Gamma,A}[\bullet_\Gamma \circ \uparrow\downarrow\gamma]$$

and we conclude by the induction case for substitutions.

Induction case for substitutions:

– In the case of the empty substitution $\Delta \vdash \langle\rangle : \varnothing$, since $\bullet_\varnothing = \langle\rangle$, we have the two following equalities

$$\langle\rangle \circ \bullet_\Delta = \langle\rangle$$
$$\bullet_\varnothing \circ \uparrow\downarrow\langle\rangle = \langle\rangle$$

– In the case of a substitution of the form $\Delta \vdash \langle\gamma, x \mapsto t\rangle : (\Gamma, x : \star)$, since $\uparrow\downarrow\langle\gamma, x \mapsto t\rangle = \uparrow\downarrow\gamma$, we have the following equalities

$$\langle\gamma, x \mapsto t\rangle \circ \bullet_\Delta = \langle\gamma \circ \bullet_\Delta, x \mapsto t[\bullet_\Delta]\rangle$$
$$\langle\bullet_\Gamma, x \mapsto \bullet_0\rangle \circ \uparrow\downarrow\gamma = \langle\bullet_\Gamma \circ \uparrow\downarrow\gamma, x \mapsto \bullet_0[\uparrow\downarrow\gamma]\rangle$$

By the induction case for substitutions, we have $\gamma \circ \bullet_\Delta = \bullet_\Gamma \circ \uparrow\downarrow\gamma$, by Lemma 130, we have the equality $\bullet_0[\uparrow\downarrow\gamma] = \bullet_0$, and finally since t is a term of dimension 0 in the theory CaTT, it is necessarily a variable of type $\star$ in Δ, and by definition of $\bullet_\Delta$, the mapping $t \mapsto \bullet_0$ appears in $\bullet_\Delta$, hence $t[\bullet_\Delta] = \bullet_0$. this proves that $\langle\gamma, x \mapsto t\rangle \circ \bullet_\Delta = \bullet_\Gamma \circ \uparrow\downarrow\langle\gamma, x \mapsto t\rangle$.

– In the case of a substitution of the form $\Delta \vdash \langle\gamma, x \mapsto t\rangle : (\Gamma, x : A)$ with A distinct of $\star$, we have the following equalities

$$\langle\gamma, x \mapsto t\rangle \circ \bullet_\Delta = \langle\gamma \circ \bullet_\Delta, x \mapsto t[\bullet_\Delta]\rangle$$
$$\langle\bullet_\Gamma, x \mapsto x\rangle \circ \langle\uparrow\downarrow\gamma, x \mapsto \uparrow\downarrow t\rangle = \langle\bullet_\Gamma \circ \langle\uparrow\downarrow\gamma, x \mapsto \uparrow\downarrow t\rangle, x \mapsto x[\langle\uparrow\downarrow\gamma, x \mapsto \uparrow\downarrow t\rangle]\rangle$$

Since the substitution $\bullet_\Gamma$ has source $\uparrow\downarrow\Gamma$ that do not use the variable x, we have

$$\bullet_\Gamma \circ \langle\uparrow\downarrow\gamma, x \mapsto \uparrow\downarrow t\rangle = \bullet_\Gamma \circ \uparrow\downarrow t$$

Moreover the expression $x[\langle\uparrow\downarrow\gamma, x \mapsto \uparrow\downarrow t\rangle]$ simplifies to $\uparrow\downarrow t$. Hence we have the equality

$$\langle\bullet_\Gamma, x \mapsto x\rangle \circ \langle\uparrow\downarrow\gamma, x \mapsto \uparrow\downarrow t\rangle = \langle\bullet_\Gamma \circ \uparrow\downarrow\gamma, x \mapsto \uparrow\downarrow t\rangle$$

the induction case for substitutions then shows that $\gamma \circ \bullet_\Delta = \bullet_\Gamma \circ \uparrow\downarrow\gamma$, and the induction case for terms shows, since t is of dimension non-zero, that $t[\bullet_\Delta] = \uparrow\downarrow t$, which proves the equality

$$\langle\gamma, x \mapsto t\rangle \circ \bullet_\Delta = \langle\bullet_\Gamma, x \mapsto x\rangle \circ \langle\uparrow\downarrow\gamma, x \mapsto \uparrow\downarrow t\rangle$$

□

Lemma 134. *For any context $\Gamma \vdash$ in GSeTT, there is a well defined substitution $\uparrow\downarrow\Gamma \vdash \bullet_\Gamma : \Gamma$.*

Proof. This result is proved by induction on the context Γ.

– For the empty context $\varnothing$, we have already proven that $\uparrow\downarrow\varnothing \vdash$. Hence the rule (ES) proves that $\uparrow\downarrow\Gamma \vdash \langle\rangle : \varnothing$.

– For a context of the form $(\Gamma, x : \star) \vdash$, we have $\uparrow\downarrow(\Gamma, x : \star) = \uparrow\downarrow\Gamma$, thus the induction shows that we have $\uparrow\downarrow(\Gamma, x : \star) \vdash \bullet_\Gamma : \Gamma$. Moreover, we have $\uparrow\downarrow(\Gamma, x : \star) \vdash$, and the association $(\bullet_0 : \star)$ appears in $\uparrow\downarrow\Gamma, x : \star$, hence the rule (VAR) applies and provides a derivation of $\uparrow\downarrow(\Gamma, x : \star) \vdash \bullet_0 : \star$. This lets us apply the rule (SE)to build the following derivation of the judgment $\uparrow\downarrow(\Gamma, x : \star) \vdash \bullet_{(\Gamma,x:\star)} : (\Gamma, x : \star)$:

$$\frac{\uparrow\downarrow(\Gamma, x : \star) \vdash \bullet_\Gamma : \Gamma \qquad (\Gamma, x : \star) \vdash \qquad \uparrow\downarrow\Gamma, x : \star \vdash \bullet_0 : \star}{\uparrow\downarrow(\Gamma, x : \star) \vdash \langle \bullet_\Gamma, x :\mapsto \bullet_0 \rangle : (\Gamma, x : \star)}(\text{SE})$$

– For a context of the form $(\Gamma, x : A) \vdash$ with A distinct from $\star$, we have by induction that $\uparrow\downarrow\Gamma \vdash \bullet_\Gamma$, and hence by weakening (c.f. Proposition 2) this shows that we also have $\uparrow\downarrow(\Gamma, x : A) \vdash \bullet_\Gamma : \Gamma$. Moreover, by Lemma 132, we have that $\uparrow\downarrow(\Gamma, x : A) \vdash x : \uparrow\downarrow A$, and since A is distinct from $\star$, Lemma 133 then shows that we have $\uparrow\downarrow(\Gamma, x : A) \vdash x : A[\bullet_\Gamma]$. This lets us apply the rule (SE) to construct a derivation of $\uparrow\downarrow(\Gamma, x : A) \vdash \bullet_{(\Gamma,x:A)} : (\Gamma, x : A)$ as follows

$$\frac{\uparrow\downarrow(\Gamma, x : A) \vdash \bullet_\Gamma : \Gamma \qquad (\Gamma, x : A) \vdash \qquad \uparrow\downarrow(\Gamma, x : A) \vdash x : A[\bullet_\Gamma]}{\uparrow\downarrow(\Gamma, x : A) \vdash \langle \bullet_{(\Gamma,x:A)}, x \mapsto x \rangle : (\Gamma, x : A)}(\text{SE})$$

□

Correctness of the reduced suspension. We are now equipped to study the reduced suspension operation in its full generality, as an operation from the theory MCaTT to the theory CaTT. In particular we prove the following correctness result showing that this operation is a well defined translation between these two theories.

Proposition 135. *The reduced suspension operation preserves derivability. More precisely, we have*

– *For any context $\Delta \vdash$ derivable in the theory MCaTT, $\uparrow\Delta \vdash$ is derivable in the theory CaTT.*

– *For any type $\Delta \vdash A$ derivable in the theory MCaTT, $\uparrow\Delta \vdash \uparrow A$ is derivable in the theory CaTT.*

– *For any term $\Delta \vdash t : A$ derivable in the theory MCaTT, $\uparrow\Delta \vdash \uparrow t : \uparrow A$ is derivable in the theory CaTT.*

– *For any substitution $\Delta \vdash \gamma : \Gamma$ derivable in the theory MCaTT, $\uparrow\Delta \vdash \uparrow\gamma : \uparrow\Gamma$ is derivable in the theory CaTT.*

Proof. The proof of this result is essentially the same than the proof of Lemma 132 by mutual induction, and adding additional cases for the term constructors we added. For the sake of completeness and to avoid the reader to constantly refer to the previous proof, we still give the entire proof here, by mutual induction on contexts, types, terms and substitutions.

Induction for contexts:

– In the case of the empty context $\varnothing$, we have $\uparrow\varnothing = (\bullet_0 : \star)$, and the following derivation shows that $\uparrow\varnothing \vdash$

$$\cfrac{\cfrac{\cfrac{}{\varnothing \vdash}(\text{EC})}{\varnothing \vdash \star}(\star\text{-INTRO})}{\bullet_0 : \star \vdash}(\text{CE})$$

– In the case of a context of the form $(\Gamma, x : A) \vdash$, we can extract a derivation of $\Gamma \vdash A$ and by the induction case for types, it gives a derivation for $\uparrow\Gamma \vdash \uparrow A$. by applying the rule (CE), we get a derivation for $(\uparrow\Gamma, x : \uparrow A) \vdash$

Induction for types:

– In the case of the type $\Gamma \vdash \star$, we can extract a derivation of $\Gamma \vdash$, which by the induction rule for contexts shows $\uparrow\Gamma \vdash$. Since moreover $(\bullet_0 : \star) \in \uparrow\Gamma$, we can construct a derivation of $\uparrow\Gamma \vdash \uparrow\star$ as follows

$$\dfrac{\dfrac{\uparrow\Gamma \vdash}{\uparrow\Gamma \vdash \star}(\star\text{-INTRO}) \qquad \dfrac{\uparrow\Gamma \vdash \qquad (\bullet_0 : \star) \in \uparrow\Gamma}{\uparrow\Gamma \vdash \bullet_0 : \star \qquad \uparrow\Gamma \vdash \bullet_0 : \star}(\text{VAR})}{\uparrow\Gamma \vdash \bullet_0 \underset{\star}{\to} \bullet_0}(\to\text{-INTRO})$$

– In the case of a type of the form $\Gamma \vdash t \underset{A}{\longrightarrow} u$, we have a derivation for $\Gamma \vdash A$, which by the induction case for types gives a derivation for $\uparrow\Gamma \vdash \uparrow A$ and two derivations for $\Gamma \vdash t : A$ and $\Gamma \vdash u : A$, which by the induction case for terms gives $\uparrow\Gamma \vdash \uparrow t : \uparrow A$ and $\uparrow\Gamma \vdash \uparrow u : \uparrow A$. These three derivations assemble with the rule ($\to$-INTRO) in order to produce a derivation for the judgment $\uparrow\Gamma \vdash \uparrow t \underset{\uparrow A}{\longrightarrow} \uparrow u$.

Induction for terms:

– For a variable $\Gamma \vdash x : A$, we necessarily have $\Gamma \vdash$ which gives by the induction case for contexts $\uparrow\Gamma \vdash$. Moreover, the condition $(x : A) \in \Gamma$ is also necessarily satisfied, which implies that $(x : \uparrow A) \in \uparrow\Gamma$. Hence, applying the rule (VAR) yields a derivation for $\uparrow\Gamma \vdash x : \uparrow A$.

– For a term of the form $\mathsf{mop}_{\Gamma,A}[\gamma]$ (resp. $\mathsf{mcoh}_{\Gamma,A}[\gamma]$), the pair (Γ, A) declares a valid operation (resp. declares a valid coherence cut), and we necessarily have a derivation of $\Delta \vdash \gamma : \downarrow\Gamma$. By the induction case for substitutions, this gives a derivation for $\uparrow\Delta \vdash \uparrow\gamma : \uparrow\downarrow\Gamma$. Moreover, Lemma 134 ensures that we have $\uparrow\downarrow\Gamma \vdash \bullet_\Gamma : \Gamma$, hence we have a substitution $\uparrow\Delta \vdash \bullet_\Gamma \circ \uparrow\gamma : \Gamma$. By applying the rule (OP) (resp. the rule (COH)), this provides a derivation for the judgment $\uparrow\Delta \vdash \mathsf{op}_{\Gamma,A}[\bullet_\Gamma \circ \uparrow\gamma] : A[\bullet_\Gamma \circ \uparrow\gamma]$ (resp. for the judgment $\uparrow\Delta \vdash \mathsf{coh}_{\Gamma,A}[\bullet_\Gamma \circ \uparrow\gamma] : A[\bullet_\Gamma \circ \uparrow\gamma]$). We then have the following equalities, proving that the type is the reduced suspension of $(\downarrow A)[\gamma]$

$$\begin{aligned} A[\bullet_\Gamma \circ \uparrow\gamma] &= A[\bullet_\Gamma \circ \uparrow\gamma] \\ &= \uparrow\downarrow A[\uparrow\gamma] && \text{By Lemma 133} \\ &= \uparrow((\downarrow A)[\gamma]) && \text{By Lemma 131} \end{aligned}$$

Induction for substitutions:

– In the case of the empty substitution $\Delta \vdash \langle\rangle : \varnothing$, we necessarily have a derivation of $\Delta \vdash$, which by the induction case for contexts gives a derivation of $\uparrow\Delta \vdash$. Moreover, we also have by definition that $(\bullet_0 : \star) \in \uparrow\Delta$. This lets us construct a derivation for $\uparrow\Delta \vdash \uparrow\langle\rangle : \uparrow\varnothing$ as follows

$$\dfrac{\dfrac{\uparrow\Delta \vdash}{\uparrow\Delta \vdash \langle\rangle : \varnothing}(\text{ES}) \qquad \bullet_0 : \star \vdash \qquad \dfrac{\uparrow\Delta \vdash \qquad (\bullet_0 : \star) \in \uparrow\Delta}{\uparrow\Delta \vdash \bullet_0 : \star}(\text{VAR})}{\uparrow\Delta \vdash \langle \bullet_0 \mapsto \bullet_0 \rangle : (\bullet_0 : \star)}(\text{SE})$$

– In the case of a substitution of the form $\Delta \vdash \langle \gamma, x \mapsto t \rangle : (\Gamma, x : A)$, we necessarily have $\Delta \vdash \gamma : \Gamma$, which gives by the induction case for substitutions, $\uparrow\Delta \vdash \uparrow\gamma : \uparrow\Gamma$. Moreover, we have a derivation of $(\Gamma, x : A) \vdash$, which gives by the induction case for contexts a derivation of $\uparrow\Gamma, x : \uparrow A \vdash$, and a derivation of $\Delta \vdash t : A[\gamma]$, which gives by the induction case for terms a derivation of $\uparrow\Delta \vdash \uparrow t : \uparrow A[\gamma]$. Lemma 131 allows to rewrite this into a derivation for $\uparrow\Delta \vdash \uparrow t : \uparrow A[\uparrow\gamma]$. Thus these derivations can be assembled using the rule (SE) into a derivation of $\uparrow\Delta \vdash \uparrow\langle \gamma, x \mapsto t \rangle : \uparrow(\Gamma, x : A)$ as follows

$$\frac{\uparrow\Delta \vdash \uparrow\gamma : \uparrow\Gamma \qquad \uparrow\Gamma, x : \uparrow A \vdash \qquad \uparrow\Delta \vdash \uparrow t : \uparrow A[\uparrow\gamma]}{\uparrow\Delta \vdash \langle \uparrow\gamma, x \mapsto \uparrow t \rangle : (\uparrow\Gamma, x : \uparrow A)}\,(\text{SE})$$

□

Reduced suspension as a functor. By combining the syntactic aspects with the correctness properties of the reduced suspension, we can reformulate our construction in more categorical terms. First note that the reduced suspension sends well-defined contexts onto well-defined contexts, well-defined substitutions onto well-defined substitutions and preserves the composition of substitutions (as well as the identity substitutions, which we can prove immediately by induction). This exactly means that we have defined a functor $\uparrow : \mathcal{S}_{\mathsf{MCaTT}} \to \mathcal{S}_{\mathsf{CaTT}}$. But we have also proved that the reduced suspension sends valid types and terms onto valid types and terms, while respecting the action of the substitutions on those. This can be reformulated as saying that $\uparrow$ actually define a morphism $\mathcal{S}_{\mathsf{MCaTT}} \to \mathcal{S}_{\mathsf{CaTT}}$ in the slice category **Cat**/**Fam**. Moreover, this morphism by definition preserves context extensions on the nose, that is $\uparrow(\Gamma, x : A) = (\uparrow\Gamma, x : \uparrow A)$ and $\uparrow\langle \gamma, x \mapsto t \rangle = \langle \uparrow\gamma, x \mapsto \uparrow t \rangle$, so at this point it might seem like $\uparrow$ defines a morphism of categories with families. This is in fact not quite the case, since it fails to preserve the terminal object. Indeed, the terminal object in MCaTT is the empty context $\varnothing$, which is mapped by $\uparrow$ onto the context $(\bullet_0 : \star)$ which is not terminal in CaTT. This is the only obstruction for $\uparrow$ to be a morphism of categories with families, this functor preserves all the structure but the terminal object.

5.2.4 Interaction between desuspension and reduced suspension

We have defined two translations between the theories MCaTT and CaTT, that define functors between their syntactic categories. We now study how these two translation relate to each other syntactically and translate it as a categorical result about the aforementioned functors. In order to understand the interaction between the desuspension and the reduced suspension, we first show the following result, describing the action of the desuspension on the substitution $\bullet_\Gamma$.

Lemma 136. *For every context $\Gamma \vdash$ in the theory CaTT, we have $\downarrow\bullet_\Gamma = \mathrm{id}_{\downarrow\Gamma}$*

Proof. We prove this by induction on the context Γ

– For the empty context $\varnothing$, we have $\downarrow\bullet_\Gamma = \langle\rangle$ and moreover $\downarrow\varnothing = \varnothing$, hence $\mathrm{id}_{\downarrow\varnothing} = \langle\rangle$.

– For a context of the form $(\Gamma, x : \star)$, we have $\bullet_{(\Gamma,x:\star)} = \langle \bullet_\Gamma, x \mapsto \bullet_0 \rangle$,and $\downarrow\bullet_{\Gamma,x:\star} = \downarrow\bullet_\Gamma$. On the other hand, we have that $\downarrow(\Gamma, x : \star) = \downarrow\Gamma$, hence $\mathrm{id}_{\downarrow(\Gamma,x:\star)} = \mathrm{id}_{\downarrow\Gamma}$. We then conclude by induction, using the fact that $\downarrow\bullet_\Gamma = \mathrm{id}_{\downarrow\Gamma}$.

– For a context of the form $(\Gamma, x : A)$ where $A \neq \star$, we have $\bullet_{(\Gamma,x:A)} = \langle \bullet_\Gamma, x \mapsto x \rangle$, and thus $\downarrow \bullet_{(\Gamma,x:A)} = \langle \downarrow \bullet_\Gamma, x \mapsto x \rangle$. Moreover, $\downarrow(\Gamma, x : A) = (\downarrow\Gamma, x : \downarrow A)$, and thus $\mathrm{id}_{\downarrow(\Gamma,x:A)} = \langle \mathrm{id}_\Gamma, x \mapsto x \rangle$. We again conclude by induction using the fact that $\downarrow \bullet_\Gamma = \mathrm{id}_{\downarrow\Gamma}$.

□

Desuspension of the reduced suspension. We have the following result

Proposition 137. *The equality between functors* $\downarrow \circ \uparrow = \mathrm{id}_{\mathcal{S}_{\mathsf{MCaTT}}}$ *holds.*

Proof. We show by mutual induction the following properties of the theory MCaTT

– For all context $\Gamma \vdash$, $\downarrow\uparrow\Gamma = \Gamma$

– For all type $\Gamma \vdash A$, $\downarrow\uparrow A = A$

– For all term $\Gamma \vdash t : A$, $\downarrow\uparrow t = t$

– For all substitution $\Delta \vdash \gamma : \Gamma$, $\downarrow\uparrow\gamma = \gamma$

Induction for contexts:

– For the empty context $\varnothing$, we have $\uparrow\varnothing = (\bullet_0 : \star)$, and thus $\downarrow\uparrow\varnothing = \varnothing$.

– For a context of the form (Γ, A), we have $\uparrow(\Gamma, x : A) = (\uparrow\Gamma, x : \uparrow A)$. Since $\uparrow A$ is necessarily distinct from $\star$, we have $\downarrow\uparrow(\Gamma, x : A) = (\downarrow\uparrow\Gamma, x : \downarrow\uparrow A)$, and the induction case for context shows that $\downarrow\uparrow\Gamma = \Gamma$ and the induction case for types shows $\downarrow\uparrow A = A$. This proves that $\downarrow\uparrow(\Gamma, x : A) = (\Gamma, x : A)$.

Induction for types:

– For the type $\star$, we have $\uparrow\star = \bullet_0 \underset{\star}{\to} \bullet_0$, and thus $\downarrow\uparrow\star = \star$.

– For a type of the form $t \underset{A}{\to} u$, we have $\uparrow t \underset{A}{\to} u = \uparrow t \underset{\uparrow A}{\longrightarrow} \uparrow u$, and since $\uparrow A$ is distinct from $\star$, this implies that $\downarrow\uparrow t \underset{A}{\to} u = \downarrow\uparrow t \underset{\downarrow\uparrow A}{\longrightarrow} \downarrow\uparrow u$. The induction case for types then shows that $\downarrow\uparrow A = A$, and the induction case for terms shows $\downarrow\uparrow t = t$ and $\downarrow\uparrow u = u$, which lets us conclude.

Induction for terms:

– For a variable x, we have $\uparrow x = x$ and thus $\downarrow\uparrow x = x$.

– For a term of the form $\mathsf{mop}_{\Gamma,A}[\gamma]$, we have $\uparrow \mathsf{mop}_{\Gamma,A}[\gamma] = \mathsf{op}_{\Gamma,A}[\bullet_\Gamma \circ \uparrow\gamma]$, and thus we have the following equalities

$$\begin{aligned}
\downarrow\uparrow \mathsf{mop}_{\Gamma,A}[\gamma] &= \mathsf{mop}_{\Gamma,A}[\downarrow(\bullet_\Gamma \circ \uparrow\gamma)] \\
&= \mathsf{mop}_{\Gamma,A}[\downarrow \bullet_\Gamma \circ \downarrow\uparrow\gamma] && \text{by Lemma 128} \\
&= \mathsf{mop}_{\Gamma,A}[\downarrow\uparrow\gamma] && \text{by Lemma 136} \\
&= \mathsf{mop}_{\Gamma,A}[\gamma] && \text{by the induction case for substitution}
\end{aligned}$$

– The case for a term of the form $\mathsf{mop}_{\Gamma,A}[\gamma]$ follows the exact same steps.

Induction for substitutions:

– For the empty substitution $\langle\rangle$, we have $\uparrow\langle\rangle = \langle \bullet_0 \mapsto \bullet_0 \rangle$, and thus $\downarrow\uparrow\langle\rangle = \langle\rangle$.

– For a substitution of the form $\langle \gamma, x \mapsto t \rangle$, we have $\uparrow\langle \gamma, x \mapsto t\rangle = \langle \uparrow\gamma, x \mapsto \uparrow t\rangle$. Necessarily $\uparrow t$ is a well defined term of dimension non-zero, hence $\downarrow\uparrow\langle \gamma, x \mapsto t\rangle = \langle \downarrow\uparrow\gamma, x \mapsto \downarrow\uparrow t\rangle$. Then the induction case for substitutions gives the equality $\downarrow\uparrow\gamma = \gamma$ and the induction case for terms gives $\downarrow\uparrow t = t$, which lets us conclude.

□

A natural transformation. In order to study the reduced suspension, we have introduced the family of substitutions $\bullet_\Gamma$ for all contexts Γ, and we have proved in Lemma 134, that it defines a family of morphisms $\bullet_\Gamma : \uparrow\downarrow\Gamma \to \Gamma$. Moreover, the equality $\gamma \circ \bullet_\Delta = \bullet_\Gamma \circ \uparrow\downarrow\gamma$ that we proved for all substitution γ in Lemma 133 can be expressed by the commutation of the following diagram

$$\begin{array}{ccc} \uparrow\downarrow\Delta & \xrightarrow{\bullet_\Delta} & \Delta \\ {\scriptstyle \uparrow\downarrow\gamma}\downarrow & & \downarrow{\scriptstyle \gamma} \\ \uparrow\downarrow\Gamma & \xrightarrow[\bullet_\Gamma]{} & \Gamma \end{array}$$

So we have in fact already proven the following proposition, which is simply a categorical reformulation of our previous fact

Proposition 138. *The family of morphisms $\bullet_\Gamma : \uparrow\downarrow\Gamma \to \Gamma$ define a natural transformation $\uparrow \circ \downarrow \Rightarrow \mathrm{id}_{\mathcal{S}_{\mathsf{CaTT}}}$.*

Adjunction between desuspension and reduced suspension. Combining these two previous theorems, we have in fact proved the following result, which is very important categorically

Theorem 139. *The functor $\uparrow$ is left adjoint to $\downarrow$, the counit is given by the family $\bullet_\Gamma$.*

This adjunction is to be understood as an analogue in the world of categories of the topological adjunction between the reduced suspension and the loop space. In our terminology, the desuspension corresponds to the loop space, and the reduced suspension corresponds to the reduced suspension. This is also analogous to the relation between the loop space and the reduced suspension in topology.

Remark 140. In fact we even have a more precise result: Since the unit of the adjunction is the identity, it exhibits $\mathcal{S}_{\mathsf{MCaTT}}$ as a coreflective subcategory of $\mathcal{S}_{\mathsf{CaTT}}$. Moreover the essential image of $\uparrow$ is exactly the category $\mathcal{S}_{\mathsf{CaTT},\bullet}$. So a way to understand our type theory MCaTT is that it achieves a structure of category with families on $\mathcal{S}_{\mathsf{CaTT},\bullet}$, which coincide with the structure on $\mathcal{S}_{\mathsf{CaTT}}$ for all the operation that are defined in both structures.

5.2.5 Models of the type theory MCaTT.

The definition of the desuspension and reduced suspension operations that we have defined relate very tightly the models of the theory MCaTT and the models of the theory CaTT.

Reduced suspension acting on the models of CaTT. Given a model $F : \mathsf{CaTT} \to \mathbf{Set}$, we define a functor $\uparrow^* F : \mathsf{MCaTT} \to \mathbf{Set}$ by precomposing with the functor $\uparrow$, hence we have $\uparrow^* F(\Gamma) = F(\uparrow \Gamma)$ and $\uparrow^* F(\gamma) = F(\uparrow \gamma)$. Since both $\uparrow$ and F preserve the context comprehension on the nose, so does $\uparrow^* F$, so $\uparrow^* F$ is a model of $\mathcal{S}_{\mathsf{MCaTT}}$ if and only if it preserves the terminal object. But $\uparrow^* F(\varnothing) = F(D^0)$, hence $\uparrow^* F$ is a model of $\mathcal{S}_{\mathsf{MCaTT}}$ if and only if $F(D^0) = \{\bullet\}$, that is if $F \in \mathbf{Mod}_\bullet(\mathcal{S}_{\mathsf{CaTT}})$. This shows that the reduced suspension translation from MCaTT to CaTT induces a functor $\uparrow^* : \mathbf{Mod}_\bullet(\mathcal{S}_{\mathsf{CaTT}}) \to \mathbf{Mod}(\mathcal{S}_{\mathsf{MCaTT}})$. Importantly, the reduced suspension induces on the models an operation akin to the loop category.

Desuspension on the models of MCaTT. Similarly, for a model $(F, \phi) : \mathsf{MCaTT} \to \mathbf{Set}$, and define the functor $\downarrow^* F : \mathsf{CaTT} \to \mathbf{Set}$ by precomposing with the functor $\downarrow$. Note that we necessarily have

$$\begin{aligned} \downarrow^* F(D^0) &= F(\downarrow D^0) \\ &= F(\varnothing) \\ &= \{\bullet\} \end{aligned}$$

so if $\downarrow^* F$ can be completed into a model of CaTT, this model is in $\mathbf{Mod}_\bullet(\mathcal{S}_{\mathsf{CaTT}})$. Consider a pullback along a generating display map in the category $\mathcal{S}_{\mathsf{CaTT}}$, it is then of the form

$$\begin{array}{ccc} (\Gamma, x : A) & \longrightarrow & D^n \\ \downarrow & \lrcorner & \downarrow \\ \Gamma & \xrightarrow[\chi_A]{} & S^{n-1} \end{array}$$

If $n > 0$, then we have

$$\begin{aligned} \downarrow^* F(\Gamma, x : A) &= F(\downarrow(\Gamma, x : A)) \\ &= F(\downarrow\Gamma, x : \downarrow A) \\ &= (F(\downarrow\Gamma), x : \phi_\Gamma(\downarrow A)) \end{aligned}$$

$\downarrow^* F$ of the previous pullback square is the following square, that is again a pullback

$$\begin{array}{ccc} (\downarrow^* F(\Gamma), x : \phi_\Gamma(\downarrow A)) & \longrightarrow & \downarrow^* F(D^n) \\ \downarrow & \lrcorner & \downarrow \\ \downarrow^* F(\Gamma) & \xrightarrow[\downarrow^* F(\chi_A)]{} & \downarrow^* F(S^{n-1}) \end{array}$$

In the case of $n = 0$, the image of the previous square is the following square, that is again a pullback

$$\begin{array}{ccc} \downarrow^* F(\Gamma) & \longrightarrow & \{\bullet\} \\ \downarrow & \lrcorner & \downarrow \\ \downarrow^* F(\Gamma) & \longrightarrow & \{\bullet\} \end{array}$$

This shows that $\downarrow^* F$ preserves all pullback along display maps, and thus it defines a unique model of CaTT. Note however that by composing the $\downarrow$ and ϕ, we get a natural transformation

$\downarrow^*\phi$ associating to each type $\Gamma \vdash A$ in CaTT an element of $\mathrm{Ty}^{\downarrow^* F(\Gamma)}$, and to each term $\Gamma \vdash t : A$, an element of $\mathrm{Tm}^{\downarrow^* F\Gamma}_{\downarrow^*\phi_\Gamma(A)}$. This natural transformation is not the one that makes $\downarrow^* F$ into a model, but is isomorphic to it. Indeed, equipped with this natural transformation, the preservation of context comprehension may not be strict for contexts of the form $(\Gamma, x : \star)$. However, since it makes the context comprehension strict for all other contexts, $\downarrow^*\phi$ coincide with the transformation that makes $\downarrow^* F$ into a model on all types distinct from $\star$ and all terms of type distinct from $\star$. For the sake of simplicity, we just denote $\downarrow^* F$ the induced model of CaTT that is necessarily in $\mathbf{Mod}_\bullet(\mathcal{S}_{\mathsf{CaTT}})$. This then defines a functor $\downarrow^* : \mathcal{S}_{\mathsf{MCaTT}} \to \mathbf{Mod}_\bullet(\mathcal{S}_{\mathsf{CaTT}})$. The desuspension induces on the models an operation similar to the looping.

Models of the category $\mathcal{S}_{\mathsf{MCaTT}}$. We have now defined a pair of functors $\downarrow^*$ and $\uparrow^*$ between the categories $\mathbf{Mod}(\mathcal{S}_{\mathsf{MCaTT}})$ and $\mathbf{Mod}_\bullet(\mathcal{S}_{\mathsf{CaTT}})$.

Theorem 141. *The functors $\downarrow^*$ and $\uparrow^*$ define an equivalence of categories between $\mathbf{Mod}(\mathcal{S}_{MCaTT})$ and $\mathbf{Mod}_\bullet(\mathcal{S}_{CaTT})$*

Proof. First, Proposition 137 implies that

$$\uparrow^* \circ \downarrow^* = \left(\downarrow \circ \uparrow\right)^* = \mathrm{id}_{\mathcal{S}_{\mathbf{MCaTT}}}$$

and proposition 138 shows that there is a natural transformation, obtained by whiskering

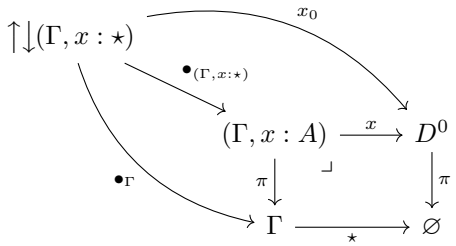
$$\downarrow^* \circ \uparrow^* = \left(\uparrow \circ \downarrow\right)^* \Rightarrow \mathrm{id}_{\mathbf{Mod}_\bullet(\mathcal{S}_{\mathbf{CaTT}})}$$

So it suffices to show that this natural transformation is a natural isomorphism, that is for any $F \in \mathbf{Mod}_\bullet(\mathcal{S}_{\mathsf{CaTT}})$ and all context Γ in CaTT

$$F(\bullet_\Gamma) : F(\uparrow\downarrow\Gamma) \to F(\Gamma)$$

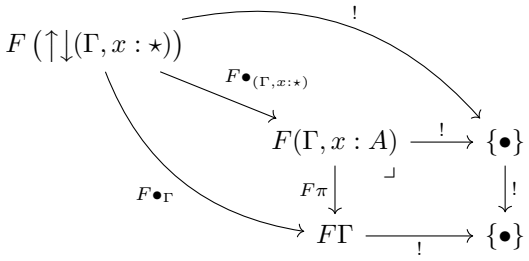
is an isomorphism. We prove this property by induction on the context Γ.

- For the empty context $\varnothing$, we necessarily have that $F(\varnothing) = \{\bullet\}$, and since $\uparrow\downarrow\varnothing = D^0$ and $F \in \mathbf{Mod}_\bullet(\mathcal{S}_{\mathsf{CaTT}})$, we also have that $F(\uparrow\downarrow\varnothing) = \{\bullet\}$. Hence $F(\bullet_\varnothing)$ is the unique map between the singleton to itself, which is an isomorphism.
- For a context of the form $(\Gamma, x : \star)$, it is obtained as a (trivial) pullback, and the map $\bullet_{(\Gamma,x:\star)}$ is obtained by universal property of the pullback, as follows



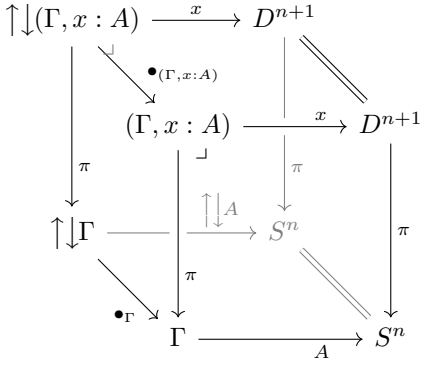
Taking the image by F on this pullback yields another pullback in **Set** (since F is a model

of CaTT), as follows

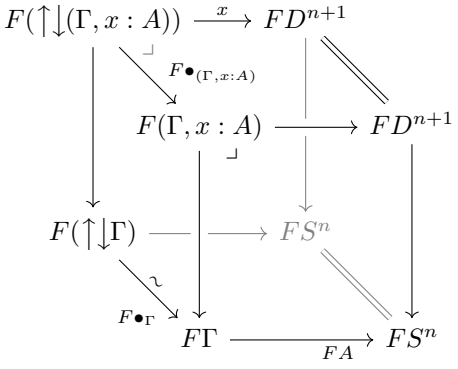


Since the square is a pullback, $F\pi$ is an isomorphism, and since by induction $F\bullet_\Gamma$ is also an isomorphism, necessarily $F\bullet_{\Gamma,x:\star}$ is an isomorphism.

– For a context of the form $(\Gamma, x : A)$ where A is a type distinct from $\star$, the context is obtained as a pullback, the context $\uparrow\downarrow(\Gamma, x : A)$ is also a pullback and the substitution $\bullet_{(\Gamma,x:A)}$ is obtained by universal property as described in the following diagram which has the shape of a cube whose faces are commutative and whose front and back face are pullbacks



Taking the image by F of this diagram yields another cube whose faces are all commutative square, and whose front and back square are again pullback squares, since as a model, F preserves the pullbacks along the display maps (in the following figure, we have left implicit most of the arrows, they are simply the image by F of the ones of the previous figure).



By induction, the map $F(\bullet_\Gamma)$ is an isomorphism, making the span defining $F(\Gamma, x : A)$ and the span defining $F(\uparrow\downarrow(\Gamma, x : A))$ isomorphic. This proves that the map $F(\bullet_{(\Gamma,x:A)})$ is also an isomorphism, by uniqueness of the pullback up to isomorphism.

□

A correspondence for free. Note that our general framework for globular theories shows that the models of MCaTT are equivalent to the presheaves on Θ_{mon} preserving the sums and the globular sums, and our syntactic translation using the desuspension and reduced suspension show that the models of MCaTT are equivalent to $\mathbf{Mod}_{\bullet}(\mathcal{S}_{\mathsf{CaTT}})$. Composing these two equivalence yields the following correspondence

Theorem 142. *The category of sum and globular sum preserving presheaves on Θ_{mon} is equivalent to the category of globular sum preserving presheaves on Θ_{∞} which send $\mathrm{Y}(0)$ onto the terminal object.*

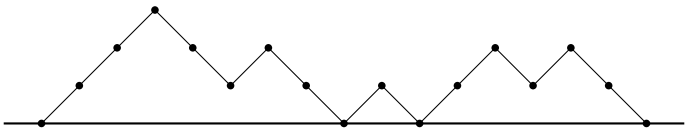
5.2.6 Interpretation

The way we have stated the equivalence may seem a bit surprising, to the reader familiar with homotopy type theory, since having a single object is not a property that is expected to by invariant by any good notion of equivalence of weak ω-categories. However, we expect the stated equivalence to lift to an equivalence between the monoidal weak ω-categories and the weak ω-categories that are weakly equivalent to a weak ω-category with only one object, when considering all these up to weak equivalence, in their appropriate higher structure. Our intuition is that our speculative notion of weak equivalence should let us characterize the weak ω-categories that are weakly equivalent to a weak ω-categories with only one object as the weak ω-categories whose groupoidal core is 0-connected.

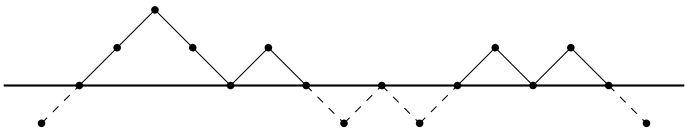
5.3 An alternative presentation of MCaTT

We have introduced the type theory MCaTT in the framework of a globular type theory. However, relies on types in the theory CaTT together with the looping operation. We present here a more standalone version, in which we give a direct combinatorial description of the introduction rules for the terms. In order to achieve this, we provide a calculus for the contexts of the form $\downarrow\Gamma$, with Γ a ps-context, together with an appropriate notion of types, terms and substitutions.

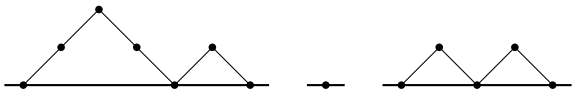
Intuitive description. In order to describe this new kind of contexts, we use the structure of a *list of contexts*. Intuitively, starting with a ps-context such as the one represented on the following picture



We compute its looping, as follows



and remove the (-1)-cells in order to finally obtain the following list of ps-contexts



This operation is closely related to computing $\downarrow\Gamma$, the difference is that it does not produce a single context, but a list of them, which we think of as encoding a product. This lets us axiomatize directly which are the lists of contexts that are obtained in this way.

5.3.1 A framework for type theories with local exchange

Our aim is to present a type theory to compute the coherator for monoidal weak ω-categories. Contrary to the coherator for weak ω-categories, this coherator has objects which admit automorphisms, such as the globular set composed by only two 0-cells, as in the following diagram

As a consequence, we need to change the structural rules for this type theory to eliminate these automorphisms.

Lists of contexts. Our new structural rules for this type theory require us to change the syntax and define new notions of contexts and substitutions, supported respectively by lists of contexts and lists of substitutions. As we only use this syntax to describe the coherator for monoidal categories, we call these new contexts and substitution respectively *monoidal contexts* and *monoidal substitutions*. In the following, we always denote with an over lined character $\overline{a}$ the lists and [] the empty list. We also respect our previous conventions, so that $\overline{\Gamma}$ denotes a list of contexts, and $\overline{\gamma}$ denotes a list of substitutions. We denote $[\overline{a}; b]$ for the list whose tail is $\overline{a}$ and whose head is b. Moreover, given two lists $\overline{a}$ and $\overline{b}$, we denote $\overline{a}@\overline{b}$ the concatenation of these two lists, and given a list $\overline{a}$, we denote $\ell(\overline{a})$ its length. Finally, given a non-empty list of contexts $\overline{\Gamma}$, we denote $(\overline{\Gamma}, x : A)$ the list of context whose tail is the tail of $\overline{\Gamma}$, and whose head is $\Gamma, x : A$ where Γ is the head of $\overline{\Gamma}$, and similarly, given a non empty list of substitutions $\overline{\gamma}$, we denote $\langle\overline{\gamma}, x \mapsto t\rangle$ the substitution whose tail is the tail of $\overline{\gamma}$ and whose head is $\langle\gamma, x \mapsto t\rangle$, where γ is the head of $\overline{\gamma}$.

Monoidal judgments. The type theory that we present has the regular notion of contexts, types, terms and substitutions, to which we add an additional notion of contexts, types, terms and substitutions. To distinguish the two kinds, we call the new ones with the adjective "monoidal" as we use them to describe monoidal categories. Monoidal contexts are supported by lists of regular contexts, and monoidal substitutions are supported by lists of regular substitutions, whereas monoidal types and terms are supported by the same syntactic expression as regular types and terms. We introduce four new kinds of judgments, that we call *monoidal judgments* and are analogous to the regular judgment, but for the monoidal part of the theory

$\overline{\Gamma} \vdash$	$\overline{\Gamma}$ is a valid monoidal context
$\overline{\Gamma} \vdash A$	A is a valid monoidal type in $\overline{\Gamma}$
$\overline{\Gamma} \vdash t : A$	t is a valid monoidal term of type A in $\overline{\Gamma}$
$\overline{\Delta} \vdash \overline{\gamma} : \overline{\Gamma}$	$\overline{\gamma}$ is a valid monoidal substitution from $\overline{\Delta}$ to $\overline{\Gamma}$

We use similar notations for these judgments and for regular judgments, but they are distinct judgments, we rely in the notation in the fact that the we can disambiguate by looking at the left-hand side: a regular context indicates a regular judgment while a monoidal context indicates a

monoidal judgment. We also introduce two kinds of substitutions that connect the monoidal part of the theory to the regular part, one of them is supported by regular substitution expressions, and the other one by a list of regular substitution expressions. We again give the associated judgments for these new kinds of substitutions

$$\begin{array}{ll} \overline{\Delta} \vdash \gamma : \Gamma & \gamma \text{ is a valid substitution from } \overline{\Delta} \text{ to } \Gamma \\ \Delta \vdash \overline{\gamma} : \overline{\Gamma} & \overline{\gamma} \text{ is a valid substitution from } \Delta \text{ to } \overline{\Gamma} \end{array}$$

Concatenation and action of substitutions. We define the operation of *concatenation* and that we denote Π as a translation from the monoidal part of the theory to the regular part of the theory: It consists in, given a list of contexts (resp. a list of substitutions), concatenate all of its components to produce a single context (resp. a single substitution), and can be defined inductively as follows

$$\begin{array}{lll} \Pi[] = \varnothing & \Pi[\overline{\Gamma}; \varnothing] = \Pi\overline{\Gamma} & \Pi[\overline{\Gamma}; (\Gamma, x : A)] = (\Pi[\overline{\Gamma}; \Gamma], x : A) \\ \Pi[] = \langle\rangle & \Pi[\overline{\gamma}; \langle\rangle] = \Pi\overline{\gamma} & \Pi[\overline{\gamma}; \langle\gamma, x \mapsto t\rangle] = \langle\Pi[\overline{\gamma}; \gamma], x \mapsto tx\rangle \end{array}$$

Using this operation, we can define an action of the monoidal substitution on the types, terms and regular substitutions by reducing it to the action of a regular substitutions

$$A[\overline{\gamma}] = A[\Pi\overline{\gamma}] \qquad t[\overline{\gamma}] = t[\Pi\overline{\gamma}] \qquad \delta \circ \overline{\gamma} = \delta \circ \Pi\overline{\gamma}$$

We also define the notation $(x : A) \in \overline{\Gamma}$ as a shorthand for $(x : A) \in \Pi\overline{\Gamma}$, and $(x \mapsto t) \in \overline{\gamma}$ as a shorthand for $(x \mapsto t) \in \Pi\overline{\gamma}$

Structural rules. To define our type theory, we have added an extra part on top of our theory, that we call the monoidal part. This part is subject to structural rules similar to the structural rules for the regular judgments. Since we are only interested in a type theory describing globular sets with extra structure, we present here the type introduction rules together with the structural rules. Our theory MCaTT satisfies, in addition to the structural rules of a cut-free type theory

for the regular judgments, the following rules:

For monoidal judgments:

$$\frac{}{[] \vdash}(\text{MEC}) \qquad \frac{\overline{\Gamma} \vdash}{[\overline{\Gamma}; \varnothing] \vdash}(\text{MC}+)$$

$$\frac{[\overline{\Gamma}; \Gamma] \vdash \qquad [\overline{\Gamma}; \Gamma] \vdash A}{[\overline{\Gamma}, (\Gamma, x : A)] \vdash}(\text{MCE})$$

where in the rule (MC+) we assume that $x \notin \mathrm{Var}([\overline{\Gamma}; \Gamma])$

$$\frac{\overline{\Gamma} \vdash}{\overline{\Gamma} \vdash \star} \qquad \frac{\overline{\Gamma} \vdash A \qquad \overline{\Gamma} \vdash t : A \qquad \overline{\Gamma} \vdash u : A}{\overline{\Gamma} \vdash t \underset{A}{\rightarrow} u}$$

$$\frac{\overline{\Gamma} \vdash \qquad (x : A) \in \overline{\Gamma}}{\overline{\Gamma} \vdash x : A}$$

$$\frac{}{[] \vdash [] : []}(\text{EMS}) \qquad \frac{\overline{\Delta} \vdash \overline{\gamma} : \overline{\Gamma} \qquad \overline{\Delta}' \vdash}{\overline{\Delta} @ \overline{\Delta}' \vdash [\gamma ;_{\overline{\Delta}} \langle\rangle] : [\overline{\Gamma}; \varnothing]}(\text{MS}+)$$

$$\frac{\overline{\Delta} @ \overline{\Delta}' \vdash [\overline{\gamma} ;_{\overline{\Delta}} \gamma] : [\overline{\Gamma}; \Gamma] \qquad [\overline{\Gamma}'; (\Gamma, x : A)] \vdash \qquad \overline{\Delta}' \vdash t : A[\overline{\gamma} ;_{\overline{\Delta}} \gamma]}{\overline{\Delta} @ \overline{\Delta}' \vdash [\gamma' ;_{\overline{\Delta}} \langle \gamma, x \mapsto t \rangle] : [\overline{\Gamma}; (\Gamma, x : A)]}(\text{MSE})$$

For substitutions between monoidal and regular contexts:

$$\frac{\Delta \vdash}{\Delta \vdash [] : []} \qquad \frac{\Delta \vdash \overline{\gamma} : \overline{\Gamma} \qquad \Delta \vdash \gamma : \Gamma}{\Delta \vdash [\overline{\gamma}; \gamma] : [\overline{\Gamma}; \Gamma]}$$

$$\frac{\overline{\Delta} \vdash}{\overline{\Delta} \vdash \langle\rangle : \varnothing} \qquad \frac{\overline{\Delta} \vdash \gamma : \Gamma \qquad \Gamma, x : A \vdash \qquad \overline{\Delta} \vdash t : A[\gamma]}{\overline{\Delta} \vdash \langle \gamma, x \mapsto t \rangle : (\Gamma, x : A)}$$

Assumptions on term constructors. In this presentation, we have not given the introduction rules for term constructors, as these rules are not easy to state and we need some preparatory work. However, the properties that we are going to study are independent of the term constructors, and can thus be defined independently. However, we still require for the theory to be well-behaved that the term constructors satisfy some properties, in particular, they must make the following rule admissible

$$\frac{\Delta \vdash \overline{\gamma} : \overline{\Gamma} \qquad \overline{\Gamma} \vdash t : A}{\Delta \vdash t[\overline{\gamma}] : A[\overline{\gamma}]}$$

We also require that the term constructors are such that for all term t, we have $t[\mathrm{id}_{\overline{\Gamma}}] = t$, where $\mathrm{id}_{\overline{\Gamma}}$ is the monoidal substitution defined inductively as follows

$$\mathrm{id}_{[]} = [] \qquad \mathrm{id}_{[\overline{\Gamma}; \Gamma]} = [\mathrm{id}_{\overline{\Gamma}}; \mathrm{id}_{\Gamma}]$$

This condition implies that we also have for any type A the equality $A[\mathrm{id}_{\overline{\Gamma}}] = A$.

Interpretation. We interpret the monoidal contexts along with their structural rules, as usual contexts with extra structure added. A way to understand them is to picture them as contexts that have two extension operation. The usual extension, that we still denote $(\Gamma, x : A)$ still behaves like the context extension, but we have added another extension that comes from the "cons" operation of the lists. For instance $[\overline{\Gamma}; (x : \star)]$ defines a monoidal contexts, that is the monoidal context $\overline{\Gamma}$, extended by the variable x of type $\star$ with the second extension operation.

This extension has different properties, and in particular the order fundamentally matters, even for non-dependent types. For instance there is no non-trivial isomorphism of the monoidal context $[(x : \star); (y : \star)]$, even though there is a non-trivial isomorphism of the context $(x : \star, y : \star)$ obtained by taking the substitution that exchanges x and y as follows

$$(x : \star, y : \star) \vdash \langle x \mapsto y, y \mapsto x \rangle : (x : \star, y : \star)$$

Uniqueness of derivations. In this theory, every derivable judgment is derivable in a unique way. Indeed, every rule corresponds to exactly one syntactic entity. There is however a rule that could be applied in different ways: The rule (MSE). Indeed given a list, it might decompose in many ways into a concatenation of lists, and without further assumptions, several decompositions may satisfy the premises of the rule. This is why we annotate the lists with the contexts in the case of substitutions.

Categorical structure. Like for usual type theories, the contexts, together with the substitutions form a category, for an appropriate notion of composition of substitutions, and identity substitution. More precisely, we have introduced a formula to compose regular substitutions expressions, and proved that the second of the two following rules is admissible. We admit that the second one is also admissible.

$$\frac{\overline{\Delta} \vdash \gamma : \Gamma \qquad \Gamma \vdash \xi : \Xi}{\overline{\Delta} \vdash \xi \circ \gamma : \Xi} \qquad \frac{\Delta \vdash \gamma : \Gamma \qquad \Gamma \vdash \xi : \Xi}{\Delta \vdash \xi \circ \gamma : \Xi}$$

This composition lets us also define the composition of a regular substitution expression with a monoidal substitution expression on the left as follows

$$[] \circ \gamma = [] \qquad [\overline{\delta}, \delta] \circ \gamma = [\overline{\delta} \circ \gamma; \delta \circ \gamma]$$

and we admit that this definition makes the following rules admissible, as is the case in a regular type theory

$$\frac{\overline{\Delta} \vdash \gamma : \Gamma \qquad \Gamma \vdash \overline{\xi} : \overline{\Xi}}{\overline{\Delta} \vdash \overline{\xi} \circ \gamma : \overline{\Xi}} \qquad \frac{\Delta \vdash \gamma : \Gamma \qquad \Gamma \vdash \overline{\xi} : \overline{\Xi}}{\Delta \vdash \overline{\xi} \circ \gamma : \overline{\Xi}}$$

In this theory, monoidal substitutions also act on term expressions, and thus we can define a composition of a monoidal substitution expression with a regular substitution expression expression on the left as follows

$$\langle\rangle \circ \overline{\gamma} = \langle\rangle \qquad \langle \delta, x \mapsto t \rangle \circ \overline{\gamma} = \langle \delta \circ \overline{\gamma}, x \mapsto t[\overline{\gamma}] \rangle$$

This composition yields well-defined expressions and we admit that the following rules are admissible

$$\frac{\overline{\Delta} \vdash \overline{\gamma} : \overline{\Gamma} \qquad \overline{\Gamma} \vdash \xi : \Xi}{\overline{\Delta} \vdash \xi \circ \overline{\gamma} : \Xi} \qquad \frac{\Delta \vdash \overline{\gamma} : \overline{\Gamma} \qquad \overline{\Gamma} \vdash \xi : \Xi}{\Delta \vdash \xi \circ \overline{\gamma} : \Xi}$$

Moreover, this composition also lets us define the composition of two monoidal substitutions together, with the following formula

$$[] \circ \overline{\gamma} = [] \qquad [\overline{\delta}; \delta] \circ \overline{\gamma} = [\overline{\delta} \circ \overline{\gamma}; \delta \circ \overline{\gamma}]$$

Similarly, we admit that these definitions make the following rules admissible

$$\frac{\overline{\Delta} \vdash \overline{\gamma} : \overline{\Gamma} \qquad \overline{\Gamma} \vdash \overline{\xi} : \overline{\Xi}}{\overline{\Delta} \vdash \overline{\xi} \circ \overline{\gamma} : \overline{\Xi}} \qquad \frac{\Delta \vdash \overline{\gamma} : \overline{\Gamma} \qquad \overline{\Gamma} \vdash \overline{\xi} : \overline{\Xi}}{\Delta \vdash \overline{\xi} \circ \overline{\gamma} : \overline{\Xi}}$$

Importantly, we have not given the index for a composition of two monoidal substitutions. In fact, we can only state for now that there exists a correct indexing, without making it explicit. However, we are introducing this theory to study monoidal ps-contexts that we introduce later on and, which have only one possible indexing, so this definition becomes unambiguous.

Correctness of the concatenation. We have introduced the concatenation as an operation on the syntax of the theory, we now show that it respects the derivability of the judgments.

Lemma 143. *In the theory that we present the following results hold*

- *For any monoidal context* $\overline{\Gamma} \vdash$*, the judgment* $\Pi\overline{\Gamma} \vdash$ *is also derivable.*
- *For any monoidal type* $\overline{\Gamma} \vdash A$*, the judgment* $\Pi\overline{\Gamma} \vdash A$ *is derivable.*
- *For any monoidal term* $\overline{\Gamma} \vdash t : A$*, the judgment* $\Pi\overline{\Gamma} \vdash t : A$ *is derivable.*
- *For any monoidal substitution* $\overline{\Delta} \vdash \overline{\gamma} : \overline{\Gamma}$*, the judgment* $\Pi\overline{\Delta} \vdash \Pi\overline{\gamma} : \Pi\overline{\Gamma}$ *is also derivable.*
- *For any monoidal context* $\overline{\Gamma} \vdash$*, the substitution* $\Pi\overline{\Gamma} \vdash \mathrm{id}_{\overline{\Gamma}} : \overline{\Gamma}$ *is a derivable substitution from a regular context to a monoidal context.*

Proof. We prove these results by mutual induction

Induction for monoidal contexts:

- For the monoidal context $[]$, we have by definition $\Pi[] = \varnothing$, and the rule (EC) gives a derivation of $\varnothing \vdash$.
- For a monoidal context of the form $[\overline{\Gamma}; \varnothing]$, we have $\Pi[\overline{\Gamma}; \varnothing] = \Pi\overline{\Gamma}$, and the induction case for context gives a derivation of $\Pi\overline{\Gamma} \vdash$.
- For a monoidal context of the form $[\overline{\Gamma}; (\Gamma, x : A)]$, by the induction case for context we have a derivation of $\Pi[\overline{\Gamma}; \Gamma]$. Moreover, we necessarily have a derivation of $\Gamma \vdash A$, hence this provides a derivation of $[\overline{\Gamma}; \Gamma] \vdash A$, and by the induction case for types, this proves that $\Pi[\overline{\Gamma}; \Gamma] \vdash A$. Applying the rule (CE) then gives a derivation of $(\Pi[\overline{\Gamma}; \Gamma], x : A) \vdash$.

Induction for monoidal types:

- For the type $\overline{\Gamma} \vdash \star$, we necessarily have $\overline{\Gamma} \vdash$, which by the induction case for contexts gives a derivation of $\Pi\overline{\Gamma} \vdash$. Applying the rule ($\star$-INTRO) then provides a derivation of $\Pi\overline{\Gamma} \vdash \star$.
- For the type $\overline{\Gamma} \vdash t \underset{A}{\longrightarrow} u$, we necessarily have a derivation of $\overline{\Gamma} \vdash A$, which by the induction case for types provides a derivation of $\Pi\overline{\Gamma} \vdash A$. Moreover, we also have a derivation of $\overline{\Gamma} \vdash t : A$ and of $\overline{\Gamma} \vdash u : A$, which by the induction case for terms provides a derivation for $\Pi\overline{\Gamma} \vdash t : A$ and $\Pi\overline{\Gamma} \vdash u : A$. By applying the rule ($\to$-INTRO), we then get a derivation of $\Pi\overline{\Gamma} \vdash t \underset{A}{\longrightarrow} u$.

Induction for monoidal terms:

- For a variable term $\overline{\Gamma} \vdash x : A$, we necessarily have $\overline{\Gamma} \vdash$, which by the induction case for contexts shows that $\Pi\overline{\Gamma} \vdash x : A$. Moreover, we also have $(x : A) \in \overline{\Gamma}$, which implies $(x : A) \in \Pi\overline{\Gamma}$, hence by the rule (VAR), we have a derivation of $\Pi\overline{\Gamma} \vdash x : A$.
- For a generic term $\overline{\Gamma} \vdash t : A$, by induction, we have a derivation of $\Pi\overline{\Gamma} \vdash \mathrm{id}_{\overline{\Gamma}} : \overline{\Gamma}$, and our assumptions on terms constructors certifies that we have $\Pi\overline{\Gamma} \vdash t[\mathrm{id}_{\overline{\Gamma}}] : A[\mathrm{id}_{\overline{\Gamma}}]$, hence $\Pi\overline{\Gamma} \vdash t : A$

Induction case for monoidal substitutions:

- For the monoidal substitution $\overline{\Delta} \vdash [] : []$, we necessarily have a derivation of $\overline{\Delta} \vdash$, which by the induction case for contexts gives a derivation of $\Pi\overline{\Delta} \vdash$. Applying the rule (ES) then gives a derivation of $\Pi\overline{\Delta} \vdash \langle\rangle : \varnothing$.
- For a monoidal substitution of the form $\overline{\Delta} \vdash [\overline{\gamma}; \langle\rangle] : [\overline{\Gamma}; \varnothing]$, the induction case for substitution gives a derivation of $\overline{\Delta} \vdash \overline{\gamma} : \overline{\Gamma}$.
- For a monoidal substitution of the form $\overline{\Delta} \vdash [\overline{\gamma}; \langle\gamma, x \mapsto t\rangle] : [\overline{\Gamma}; (\Gamma, x : A)]$, we necessarily have $\overline{\Delta} \vdash t : A$, hence by the induction case for terms, this gives a derivation of $\Pi\overline{\Delta} \vdash t : A$. Moreover, we have a derivation of $\overline{\Delta} \vdash \overline{\gamma} : \overline{\Gamma}$, and the induction case for substitution provides a derivation of $\Pi\overline{\Delta} \vdash \Pi\overline{\gamma} : \Pi\overline{\Gamma}$. The rule (SE) then applies and provides a derivation of $\Pi\overline{\Delta} \vdash \langle\Pi\overline{\gamma}, x \mapsto t\rangle : (\Pi\overline{\Gamma}, x : A)$

For the identity monoidal substitution:

- For the monoidal context $[]$, we have $\mathrm{id}_{[]} = []$, and the rule (EMS) gives a derivation of $\varnothing \vdash [] : []$.
- For a monoidal context of the form $[\overline{\Gamma}; \varnothing]$, we have $\mathrm{id}_{[\overline{\Gamma};\varnothing]} = [\mathrm{id}_{\overline{\Gamma}}; \langle\rangle]$. Since $[\overline{\Gamma}; \varnothing] \vdash$, we have derivation of $\overline{\Gamma} \vdash$, which by the induction case for contexts gives a derivation of $\Pi\overline{\Gamma} \vdash$. Moreover, the rule (ES) applies to give a derivation of $\Pi\overline{\Gamma} \vdash \langle\rangle : \varnothing$. By the induction case for the identity monoidal substitution, we also have a derivation of $\Pi\overline{\Gamma} \vdash \mathrm{id}_{\overline{\Gamma}} : \overline{\Gamma}$, hence the rule (MS+) applies and gives a derivation of $\Pi\overline{\Gamma} \vdash [\mathrm{id}_{\overline{\Gamma}}; \langle\rangle] : [\overline{\Gamma}; \varnothing]$.
- For a monoidal context of the form $[\overline{\Gamma}; (\Gamma, x : A)]$, the induction case for the identity monoidal substitution gives a derivation of the judgment $\Pi[\overline{\Gamma}; \Gamma] \vdash \mathrm{id}_{[\overline{\Gamma};\Gamma]} : [\overline{\Gamma}; \Gamma]$, which by weakening provides a derivation of $(\Pi[\overline{\Gamma}; \Gamma], x : A) \vdash \mathrm{id}_{[\overline{\Gamma};\Gamma]} : [\overline{\Gamma}; \Gamma]$. Moreover, we have a derivation of $(\Pi[\overline{\Gamma}; \Gamma], x : A) \vdash x : A$ obtained by application of the rule (VAR). This lets us build a derivation for the judgment

$$(\Pi[\overline{\Gamma}; \Gamma], x : A) \vdash [\mathrm{id}_{\overline{\Gamma}}; \langle \mathrm{id}_{\Gamma}, x \mapsto t\rangle] : [\overline{\Gamma}; (\Gamma, x : A)]$$

□

This proposition formalizes the fact that monoidal contexts are regular contexts with extra structure, in the sense that it defines a way to forget this extra structure, and take any monoidal judgment to a regular judgment.

The unit. Conversely, we define an associated operation to the concatenation, that we call the *unit* (as it is the operation induced by the unit of the lists monad) and denote η. It takes a regular context (resp. a regular substitution) and produces a monoidal context (resp. monoidal substitution) by associating the one-element list whose element is the argument.

$$\eta(\Gamma) = [\Gamma] \qquad\qquad \eta\gamma = [\gamma]$$

We can also give an inductive definition of η, that is useful for the proofs by induction

$$\eta(\varnothing) = [\varnothing] \qquad\qquad \eta(\Gamma, x : A) = (\eta(\Gamma), x : A)$$
$$\eta(\langle\rangle) = [\langle\rangle] \qquad\qquad \eta(\langle\gamma, x \mapsto t\rangle) = \langle\eta(\gamma), x \mapsto t\rangle$$

Lemma 144. *This operation on the syntax respects the derivability:*

– *for any context* $\Gamma \vdash$*, the judgment* $\eta(\Gamma) \vdash$ *is derivable*

– *for any type* $\Gamma \vdash A$*, the judgment* $\eta(\Gamma) \vdash A$ *is derivable*

– *for any terms* $\Gamma \vdash t : A$*, the judgment* $\eta(\Gamma) \vdash t : A$ *is derivable*

– *for any substitution* $\Delta \vdash \gamma : \Gamma$*, the judgment* $\eta(\Delta) \vdash \eta(\gamma) : \eta(\Gamma)$ *is derivable*

– *the expression* id_Γ *defines a substitution* $\eta(\Gamma) \vdash \mathrm{id}_\Gamma$

Proof. We prove these result by mutual induction

Induction for contexts:

– For the empty context $\varnothing$, we have a derivation of $\eta(\varnothing)$ as follows

$$\dfrac{\dfrac{}{[] \vdash}(\textsc{mec})}{[\varnothing] \vdash}(\textsc{mc}+)$$

– For a context $(\Gamma, x : A) \vdash$, we necessarily have a derivation of $\Gamma \vdash A$, which by the induction case for types gives a derivation of $\eta(\Gamma) \vdash A$, and since $x \notin \mathrm{Var}(\Gamma)$, we also have $x \notin \mathrm{Var}(\eta(\Gamma))$, which lets us apply the rule (MCE) to get a derivation of $(\eta(\Gamma), x : A) \vdash$.

Induction for types:

– For the type $\Gamma \vdash \star$, we have a derivation of $\Gamma \vdash$, which gives by the induction case for contexts a derivation of $\eta(\Gamma) \vdash$, and hence the rule ($\star$-MINTRO) applies to give a derivation of $\eta(\Gamma) \vdash \star$.

– For the type $\Gamma \vdash t \underset{A}{\longrightarrow} u$, we necessarily have a derivation of $\Gamma \vdash A$, of $\Gamma \vdash t : A$ and of $\Gamma \vdash u : A$. By the induction cases for types and terms, these provide derivations of $\eta(\Gamma) \vdash A$, of $\eta(\Gamma) \vdash t : A$ and of $\eta(\Gamma) \vdash u : A$. The rule ($\to$-MINTRO) then applies and gives a derivation of $\eta(\Gamma) \vdash t \underset{A}{\longrightarrow} u$.

Induction for terms:

– For a variable term $\Gamma \vdash x : A$, we have a derivation of $\Gamma \vdash$, which by the induction case for contexts gives a derivation of $\eta(\Gamma) \vdash$. Moreover, the condition $(x : A) \in \Gamma$ implies that $(x : A) \in \eta(\Gamma)$, and then by the rule (MVAR), we have a derivation of $\eta(\Gamma) \vdash x : A$.

– For a generic term $\Gamma \vdash t : A$, then we have the substitution $\eta(\Gamma) \vdash \mathrm{id}_\Gamma : \Gamma$, and by our assumption on term constructors, this ensures that we have $\eta(\Gamma) \vdash t[\mathrm{id}_\Gamma] : A[\mathrm{id}_\Gamma]$, hence $\eta(\Gamma) \vdash t : A$.

Induction for substitutions:

– For the substitution $\Delta \vdash \langle\rangle : \varnothing$, we have a derivation of $\Delta \vdash$, which by induction provides a derivation of $\eta(\Delta) \vdash$. This lets us build a derivation of $\eta(\Delta) \vdash \eta(\langle\rangle) : \eta(\varnothing)$ as follows

$$\dfrac{\dfrac{}{[] \vdash [] : []}(\textsc{mes}) \qquad \eta(\Delta) \vdash}{\eta(\Delta) \vdash [][];_{[]} \langle\rangle] : [\varnothing]}(\textsc{ms}+)$$

– For the substitution $\Delta \vdash \langle \gamma, x \mapsto t \rangle : (\Gamma, x : A)$, we have a derivation of $\Delta \vdash \gamma : \Gamma$, which by the induction case for substitution gives a derivation of $\eta(\Delta) \vdash \eta(\gamma) : \eta(\Gamma)$. Moreover, we also have a derivation of $\Delta \vdash t : A[\gamma]$, which by the induction case for terms gives a derivation of $\eta(\Delta) \vdash t : A[\gamma]$, and since $A[\gamma] = A[\eta(\gamma)]$ this lets us apply the rule (MSE) in order to get a derivation as follows

$$\frac{\eta(\Delta) \vdash [[];_{[]} \gamma] : \eta(\Gamma) \qquad (\eta(\Gamma), x : A) \vdash \qquad \eta(\Delta) \vdash t : A[\eta(\gamma)]}{\Delta \vdash [[];_{[]} \langle \gamma, x \mapsto t \rangle] : (\eta(\Gamma), x : A)}(\text{MSE})$$

Induction for the identity:

– For the context $\varnothing$, we have by the induction case on context a derivation of $\eta(\varnothing) \vdash$, and thus we have a derivation of $\eta(\varnothing) \vdash \langle\rangle : \varnothing$

– For the context $(\Gamma, x : A)$, we necessarily have a derivation of $\Gamma \vdash$, so the induction case for the identity gives a derivation of $\eta(\Gamma) \vdash \mathrm{id}_\Gamma : \Gamma$, and since we also have a derivation of $\eta(\Gamma, x : A) \vdash$ by the induction case for contexts, we get by weakening (which we assume to hold in this framework as well) a derivation of $\eta(\Gamma, x : A) \vdash \mathrm{id}_\Gamma : \Gamma$. Moreover, we have a derivation of $\eta(\Gamma, x : A) \vdash x : A$ by the induction case for terms, which lets us build a derivation as follows

$$\frac{\eta(\Gamma, x : A) \vdash \mathrm{id}_\Gamma : \Gamma \qquad (\Gamma, x : A) \vdash \qquad \eta(\Gamma, x : A) \vdash x : A}{\eta(\Gamma, x : A) \vdash \langle \mathrm{id}_\Gamma, x \mapsto x \rangle : (\Gamma, x : A)}$$

□

It is immediate from the definition that the unit and the concatenation interact nicely, in the sense that for all context Γ, we have $\Pi(\eta(\Gamma)) = \Gamma$.

5.3.2 Monoidal ps-contexts

Inference rules for monoidal ps-contexts. We call *monoidal ps-contexts* the lists of contexts that are obtained from a ps-context by the operation we just described, and we introduce a new judgment in order to characterize them, as well as a judgment for partially constructed monoidal ps-contexts as before.

$\overline{\Gamma} \vdash_{\mathsf{ps}}$	$\overline{\Gamma}$ is a well-formed monoidal ps-context
$\overline{\Gamma} \vdash_{\mathsf{ps}} x : A$	$\overline{\Gamma}$ is a monoidal ps-context with dangling variable x of type A

The derivation rules for the monoidal ps-contexts are reminiscent of the derivation rules of the judgment for regular ps-contexts, but allow for more derivations.

$$\frac{\dfrac{}{[] \vdash_{\mathsf{ps}}}(\text{MPSNIL})}{[\overline{\Gamma}; (x : \star)] \vdash_{\mathsf{ps}} x : \star}(\text{MPSS}) \qquad \frac{\overline{\Gamma} \vdash_{\mathsf{ps}} x : A}{(\overline{\Gamma}, y : A, f : x \underset{A}{\longrightarrow} y) \vdash_{\mathsf{ps}} f : x \underset{A}{\longrightarrow} y}(\text{MPSE})$$

$$\frac{\overline{\Gamma} \vdash_{\mathsf{ps}} f : x \underset{A}{\longrightarrow} y}{\overline{\Gamma} \vdash_{\mathsf{ps}} y : A}(\text{MPSD}) \qquad \frac{\overline{\Gamma} \vdash_{\mathsf{ps}} x : \star}{\overline{\Gamma} \vdash_{\mathsf{ps}}}(\text{MPS})$$

where in the rule (MPSS) we assume that x is not a variable in $\overline{\Gamma}$, and in the rule (MPSE) we assume that neither y nor f are variables in $\overline{\Gamma}$. Intuitively, these rules simply describe lists of

ps-contexts, indeed a derivation of $\overline{\Gamma} \vdash_{\mathsf{ps}}$ can only be a succession of application of rule (MPSS) followed by (MPS) with in between a proof that a component of $\overline{\Gamma}$ is a ps-context. The only added constraint is that these ps-contexts have non-clashing variable names.

Source and target of a monoidal ps-context. We define operations of *i-source* and *i-target* suited for monoidal ps-contexts. In this operation, we need to treat the case $i = 0$ separately, we define in this case $\partial_0^-(\overline{\Gamma}) = []$ and $\partial_0^+(\overline{\Gamma}) = []$, and consider a number $i > 0$. Then we pose

$$\partial_i^-([]) = [] \qquad \partial_i^-([\overline{\Gamma}; (x : \star)]) = [\partial_i^-(\overline{\Gamma}); (x : \star)]$$

$$\partial_i^-(\overline{\Gamma}, y : A, f : x \underset{A}{\longrightarrow} y) = \begin{cases} \partial_i^-(\overline{\Gamma}) & \text{if } \dim y \geq i \\ (\partial_i^-(\overline{\Gamma}), y : A, f : x \underset{A}{\longrightarrow} y) & \text{otherwise} \end{cases}$$

For the i-source, and for the i-target, we pose

$$\partial_i^+([]) = [] \qquad \partial_i^+([\overline{\Gamma}; (x : \star)]) = [\partial_i^+(\overline{\Gamma}); (x : \star)]$$

$$\partial_i^+(\overline{\Gamma}, y : A, f : x \underset{A}{\longrightarrow} y) = \begin{cases} \partial_i^+(\overline{\Gamma}) & \text{if } \dim y > i \\ (\mathrm{drop}(\partial_i^+(\overline{\Gamma})), y : A) & \text{if } \dim y = i \\ (\partial_i^+(\overline{\Gamma}), y : A, f : x \underset{A}{\longrightarrow} y) & \text{otherwise} \end{cases}$$

where $\mathrm{drop}\,\overline{\Gamma}$ is the operator drop applied to the last context in $\overline{\Gamma}$

This lets us define the general notion of source for monoidal ps-contexts, which are useful for defining the introduction rules for coherences

$$\partial^-\overline{\Gamma} = \partial^-_{\dim\overline{\Gamma}}\overline{\Gamma} \qquad \partial^+\overline{\Gamma} = \partial^+_{\dim\overline{\Gamma}}\overline{\Gamma}$$

Note that in the case of a monoidal ps-context of dimension 0, (i.e., a list of ps-contexts all isomorphic to D^0), this definition does not make sense and we then use the convention that $\partial^-\overline{\Gamma} = []$. In general the case of the monoidal ps-context of dimension 0 is a limit case, and it will appear in various places. Similarly, monoidal ps-contexts come equipped with a notion of target $\partial^+\overline{\Gamma}$, defined the same way as the source from the corresponding notion on ps-contexts. The following lemma is immediate by definition of the source and the target,

Lemma 145. *Any monoidal ps-context $\overline{\Gamma}$ of dimension non-zero has the same length than its source and its target $\ell(\overline{\Gamma}) = \ell(\partial^-(\overline{\Gamma})) = \ell(\partial^+(\overline{\Gamma}))$.*

Term constructors. With the help of the monoidal ps-contexts, we can define new term constructors, analogous to mop and mcoh, and that we denote mop' and mcoh'. Formally, we define terms to be of the form $\mathsf{mop}'_{\overline{\Gamma},A}[\overline{\gamma}]$, where $\overline{\Gamma}$ is a monoidal ps-context, A is a monoidal type is $\overline{\Gamma}$ and γ is a list of regular substitutions. Additionally, we require as a side condition that either $\dim\overline{\Gamma} = 0$ and $A = \star$, or A is of the form $t \underset{B}{\longrightarrow} u$ with $\mathrm{Var}(t) \cup \mathrm{Var}(B) = \partial^-(\overline{\Gamma})$ and $\mathrm{Var}(u) \cup \mathrm{Var}(B) = \partial^+(\overline{\Gamma})$. When these conditions are met, we say that A and $\overline{\Gamma}$ satisfy $(\mathrm{C}_{\mathsf{mop}'})$. Similarly, we add terms of the form $\mathsf{mcoh}'_{\overline{\Gamma},A}[\overline{\gamma}]$, where $\overline{\Gamma}$ is a monoidal ps-context, A is a type expression, and $\overline{\gamma}$ is a list of regular substitution expressions. Additionally, we also require as a side condition that either $\overline{\Gamma} = []$ and $A = \star$, or A is of the form $t \underset{B}{\longrightarrow} u$ with

$\mathrm{Var}(B) \cup \mathrm{Var}(t) = \mathrm{Var}(\overline{\Gamma})$ and $\mathrm{Var}(B) \cup \mathrm{Var}(u) = \mathrm{Var}(\overline{\Gamma})$, and when these condition are met, we say that A and $\overline{\Gamma}$ satisfy ($\mathrm{C}_{\mathsf{mcoh}'}$). In addition to these term constructors, we have to define the action of substitution and lists of substitutions, that we pose as follows

$$\mathsf{mop}'_{\overline{\Gamma},A}[\overline{\gamma}][\delta] = \mathsf{mop}'_{\overline{\Gamma},A}[\overline{\gamma} \circ \delta] \qquad \mathsf{mop}'_{\overline{\Gamma},A}[\overline{\gamma}][\overline{\delta}] = \mathsf{mop}'_{\overline{\Gamma},A}[\overline{\gamma} \circ \overline{\delta}]$$
$$\mathsf{mcoh}'_{\overline{\Gamma},A}[\overline{\gamma}][\delta] = \mathsf{mcoh}'_{\overline{\Gamma},A}[\overline{\gamma} \circ \delta] \qquad \mathsf{mcoh}'_{\overline{\Gamma},A}[\overline{\gamma}][\overline{\delta}] = \mathsf{mcoh}'_{\overline{\Gamma},A}[\overline{\gamma} \circ \overline{\delta}]$$

Note that these definition rely on the compositions, which themselves rely on the application of the substitution on terms, hence making these two definitions mutually inductive. For the sake of simplicity, and to present our motivation, we have chosen to present them separated, and we admit that the properties that we have proved still hold when these definitions become mutually inductive.

Term introduction rules. These new term constructors come with associated introduction rules in order to create new monoidal terms, and hence new monoidal terms. This gives the recursive structure of the theory. Formally, these rules mirror the term introduction rules of CaTT, but in the monoidal fragment of the theory. These rules are the following

$$\frac{\overline{\Gamma} \vdash_{\mathsf{ps}} \qquad \overline{\Gamma} \vdash A \qquad \overline{\Delta} \vdash \overline{\gamma} : \overline{\Gamma}}{\overline{\Delta} \vdash \mathsf{mop}'_{\overline{\Gamma},A}[\overline{\gamma}]}$$

whenever $\overline{\Gamma}$ and A satisfy ($\mathrm{C}_{\mathsf{mop}'}$), and

$$\frac{\overline{\Gamma} \vdash_{\mathsf{ps}} \qquad \overline{\Gamma} \vdash A \qquad \overline{\Delta} \vdash \overline{\gamma} : \overline{\Gamma}}{\overline{\Delta} \vdash \mathsf{mcoh}'_{\overline{\Gamma},A}[\overline{\gamma}]}$$

whenever $\overline{\Gamma}$ and A satisfy ($\mathrm{C}_{\mathsf{mcoh}'}$). Additionally, we add introduction rules for regular terms in the theory, that are intuitively nothing else than a monoidal term that has been transferred to a regular context. Formally these rules are the following

$$\frac{\overline{\Gamma} \vdash_{\mathsf{ps}} \qquad \overline{\Gamma} \vdash A \qquad \Delta \vdash \overline{\gamma} : \overline{\Gamma}}{\Delta \vdash \mathsf{mop}'_{\overline{\Gamma},A}[\overline{\gamma}]}(\text{MOP'})$$

whenever $\overline{\Gamma}$ and A satisfy ($\mathrm{C}_{\mathsf{mop}'}$), and

$$\frac{\overline{\Gamma} \vdash_{\mathsf{ps}} \qquad \overline{\Gamma} \vdash A \qquad \Delta \vdash \overline{\gamma} : \overline{\Gamma}}{\Delta \vdash \mathsf{mcoh}'_{\overline{\Gamma},A}[\overline{\gamma}]}(\text{MCOH'})$$

whenever $\overline{\Gamma}$ and A satisfy ($\mathrm{C}_{\mathsf{mcoh}'}$). We refer the reader to Appendix A.4 for a complete presentation of all the rules of MCaTT'.

Substitutions satisfying the side conditions. Consider a monoidal substitution $\overline{\Delta} \vdash \overline{\gamma} \vdash \overline{\Gamma}$, between two monoidal ps-contexts $\overline{\Delta}$ and $\overline{\Gamma}$, such that the variables of $\overline{\gamma}$ are either $\mathrm{Var}(\Delta)$, or $\mathrm{Var}(\partial^-(\overline{\Gamma}))$ or $\mathrm{Var}(\partial^+(\overline{\Delta}))$. We admit that in this case, there is a unique way annotate the list $\overline{\gamma}$ such that $\overline{\Delta} \vdash \overline{\gamma} : \overline{\Gamma}$, so in this case we can forget the annotation and a monoidal substitution is completely determined by a list of substitutions.

5.3.3 Folding and flattening

Our objective is now to prove that theory MCaTT' that we have defined is the same as the theory MCaTT. More precisely, we consider the syntactic category of MCaTT', that is the category whose objects are regular contexts and whose morphisms are regular substitutions, and show that there is an equivalence of category with families with the syntactic category of the theory MCaTT. For this we consider syntactic operations on the theory MCaTT'.

Folding of a ps-context. We define translations between the monoidal ps-contexts and the regular contexts, that relate our ad-hoc notion of monoidal ps-context with the notion of ps-contexts that we have used in CaTT. We start by defining the operation of *folding*, that takes any regular context to a monoidal context. Although this operation is well defined for any context, we only study it in the case of a monoidal ps-contexts, as it does not have a clear interpretation in a regular context. Intuitively it forgets the variables of dimension 0 in the context and remove them, like the desuspension, but it also remembers their position by cutting the context into a list of context in every place of a variable of dimension 0. We define this operation inductively as follows

$$\varnothing^\sharp = [] \qquad\qquad (\Gamma, x : A)^\sharp = \begin{cases} [\Gamma^\sharp; \varnothing] & \text{if } A = \star \\ (\Gamma^\sharp, x : \downarrow A) & \text{otherwise} \end{cases}$$

Proposition 146. *The association $\_^\sharp$ is well-defined, for every ps-context $\Gamma \vdash_{\mathsf{ps}}$, the list $\Gamma^\sharp$ is a monoidal ps-context. In other words, the following rule is admissible*

$$\frac{\Gamma \vdash_{\mathsf{ps}}}{\Gamma^\sharp \vdash_{\mathsf{ps}}}$$

Proof. We prove by mutual induction that a derivation of the judgment $\Gamma \vdash_{\mathsf{ps}} x : \star$ induces a derivation of $\Gamma^\sharp \vdash_{\mathsf{ps}}$, and a derivation of the judgment $\Gamma \vdash_{\mathsf{ps}} x : A$ with A distinct from $\star$, induces a derivation of $\Gamma^\sharp \vdash_{\mathsf{ps}} x : \downarrow A$.

- *Induction for* $\Gamma \vdash_{\mathsf{ps}} x : \star$:
  - For the derivation of $(x : \star) \vdash_{\mathsf{ps}}$ obtained by the rule (PSS), we have $(x : \star)^\sharp = []$, and the rule (MPSNIL) then gives a derivation of $[] \vdash_{\mathsf{ps}}$.
  - For a derivation of $\Gamma \vdash_{\mathsf{ps}} y : \star$ obtained by application of the rule (PSD), we necessarily have a derivation of the judgment $\Gamma \vdash_{\mathsf{ps}} f : x \underset{\star}{\to} y$, which by the other induction case gives a derivation of $\Gamma^\sharp \vdash_{\mathsf{ps}} f : \star$. We can then apply the rule (MPS) to get a derivation of $\Gamma^\sharp \vdash_{\mathsf{ps}}$ as follows
  $$\frac{\Gamma^\sharp \vdash f : \star}{\Gamma^\sharp \vdash_{\mathsf{ps}}} (\text{MPSS})$$
- *Induction for* $\Gamma \vdash x : A$:
  - For a derivation of $(\Gamma, y : \star, f : x \underset{\star}{\to} y) \vdash_{\mathsf{ps}} f : x \underset{\star}{\to} y$ obtained by applying the rule (PSE), we necessarily have a derivation of $\Gamma \vdash_{\mathsf{ps}} x : \star$, which by induction gives a derivation of $\Gamma^\sharp \vdash_{\mathsf{ps}}$. This lets us build a derivation of $(\Gamma, y : \star, f : x \underset{\star}{\to} y)^\sharp \vdash_{\mathsf{ps}} f : \star$ by applying the rule (MPSS) as follows
  $$\frac{\Gamma^\sharp \vdash_{\mathsf{ps}}}{[\Gamma^\sharp, (f : \star)] \vdash_{\mathsf{ps}} f : \star} (\text{MPSS})$$

– For a derivation of $(\Gamma, y : A, f : x \underset{A}{\longrightarrow} y) \vdash_{\mathsf{ps}} f : x \underset{A}{\longrightarrow} y$ obtained by an application of the rule (PSE), with A distinct from $\star$, we necessarily have a derivation of $\Gamma \vdash_{\mathsf{ps}} x : A$, which by induction gives a derivation of $\Gamma^\sharp \vdash_{\mathsf{ps}} x : \downarrow A$. This lets us build a derivation of $(\Gamma, y : A, f : x \underset{A}{\longrightarrow} y) \vdash_{\mathsf{ps}} f : \downarrow x \underset{A}{\longrightarrow} y$ as follows

$$\frac{\Gamma^\sharp \vdash_{\mathsf{ps}} x : \downarrow A}{(\Gamma^\sharp, y : \downarrow A, f : x \underset{\downarrow A}{\longrightarrow} y) \vdash_{\mathsf{ps}} f : x \underset{\downarrow A}{\longrightarrow} y}(\text{MPSE})$$

– For a derivation of $\Gamma \vdash y : A$ obtained by the rule (PSD), we necessarily have a derivation of the judgment $\Gamma \vdash_{\mathsf{ps}} f : x \underset{A}{\longrightarrow} y$, which by induction gives a derivation of the judgment $\Gamma^\sharp \vdash_{\mathsf{ps}} f : x \underset{\downarrow A}{\longrightarrow} y$. We can then construct a derivation of the judgment $\Gamma^\sharp \vdash y : \downarrow A$ as follows

$$\frac{\Gamma^\sharp \vdash f : x \underset{\downarrow A}{\longrightarrow} y}{\Gamma^\sharp \vdash y : \downarrow A}(\text{MPSD})$$

□

Flattening of a monoidal ps-context. We define an operation acting as the opposite of the folding on ps-contexts , and which associate to each monoidal ps-context, a regular context obtained by suspending all the components of the ps-context and gluing them together along the new objects introduced by the suspension. We call this operation the *flattening*, and given a monoidal ps-context $\overline{\Gamma}$, we denote $\overline{\Gamma}^\flat$ the result of the flattening on $\overline{\Gamma}$. In order to define this operation, we assume that have a countable list of fresh variables $\bullet_0, \bullet_1, \bullet_2, \ldots$, that we can add to the signature of the theory, to ensure their freshness. We then define by induction the flattening of a generic monoidal context, but are only interested in the case of monoidal ps-context where this operation has good properties.

$$[]^\flat = (\bullet_0 : \star) \qquad [\overline{\Gamma}; \varnothing]^\flat = (\overline{\Gamma}^\flat, \bullet_{l+1} : \star)$$
$$[\overline{\Gamma}; (\Gamma, y : A)]^\flat = ([\overline{\Gamma}; \Gamma]^\flat, y : \Sigma^{l,l+1} A)$$

Where l is a shorthand for $\ell(\overline{\Gamma})$, and the operation $\Sigma^{i,j} A$ is defined of types as follows

$$\Sigma^{i,j} \star = \bullet_i \underset{\star}{\to} \bullet_j \qquad \Sigma^{i,j}(x \underset{A}{\longrightarrow} y) = x \underset{\Sigma^{i,j} A}{\longrightarrow} y$$

To simplify the notations, we simply write $\Sigma^i A$ instead of $\Sigma^{i,i+1} A$.

Proposition 147. *For any monoidal ps-context $\overline{\Gamma} \vdash_{\mathsf{ps}}$, the context $\overline{\Gamma}^\flat$ obtained this way is a ps-context. In other words, the following rule is admissible*

$$\frac{\overline{\Gamma} \vdash_{\mathsf{ps}}}{\overline{\Gamma}^\flat \vdash_{\mathsf{ps}}}$$

Proof. We prove by induction that a derivation of the judgment $\overline{\Gamma} \vdash_{\mathsf{ps}}$ induces a derivation of $\overline{\Gamma}^\flat \vdash_{\mathsf{ps}} \bullet_{\ell(\overline{\Gamma})} : \star$, and a derivation of the judgment $\overline{\Gamma}^\flat \vdash_{\mathsf{ps}} x : A$ induces a derivation of $\overline{\Gamma}^\flat \vdash x : \Sigma^{\ell(\overline{\Gamma})-1} A$.

Induction for the judgment $\overline{\Gamma} \vdash_{\mathsf{ps}}$:

- For the derivation $[] \vdash_{\mathsf{ps}}$ obtained by the rule (MPSNIL), we have the following derivation of $[]^\flat \vdash_{\mathsf{ps}} \bullet_0 : \star$

$$\frac{}{\bullet_0 : \star \vdash_{\mathsf{ps}} \bullet_0 : \star}(\text{PSS})$$

- For the a derivation $\overline{\Gamma} \vdash_{\mathsf{ps}}$ obtained by the rule (MPS), we necessarily have a derivation of the judgment $\overline{\Gamma} \vdash_{\mathsf{ps}} x : \star$ and by the induction case for this judgment, this provides a derivation of $\overline{\Gamma}^\flat \vdash_{\mathsf{ps}} x : \bullet_{\ell(\overline{\Gamma})-1} \underset{\star}{\to} \bullet_{\ell(\overline{\Gamma})}$. This lets us build a derivation of $\overline{\Gamma}^\flat \vdash_{\mathsf{ps}} \bullet_{\ell(\overline{\Gamma})} : \star$ by applying the rule (PSD) as follows

$$\frac{\overline{\Gamma}^\flat \vdash_{\mathsf{ps}} x : \bullet_{\ell(\overline{\Gamma})-1} \underset{\star}{\to} \bullet_{\ell(\overline{\Gamma})}}{\overline{\Gamma}^\flat \vdash_{\mathsf{ps}} \bullet_{\ell(\overline{\Gamma})} : \star}(\text{PSD})$$

Induction for the judgment $\overline{\Gamma} \vdash_{\mathsf{ps}} x : A$:

- For a derivation $[\overline{\Gamma}, (x : \star)] \vdash x : \star$ obtained by application of the rule (MPSS), we necessarily have a derivation of $\overline{\Gamma} \vdash_{\mathsf{ps}}$, which by the induction case for this judgment gives a derivation of $\overline{\Gamma}^\flat \vdash_{\mathsf{ps}} \bullet_{\ell(\Gamma)} : \star$. We then construct a derivation of $[\overline{\Gamma}; (x : \star)]^\flat \vdash_{\mathsf{ps}} x : \Sigma^{\ell(\Gamma)} \star$ as follows

$$\frac{\overline{\Gamma}^\flat \vdash_{\mathsf{ps}} \bullet_{\ell(\Gamma)}}{\overline{\Gamma}^\flat, \bullet_{\ell(\overline{\Gamma})+1} : \star, x : \bullet_{\ell(\overline{\Gamma})} \underset{\star}{\to} \bullet_{\ell(\overline{\Gamma})+1} \vdash_{\mathsf{ps}} x : \bullet_{\ell(\overline{\Gamma})} \underset{\star}{\to} \bullet_{\ell(\overline{\Gamma})+1}}(\text{PSE})$$

- For a derivation $[\overline{\Gamma}, (\Gamma, y : A, f : x \underset{A}{\to} y)] \vdash_{\mathsf{ps}} f : x \underset{A}{\to} y$ obtained by applying the rule (MPSE), we necessarily have a derivation of $[\overline{\Gamma}; \Gamma] \vdash_{\mathsf{ps}} x : A$, which by induction provides a derivation for $[\overline{\Gamma}; \Gamma]^\flat \vdash_{\mathsf{ps}} x : \Sigma^{\ell(\Gamma)} A$. This lets us construct a derivation of $[\overline{\Gamma}; (\Gamma, y : A, f : x \underset{A}{\to} y)]^\flat \vdash_{\mathsf{ps}} f : \Sigma^{\ell(\Gamma)}(x \underset{A}{\to} y)$ as follows

$$\frac{[\overline{\Gamma}; \Gamma]^\flat \vdash_{\mathsf{ps}} x : \Sigma^{\ell(\overline{\Gamma})} A}{([\overline{\Gamma}; \Gamma]^\flat, y : \Sigma^{\ell(\overline{\Gamma})} A, f : x \xrightarrow[\Sigma^{\ell(\overline{\Gamma})} A]{} y) \vdash_{\mathsf{ps}} f : x \xrightarrow[\Sigma^{\ell(\overline{\Gamma})} A]{} y}(\text{PSE})$$

- For a derivation $\overline{\Gamma} \vdash_{\mathsf{ps}} y : A$ obtained by application of the rule (MPSD), we necessarily have a derivation of a judgment on the form $\overline{\Gamma} \vdash_{\mathsf{ps}} f : x \underset{A}{\to} y$. By induction, this provides a derivation of the judgment $\overline{\Gamma}^\flat \vdash_{\mathsf{ps}} f : x \xrightarrow[\Sigma^{\ell(\overline{\Gamma})-1} A]{} y$. This lets us build a derivation of $\overline{\Gamma}^\flat \vdash_{\mathsf{ps}} y : \Sigma^{\ell(\overline{\Gamma})-1} A$ as follows

$$\frac{\overline{\Gamma}^\flat \vdash_{\mathsf{ps}} f : x \xrightarrow[\Sigma^{\ell(\overline{\Gamma})-1} A]{} y}{\overline{\Gamma}^\flat \vdash_{\mathsf{ps}} y : \Sigma^{\ell(\overline{\Gamma})-1} A}(\text{PSD})$$

We then apply this result to our derivation of $\overline{\Gamma} \vdash_{\mathsf{ps}}$, which gives a derivation of $\overline{\Gamma}^\flat \vdash_{\mathsf{ps}} \bullet_{\ell(\overline{\Gamma})} : \star$. Applying the rule (PS) gives a derivation of $\overline{\Gamma}^\flat \vdash_{\mathsf{ps}}$ □

Monoidal ps-contexts vs. ps-contexts. The two operations of folding and flattening show that the structure of monoidal ps-context is closely related to the one of ps-context, and in fact we have the following result

Lemma 148. *The folding and the flattening define a bijection between monoidal ps-contexts and ps-contexts (up to α-equivalence).*

This can by proved by following the proofs of Lemmas 146 and 147 and show that starting from a derivation of $\Gamma \vdash_{\mathsf{ps}} x : A$ the induced derivation of $\Gamma \vdash_{\mathsf{ps}} y : B$ by applying folding and then flattening has the same structure as the original derivation, and the other way around. We admit this result here, as it is apparent on the combinatorial structure. An efficient way to follow the derivations would be to give a system of equation computing a derivation of $\Gamma^\sharp \vdash_{\mathsf{ps}}$ from a derivation $\Gamma \vdash_{\mathsf{ps}}$ as well as system of equation computing $\overline{\Gamma}^\flat \vdash_{\mathsf{ps}}$ from a derivation of $\overline{\Gamma} \vdash_{\mathsf{ps}}$, and show that these systems are inverse to each other.

Concatenation of a monoidal ps-context.

Lemma 149. *For every ps-context Γ the equality $\Pi(\Gamma^\sharp) = \downarrow\Gamma$.*

Proof. We prove this by induction on the structure of the ps-contexts

- For the ps-context $(x : \star)$, we have $\downarrow(x : \star) = \varnothing$ and $(x : \star)^\flat = []$, hence $\Pi((x : \star)^\flat) = \varnothing$.
- For a ps-context of the form $(\Gamma, y : \star, f : x \underset{\star}{\to} y)$ with Γ a ps-context, we have

$$\downarrow(\Gamma, y : \star, f : x \underset{\star}{\to} y) = (\downarrow\Gamma, f : \star)$$
$$(\Gamma, y : \star, f : x \underset{\star}{\to} y)^\flat = [\Gamma^\flat; (f : \star)]$$

  hence $\Pi(\Gamma, y : \star, f : x \underset{\star}{\to} y)^\flat = (\Pi(\Gamma^\flat), f : \star)$, and the induction shows that $\Pi(\Gamma^\flat) = \downarrow\Gamma$, hence the equality.

- For a ps-context of the form $(\Gamma, y : A, f : x \underset{A}{\to} y)$ with A distinct from $\star$, we have

$$\downarrow(\Gamma, y : A, f : x \underset{A}{\to} y) = \downarrow\Gamma, y : \downarrow A, f : x \to y$$
$$(\Gamma, y : A, f : x \underset{A}{\to} y)^\flat = (\Gamma^\flat, y : \downarrow A, f : x \to y)$$

  thus $\Pi((\Gamma, y : A, f : x \underset{A}{\to} y)^\flat) = \Pi(\Gamma^\flat), y : \downarrow A, f : x \to y$. Moreover, by induction we have $\Pi(\Gamma^\flat)$, hence the equality.

□

Source and target. These definition are just the induced notions of source and target from the regular ps-context to the monoidal ps-contexts, under the bijection defined by folding and flattening. More formally, we have the following lemma

Lemma 150. *For every ps-context* Γ *and every* $i \leq \dim \Gamma$*, the following equalities hold*

$$(\partial_i^-(\Gamma))^\sharp = \partial_i^-(\Gamma^\sharp)$$
$$(\partial_i^+(\Gamma))^\sharp = \partial_i^+(\Gamma^\sharp)$$

Proof. We first treat separately the case $i = 0$, which is a special case.

– In this case the contexts $\partial_0^-(\Gamma)$ and $\partial_0^+(\Gamma)$ are both of dimension 0, and hence we have

$$(\partial_0^-(\Gamma))^\sharp = [] \qquad (\partial_0^+(\Gamma))^\sharp = []$$

on the other hand, we have by definition

$$\partial_0^-(\Gamma^\sharp) = [] \qquad \partial_0^+(\Gamma^\sharp) = []$$

We now assume that $i > 0$ and prove this result by induction on the structure of the ps-context Γ.

– For the ps-context $(x : \star)$, we have the following equalities

$$\begin{aligned}(\partial_i^-(x:\star))^\flat &= \varnothing^\flat & (\partial_i^+(x:\star))^\flat &= \varnothing^\flat\\ &= [] & &= []\end{aligned}$$

and on the other hand

$$\begin{aligned}\partial_i^-((x:\star)^\flat) &= \partial_i^-([]) & \partial_i^+((x:\star)^\flat) &= \partial_i^+([])\\ &= [] & &= []\end{aligned}$$

– For a context of the form $(\Gamma, y : \star, f : x \underset{\star}{\to} y)$, we have the equalities

$$\begin{aligned}(\partial_i^-(\Gamma, y:\star, f : x \underset{\star}{\to} y))^\sharp &= (\partial_i^-(\Gamma), y:\star, f : x \underset{\star}{\to} y)^\sharp\\ &= [(\partial_i^-(\Gamma))^\sharp; (f:\star)]\end{aligned}$$

and

$$\begin{aligned}(\partial_i^+(\Gamma, y:\star, f : x \underset{\star}{\to} y))^\sharp &= (\partial_i^+(\Gamma), y:\star, f : x \underset{\star}{\to} y)^\sharp\\ &= [(\partial_i^+(\Gamma))^\sharp; (f:\star)]\end{aligned}$$

and on the other hand

$$\begin{aligned}\partial_i^-((\Gamma, y:\star, f : x \underset{\star}{\to} y)^\sharp) &= \partial_i^-([\Gamma^\sharp; (f:\star)])\\ &= [\partial_i^-(\Gamma^\sharp); (f:\star)]\end{aligned}$$

and

$$\begin{aligned}\partial_i^+((\Gamma, y:\star, f : x \underset{\star}{\to} y)^\sharp) &= \partial_i^+([\Gamma^\sharp; (f:\star)])\\ &= [\partial_i^+(\Gamma^\sharp); (f:\star)]\end{aligned}$$

By induction, we have $(\partial_i^-(\Gamma))^\sharp = \partial_i^-(\Gamma^\sharp)$ and $(\partial_i^+(\Gamma))^\sharp = \partial_i^+(\Gamma^\sharp)$ which prove the desired equalities

– For a context of the form $(\Gamma, y : A, f : x \underset{A}{\longrightarrow} y)$, we separate into three different cases

  – If $i \leq \dim y$, we have the following equalities

$$(\partial_i^-(\Gamma, y : A, f : x \underset{A}{\longrightarrow} y))^\sharp = (\partial_i^-(\Gamma))^\sharp$$
$$(\partial_i^+(\Gamma, y : A, f : x \underset{A}{\longrightarrow} y))^\sharp = (\partial_i^+(\Gamma))^\sharp$$

  and on the other hand

$$\begin{aligned}\partial_i^-((\Gamma, y : A, f : x \underset{A}{\longrightarrow} y)^\sharp) &= \partial_i^-(\Gamma^\sharp, y : A, f : x \underset{A}{\longrightarrow} y)\\ &= \partial_i^-(\Gamma^\sharp)\end{aligned}$$

  and

$$\begin{aligned}\partial_i^+((\Gamma, y : A, f : x \underset{A}{\longrightarrow} y)^\sharp) &= \partial_i^+(\Gamma^\sharp, y : A, f : x \underset{A}{\longrightarrow} y)\\ &= \partial_i^+(\Gamma^\sharp)\end{aligned}$$

  By induction we then have $(\partial_i^-(\Gamma))^\sharp = \partial_i^-(\Gamma^\sharp)$ and $(\partial_i^+(\Gamma))^\sharp = \partial_i^+(\Gamma^\sharp)$

  – If $i = \dim y$, we have the following equalities

$$(\partial_i^-(\Gamma, y : A, f : x \underset{A}{\longrightarrow} y))^\sharp = (\partial_i^-(\Gamma))^\sharp$$

  and

$$\begin{aligned}(\partial_i^+(\Gamma, y : A, f : x \underset{A}{\longrightarrow} y))^\sharp &= (\mathrm{drop}(\partial_i^+(\Gamma)), y : A)^\sharp\\ &= (\mathrm{drop}(\partial_i^+(\Gamma))^\sharp, y : \downarrow A)\end{aligned}$$

  and on the other hand

$$\begin{aligned}\partial_i^-((\Gamma, y : A, f : x \underset{A}{\longrightarrow} y)^\sharp) &= \partial_i^-(\Gamma^\sharp, y : \downarrow A, f : x \underset{\downarrow A}{\longrightarrow} y)\\ &= \partial_i^-(\Gamma^\sharp)\end{aligned}$$

  and

$$\begin{aligned}\partial_i^+((\Gamma, y : A, f : x \underset{A}{\longrightarrow} y)^\sharp) &= \partial_i^+(\Gamma^\sharp, y : \downarrow A, f : x \underset{\downarrow A}{\longrightarrow} y)\\ &= (\mathrm{drop}(\partial_i^+(\Gamma^\sharp)), y : \downarrow A)\end{aligned}$$

  By induction we have $(\partial_i^-(\Gamma))^\sharp = \partial_i^-(\Gamma^\sharp)$ and $(\partial_i^+(\Gamma))^\sharp = \partial_i^+(\Gamma^\sharp)$, and moreover an induction shows that $\mathrm{drop}(\Delta)^\sharp = \mathrm{drop}(\Delta^\sharp)$.

– If $i > \dim y$, we have

$$(\partial_i^-(\Gamma, y : A, f : x \underset{A}{\to} y))^\sharp = (\partial_i^-(\Gamma), y : A, f : x \underset{A}{\to} y)^\sharp$$
$$= ((\partial_i^-(\Gamma))^\sharp, y : \downarrow A, f : x \underset{\downarrow A}{\longrightarrow} y)$$

and

$$(\partial_i^+(\Gamma, y : A, f : x \underset{A}{\to} y))^\sharp = (\partial_i^+(\Gamma), y : A, f : x \underset{A}{\to} y)^\sharp$$
$$= ((\partial_i^+(\Gamma))^\sharp, y : \downarrow A, f : x \underset{\downarrow A}{\longrightarrow} y)$$

and on the other hand

$$\partial_i^-((\Gamma, y : A, f : x \underset{A}{\to} y)^\sharp) = \partial_i^-(\Gamma^\sharp, y : \downarrow A, f : x \underset{\downarrow A}{\longrightarrow} y)$$
$$= (\partial_i^-(\Gamma^\sharp), y : \downarrow A, f : x \underset{\downarrow A}{\longrightarrow} y)$$

and

$$\partial_i^+((\Gamma, y : A, f : x \underset{A}{\to} y)^\sharp) = \partial_i^+(\Gamma^\sharp, y : \downarrow A, f : x \underset{\downarrow A}{\longrightarrow} y)$$
$$= (\partial_i^+(\Gamma^\sharp), y : \downarrow A, f : x \underset{\downarrow A}{\longrightarrow} y)$$

By induction we have $(\partial_i^-(\Gamma))^\sharp = \partial_i^-(\Gamma^\sharp)$ and $(\partial_i^+(\Gamma))^\sharp = \partial_i^+(\Gamma^\sharp)$, which prove the equalities.

□

Flattening of a monoidal judgment. In the particular case of a monoidal judgment in a monoidal ps-context, we can extend the operation of flattening. In order to define this operation in general, we need to provide two extra arguments, i and j that are natural number, and we always assume that $i < j$. We denote $A^\flat_{i,j}$ (resp. $t^\flat_{i,j}$, $\overline{\gamma}^\flat_{i,j}$) the result of a monoidal type A (resp. monoidal term t, monoidal substitution $\overline{\gamma}$) by this operation, and we define it by induction as follows

$$\star^\flat_{i,j} = \bullet_i \underset{\star}{\to} \bullet_j \qquad (t \underset{A}{\to} u)^\flat_{i,j} = t^\flat_{i,j} \underset{A^\flat_{i,j}}{\longrightarrow} u^\flat_{i,j}$$
$$x^\flat_{i,j} = x \qquad \mathsf{mop}'_{\overline{\Gamma},A}[\overline{\gamma}] = \mathsf{op}_{\overline{\Gamma}^\flat,A^\flat}[\overline{\gamma}^\flat_{n,m}]$$
$$\mathsf{mcoh}'_{\overline{\Gamma},A}[\overline{\gamma}] = \mathsf{coh}_{\overline{\Gamma}^\flat,A^\flat}[\overline{\gamma}^\flat_{n,m}]$$
$$[]^\flat_{i,j} = \langle \bullet_0 \mapsto \bullet_i \rangle \qquad [\overline{\gamma};_{\overline{\Delta}} \langle\rangle] = \langle \overline{\gamma}^\flat_{i,i+\ell(\overline{\Delta})}, \bullet_{\ell(\overline{\gamma})} \mapsto \bullet_j \rangle$$
$$[\overline{\gamma};_{\overline{\Delta}} \langle \gamma, x \mapsto t \rangle]^\flat_{i,j} = \langle [\overline{\gamma};_{\overline{\Delta}} \gamma]^\flat_{i,j}, t^\flat_{i+\ell(\overline{\Delta})+1,j} \rangle$$

Where for a type $\overline{\Gamma} \vdash A$ satisfying ($\mathrm{C}_{\mathsf{mop}'}$) or ($\mathrm{C}_{\mathsf{mcoh}'}$) we denote $A^\flat$ for $A^\flat_{0,\ell(\overline{\Gamma})}$

In order to prove the correctness of the flattening operation, we introduce the notion of (i,j)-*fullness* of a monoidal type, a monoidal term or a monoidal substitution. We define this notion by mutual induction for types and terms:

- A monoidal type $\overline{\Delta} \vdash A$ is (i,j)-full if it is either the type $\star$, or of the form $t \underset{B}{\longrightarrow} u$ with both the terms $\overline{\Delta} \vdash t : B$ and $\overline{\Delta} \vdash u : B$ that are (i,j)-full.
- A monoidal term $\overline{\Delta} \vdash t : A$ is (i,j)-full if $\mathrm{Var}(t) \cup \mathrm{Var}(A)$ contains variables from all the contexts at the indices $i, i+1, \dots, j-1$ in $\overline{\Delta}$, and only from those.

A substitution $\overline{\Delta} \vdash \overline{\gamma} : \overline{\Gamma}$ is said to be (i,j)-full if the terms composing it contain all variables from the contexts at the indices $i, i+1, \dots, j-1$ in $\overline{\Delta}$, and only from those.

Lemma 151. *The notion of (i,j)-fullness is well-behaved in ps-contexts, to allow for inductive proofs*

- *The type $\overline{\Delta} \vdash t \underset{A}{\longrightarrow} u$ is (i,j)-full in a monoidal context $\overline{\Delta}$ if and only if the type A and the terms t and u are (i,j)-full*
- *Given two monoidal ps-contexts $\overline{\Delta}$ and $[\overline{\Gamma}; (x : \star)]$, the monoidal substitution*

$$\overline{\Delta} \vdash [\overline{\gamma}; _{\overline{\Delta}'} \langle x \mapsto t \rangle] : [\overline{\Gamma}; (x : \star)]$$

  is (i,j)-full if and only if $\overline{\gamma}$ is $(i, \ell(\overline{\Delta}'))$-full and t is $(\ell(\overline{\Delta}'), j)$-full
- *Given two monoidal ps-contexts $\overline{\Delta}$ and $[\overline{\Gamma}; (\Gamma, y : A, f : x \to y)]$, the monoidal substitution*

$$\overline{\Delta} \vdash [\overline{\gamma}; _{\overline{\Delta}'} \langle \gamma, y \mapsto t, f \mapsto u \rangle] : [\overline{\Gamma}; (\Gamma, y : A, f : x \to y)]$$

  is (i,j)-full if and only if $[\overline{\gamma}; _{\overline{\Delta}'} \gamma]$ is (i,j)-full and t and u are $(\ell(\overline{\Delta}'), j)$-full

Proof. By definition, of (i,j)-fullness, the monoidal type $\overline{\Delta} \vdash t \underset{A}{\longrightarrow} u$ is (i,j)-full if and only if the monoidal type A and the monoidal terms t and u are (i,j)-full. The second property comes from the fact that $\overline{\gamma}$ can only contain variables in the contexts before Δ', and t can only contain variables from the context starting from Δ', but they must together contain contexts from all contexts between i and j. The last property combines these two arguments: The substitution γ and the terms t and u can only ever contain variables starting from contexts starting from $\overline{\Delta}'$, but together they have to contain variables from all contexts between $\ell(\overline{\Delta}')$ and j. The connectedness of the ps-context $(\Gamma, y : A, f : x \to y)$ then implies that all the terms of $\langle \gamma, y \mapsto t, f \mapsto u \rangle$ contain variables from all contexts between $\ell(\Delta')$ and j □

Lemma 152. *For any type $\overline{\Delta} \vdash A$ satisfying ($\mathrm{C}_{\mathsf{mop}'}$) (resp. satisfying ($\mathrm{C}_{\mathsf{mcoh}'}$)) the flattened pair $(\overline{\Delta}^\flat, A^\flat_{0,\ell(\overline{\Delta})})$ satisfies (C_{op}) (resp. satisfies ($\mathrm{C}_{\mathsf{coh}}$)).*

Proof. Consider a type $\overline{\Delta} \vdash t \underset{A}{\longrightarrow} u$ satisfying ($\mathrm{C}_{\mathsf{mop}'}$) and denote $l = \ell(\overline{\Delta})$, we have

$$\begin{aligned}\mathrm{Var}(\partial^-(\overline{\Delta}^\flat)) &= \mathrm{Var}(\partial^-(\overline{\Delta})^\flat) && \text{by Lemma 150}\\ &= \mathrm{Var}(\partial^-(\overline{\Delta})) \cup \{\bullet_0, \dots, \bullet_l\}\end{aligned}$$

Moreover, by assumption $\mathrm{Var}(t) \cup \mathrm{Var}(A)$ contains all variables of $\partial^-(\overline{\Delta})$, hence it contains a variable x_i of dimension 0 from each of the contexts $\overline{\Delta}_i$, after translations, each of the x_i has type $\bullet_i \to \bullet_{i+1}$, hence all of these variables appear in A and hence

$$\mathrm{Var}(t^\flat) \cup \mathrm{Var}(A^\flat) = \mathrm{Var}(t) \cup \mathrm{Var}(A) \cup \{\bullet_0, \dots, \bullet_{l-1}\}$$

We can perform a symmetric reasoning for $\mathrm{Var}(u) \cup \mathrm{Var}(A)$. If we consider the type $\overline{\Delta} \vdash \star$ satisfying ($\mathrm{C}_{\mathsf{mop}'}$), we check directly that $\mathrm{Var}(\partial^-(\overline{\Delta}^\flat)) = \{\bullet_0, \bullet_{l-1}\} = \mathrm{Var}(\star^\flat)$. The case for a term $\overline{\Delta} \vdash A$ satisfying ($\mathrm{C}_{\mathsf{mcoh}'}$) is similar to the first case. □

Lemma 153. *The operation $\_^\flat$ respects the action of monoidal substitutions: Given a monoidal substitution $\overline{\gamma}$,*

- *for any type $\overline{\Gamma} \vdash A$ which is $(0, \ell(\overline{\Gamma}))$-full and substitution $\overline{\Delta} \vdash \overline{\gamma} : \overline{\Gamma}$ which is (i,j)-full, we have $A[\overline{\gamma}]^\flat_{i,j} = A^\flat_{0,\ell(\overline{\Gamma})}[\overline{\gamma}^\flat_{i,j}]$*
- *for any term t which is (a,b)-full in $\overline{\Gamma}$, we have $t[\overline{\gamma}]^\flat_{i_a,j_b} = t^\flat_{a,b}[\overline{\gamma}^\flat_{i,j}]$, where i_a is the minimum index of the variables of $\overline{\Delta}$ that appears in $\overline{\gamma}$ after the index a and j_b is the maximal index of the variables of $\overline{\Delta}$ appearing in $\overline{\gamma}$ before the index b.*
- *for any monoidal substitution $\overline{\xi}$ which is (a,b)-full in $\overline{\Gamma}$ we have $(\overline{\xi} \circ \overline{\gamma})^\flat_{i_a,j_b} = \overline{\xi}^\flat_{a,b} \circ \overline{\xi}^\flat_{i,j}$ where i_a and j_b are defined as in the previous case*

Proof. We prove this by mutual induction

Induction for monoidal types:

- For the type $\star$, we have

$$\begin{aligned}(\star[\overline{\gamma}])^\flat_{i,j} &= \bullet_i \underset{\star}{\to} \bullet_j\\ \star^\flat_{0,\ell(\overline{\Gamma})}[\overline{\gamma}^\flat_{i,j}] &= \bullet_0[\overline{\gamma}^\flat_{i,j}] \underset{\star}{\to} \bullet_{\ell(\overline{\Gamma})}[\overline{\gamma}^\flat_{i,j}]\end{aligned}$$

  Since $\overline{\gamma}$ is (i,j)-full, we have $(\bullet_0 \mapsto \bullet_i) \in \overline{\gamma}^\flat_{i,j}$ and $(\bullet_{\ell(\overline{\Gamma})} \mapsto \bullet_j) \in \overline{\gamma}^\flat_{i,j}$, which proves the equality.

- For the type $t \underset{A}{\longrightarrow} u$, we have

$$((t \underset{A}{\longrightarrow} u)[\overline{\gamma}])^\flat_{i,j} = t[\overline{\gamma}]^\flat_{i,j} \xrightarrow[A[\overline{\gamma}]^\flat_{i,j}]{} u[\overline{\gamma}]^\flat_{i,j}$$

$$(t \underset{A}{\longrightarrow} u)^\flat_{0,\ell(\overline{\Gamma})}[\overline{\gamma}^\flat_{i,j}] = t^\flat_{0,\ell(\overline{\Gamma})}[\overline{\gamma}^\flat_{i,j}] \xrightarrow[A^\flat_{0,\ell(\overline{\Gamma})}[\overline{\gamma}^\flat_{i,j}]]{} u^\flat_{0,\ell(\overline{\Gamma})}[\overline{\gamma}^\flat_{i,j}]$$

  By Lemma 151, A, t and u are all $(0, \ell(\overline{\Gamma}))$-full, hence the induction applies and shows the equality.

Induction for monoidal terms:

- For a variable term x, x being (a,b)-full necessarily implies that $b = a+1$. Denote $t = x[\overline{\gamma}]$, then we have $(x \mapsto t) \in \overline{\gamma}$. By definition of i_a and j_b, we then have so necessarily $(x \mapsto t^\flat_{i_a,j_b}) \in \overline{\gamma}^\flat_{i,j}$. This shows that $t^\flat_{i_a,j_b} = x[\overline{\gamma}^\flat_{i,j}]$.
- For a term of the form $\overline{\Gamma} \vdash \mathsf{mop}'_{\Xi,A}[\overline{\xi}]$, we have

$$\mathsf{mop}'_{\Xi,A}[\overline{\xi} \circ \overline{\gamma}]^\flat_{i_a,j_b} = \mathsf{op}_{\Xi^\flat,A^\flat}[(\overline{\xi} \circ \overline{\gamma})^\flat_{i_a,j_b}]$$
$$(\mathsf{mop}'_{\Xi,A}[\overline{\xi}])^\flat_{0,\ell(\overline{\Gamma})}[\overline{\gamma}^\flat_{i,j}] = \mathsf{op}_{\Xi^\flat,A^\flat}[\overline{\xi}^\flat_{0,\ell(\overline{\Gamma})} \circ \overline{\gamma}^\flat_{i,j}]$$

  and by induction this shows that the equality
- The case for the term $\overline{\Gamma} \vdash \mathsf{mcoh}'_{\xi,A}[\overline{\xi}]$ is identical.

Induction for monoidal substitutions:

- For a substitution of the form $[]$, the (a,b)-fullness condition implies that $\overline{\Gamma} = []$ and $a = b$, and hence $\overline{\gamma} = []$ and $i = j$, then we have

$$([] \circ [])^\flat_{i,i} = \langle \bullet_a \mapsto \bullet_i \rangle$$
$$[]^\flat_{a,a} \circ []^\flat_{i,i} = \langle \bullet_a \mapsto \bullet_a \rangle \circ \langle \bullet_a \mapsto \bullet_i \rangle$$

  Hence these two expressions are equal.
- For a substitution of the form $[\overline{\xi};_{\overline{\Gamma}'} \langle\rangle]$,

$$([\overline{\xi};_{\overline{\Gamma}'} \langle\rangle] \circ \overline{\gamma})^\flat_{i_a,j_b} = \langle (\overline{\xi} \circ \overline{\gamma})^\flat_{i_a,j_b}, \bullet_{\ell(\overline{\xi})} \mapsto \bullet_{j_b} \rangle$$
$$[\overline{\xi}_{\overline{\Gamma}'} \langle\rangle]^\flat_{a,b} \circ \overline{\gamma}^\flat_{i,j} = \langle \overline{\xi}^\flat_{a,b} \circ \overline{\gamma}^\flat_{i,j}, \bullet_{\ell(\overline{\xi})} \mapsto \bullet_b[\overline{\gamma}^\flat_{i,j}] \rangle$$

  By Lemma 151, the induction applies, and by definition of j_b we have $\bullet_b[\overline{\gamma}^\flat_{i,j}] = \bullet_{j_b}$. This implies the equality.
- For a substitution of the form $[\overline{\xi};_{\overline{\Gamma}'} \langle \xi, x \mapsto t \rangle]$, we have the equalities

$$([\overline{\xi};_{\overline{\Gamma}'} \langle \xi, x \mapsto t \rangle] \circ \overline{\gamma})^\flat_{i_a,j_b} = \langle [\overline{\xi} \circ \overline{\gamma}; \xi \circ \overline{\gamma}]^\flat_{i,j}, x \mapsto t[\overline{\gamma}]^\flat_{z,j_b} \rangle$$
$$([\overline{\xi};_{\overline{\Gamma}'} \langle \xi, x \mapsto t \rangle]^\flat_{a,b} \circ \overline{\gamma}) = \langle [\overline{\xi}; \xi]^\flat_{a,b} \circ \overline{\gamma}, x \mapsto t^\flat_{a+\ell(\overline{\Gamma}')+1,b}[\overline{\gamma}_{i,j}] \rangle$$

  where $\overline{\gamma}$ is of the form $[\_;_{\overline{\Delta}'} \_]$. We admit that the index z is the appropriate index corresponding to $a + \ell(\overline{\Gamma}') + 1$ for the induction for terms applies. The induction for substitutions also applies and gives the result.

□

Lemma 154. *The general flattening operation preserves the derivability of monoidal judgment in a monoidal ps-context: For a monoidal ps-context $\overline{\Delta}$ with two indices $0 \leq i \leq j \leq \ell(\overline{\Delta})$, the following hold*

- *For any type $\overline{\Delta} \vdash A$, which is (i,j)-full, we have the derivation of $\overline{\Delta}^\flat \vdash A^\flat_{i,j}$*
- *For any term $\overline{\Delta} \vdash t : A$ which is (i,j)-full, we have a derivation of $\overline{\Delta}^\flat \vdash t^\flat_{i,j} : A^\flat_{i,j}$*

– *For any substitution* $\overline{\Delta} \vdash \overline{\gamma} : \overline{\Gamma}$ *which is* (i,j)*-full we have a derivation of* $\overline{\Delta}^\flat \vdash \overline{\gamma}^\flat_{i,j} : \overline{\Gamma}^\flat$

Proof. First note that by hypothesis, we have $\overline{\Delta} \vdash_{\mathsf{ps}}$, hence by Lemma 147 and Proposition 31 give a derivation of $\overline{\Delta}^\flat \vdash$ We prove these results by mutual induction.

Induction for types:

– For the type $\overline{\Delta} \vdash \star$, since $i, j \leq \ell(\overline{\Delta})$, we have $(\bullet_i : \star) \in \overline{\Delta}^\flat$ and $(\bullet_j : \star) \in \overline{\Delta}^\flat$, hence rule (VAR) applies twice and gives derivations of $\overline{\Delta}^\flat \vdash \bullet_i : \star$ and $\overline{\Delta}^\flat \vdash \bullet_j : \star$. We can then apply the rule ($\to$-INTRO) to give a derivation of $\overline{\Delta}^\flat \vdash \bullet_i \underset{\star}{\to} \bullet_j$.

– For the type $\overline{\Delta} \vdash t \underset{A}{\to} u$, we have a derivation of $\overline{\Delta} \vdash A$, of $\overline{\Delta} \vdash t : A$ and of $\overline{\Delta} \vdash u : A$, and since the type is (i,j)-full, so are A, t and u be Lemma 151. Then, by the induction cases for types and terms a derivation of $\overline{\Delta}^\flat \vdash A^\flat_{i,j}$, of $\overline{\Delta}^\flat \vdash t^\flat_{i,j} : A^\flat_{i,j}$ and of $\overline{\Delta}^\flat \vdash u^\flat_{i,j} : A^\flat_{i,j}$. These let us apply the rule ($\to$-INTRO) to give a derivation of $\overline{\Delta}^\flat \vdash t^\flat \underset{A^\flat}{\to} u^\flat$.

Induction for terms:

– For a variable term $\overline{\Delta} \vdash x : A$, since we have $\overline{\Delta}^\flat \vdash$, it suffices to check that x belongs to $\overline{\Delta}^\flat$ with the appropriate type. Since we have $(x : A) \in \overline{\Delta}$, there exists a index k such that $(x : A)$ belongs the to context in k-th position in the list $\overline{\Delta}$, and then x being (i,j)-full implies that $i = k$ and $j = k+1$. Then by definition of the operation $\overline{\Delta}^\flat$, the association $(x : A^\flat_{k,k+1}) \in \overline{\Delta}^\flat$, and hence the rule (VAR) applies to give a derivation of $\overline{\Delta}^\flat \vdash x : A^\flat_{k,k+1}$.

– For the term $\overline{\Delta} \vdash \mathsf{mop}'_{\overline{\Gamma},A}[\overline{\gamma}] : A[\overline{\gamma}]$, since A satisfies (C_{op}) it is necessarily $(0, \ell(\overline{\Delta}))$-full and we have by the induction case for types that $\overline{\Gamma}^\flat \vdash A^\flat$, and by Lemma 152, $\overline{\Gamma}^\flat, A^\flat$ satisfy (C_{op}). Moreover, we have a derivation of $\overline{\Delta} \vdash \overline{\gamma} : \overline{\Gamma}$, and the condition, and since $\mathrm{Var}(()\mathsf{mop}'_{\overline{\Gamma},A}[\gamma]) = \mathrm{Var}(\gamma)$, γ is (i,j)-full, so by the induction case for substitutions, we have a derivation of $\overline{\Delta}^\flat \vdash \gamma^\flat_{i,j} : \overline{\Gamma}^\flat$. The rule (MOP') then applies to give a derivation of $\overline{\Delta}^\flat \vdash \mathsf{op}_{\overline{\Gamma}^\flat, A^\flat} : A^\flat[\overline{\gamma}^\flat_{i,j}]$. By Lemma 153, we also have $A^\flat[\overline{\gamma}^\flat_{i,j}] = (A[\overline{\gamma}])^\flat_{i,j}$ which gives the result.

– The case for a term $\overline{\Delta} \vdash \mathsf{mcoh}'_{\overline{\Gamma},A}[\overline{\gamma}] : A[\overline{\gamma}]$ is exactly similar: Necessarily $\overline{\Gamma}^\flat \vdash A^\flat$ is derivable and satisfies ($\mathrm{C}_{\mathsf{coh}}$), and $\overline{\Delta}^\flat \vdash \overline{\gamma}^\flat : \Gamma^\flat$ is derivable, thus the rule (MCOH') applies, and Lemma 153 gives the result.

Induction for substitutions:

– For the empty substitution $\overline{\Delta} \vdash [] : []$. Since the substitution $[]$ has no variable and is (i,j)-full, it implies $i = j < \ell(\overline{\Delta})$, so we have $(\bullet_i : \star) \in \overline{\Delta}^\flat$. Since we also have a derivation of $\overline{\Delta}^\flat \vdash$, the rule (VAR) gives a derivation of $\overline{\Delta}^\flat \vdash \bullet_0 : \star$, and we can then construct a derivation of $\overline{\Delta}^\flat \vdash []^\flat_{0,0} : []^\flat$ as follows

$$\dfrac{\dfrac{\overline{\Delta} \vdash}{\overline{\Delta} \vdash \langle\rangle : \varnothing}(\mathrm{ES}) \qquad (\bullet_0 : \star) \vdash \qquad \overline{\Delta} \vdash \bullet_0 : \star}{\overline{\Delta} \vdash \langle \bullet_0 \mapsto \bullet_0 \rangle : (\bullet_0 : \star)}(\mathrm{SE})$$

– For the substitution $\overline{\Delta} \vdash [\overline{\gamma}; \langle x \mapsto t \rangle] : [\overline{\Gamma}; \overline{\Delta}' \, (x : \star)]$, we denote l for $\ell(\overline{\Delta}')$ since by hypothesis this substitution is (i,j)-full, Lemma 151 shows that $\overline{\gamma}$ is (i,l)-full, and that t is (l,j)-full in $\overline{\Delta}$. Hence by the induction case for substitutions, we get a derivation of $\overline{\Delta}^\flat \vdash \overline{\gamma}^\flat_{i,l} : \overline{\Gamma}^\flat$. For simplification, we denote $k = \ell(\overline{\Gamma})$. Since by definition $j < \ell(\overline{\Delta})$, we necessarily have $(\bullet_j : \star) \in \overline{\Delta}$, and by the induction case on context we have a derivation of $\overline{\Delta}^\flat \vdash$. This lets us build a derivation as follows

$$\frac{\overline{\Delta}^\flat \vdash \overline{\gamma}^\flat_{i,l} : \overline{\Gamma}^\flat \qquad (\overline{\Gamma}^\flat, \bullet_k : \star) \vdash \qquad \dfrac{\overline{\Delta}^\flat \vdash \qquad (\bullet_j : \star) \in \overline{\Delta}}{\overline{\Delta}^\flat \vdash \bullet_j : \star}\,(\text{VAR})}{\overline{\Delta}^\flat \vdash \langle \overline{\gamma}^\flat_{i,l}, \bullet_k \mapsto \bullet_j \rangle : (\overline{\Gamma}^\flat, \bullet_k : \star)}\,(\text{SE})$$

Moreover, the induction case for terms applies and gives a derivation of $\overline{\Delta}^\flat \vdash t^\flat_{l,j} : \bullet_l \to \bullet_j$, and since we also have $\bullet_{k-1}[\langle \overline{\gamma}^\flat_{i,l}, \bullet_k \mapsto \bullet_j \rangle] = \bullet_l$ and $\bullet_k[\langle \overline{\gamma}^\flat_{i,l}, \bullet_k \mapsto \bullet_j \rangle] = \bullet_j$, this lets us build the following derivation

$$\frac{\overline{\Delta} \vdash \langle \overline{\gamma}^\flat_{i,l}, \bullet_k \mapsto \bullet_j \rangle : (\overline{\Gamma}^\flat, \bullet_k : \star) \qquad (\overline{\Gamma}^\flat, \bullet_k : \star, x : \bullet_{k-1} \underset{\star}{\to} \bullet_k) \vdash \qquad \overline{\Delta}^\flat \vdash t^\flat_{l,j} : \bullet_l \underset{\star}{\to} \bullet_j}{\overline{\Delta}^\flat \vdash \langle \overline{\gamma}^\flat_{i,l}, \bullet_k \mapsto \bullet_l, x \mapsto t^\flat_{l,j} \rangle : (\overline{\Gamma}^\flat, \bullet_k : \star, x : \bullet_{k-1} \underset{\star}{\to} \bullet_k)}$$

– For the substitution $\overline{\Delta} \vdash [\overline{\gamma}; \overline{\Delta}' \, \langle \gamma, y \mapsto t, f \mapsto u \rangle] : [\overline{\Gamma}; (\Gamma, y : A, f : x \underset{A}{\to} y)]$, denote $l = \ell(\overline{\Delta})$, then since the substitution is (i,j)-full, so is the substitution $[\overline{\gamma}; \gamma]$, and moreover, the terms t and y are necessarily (l,j)-full by Lemma 151. Hence, by the induction case for substitutions, we have a derivation of $\overline{\Delta}^\flat \vdash [\overline{\gamma}; \gamma]^\flat : [\overline{\Gamma}; \Gamma]^\flat$, the induction case for terms gives a derivation of $\overline{\Delta}^\flat \vdash t^\flat_{l,j} : (A[\overline{\gamma}; \gamma])^\flat_{l,j}$, and one can check that $(A[\overline{\gamma}; \gamma])^\flat_{l,j} = (\Sigma^{k-1,k} A)[[\overline{\gamma}; \gamma]^\flat_{i,j}]$, this lets us build the following derivation

$$\frac{\overline{\Delta}^\flat \vdash [\overline{\gamma}; \gamma]^\flat_{i,j} : [\overline{\Gamma}; \Gamma]^\flat \qquad [\overline{\Gamma}; \Gamma]^\flat, y : \Sigma^{k-1,k} A \vdash \qquad \overline{\Delta}^\flat \vdash t^\flat_{l,j} : A[\overline{\gamma}; \gamma]^\flat_{l,j}}{\overline{\Delta}^\flat \vdash \langle [\overline{\gamma}; \gamma]^\flat_{i,j}, y \mapsto t^\flat_{l,j} \rangle : ([\overline{\Gamma}; \Gamma]^\flat, y : \Sigma^{k-1,k} A)}\,(\text{SE})$$

The induction case for terms also provides a derivation of $\overline{\Delta}^\flat \vdash u^\flat_{l,j} : t'^\flat_{l,j} \xrightarrow[A[\overline{\gamma}:\gamma]^\flat_{l,j}]{} t^\flat_{l,j}$, where $t' = x[\overline{\gamma}; \gamma]$. We then again use the equality $(\Sigma^{k-1,k} A)[[\overline{\gamma}; \gamma]^\flat_{i,j}] = A[\overline{\gamma}; \gamma]^\flat_{l,j}$ to get the following derivation

$$\frac{\overline{\Delta}^\flat \vdash \langle [\overline{\gamma}; \gamma]^\flat_{i,j}, y \mapsto t^\flat_{l,j} \rangle : \Gamma_y \qquad \Gamma_f \vdash \qquad \overline{\Delta}^\flat \vdash u^\flat_{l,j} : t'^\flat_{l,j} \xrightarrow[A[\overline{\gamma};\gamma]^\flat_{l,j}]{} t^\flat_{l,j}}{\overline{\Delta}^\flat \vdash \langle [\overline{\gamma}; \gamma]^\flat_{i,j}, y \mapsto t^\flat_{l,j}, f \mapsto u^\flat_{l,j} \rangle : \Gamma_f}$$

Where we denote Γ_y to be the context $([\overline{\Gamma}; \Gamma]^\flat, y : \Sigma^{k-1,k} A)$ and Γ_f to be the context $([\overline{\Gamma}; \Gamma]^\flat, y : \Sigma^{k-1,k} A, f : x \xrightarrow[\Sigma^{k-1,k} A]{} y)$

□

Note that this mutual induction is not structurally well-formed, as it requires to prove the induction hypothesis for any type in any given monoidal ps-context in the case of terms. However

this always decreases the depth of the terms, hence we can layer this induction by depth and dimension, in order to prove that it is well-formed, similarly to the technique we have used in Proposition 43 Importantly, along this proof, we have showed that given a monoidal type $\overline{\Gamma} \vdash A$ satisfying ($\mathrm{C}_{\mathsf{mop}'}$) (resp. ($\mathrm{C}_{\mathsf{mcoh}'}$)), we have a type $\overline{\Gamma}^\flat \vdash A^\flat$ satisfying (C_{op}) (resp. ($\mathrm{C}_{\mathsf{coh}}$)). Moreover, it is immediate from the inductive definition that given two monoidal types $\overline{\Gamma} \vdash A$ and $\overline{\Gamma} \vdash B$ both satisfying ($\mathrm{C}_{\mathsf{mop}'}$) or ($\mathrm{C}_{\mathsf{coh}}$), if $A^\flat = B^\flat$, then $A = B$, hence $\_^\flat$ defines a injection from moniodal types in $\overline{\Gamma}$ satisfying ($\mathrm{C}_{\mathsf{mop}'}$) (resp. satisfying ($\mathrm{C}_{\mathsf{mcoh}'}$)) to types in $\overline{\Gamma}^\flat$ satisfying (C_{op}) (resp. satisfying ($\mathrm{C}_{\mathsf{coh}}$)).

The general folding operation. We now show that this association in fact defines a bijection, that is we consider a type $\overline{\Delta}^\flat \vdash A$ satisfying (C_{op}) (resp. ($\mathrm{C}_{\mathsf{coh}}$)), there exists a monoidal type $\overline{\Delta} \vdash B$ satisfying ($\mathrm{C}_{\mathsf{mop}'}$) (resp. ($\mathrm{C}_{\mathsf{mcoh}'}$)) such that $B^\flat = A$. We call this type A the *folding* of A and denote it $A^\sharp$, and we define it by induction as follows

$$\star^\sharp = \star \qquad (t \underset{\star}{\to} u)^\sharp = \star \qquad (t \underset{A}{\to} u)^\sharp = t^\sharp \underset{A^\sharp}{\longrightarrow} u^\sharp$$

$$x^\sharp = x \qquad (\mathsf{op}_{\Gamma,A}[\gamma]) = \mathsf{mop}'_{\Gamma^\sharp,A^\sharp}[\gamma^\sharp] \qquad (\mathsf{coh}_{\Gamma,A}[\gamma])^\sharp = \mathsf{mcoh}'_{\Gamma^\sharp,A^\sharp}[\gamma^\sharp]$$

For the substitutions, we only define this operation in the case of a derivable whose target is a ps-context

For the substitution $\Delta \vdash \langle x \mapsto t\rangle : (x : \star)$:

$$\langle x \mapsto t\rangle^\sharp = []$$

For the substitution $\Delta \vdash \langle \gamma, y \mapsto t, f \mapsto u\rangle : (\Gamma, y : A, f : x \underset{A}{\to} y)$

$$\langle \gamma, x \mapsto t\rangle^\sharp = \begin{cases} [\gamma^\sharp ;_{\Delta_{\leq t}} \langle f \mapsto u^\sharp\rangle] & \text{if } A = \star \\ \langle \gamma^\sharp, y \mapsto t^\sharp, f \mapsto u^\sharp\rangle & \text{otherwise} \end{cases}$$

where in the first case, t is a term of dimension 0, hence it is necessarily a variable of Δ, and we denote $\Delta_{\leq t}$ the context obtained from Δ by removing all the variables that are after t.

Lemma 155. *We have the following results relating the introduction rules for the terms in* CaTT *with the introduction rules for the terms in* MCaTT′

- *For any type $\Gamma \vdash A$ derivable in a ps-context Γ in the theory* CaTT *satisfying* (C_{op})*, the monoidal ps-context $\Gamma^\sharp$ and the type $A^\sharp$ satisfy* ($\mathrm{C}_{\mathsf{mop}'}$)*.*
- *For any type $\Gamma \vdash A$ derivable in a ps-context Γ in the theory* CaTT *satisfying* ($\mathrm{C}_{\mathsf{coh}}$)*, the monoidal ps-context $\Gamma^\sharp$ and the type $A^\sharp$ satisfy* ($\mathrm{C}_{\mathsf{mcoh}'}$)*.*

Proof. We separate the two cases, for the side conditions of the two term constructors.

- Consider a ps-context $\Gamma \vdash$ together with a type $\Gamma \vdash A$ satisfying (C_{op}) in the theory CaTT. Then by definition of (C_{op}), we necessarily have $A = t \underset{B}{\to} u$.

- If $B = \star$, then $A^\sharp = \star$. Moreover, necessarily t and u are derivable and of type $\star$ in Γ, hence t and u are variables. The condition (C_{op}) then becomes $\mathrm{Var}(\partial^-(\Gamma)) = \{t\}$ and $\mathrm{Var}(\partial^+(\Gamma)) = \{u\}$, hence $\partial^-(\Gamma)$ and $\partial^+(\Gamma)$ are of dimension 0, so Γ is of dimension 1, and thus $\Gamma^\sharp$ is of dimension 0 in MCaTT'. Hence $(\Gamma^\sharp, A^\sharp)$ satisfy the condition ($\mathrm{C}_{\mathsf{mop}'}$)
- If B is not the type $\star$, then $A^\sharp = t^\sharp \underset{B^\sharp}{\longrightarrow} u^\sharp$. Then since $\partial^-(\Gamma^\sharp) = (\partial^-(\Gamma))^\sharp$, we have $\mathrm{Var}(\partial^-(\Gamma^\sharp)) = \mathrm{Var}(\partial^-(\Gamma)) - V_0$, where V_0 is the set of variables of dimension 0 in the context Γ. Similarly, $\mathrm{Var}(\partial^+(\Gamma^\sharp)) = \mathrm{Var}(\partial^+(\Gamma)) - V_0$, and $\mathrm{Var}(B^\sharp) = \mathrm{Var}(B) - V_0$, $\mathrm{Var}(t^\sharp) = \mathrm{Var}(t) - V_0$, $\mathrm{Var}(u^\sharp) = \mathrm{Var}(u) - V_0$. The equalities between sets of variables given by (C_{op}) then gives the equalities $\mathrm{Var}(t^\sharp) \cup \mathrm{Var}(B^\sharp) = \mathrm{Var}(\partial^-(\Gamma^\sharp))$ and $\mathrm{Var}(u^\sharp) \cup \mathrm{Var}(B^\sharp) = \mathrm{Var}(\partial^+(\Gamma^\sharp))$, which shows the condition ($\mathrm{C}_{\mathsf{mop}'}$).

– Consider a ps-context $\Gamma \vdash$ together with a type $\Gamma \vdash A$ satisfying ($\mathrm{C}_{\mathsf{coh}}$), then we can decompose $A = t \underset{B}{\longrightarrow} u$.

- If $B = \star$ then the condition ($\mathrm{C}_{\mathsf{coh}}$) imply that both t and u are the same variable x, and that Γ is actually the ps-context $(x : \star)$. In this case, the folded judgment is $[] \vdash \star$, which by definition satisfies ($\mathrm{C}_{\mathsf{coh}}$).
- If B is not the type $\star$, then we have the equalities between the set of variables

$$\mathrm{Var}(\Gamma^\sharp) = \mathrm{Var}(\Gamma) - V_0 \qquad \mathrm{Var}(B^\sharp) = \mathrm{Var}(B) - V_0$$
$$\mathrm{Var}(t^\sharp) = \mathrm{Var}(t) - V_0 \qquad \mathrm{Var}(u^\sharp) = \mathrm{Var}(u) - V_0$$

where V_0 is the set of variable of dimension 0 in Γ, The condition ($\mathrm{C}_{\mathsf{coh}}$) then implies the equalities $\mathrm{Var}(t^\sharp) \cup \mathrm{Var}(B^\sharp) = \mathrm{Var}(\Gamma^\sharp)$ and $\mathrm{Var}(u^\sharp) \cup \mathrm{Var}(\Gamma^\sharp)$, which imply ($\mathrm{C}_{\mathsf{mcoh}'}$).

□

In order to prove the correctness of the folding operation, we introduce a notion that is analogous to the (i, j)-fullness of a monoidal type, for a regular type in a context of the form $\overline{\Delta}^\flat$. We say that a type $\overline{\Delta} \vdash A$ (resp. a term $\overline{\Delta}^\flat \vdash t : A$ or a substitution $\overline{\Delta}^\flat \vdash \gamma : \Gamma$) is *full* if in contains at least a variable from each of the contexts in $\overline{\Delta}$. Contrarily to our prior convention, the type $\star$ is now not full in any context.

Lemma 156. *The fullness condition allows to perform induction, more precisely, we have*

- *The type $\overline{\Delta}^\flat \vdash t \underset{A}{\longrightarrow} u$ with A distinct from $\star$ is full if and only if A, t and u are full.*
- *The substitution $(\overline{\Delta}_{\leq k} @ \overline{\Delta}_{>k})^\flat \vdash [\overline{\gamma};_{\overline{\Delta}_{\leq k}} \gamma] : \Gamma$ is full if and only if $\overline{\Gamma}$ is full and all the terms of Gg are full for $\overline{\Delta}_{>k}$*

Proof. If the type $t \underset{A}{\longrightarrow} u$, then it contains in particular a variable of dimension 0 in $\overline{\Delta}$ which for each of the contexts in $\overline{\Delta}$. these variables are of dimension 1 in $\overline{\Delta}^\flat$ and since by assumption A is not $\star$, they are necessarily of degree strictly positive, hence they appear in $\mathrm{Var}(A)$, $\mathrm{Var}(t)$ and $\mathrm{Var}(u)$. Similarly, If the substitution $(\overline{\Delta}_{\leq k} @ \overline{\Delta}_{>k})^\flat \vdash [\overline{\gamma};_{\overline{\Delta}_{\leq k}} \gamma] : \Gamma$ is full in $\overline{\Delta}_{\leq k}$, then since all the variables in $\mathrm{Var}(\gamma)$ are in $\overline{\Delta}_{>k}$, necessarily $\overline{\gamma}$ contains at least a variable from each of the contexts in $\overline{\Delta}_{\leq k}$, so it is full for this context. Moreover, since γ contains at least a variable from each of the context in $\overline{\Delta}_{>k}$, it has to contain the variables $\bullet_k$ and $\bullet_{\ell(\overline{\Delta})}$ and these are the only variables of dimension 0 possible in γ. Hence all the terms from γ have to contain a math between these variables, and such a path is necessarily full in $\overline{\Delta}_{>k}$ □

Lemma 157. *The folding respects the action of substitutions. More precisely, given a substitution* $\Delta \vdash \gamma : \Gamma$

- *For every type* $\Gamma \vdash A$*, we have* $A[\gamma]^\sharp = A^\sharp[\gamma^\sharp]$
- *For every term* $\Gamma \vdash t : A$ *of dimension non-zero we have* $t[\gamma]^\sharp = t^\sharp[\gamma^\sharp]$.
- *For every substitution* $\Gamma \vdash \xi : \Xi$ *whose target is a ps-context, we have* $(\xi \circ \gamma)^\sharp = \xi^\sharp \circ \gamma^\sharp$.

Proof. We prove these by mutual induction

Induction for types:

- For the type $\Gamma \vdash \star$, we have

$$\star[\gamma]^\sharp = \star$$
$$\star[\gamma^\sharp] = \star$$

- For the type $\Gamma \vdash t \underset{\star}{\to} u$, we have

$$((t \underset{\star}{\to} u)[\gamma])^\sharp = \star$$
$$(t \underset{\star}{\to} u)^\sharp[\gamma] = \star$$

- For the type $\Gamma \vdash t \underset{A}{\longrightarrow} u$ with A distinct from $\star$ we have

$$((t \underset{A}{\to} u)[\gamma])^\sharp = t[\gamma]^\sharp \xrightarrow[A[\gamma]^\sharp]{} u[\gamma]^\sharp$$
$$(t \underset{A}{\to} u)^\sharp[\gamma^\sharp] = t^\sharp[\gamma^\sharp] \xrightarrow[A^\sharp[\gamma^\sharp]]{} u^\sharp[\gamma^\sharp]$$

  and the result is given by induction, since the target of ξ is the ps-context Ξ

Induction for terms:

- For a variable term $\Gamma \vdash x : A$, denote $t = x[\gamma]$, such that $(x \mapsto t) \in \gamma$. Then since x is of dimension non-zero, we necessarily have $(x \mapsto t^\sharp) \in \gamma^\sharp$, which proves that $t^\sharp = x[\gamma^\sharp]$.
- For a term of the form $\Gamma \vdash \mathsf{op}_{\Xi,A}[\xi] : A[\xi]$, we have

$$(\mathsf{op}_{\Xi,A}[\xi][\gamma])^\sharp = \mathsf{mop}'_{\Xi^\sharp,A^\sharp}[(\xi \circ \gamma)^\sharp]$$
$$(\mathsf{op}_{\Xi,A}[\xi])^\sharp[\gamma^\sharp] = \mathsf{mop}'_{\Xi;A}[\xi^\sharp][\gamma^\sharp]$$

  And the equality comes from the induction.
- The case for a term of the from $\mathsf{coh}_{\Xi,A}[\xi]$ is exactly identical.

Induction for substitutions:

- For the substitution $\Gamma \vdash \langle x \mapsto t \rangle : (x : \star)$, we have

$$(\langle x \mapsto t \rangle \circ \gamma)^\sharp = []$$
$$\langle x \mapsto t \rangle^\sharp \circ \gamma^\sharp = []$$

– For the substitution $\Gamma \vdash \langle \xi, y \mapsto t, f \mapsto u\rangle : (\Xi, y : \star, f : x \to y)$, we have

$$(\langle \xi, y \mapsto t, f \mapsto u\rangle \circ \gamma)^\sharp = [(\xi \circ \gamma)^\sharp; \langle f \mapsto u[\gamma]^\sharp\rangle]$$
$$\langle \xi, y \mapsto t, f \mapsto u\rangle^\sharp \circ \gamma^\sharp = [\xi^\sharp \circ \gamma^\sharp; f \mapsto u^\sharp[\gamma^\sharp]]$$

The induction for substitutions and for terms then gives the equality.

– For the substitution $\Gamma \vdash \langle \xi, y \mapsto t, f \mapsto u\rangle : (\Xi, y : A, f : x \to y)$ with A distinct from $\star$, we have

$$(\langle \xi, y \mapsto t, f \mapsto u\rangle \circ \gamma)^\sharp = \langle (\xi \circ \gamma)^\sharp, y \mapsto t[\gamma]^\sharp, f \mapsto u[\gamma]^\sharp\rangle$$
$$\langle \xi, y \mapsto t, f \mapsto u\rangle^\sharp \circ \gamma^\sharp = \langle \xi^\sharp \circ \gamma^\sharp, y \mapsto t^\sharp[\gamma^\sharp], f \mapsto u^\sharp[\gamma^\sharp]\rangle$$

The induction cases for substitutions and terms then give the equality.

□

Lemma 158. *The folding operation preserves the derivability of the full judgments in ps-contexts.*

– *For any full type* $\overline{\Delta}^\flat \vdash A$*, we have a derivation of* $\overline{\Delta} \vdash A^\sharp$

– *For any full term* $\overline{\Delta}^\flat \vdash t : A$*, we have a derivation of* $\overline{\Delta} \vdash t^\sharp : A^\sharp$

– *For any full substitution* $\overline{\Delta}^\flat \vdash \gamma : \Gamma$*, such that* Γ *is a ps-context, we have a derivation of* $\overline{\Delta} \vdash \gamma^\sharp : \Gamma^\sharp$*.*

Proof. We suppose given a monoidal ps-contexts $\overline{\Delta} \vdash_{\mathsf{ps}}$, and proceed by mutual induction

Induction for types: Note that the type $\star$ is not full, so the induction is in fact the following

– For the type $\overline{\Delta}^\flat \vdash t \underset{\star}{\to} u$, we have $A^\sharp = \star$. Since we have a derivation of $\overline{\Delta} \vdash$ the rule ($\star$-INTRO) gives a derivation of $\overline{\Delta} \vdash \star$.

– For the type $\overline{\Delta}^\flat \vdash t \underset{A}{\to} u$ with A distinct from $\star$, we necessarily have derivations of the judgments $\overline{\Delta}^\flat \vdash A$, $\overline{\Delta}^\flat \vdash t : A$ and $\overline{\Delta}^\flat \vdash u : A$. Since $t \underset{A}{\to} u$ are full by Lemma 156, so are A, t and u. Hence the induction case for types gives a derivation of $\overline{\Delta} \vdash A^\sharp$ and the induction case for terms provides a derivation of $\overline{\Delta} \vdash t^\sharp : A^\sharp$ and of $\overline{\Delta} \vdash u^\sharp : A^\sharp$. This lets us apply the rule ($\to$-INTRO) to get a derivation of $\overline{\Delta} \vdash t^\sharp \underset{A^\sharp}{\longrightarrow} u^\sharp$.

Induction for terms:

– For a variable term $\overline{\Delta} \vdash x : A$ which is full, necessarily $\overline{\Delta}$ is a single context $[\Delta]$, with $(x : B) \in \Delta$, such that $B^\flat_{0,1} = A$. Hence by definition of the folding, we have $A^\sharp = B$, and hence $(x : A^\sharp) \in \overline{\Delta}$. The rule (VAR) then applies to give a derivation of $\overline{\Delta} \vdash x : A^\sharp$.

– For the term $\overline{\Delta} \vdash \mathsf{op}_{\Gamma,A}[\gamma]$, then we necessarily have a derivation of $\Gamma \vdash A$, with A satisfying (C_{op}). Up to renaming of the variables, we have that ${\Gamma^\sharp}^\flat = \Gamma$, and we denote A' the type obtained from A by renaming the variables accordingly, so that we have a derivation of ${\Gamma^\sharp}^\flat \vdash A'$ satisfying (C_{op}). This implies in particular that A' is full for the monoidal ps-context $\Gamma^\sharp$, and hence the induction case for types gives a derivation of $\Gamma^\sharp \vdash A'^\sharp$, and by Lemma 155, this satisfies ($\mathrm{C}_{\mathsf{mop}'}$). Moreover, since the only variables

in A that were renamed to get A' are of dimension 0, we have that $A^\sharp = A'^\sharp$, hence we have in fact a derivation of $\Gamma^\sharp \vdash A^\sharp$. Moreover, since the variables of the terms are exactly the variables of γ, and the term is full, so is γ, hence the induction case for substitution gives a derivation of $\overline{\Delta} \vdash \gamma^\sharp : \Gamma^\sharp$. The rule (MOP') then applies and gives a derivation of $\overline{\Delta} \vdash \mathsf{mop}'_{\Gamma^\sharp, A^\sharp}[\gamma^\sharp] : A^\sharp[\gamma^\sharp]$. Then result is then given by Lemma 157

– The case for a term of the form $\mathsf{coh}_{\Gamma,A}[\gamma]$ is exactly identical.

Induction for substitutions:

– For the substitution $\overline{\Delta} \vdash \langle x \mapsto t \rangle : (x : \star)$,we have by definition $\langle x \mapsto t \rangle^\sharp = []$ and $(x : \star)^\sharp = []$. Then t is a term of dimension 0 in $\overline{\Delta}^\flat$, hence it is a variable, which cannot appear in $\overline{\Delta}$. The fullness condition then implies that $\overline{\Delta} = []$, and the rule (MES) gives a derivation of $[] \vdash [] : []$.

– For a substitution of the form $\overline{\Delta} \vdash \langle \gamma, y \mapsto t, f \mapsto u \rangle : (\Gamma, y : \star, f : x \underset{\star}{\to} y)$, we have the following equalities

$$\langle \gamma, y \mapsto \bullet_k, f \mapsto u \rangle^\sharp = [\gamma^\sharp ;_{\overline{\Delta}_{\leq k}} \langle f \mapsto u^\sharp \rangle]$$
$$(\Gamma, y : \star, f : x \underset{\star}{\to} y)^\sharp = [\Gamma^\sharp ; (f : \star)]$$

where we denote $\overline{\Delta}_{\leq k}$ the list constituted in the k first elements of the list $\overline{\Delta}$. The fullness condition implies that γ is full for $\overline{\Delta}^\flat_{\leq k}$, and we necessarily have a derivation of $\overline{\Delta}^\flat_{\leq k} \vdash \gamma$, hence the induction case for substitution gives a derivation of $\overline{\Delta}_{\leq k} \vdash \gamma^\sharp : \Gamma^\sharp$. Moreover, we have $\overline{\Delta} \vdash$, that we can split into $\overline{\Delta} = \overline{\Delta}_{\leq k} @ \overline{\Delta}_{>k}$, and we necessarily have $\overline{\Delta}_{>k} \vdash$, so we can build the following derivation

$$\frac{\overline{\Delta}_{\leq k} \vdash \gamma^\sharp : \Gamma : \sharp \qquad \overline{\Delta}_{>k}}{\overline{\Delta} \vdash [\gamma^\sharp ; \langle\rangle] : [\Gamma^\sharp ; \varnothing]} (\text{MS}+)$$

Moreover, we necessarily have a derivation of $\overline{\Delta}^\flat \vdash u : x[\gamma] \underset{\star}{\to} t$, and since it does not use variables of $\overline{\Delta}_k$, u is full in $\overline{\Delta}_{>k}$. More precisely, we can rename the variables of dimension 0 of u by the transformation $\bullet_n \rightsquigarrow \bullet_{n-k}$ and get the term u', which is such that $\overline{\Delta}^\flat_{>k} \vdash u'$ and u' is full. Applying the induction case for terms, we have a derivation of $\overline{\Delta}_{>k} \vdash u'^\sharp : \star$, and since $u'^\sharp = u^\sharp$, we have in fact a derivation of $\overline{\Delta}_{>k} \vdash u^\sharp : \star$. This lets us build the following derivation

$$\frac{\overline{\Delta} \vdash [\gamma^\sharp ;_{\overline{\Delta}_{\leq k}} \langle\rangle] \qquad [\Gamma^\sharp ; (f : \star)] \vdash \overline{\Delta}_{>k} \vdash u^\sharp : \star}{\overline{\Delta} \vdash [\gamma^\sharp ;_{\overline{\Delta}_{\leq k}} \langle f \mapsto u^\sharp \rangle] : [\Gamma^\sharp ; (f : \star)]} (\text{MSE})$$

– For a substitution of the form $\overline{\Delta} \vdash \langle \gamma, y \mapsto t, f \mapsto u \rangle : (\Gamma, y : A, f : x \underset{A}{\to} y)$, with A distinct from the type $\star$, we use a similar reasoning. We first decompose $\langle \gamma \rangle^\sharp$ as $[\overline{\gamma} ;_{\overline{\Delta}_{\leq k}} \gamma']$, and split the context $\overline{\Delta}$ into $\overline{\Delta} = \overline{\Delta}_{\leq k} @ \overline{\Delta}_{>k}$. Then we have a derivation of $\overline{\Delta}^\flat \vdash \gamma : \Gamma$ which is full by Lemma 156, hence the induction case for substitutions gives a derivation of $\overline{\Delta} \vdash \gamma^\sharp : \Gamma^\sharp$. Moreover, by Lemma 156 the terms t and u are also necessarily full for $\overline{\Delta}_{>k}$. Denote $B = A[\gamma]$, then we can perform the renaming of the variables of dimension 0 given by $\bullet_n \rightsquigarrow \bullet_{n-k}$ on the type B and the terms t and u in order to get the type B' and

the terms t' and u'. Then we have a derivation of $\overline{\Delta}_{>k} \vdash t' : B'$ which is full, and by the induction case for terms provides a derivation of $\overline{\Delta} \vdash t'^\sharp : B'^\sharp$. Since $t'^\sharp = t^\sharp$ and $B'^\sharp = B^\sharp$, and by Lemma 157, we have $B^\sharp = A^\sharp[\gamma^\sharp]$, so this in fact is a derivation of $\overline{\Delta} \vdash t^\sharp : A^\sharp[\gamma^\sharp]$. This lets us build a derivation as follows

$$\frac{\overline{\Delta} \vdash \gamma^\sharp : \Gamma^\sharp \qquad (\Gamma^\sharp, y : A) \vdash \qquad \overline{\Delta}_{>k} \vdash t^\sharp : A^\sharp[\gamma^\sharp]}{\overline{\Delta} \vdash \langle \gamma^\sharp, y \mapsto t^\sharp \rangle : (\Gamma^\sharp, y : A^\sharp)} (\text{MSE})$$

We can now iterate this construction: denote $s = x[\gamma]$, then we have a derivation of $\overline{\Delta}^\flat_{>k} \vdash u' : s' \underset{B'}{\longrightarrow} t'$ which is full, thus it provides by induction a derivation of the judgment $\overline{\Delta}_{>k} \vdash u'^\sharp : s'^\sharp \underset{B'^\sharp}{\longrightarrow} t'^\sharp$, which is also a derivation of $\overline{\Delta}_{>k} \vdash u^\sharp : x[\gamma^\sharp] \underset{A^\sharp[\gamma^\sharp]}{\longrightarrow} t^\sharp$. This lets us apply the rule (MSE) and get a substitution as follows

$$\frac{\overline{\Delta} \vdash \langle \gamma^\sharp, y \mapsto t^\sharp \rangle : (\Gamma^\sharp, y : A) \qquad \Gamma^\sharp, y : A, f : x \to y \vdash \qquad \overline{\Delta}_{>k} \vdash u^\sharp : x[\gamma^\sharp] \underset{A^\sharp[\gamma^\sharp]}{\longrightarrow} t^\sharp}{\overline{\Delta} \vdash \langle \gamma^\sharp, y \mapsto t^\sharp, f \mapsto u^\sharp \rangle : (\Gamma^\sharp, y : A, f : x \underset{A}{\longrightarrow} y)} (\text{MSE})$$

□

Note that again this mutual induction is not structurally well defined, but we can layer it using the depth of the judgments, in order to make it correct.

Flattening and folding. The two operations of flattening and folding that we have defined are inverse to each other. More specifically, we can check that whenever we have a derivation of $\overline{\Gamma} \vdash A$ satisfying ($\mathrm{C}_{\mathsf{mop}'}$) or ($\mathrm{C}_{\mathsf{mcoh}'}$), then we have $(A^\flat)^\sharp = A$. Conversely, whenever we have $\Gamma \vdash A$ satisfying (C_{op}) or ($\mathrm{C}_{\mathsf{coh}}$), then we also have $(\Gamma^\sharp)^\flat \vdash (A^\sharp)^\flat$ satisfying the same condition, and additionally, we have proved that $(\Gamma^\sharp)^\flat$ is equal to Γ up to a renaming of its variables of dimension 0, up to the same renaming, $(A^\sharp)^\flat$ is equal to A. These proofs are straightforward, but one seeds to take the same care about the indices in monoidal substitutions and the names of the variables, as we have presented before, so we admit this result here. Since by convention the term constructors do not distinguish between the renamings of the variables, the situation can be described by the following equations

$$\mathsf{op}_{\Gamma,A} = \mathsf{op}_{(\Gamma^\sharp)^\flat,(A^\sharp)^\flat} \qquad \mathsf{coh}_{\Gamma,A} = \mathsf{coh}_{(\Gamma^\sharp)^\flat,(A^\sharp)^\flat}$$
$$\mathsf{mop}'_{\overline{\Gamma},A} = \mathsf{mop}'_{(\overline{\Gamma}^\flat)^\sharp,(A^\sharp)^\flat} \qquad \mathsf{mcoh}'_{\overline{\Gamma},A} = \mathsf{mcoh}'_{(\overline{\Gamma}^\flat)^\sharp,(A^\sharp)^\flat}$$

5.3.4 Equivalence between MCaTT and MCaTT′

With the help of the folding and the flattening operation, we can define syntactic translations between the theories MCaTT and MCaTT′. These translations are defined only for regular judgments of the theory MCaTT′, and respect completely the structure of category with families,

in fact they are merely a renaming of the term constructors.

$$\begin{aligned}
F\varnothing &= \varnothing & F(\Gamma, x : A) &= (F\Gamma, x : FA)\\
F\star &= \star & F(t \underset{A}{\to} u) &= Ft \underset{FA}{\longrightarrow} Fu\\
Fx &= x & F(\mathsf{mop}_{\Gamma,A}[\gamma]) &= \mathsf{mop}'_{\Gamma\sharp,A\sharp}[\mathrm{id}_{\Gamma\sharp} \circ F\gamma]\\
& & F(\mathsf{mcoh}_{\Gamma,A}[\gamma]) &= \mathsf{mcoh}'_{\Gamma\sharp,A\sharp}[\mathrm{id}_{\Gamma\sharp} \circ F\gamma]\\
F\langle\rangle &= \langle\rangle & F\langle\gamma, x \mapsto t\rangle &= \langle F\gamma, x \mapsto Ft\rangle
\end{aligned}$$

and conversely

$$\begin{aligned}
G\varnothing &= \varnothing & G(\Gamma, x : A) &= (G\Gamma, x : GA)\\
G\star &= \star & G(t \underset{A}{\to} u) &= Gt \underset{GA}{\longrightarrow} Gu\\
Gx &= x & G(\mathsf{mcoh}'_{\Gamma,A}[\overline{\gamma}]) &= \mathsf{mcoh}_{\overline{\Gamma}^\flat,A^\flat}[G(\Pi\overline{\gamma})]\\
& & G(\mathsf{mop}'_{\Gamma,A}[\overline{\gamma}]) &= \mathsf{mop}_{\overline{\Gamma}^\flat,A^\flat}[G(\Pi\overline{\gamma})]\\
G\langle\rangle &= \langle\rangle & G\langle\gamma, x \mapsto t\rangle &= \langle F\gamma, x \mapsto Gt\rangle
\end{aligned}$$

Correctness of the translation F. The translation F takes a judgment of the category MCaTT to a judgment of the category MCaTT', in order to prove this we use the following intermediate result

Lemma 159. *The translation F respects the action of substitutions, more precisely, given a substitution $\Delta \vdash \gamma : \Gamma$,*

- *For any type $\Gamma \vdash A$, we have $F(A[\gamma]) = FA[F\gamma]$.*
- *For any term $\Gamma \vdash t : A$, we have $F(t[\gamma]) = Ft[F\gamma]$.*
- *For any substitution $\Gamma \vdash \theta : \Theta$, we have $F(\theta \circ \gamma) = F\theta \circ F\gamma$.*

Proof. We prove this result by mutual induction

Induction for types:

- For the type $\Gamma \vdash \star$, we have $\star[\gamma] = \star$, and hence $F(\star[\gamma]) = \star$, but we also have $F(\star)[\gamma] = \star$.
- For the type $\Gamma \vdash t \underset{A}{\to} u$, we have the equalities

$$F((t \underset{A}{\to} u)[\gamma]) = F(t[\gamma]) \xrightarrow[F(A[\gamma])]{} F(u[\gamma])$$

$$(F(t \underset{A}{\to} u))[\gamma] = (Ft)[\gamma] \xrightarrow[(FA)[\gamma]]{} (Fu)[\gamma]$$

  The induction cases for types and terms then prove the equality, by showing that we have $F(A[\gamma]) = (FA)[\gamma]$, $F(t[\gamma]) = (Ft)[\gamma]$ and $F(u[\gamma]) = (Fu)[\gamma]$

Induction for terms:

- For a variable $\Gamma \vdash x : A$, we necessarily have $(x : A) \in \Gamma$, hence there is a mapping of the form $x \mapsto t$ in γ, ans we have $x[\gamma] = t$, thus $F(x[\gamma]) = Ft$. But the mapping $x \mapsto t$ in γ also implies the existence of the mapping $x \mapsto Ft$ in $F\gamma$, hence $x[F\gamma] = Ft$, which proves the desired equality.

– For a term of the form $\Gamma \vdash \mathsf{mop}_{GTH,A}[\theta]$, we have the following equalities

$$F(\mathsf{mop}_{\Theta,A}[\theta][\gamma]) = \mathsf{mop}'_{\Theta\sharp,A\sharp}[\mathrm{id}_{\Theta\sharp} \circ F(\theta \circ \gamma)]$$
$$(F\mathsf{mop}_{\Theta,A}[\theta])[\gamma] = \mathsf{mop}'_{\Theta\sharp,A\sharp}[(\mathrm{id}_{\Theta\sharp} \circ F\theta) \circ F\gamma]$$

The induction case for substitutions, along with the associativity of substitution then shows the equality between these two expressions.

– Similarly for a term of the form $\Gamma \vdash \mathsf{mcoh}_{GTH,A}[\theta]$, we have the following equalities

$$F(\mathsf{mcoh}_{\Theta,A}[\theta][\gamma]) = \mathsf{mcoh}'_{\Theta\sharp,A\sharp}[\mathrm{id}_{\Theta\sharp} \circ F(\theta \circ \gamma)]$$
$$(F\mathsf{mcoh}_{\Theta,A}[\theta])[\gamma] = \mathsf{mcoh}'_{\Theta\sharp,A\sharp}[(\mathrm{id}_{\Theta\sharp} \circ F\theta) \circ F\gamma]$$

The induction case for substitutions, along with the associativity of substitution then shows the equality between these two expressions.

Induction for substitutions:

– For the empty substitution $\Gamma\langle\rangle : \varnothing$, we have $\langle\rangle \circ \gamma = \langle\rangle$, and hence $F(\langle\rangle \circ \gamma) = \langle\rangle$ and also $F\langle\rangle \circ \gamma = \langle\rangle$.

– For a substitution of the form $\Gamma \vdash \langle\theta, x \mapsto t\rangle : (\Theta, x : A)$, we have the following equalities

$$F(\langle\theta, x \mapsto t\rangle \circ \gamma) = \langle F(\theta \circ \gamma), x \mapsto F(t[\gamma])\rangle$$
$$F(\langle\theta, x \mapsto t\rangle) \circ \gamma = \langle F\theta \circ \gamma, x \mapsto (Ft)[\gamma]\rangle$$

The induction cases for substitution and terms show that $F(\theta \circ \gamma) = F\theta \circ F\gamma$ and $F(t[\gamma]) = (Ft)[\gamma]$, which proves the equality between the two previous expressions.

□

Lemma 160. *The translation F preserves derivability*

– *For any context $\Gamma \vdash$ in the theory MCaTT, the context $F\Gamma \vdash$ is derivable in MCaTT′*

– *For any type $\Gamma \vdash A$ in the theory MCaTT, the type $F\Gamma \vdash FA$ is derivable in MCaTT′*

– *For any term $\Gamma \vdash t : A$ in the theory MCaTT, the term $F\Gamma \vdash Ft : FA$ is derivable in MCaTT′*

– *For any substitution $\Delta \vdash \gamma : \Gamma$ in the theory MCaTT, the substitution $F\Gamma \vdash F\gamma : F\Gamma$ is derivable in MCaTT′*

Proof. We prove this by mutual induction

Induction for contexts:

– For the context $\varnothing \vdash$, we have $F\varnothing = \varnothing$, and the rule (EC) gives a derivation of $\varnothing \vdash$.

– For the context $(\Gamma, x : A)$, we have a derivation of $\Gamma \vdash$ and a derivation of $\Gamma \vdash A$, which by the induction cases for contexts and types give a derivation of $F\Gamma \vdash$ and of $F\Gamma \vdash FA$. The rule (CE) then applies to provide a derivation of $(F\Gamma, x : FA) \vdash$.

Induction for types:

– For the type $\Gamma \vdash \star$, we have a derivation of $\Gamma \vdash$, which by the induction for contexts gives a derivation of $F\Gamma$, and the rule ($\star$-INTRO) then applies to give a derivation of $F\Gamma \vdash \star$.

– For the type $\Gamma \vdash t \underset{A}{\longrightarrow} u$, we necessarily have a derivation of $\Gamma \vdash A$, which by the induction case for types gives a derivation of $F\Gamma \vdash FA$. Moreover we have derivations of $\Gamma \vdash t : A$ and $\Gamma \vdash u : A$, which by the induction case for the terms gives a derivation of $F\Gamma \vdash Ft : FA$ and of $F\Gamma \vdash Fu : FA$. These let us apply the rule (→-INTRO) to construct a derivation of $F\Gamma \vdash Ft \underset{FA}{\longrightarrow} Fu$.

Induction for terms:

– For a variable term $\Gamma \vdash x : A$, we have a derivation of $\Gamma \vdash$, which by the induction case for contexts gives a derivation of $F\Gamma \vdash$. Moreover the condition $(x : A) \in \Gamma$ implies that $(x : FA) \in F\Gamma$, hence the rule (VAR) applies and gives a derivation of $F\Gamma \vdash x : FA$.

– For a term of the form $\Delta \vdash \mathsf{mop}_{\Gamma,A}[\gamma]$, we have a derivation of $\Gamma \vdash A$ satisfying $(\mathrm{C}_{\mathsf{op}})$, hence, by Lemma 158, this gives a derivation of $\Gamma^\sharp \vdash A^\sharp$ satisfying $(\mathrm{C}_{\mathsf{mop}'})$. Moreover, we have a derivation of $\Delta \vdash \gamma : \lfloor\Gamma$, which by the induction case for substitution gives a derivation of $F\Delta \vdash F\gamma : F(\lfloor\Gamma)$. By Lemma 149, we have $F(\lfloor\Gamma) = \lfloor\Gamma = \Pi(\Gamma^\sharp)$, and Lemma 143 shows that we have a derivation of $\Pi(\Gamma^\sharp) \vdash \mathrm{id}_{\Gamma^\sharp} : \Gamma^\sharp$, hence the substitutions compose and give a derivation of $F\Delta \vdash \mathrm{id}_{\Gamma^\sharp} \circ F\gamma : \Gamma^\sharp$. This lets us apply the rule (MOP') to get a derivation of $F\Delta \vdash \mathsf{mop}'_{\Gamma^\sharp,A^\sharp}[\mathrm{id}_{\Gamma^\sharp} \circ F\gamma]$.

– Similarly for a term of the form $\Delta \vdash \mathsf{mcoh}_{\Gamma,A}[\gamma]$, we have a derivation of $\Gamma \vdash A$ satisfying $(\mathrm{C}_{\mathsf{coh}})$, hence, by Lemma 158, this gives a derivation of $\Gamma^\sharp \vdash A^\sharp$ satisfying $(\mathrm{C}_{\mathsf{mcoh}'})$. Moreover, we have a derivation of $\Delta \vdash \gamma : \lfloor\Gamma$, and the induction case for substitutions together with Lemmas 149 and 143 give a derivation of $F\Delta \vdash \mathrm{id}_{\Gamma^\sharp} \circ F\gamma : \Gamma^\sharp$. The rule (MCOH') then gives a derivation of $F\Delta \vdash \mathsf{mcoh}'_{\Gamma^\sharp,A^\sharp}[\mathrm{id}_{\Gamma^\sharp} \circ F\gamma]$.

Induction for substitutions:

– For the substitution $\Delta \vdash \langle\rangle : \varnothing$, we have a derivation of $\Delta \vdash$ which by induction gives a derivation of $F\Delta \vdash$, and applying the rule (ES) provides a derivation of $F\Delta \vdash \langle\rangle : \varnothing$.

– For the substitution $\Delta \vdash \langle\gamma, x \mapsto t\rangle : (\Gamma, x : A)$, we necessarily have a derivation of $\Delta \vdash \gamma : \Gamma$, of $(\Gamma, x : A) \vdash$, and of $\Delta \vdash t : A[\gamma]$. By the induction cases respectively for substitution, for contexts and for terms, these give a derivation of $F\Delta \vdash F\gamma : F\Gamma$, of $(F\Gamma, x : FA)$ and of $F\Delta \vdash Ft : F(A[\gamma])$. Since by Lemma 159 we also have $F(A[\gamma]) = FA[F\gamma]$, the rule (SE) applies and gives a derivation of $F\Delta \vdash \langle F\gamma, x \mapsto Ft\rangle : (F\Gamma, x : FA)$.

□

We can reformulate this result in a more categorical way. Indeed all these results show exactly that F defines a morphism of category with families between the syntactic category $\mathcal{S}_{\mathsf{MCaTT}}$ and the syntactic category $\mathcal{S}_{\mathsf{MCaTT}'}$.

Correctness of the translation G. We follow the same reasoning as for the translation F to show that the translation G also respects the derivability, and defines a morphism of categories with families. The proof follows the exact same steps as for F, since F and G are defined the exact same way on the syntax, replacing the folding by the flattening, and we have proved that both the folding and the flattening satisfy the same lemmas, so we do not present it here.

Cancellation of F and G. We now prove that the translations F and G are exactly inverse to each other

Lemma 161. *The following result hold*

- *For every context Γ in MCaTT, $GF\Gamma = \Gamma$*
- *For every type A in MCaTT, $GFA = A$*
- *For every term t in MCaTT, $GFt = t$*
- *For every substitution γ in MCaTT, $GF\gamma = \gamma$*

Proof. We prove this by mutual induction. The structure of the proof is exactly similar to the one of the previous proofs, as F and G respect exactly all the structure of categories with families, and the type constructors, so we only present the induction case for the term constructors, and since the two cases for term constructors are very similar, we only present the case of the constructor mop.For the term constructor $\mathsf{mop}_{\Gamma,A}[\gamma]$, we have

$$\begin{aligned} GF(\mathsf{mop}_{\Gamma,A}[\gamma]) &= G(\mathsf{mop}'_{\Gamma^\sharp,A^\sharp}[\mathrm{id}_{\Gamma^\sharp} \circ F\gamma]) \\ &= \mathsf{mop}_{(\Gamma^\sharp)^\flat,(A^\sharp)^\flat}[G(\Pi(\mathrm{id}_{\Gamma^\sharp} \circ F\gamma))] \end{aligned}$$

Since moreover we have

$$\begin{aligned} \Pi(\mathrm{id}_{\Gamma^\sharp} \circ F\gamma) &= \Pi(\mathrm{id}_{\Gamma^\sharp}) \circ F\gamma \\ &= \mathrm{id}_\Gamma \circ F\gamma \\ &= F\gamma \end{aligned}$$

the previous expression simplifies to

$$GF(\mathsf{mop}_{\Gamma,A}[\gamma]) = \mathsf{mop}_{(\Gamma^\sharp)^\flat,(A^\sharp)^\flat}[GF\gamma]$$

The induction case for substitutions and the equalities for the term constructors then give

$$GF(\mathsf{mop}_{\Gamma,A}[\gamma]) = \mathsf{mop}_{\Gamma,A}[\gamma]$$

Importantly, we consider the indexes Γ, A for the term constructors mop and mcoh up to renaming of the variables, so even though the contexts Γ and $(\Gamma^\sharp)^\flat$ have different variables they have the same structure. □

Lemma 162. *The following result hold*

- *For every context Γ in MCaTT′, $FG\Gamma = \Gamma$*
- *For every type A in MCaTT′, $FGA = A$*
- *For every term t in MCaTT′, $FGt = t$*
- *For every substitution γ in MCaTT′, $FG\gamma = \gamma$*

Proof. Again, we prove this by mutual induction, and only present the case of the term constructor mop', since the case for the other term constructor is extremely similar and all the other cases boil down to the fact that F and G preserve all the structural operation of the type theory, along with the rules for types. For the term constructor $\mathsf{mop}'_{\overline{\Gamma},A}[\overline{\gamma}]$, we have

$$\begin{aligned} FG(\mathsf{mop}'_{\overline{\Gamma},A}[\overline{\gamma}]) &= F(\mathsf{mop}_{\overline{\Gamma}^\flat,A^\flat}[G(\Pi\overline{\gamma})]) \\ &= \mathsf{mop}'_{(\overline{\Gamma}^\flat)^\sharp,(A^\flat)^\sharp}[\mathrm{id}_{(\overline{\Gamma}^\flat)^\sharp} \circ FG(\Pi\overline{\gamma})] \end{aligned}$$

By our previous discussion, this simplifies to

$$FG(\mathsf{mop}'_{\overline{\Gamma},A}[\overline{\gamma}]) = \mathsf{mop}'_{\overline{\Gamma},A}[\mathrm{id}_{\overline{\Gamma}} \circ FG(\Pi\overline{\gamma})]$$

The induction case for substitution then shows

$$\begin{aligned} FG(\mathsf{mop}'_{\overline{\Gamma},A}[\overline{\gamma}]) &= \mathsf{mop}'_{\overline{\Gamma},A}[\mathrm{id}_{\overline{\Gamma}} \circ (\Pi\overline{\gamma})] \\ &= \mathsf{mop}'_{\overline{\Gamma},A}[\overline{\gamma}] \end{aligned}$$

□

These two previous lemmas can be reformulated in a more categorical way, as the following theorem

Theorem 163. *The morphisms of categories with families F and G are inverse isomorphisms between $\mathcal{S}_{\mathsf{MCaTT}}$ and $\mathcal{S}_{\mathsf{MCaTT}'}$*

This shows that the theories MCaTT and MCaTT' can be considered to be the same in the strictest sense possible, and from now on we consider that the difference between MCaTT and MCaTT' is merely a choice of implementation.

5.3.5 Examples of derivations

Here are few examples and counter-example of derivable terms in MCaTT, to illustrate how they describe monoidal weak ω-categories.

– Monoidal product: we can define the monoidal tensor product in MCaTT as the coherence

$$\mathtt{prod} := \mathsf{coh}_{[(x:\star);(y:\star)],\star}$$

By composing with a substitution, for every objects t and u (of type $\star$) in a given context Γ, we can form their tensor product as

$$\Gamma \vdash \mathtt{prod}\ t\ u : \star$$

– Associativity of monoidal product: similarly, we can define the valid coherence

$$\mathtt{assoc} := \mathsf{coh}_{[(x:\star);(y:\star);(z:\star)],\mathtt{prod}\ x\ (\mathtt{prod}\ y\ z) \to \mathtt{prod}\ (\mathtt{prod}\ x\ y)\ z}$$

Using this coherence, for every object t, u and v (of type $\star$) in a given context Γ, we can form a witness of associativity

$$\Gamma \vdash \mathtt{assoc}\ t\ u\ v : \mathtt{prod}\ t\ (\mathtt{prod}\ u\ v) \underset{\star}{\to} \mathtt{prod}\ (\mathtt{prod}\ t\ u)\ v$$

– Neutral element: it is also possible to derive the neutral element for the monoidal product, which can be viewed as a nullary monoidal product. It is defined as a coherence

$$\mathsf{e} := \mathsf{coh}_{[],\star}$$

In any context, one compose with the empty substitution, in order to get the term witnessing the neutral element.

$$\Gamma \vdash \mathsf{e}\,[] : \star$$

In order to simplify the notations, we may simply denote e in any context, and omit the empty substitutions [] which carries no information.

– Cancellation witness: one can also prove in MCaTT that the term e is indeed a neutral element for the monoidal product, here for instance on the left. This is done by the coherence

$$\texttt{l-unit} := \mathsf{coh}_{[(x:\star)],(\texttt{prod e } x)\to x}$$

This coherence can be used to derive, for any object u in a context Γ, the following witness of left unitality of e

$$\Gamma \vdash \texttt{l-unit}\ u : (\texttt{prod e } u) \to u$$

– Functoriality of monoidal product: The following coherence defines the functoriality for monoidal product

$$\texttt{funl} := \mathsf{coh}_{[(x:\star,y:\star,f:x\to y);(z:\star)],(\texttt{prod } x\ z)\to(\texttt{prod } y\ z)}$$

Given any three objects u, u', v together with a term t of type $u \to u'$ in a context Γ, this coherence can be used to derive a witness for the functoriality of the monoidal tensor product on the left

$$\Gamma \vdash \texttt{funl}\ t\ v : (\texttt{prod}\ u\ v) \to (\texttt{prod}\ u'\ v)$$

– Symmetry of the monoidal product: note that the same idea does not apply to derive the symmetry of the monoidal product. Indeed, if we try to build a witness for symmetry, it would be a term

$$\mathsf{sym} := \mathsf{coh}_{[(x:\star);(y:\star)],\texttt{prod } x\ y\to\texttt{prod } y\ x}$$

It turns out that the list $[(x:\star);(y:\star)] \vdash [\langle y\rangle;\langle x\rangle] : [(x:\star);(y:\star)]$ is not a valid substitution, which makes the judgment $[(x:\star);(y;\star)] \vdash \texttt{prod}\ y\ x$ not derivable. So the term sym is not derivable in MCaTT. This is a safety check, since the internal language for monoidal categories is supposed to be unable to derive a witness for commutativity.

5.4 Towards k-tuply monoidal weak ω-category

We generalize the type theory MCaTT to model k-tuply monoidal categories, following the exact same principle as before, and call k-MCaTT the globular type theory we obtain this way. However, proving that the models of k-MCaTT are equivalent to the models of CaTT that have only one object in all the dimensions up to k turns out to me significantly more complicated, and we do not provide a proof of this fact here.

Syntax. We can define the theory k-MCaTT from CaTT by iterating the construction we have presented to obtain the theory MCaTT from CaTT. We denote ${}_k\mathsf{mop}$ and ${}_k\mathsf{mcoh}$ the two term constructors corresponding to op and *coh* for k-tuply monoidal categories.

The k-fold desuspension on $\mathcal{S}_{\mathsf{PS},0}$. We start by defining the k-fold desuspension as an operation on the syntax, that is only valid for derivable judgments in the theory CaTT

$$\begin{array}{cc}
\text{For the context } \varnothing \vdash & \text{For the context } (\Gamma, x : A) \vdash \\
\downarrow^k \varnothing = \varnothing & \downarrow^k (\Gamma, x : A) = \begin{cases} \downarrow^k \Gamma & \text{if } \dim A \leq k-1 \\ \downarrow^k \Gamma, x : \downarrow^k A & \text{otherwise} \end{cases} \\
\\
\text{For the type } \Gamma \vdash \star & \text{For the type } \Gamma \vdash t \underset{A}{\longrightarrow} u \\
\downarrow^k \star = \star & \downarrow^k (t \underset{A}{\longrightarrow} u) = \begin{cases} \star & \text{if } \dim A \leq k-1 \\ \downarrow^k t \underset{\downarrow^k A}{\longrightarrow} \downarrow^k u & \text{otherwise} \end{cases} \\
\\
\text{For a variable } \Gamma \vdash x : A & \text{For the term } \Delta \vdash \mathsf{op}_{\Gamma,A}[\gamma] : A[\gamma] \\
\downarrow^k x = x & \downarrow^k \mathsf{op}_{\Gamma,A}[\gamma] = {}_k\mathsf{mop}_{\Gamma,A}[\downarrow^k \gamma] \\
\\
 & \text{For the term } \Delta \vdash \mathsf{coh}_{\Gamma,A}[\gamma] : A[\gamma] \\
 & \downarrow^k \mathsf{coh}_{\Gamma,A}[\gamma] = {}_k\mathsf{mcoh}_{\Gamma,A}[\downarrow^k \gamma] \\
\\
\text{For } \Delta \vdash \langle\rangle : \varnothing & \text{For } \Delta \vdash \langle \gamma, x \mapsto t \rangle : (\Gamma, x : A) \\
\downarrow^k \langle\rangle = \langle\rangle & \downarrow^k \langle \gamma, x \mapsto t \rangle = \begin{cases} \downarrow \gamma & \text{if } \dim A \leq k-1 \\ \langle \downarrow^k \gamma, x \mapsto \downarrow^k t \rangle & \text{otherwise} \end{cases}
\end{array}$$

Introduction rules for the theory k-MCaTT. The introduction rules for these term constructors can be expressed using the k-fold desuspension operation.

$$\frac{\Gamma \vdash_{\mathsf{ps}} \qquad \Gamma \vdash A \qquad \Delta \vdash \gamma : \downarrow^k \Gamma}{\Delta \vdash {}_k\mathsf{mop}_{\Gamma,A} : (\downarrow^k A)[\gamma]}$$

whenever A satisfies (C_{op}),

$$\frac{\Gamma \vdash_{\mathsf{ps}} \qquad \Gamma \vdash A \qquad \Delta \vdash \gamma : \downarrow^k \Gamma}{\Delta \vdash {}_k\mathsf{mcoh}_{\Gamma,A} : (\downarrow^k A)[\gamma]}$$

whenever A satisfies ($\mathrm{C}_{\mathsf{coh}}$).

Chapter 6

Cubical weak ω-categories

6.1 Type theory for pre-cubical sets

We now present another variation of the theory CaTT that aims to define structures similar to weak ω-categories, but based on cubical shapes rather than on globular ones. To achieve this, we first present the pre-cubical sets, that we use as support for these structures. Our strategy is then to carry over the intuitions that we can grow from the theory CaTT to this structure, and propose directly a type theoretic definition. We then study this definition to extract a mathematical description of its models that serves as a new definition of cubical weak ω-categories.

6.1.1 The category of pre-cubical sets

We define the pre-cube category and the category of pre-cubical set as an alternate shape for working with higher structures. Our presentation is inspired by [38].

The pre-cube category. We denote by $\square$ the *pre-cube category*. It is the free completion by cartesian product of the category

$$[0] \underset{\tau}{\overset{\sigma}{\rightrightarrows}} [1]$$

One can give a more explicit definition of the pre-cube category, by giving a presentation of this category.

$$\square : [0] \underset{\tau_0}{\overset{\sigma_0}{\rightrightarrows}} [1] \underset{\tau_1}{\overset{\sigma_0}{\underset{\tau_0}{\overset{\sigma_1}{\rightrightarrows}}}} [2] \underset{\tau_2}{\overset{\sigma_0}{\underset{\tau_1}{\overset{\sigma_1}{\underset{\tau_0}{\overset{\sigma_2}{\rightrightarrows}}}}}} [3] \qquad \cdots$$

subject to the *cocubical relations*, for all $j < i$

$$\sigma_j\sigma_i = \sigma_{i+1}\sigma_j \qquad \sigma_j\tau_i = \tau_{i+1}\sigma_j$$
$$\tau_j\sigma_i = \sigma_{i+1}\tau_j \qquad \tau_j\tau_i = \tau_{i+1}\tau_j$$

Shifts in $\square$. Notice that the category $\square$ has a non-trivial faithful endofunctor shift.

$$\begin{array}{rccc} \text{shift} & : & \square & \to & \square \\ & & [i] & \mapsto & [i+1] \\ & & \sigma_i & \mapsto & \sigma_{i+1} \\ & & \tau_i & \mapsto & \tau_{i+1} \end{array}$$

The well-definedness of this functor comes from the fact that the cocubical relations are invariant under the transformation $(i,j) \mapsto (i+1, j+1)$

Pre-cubical sets. We now define the category of pre-cubical sets **CSet** as the presheaf category over the category $\square$

$$\mathbf{CSet} = \widehat{\square} = \mathbf{Set}^{\square^{\mathrm{op}}}$$

more explicitly, a pre-cubical set $X_\bullet$ is a collection of sets $X_0, X_1, X_2, \cdots$, together with families of maps $\partial_i^+, \partial_i^- : X_n \to X_{n-1} \quad (0 \leq i \leq n)$, satisfying the following *cubical relations*, that are dual to the cubical relations:

$$\begin{array}{ll} \partial_i^- \partial_j^- = \partial_j^- \partial_{i+1}^- & \partial_i^- \partial_j^+ = \partial_j^+ \partial_{i+1}^- \\ \partial_i^+ \partial_j^- = \partial_j^- \partial_{i+1}^+ & \partial_i^+ \partial_j^+ = \partial_j^+ \partial_{i+1}^+ \end{array}$$

The elements of the set X_n will be called the *n-cells* of the pre-cubical set X. The 0-cells are also called *objects* of X, and the 1-cells are also called *arrows* of X.We will refer to finite cubical sets as *(cubical) diagrams*, and we may give a graphical representation for such diagrams. For instance, the following diagram

$$\begin{array}{ccccc} x & \xrightarrow{f} & y & \xrightarrow{g} & z \\ & & {\scriptstyle k}\downarrow & \Downarrow\alpha & \downarrow{\scriptstyle l} \\ & & y' & \xrightarrow[g']{} & z' \xrightarrow[h']{} w' \end{array}$$

represents the pre-cubical set X with $X_0 = \{x, y, z, y', z', w'\}$, $X_1 = \{f, g, g', h', k, l\}$, $X_2 = \{\alpha\}$, and $X_{>2} = \emptyset$, together with the maps given by

$$\partial_0^- : \begin{array}{l} f \mapsto x \\ g \mapsto y \\ g' \mapsto y' \\ h' \mapsto z' \\ k \mapsto y \\ l \mapsto z \end{array} \qquad \partial_0^+ : \begin{array}{l} f \mapsto y \\ g \mapsto z \\ g' \mapsto z' \\ h' \mapsto w' \\ k \mapsto y' \\ l \mapsto z' \end{array} \qquad \begin{array}{l} \partial_0^-(\alpha) = k \\ \partial_0^+(\alpha) = l \\ \partial_1^-(\alpha) = g \\ \partial_1^+(\alpha) = g' \end{array}$$

Shifts of pre-cubical sets. Let $X : \square^{\mathrm{op}} \to \mathbf{Set}$ be a pre-cubical set. The endofunctor $\mathrm{shift} : \square \to \square$ induces another pre-cubical set $\mathrm{shift}^*(X) = X \circ \mathrm{shift}$. The object of $\mathrm{shift}^*(X)$ are exactly the arrows of X, and more generally the n-cells of $\mathrm{shift}^*(X)$ are exactly the $n+1$-cells of X.

$$\mathrm{shift}^*(X)_n = X_{n+1}$$

6.1.2 Type theory for pre-cubical sets

From our presentation of the direct category $\square$, we can extract the type theory $T_\square$. For integer $i \in \mathbb{N}$, there is a type constructor $\to_n$ of arity $2n$, that we denote with n arguments on the left and n arguments on the right like in the following expression $(t_1, \dots, t_n) \to_n (u_1, \dots, u_n)$, and similarly to the type theory GSeTT, we denote $\star$ for the type $\to_0$. These type constructors are then subject to the following introduction rules

$$\frac{\Gamma \vdash}{\Gamma \vdash \star} \qquad \frac{\Gamma \vdash t : \star \qquad \Gamma \vdash u : \star}{\Gamma \vdash t \to_1 u}$$

$$\frac{\Gamma \vdash t_0 : x \to_1 y \qquad \Gamma \vdash t_1 : x \to_1 z \qquad \Gamma \vdash u_0 : z \to_1 w \qquad \Gamma \vdash u_1 : y \to_1 w}{\Gamma \vdash (t_0, t_1) \to_2 (u_0, u_1)}$$

and so on. The rules becoming increasingly complicated as the dimension increases, this type theory is complicated to study and to implement. We give an alternative presentation for the type theory $T_\square$, inspired from cubical type theory [29], and which treats the dimensions more uniformly.

Signature. Our new presentation of the type theory $T_\square$ uses a new kind of variables, that we call *dimension variables* or *direction variables* of which we suppose given a set $\mathcal{D}$. The terms and types may now have free dimension variables, and the constructors Path binds those variables. In our presentation I always denotes a list of dimension variables, and the dimension variables are always denoted $i, j \cdots$. We define a *term* t to be either a variable $x \in \mathcal{V}$, or of the form ti, with t a term, and $i \in \mathcal{D}$, and a *type*, to be either $\star$, or of the form $\mathsf{Path}^i\ A\ t\ u$ where $i \in \mathbb{D}$, A is a type, and t, u are terms.

Judgments. We introduce four kinds of judgments that are similar from our previous judgments, except that they account for the dimension variables. This is done by indexing the judgment by a list of dimension variables I, and we then denote $\vdash_I$ the corresponding judgments.

$\Gamma \vdash$	The context Γ is valid
$\Gamma \vdash_I A$	The type A is valid in the context Γ, with direction variables I
$\Gamma \vdash_I t : A$	The term t is of type A in the context Γ, with direction variables I
$\Gamma \vdash \sigma : \Delta$	The substitution σ is from Γ to Δ

When the list I is the empty list, we will write $\vdash$ instead of $\vdash_I$, and we call a type A such that $\Gamma \vdash A$ is valid (resp. a term t such that $\Gamma \vdash t : A$ is valid) a *closed type* (resp. *closed term*) in the context Γ. We are interested in the closed judgments in this theory, and see the judgment with free dimension variables as intermediate steps to compute closed ones.

Variable replacement. For a term t (resp. a type A) and two dimension variables i, r, we introduce the term $t[r/i]$ (resp. the type $A[r/i]$) obtained by formally replacing all occurrences

of the dimension variable i by r. This is inductively defined by

$$
\begin{array}{lcll}
\text{For terms:} & & & \\
x[r/i] & = & x & (x \text{ is a variable}) \\
(t\ i)[r/i] & = & t\ r & \\
(t\ j)[r/i] & = & (t[r/i])\ j & (j \neq i) \\
\text{For types:} & & & \\
\star[r/i] & = & \star & \\
(\mathsf{Path}^j\ B\ u\ t)[r/i] & = & \mathsf{Path}^j\ B\ u[r/i]\ t[r/i] & (j \neq i) \\
(\mathsf{Path}^i\ B\ u\ t)[r/i] & = & \mathsf{Path}^r\ B[r/i]\ t\ u &
\end{array}
$$

Typing rules. We consider the following typing rules for our type theory. These rules are very similar to the general rules for a type theory, but we add the possibility to have free dimension variables. We also add a rewriting rule that makes explicit the computation on dimension variables.

For contexts:

$$\frac{}{\varnothing \vdash} \qquad \frac{\Gamma \vdash A \qquad x \notin \mathrm{Var}(\Gamma)}{(\Gamma, x : A) \vdash}$$

For types:

$$\frac{\Gamma \vdash}{\Gamma \vdash_I \star} \qquad \frac{\Gamma \vdash_{I,i} A \qquad \Gamma \vdash_I t_0 : A[0/i] \qquad \Gamma \vdash_I t_1 : A[1/i]}{\Gamma \vdash_I \mathsf{Path}^i\ A\ t_0\ t_1}$$

For terms:

$$\frac{\Gamma \vdash \qquad (x : A) \in \Gamma}{(\Gamma, x : A) \vdash x : A} \qquad \frac{\Gamma \vdash_I t : \mathsf{Path}^i\ A\ u_0\ u_1 \qquad r \notin I}{\Gamma \vdash_{I,r} t\ r : A[r/i]}$$

For substitutions:

$$\frac{\Delta \vdash}{\Delta \vdash \langle\rangle : \varnothing} \qquad \frac{\Delta \vdash \sigma : \Gamma \qquad \Gamma \vdash A \qquad \Delta \vdash t : A\,[\sigma]}{\Delta \vdash \langle \sigma, x \mapsto t \rangle : (\Gamma, x : A)}$$

Rewriting:

$$\frac{\Gamma \vdash_I t : \mathsf{Path}^i\ A\ u_0\ u_1}{\Gamma \vdash_I t\ \varepsilon \rightsquigarrow u_\varepsilon : A[\varepsilon/i]} (\varepsilon = 0, 1)$$

Dimension. A type A of this theory come along with a notion of *dimension* denoted $\dim(A)$, which is the height of the nested types. It is formally defined inductively as follows

$$\dim(\star) = -1 \qquad \dim(\mathsf{Path}^i\ A\ t\ u) = \dim A\ + 1$$

Given a term t, together with a context Γ such that $\Gamma \vdash t : A$ holds, then we define the dimension of term t in the context Γ. In general the context Γ is clear, and we will often write $\dim t$, unless we want to emphasize that the definition of this dimension fundamentally relies on the context. We also define the dimension $\dim(\Gamma)$ of a context Γ to be the maximal dimension of all its variables

$$\dim_\Gamma(t) = \dim A\ + 1 \qquad \dim(x_0 : A_0, \cdots, x_n : A_n) = \max_i \{\dim(A_i)\}$$

Terms with free dimension variable. Notice that if we have a term t, such that $\Gamma \vdash_i t : A$ holds, the only way possible for this judgment to be derivable is if there is a term t' such that $\Gamma \vdash t' : \mathsf{Path}^i\ B\ t_0\ t_1$ holds, and $t = t'\ i$. This allows us to define an *extraction function* ϵ, which from a term with a free dimension variable extracts a closed term. We define ϵ as a partial function

$$\epsilon(t\ \_) = t$$

and by the above discussion, the following rule is admissible

$$\frac{\Gamma \vdash_i t : A}{\Gamma \vdash \epsilon(t) : B}$$

and by the same argument, there is a more general rule

$$\frac{\Gamma \vdash_{I,r} t : A}{\Gamma \vdash_I \epsilon(t) : B}$$

In particular, from this composing this function n times defines a one to one correspondence between the terms of dimension 0 with n free dimension variables and the closed terms of dimension n in Γ.

Sources and target of a term. Given a type $A = \mathsf{Path}^i\ B\ t\ u$, we define inductively its sources and targets, in various directions. In the categorical point of view, the number of source and target function depends on the dimension of the type, but here we will instead define the sources and targets for all indexes as partial functions on types, in order to have an easier inductive definition (the cases that are not specified here correspond to cases where the function is undefined).

$$\partial^-_n(\mathsf{Path}^i\ B\ t\ u) = \begin{cases} t & \text{If } n\ = \dim B + 1 \\ \epsilon(\partial^-_n(B)) & \text{Otherwise} \end{cases}$$

$$\partial^+_n(\mathsf{Path}^i\ B\ t\ u) = \begin{cases} u & \text{If } n\ = \dim B + 1 \\ \epsilon(\partial^+_n(B)) & \text{Otherwise} \end{cases}$$

Notation. For the sake of simplicity, given terms $t_0, \cdots, t_n$ and $u_0, \cdots, u_n$, we may write $(t_0, \cdots, t_n) \to (u_0, \cdots, u_n)$ for the iterated path type

$$\mathsf{Path}^i\ (\mathsf{Path}^j\ \cdots\ (t_{n-1}\ i)\ (u_{n-1}\ i))\ t_n\ u_n$$

so that the dimension of this type is given by $\dim\left((t_0, \cdots, t_n) \to (u_0, \cdots, u_n)\right) = n$ and all its sources and targets are given by the following equations

$$\partial^-_i((t_0, \cdots, t_n) \to (u_0, \cdots, u_n)) = t_i$$

$$\partial^+_i((t_0, \cdots, t_n) \to (u_0, \cdots, u_n)) = u_i$$

We can then check that the introduction rules for the types $\to$ are exactly the introduction for the type $\to_n$ obtained as the theory $T_\square$ constructed from the direct category $\square$. We illustrate this fact in dimension 2: Consider the type

$$\Gamma \vdash \mathsf{Path}^i\ \mathsf{Path}^j\ \star\ (t_0\ i)\ (u_0\ i)\ t_1\ u_1$$

then we necessarily have a derivation for the following judgments

$$\Gamma \vdash_i \mathsf{Path}^j \ \star \ (t_0\ i)\ (u_0\ i)$$
$$\Gamma \vdash t_1 : \mathsf{Path}^j \ \star \ (t_0\ 0)\ (u_0\ 0)$$
$$\Gamma \vdash u_1 : \mathsf{Path}^j \ \star \ (t_0\ 1)\ (u_0\ 1)$$

The first of these derivations implies that we have a derivation of

$$\Gamma \vdash_i t_0\ i : \star \qquad \Gamma \vdash_i u_0\ i : \star$$

Which necessarily gives t_0 and u_0 of the following form

$$\Gamma \vdash t_0 : \mathsf{Path}^j \ \star \ t_0^-\ t_0^+ \qquad \Gamma \vdash u_0 : \mathsf{Path}^j \ \star \ u_0^-\ u_0^+$$

The second and the third of the previous derivations then rewrites as

$$\Gamma \vdash t_1 : \mathsf{Path}^j \ \star \ t_0^-\ u_0^- \qquad \Gamma \vdash u_1 : \mathsf{Path}^j \ \star \ t_0^+\ u_0^+$$

Hence the terms t_0, u_0, t_1, u_1 satisfy the cubical relations

$$\begin{array}{ccc} t_0^- & \xrightarrow{t_1} & u_0^- \\ {\scriptstyle t_0}\downarrow & & \downarrow{\scriptstyle u_0} \\ t_0^+ & \xrightarrow[u_1]{} & u_0^+ \end{array}$$

Conversely, given four terms satisfying these relations, we can reconstruct the type

$$\Gamma \vdash \mathsf{Path}^i\ (\mathsf{Path}^i\ (t\ {}_0 i)\ (u\ {}_0 i)\ )\ t_1\ u_1$$

This correspondence works the same way for terms of higher dimension, but is more involved to follow.

Syntactic category. We consider the syntactic category $\mathcal{S}_{T_\square}$ to be the category whose objects are all contexts $\Gamma \vdash$, and whose morphisms $\Delta \to \Gamma$ are closed substitutions $\Delta \vdash \gamma : \Gamma$. This category is equipped with a structure of category with families, by choosing Ty^Γ to be the set of all closed types $\Gamma \vdash A$, and Tm_A^Γ to be the set if all closed terms $\Gamma \vdash t : A$. Our previous remark shows that this syntactic category is isomorphic to the syntactic category of the theory $T_\square$ associated to the direct category $\square$. For this reason we denote $\mathcal{S}_{T_\square}$ both syntactic categories. Our study of the theory T_I for any I shows the following, as an application of Theorem 113

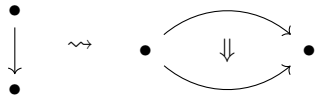
Theorem 164. *The syntactic category $\mathcal{S}_{T_\square}$ is equivalent to the opposite of the category of finite pre-cubical sets.*

Models. Again, our study of the theory T_I in general, shows the following as a special case of Theorem 114

Theorem 165. *The category of models* $\mathbf{Mod}(\mathcal{S}_{T_\square})$ *is equivalent to the category of pre-cubical sets*

6.1.3 Functorialization in the theory $T_\square$

We introduce two meta operations in the theory $T_\square$ that are the cubical analogue to the suspension and the functorialization. Note that in a cubical setting, the suspension does not make immediate sense, as it would generate cells that are not allowed, as we show on the following diagram



Instead we replace our notion of suspension by another operation that we call extrusion. We first describe the notion of functorialization, as it stays closer to the definition we have presented in CaTT, and gives intuition to understand this new operation.

Intuitive description. The intuition for the functorialization is similar to the intuition of the functorialization in GSeTT: we duplicate a cell, and create a higher cell whose source is our original cell and whose target is the duplicated cell. Given a cell x, we denote x^+ its duplicate and $\vec{x}$ the higher cell that connects them. The difference however is that in order to functorialize with respect to a cell, we need to functorialize inductively the lower cells from which it is built. We give a pictorial example of the process of functorialization, along with its result

$$x \xrightarrow{f} y \quad \rightsquigarrow \quad \begin{array}{ccc} x & \xrightarrow{f} & y \\ {\scriptstyle \vec{x}}\downarrow & \Downarrow \vec{f} & \downarrow{\scriptstyle \vec{y}} \\ x^+ & \xrightarrow[f^+]{} & y^+ \end{array}$$

Definition. In order to define the functorialization, we suppose that we have enough fresh variable names at our disposal, and we chose a set X of variables, with respect to which we perform the functorialization. We then define a context $\Gamma^{\vec{X}}$ that we call the functorialization of Γ with respect to the set X, together with two substitutions $\iota_x(\Gamma)$ and $\mathfrak{s}_x(\Gamma)$

$$\varnothing^{\vec{X}} = \varnothing$$
$$(\Gamma, x : A)^{\vec{X}} = \begin{cases} (\Gamma^{\vec{X}_A}, x : A, x^+ : A[\mathfrak{s}_{X_A}(\Gamma)], \vec{x} : \mathsf{Path}^i\ \overrightarrow{A}^i\ x\ x^+) & \text{If } x \in X \\ (\Gamma^{\vec{X}}, x : A) & \text{Otherwise} \end{cases}$$

where $X_A = X \cup \mathrm{Var}(A)$, and the operation $\overrightarrow{A}^i$ on types is defined by

$$\overrightarrow{\star}^i = \star \qquad \begin{array}{l} \overrightarrow{\mathsf{Path}^j\ A\ x\ y}^i = \mathsf{Path}^j\ \overrightarrow{A}^i\ (x\ i)\ (y\ i) \\ \overrightarrow{\mathsf{Path}^i\ A\ x\ y}^i = \mathsf{Path}^k\ \overrightarrow{A[k/i]}^i\ (x\ i)\ (y\ i) \end{array}$$

and the substitutions ι and $\mathfrak{s}$ are defined by

$$\iota_X(\varnothing) = \langle\rangle \qquad \iota_x(\Gamma, x : A) = \langle \iota_x(\Gamma), x \mapsto x \rangle$$
$$\mathfrak{s}_X(\varnothing) = \varnothing \qquad \mathfrak{s}_X(\Gamma, x : A) = \langle \mathfrak{s}_X(\Gamma), x \mapsto x^+ \rangle$$

Since we do not use this operation, any further and only present it to illustrate possible constructions in this theory, we admit without giving a proof that it only produces valid terms.

6.1.4 Extrusion of a variable

We now describe the other meta-theoretic operation that we use on the theory $T_\square$. This operation that we call *extrusion* is a bit analogous to the suspension in the globular setting, but has to be adapted a lot to be definable in the cubical setting.

Intuitive description. Intuitively, the idea is similar to the suspension, in that we increase all the dimensions of the cells by 1, except that we cannot perform this anymore by simply adding two new points. Instead, we add two new entire copies of our context, and we reinterpret our context as higher cells in between these two copies. In order to emphasize the similarity with the functorialization, we reformulate this procedure as adding a copy of our context and higher cells to fill the gap thus created. The difference with the functorialization is the direction of the higher cells we add. We again denote for a cell x, its new copy x^+, and the higher cell $\vec{x}$. This operation can be described pictorially as follows

$$\begin{array}{ccc} x & & x & \xrightarrow{\vec{x}} & x^+ \\ {\scriptstyle f}\downarrow & \rightsquigarrow & {\scriptstyle f}\downarrow & \Downarrow_{\vec{f}} & \downarrow{\scriptstyle f^+} \\ y & & y & \xrightarrow[\vec{y}]{} & y^+ \end{array}$$

Definition. We again assume that we have as many fresh variable names at our disposal, and define a fixed set of variables with respect to which we perform the extrusion. For a context Γ, we consider a set of variables X that is downwards closed, i.e., , if a variable $x \in X$ is in $\mathrm{Var}(\Gamma)$, so are all the variables of its type are in X. We denote $\Sigma_X\Gamma$ the context obtained by extrusion of Γ with respect to the variables of x, that we define as follows

$$\begin{array}{lll} \Sigma_X\varnothing & = \varnothing & \\ \Sigma_X(\Gamma, x : A) = \begin{cases} (\Sigma_X\Gamma, x : A, x^+ : A^+, \vec{x} : \Sigma^{x,x^+}A) & \text{If } x \in X \\ (\Sigma_X\Gamma, x : A) & \text{Otherwise} \end{cases} & & \\ \Sigma^{a,b}\star & = \mathsf{Path}^i\ \star\ a\ b & \Sigma^{a,b}(\mathsf{Path}^i\ A\ t\ u) = \mathsf{Path}^i\ (\Sigma^{(a\ i),(b\ i)}A)\ \Sigma t\ \Sigma u \\ \Sigma x & = \vec{x} & \Sigma(t\ i) = (\Sigma t)\ i \\ \text{where } A^+ \text{ is defined by} & & \\ \star^+ & = \star & (\mathsf{Path}^i\ A\ t\ u)^+ = \mathsf{Path}^i\ A^+\ t^+\ u^+ \\ x^+ & = x^+ & (t\ i)^+ = t^+\ i \end{array}$$

Correctness. We have defined this operation purely syntactically, but we can check that it only yields valid judgments in the theory $T_\square$. More precisely, we have the following result

Lemma 166. *The extrusion preserves the derivability of judgments, for a context $\Gamma \vdash$ together with a set X downwards closed in Γ:*

- *There is a derivation of $\Sigma_X\Gamma \vdash$.*
- *For any type $\Gamma \vdash_I A$, there is a derivation of $\Sigma_X\Gamma \vdash_I A$.*
- *For any term $\Gamma \vdash_I t : A$, there is a derivation of $\Sigma_X\Gamma \vdash_I t : A$.*
- *For any type $\Gamma \vdash_I A$ with $\mathrm{Var}(A) \subseteq X$, there is a derivation of $\Sigma_X\Gamma \vdash_I A^+$.*
- *For any type $\Gamma \vdash_I t : A$ with $\mathrm{Var}(t) \subseteq X$, there is a derivation of $\Sigma_X\Gamma \vdash_I t^+ : A^+$ and of $\Sigma_X\Gamma \vdash_I \Sigma t : \Sigma^{t,t^+}A$.*

Proof. We prove these results by mutual induction, the presence of the dimension variables make the proofs a little more involved.

Induction for $\Gamma \vdash$:

- For the context $\varnothing$, the derivation is given by the rule (EC).
- For a context of the form $(\Gamma, x : A) \vdash$ with $x \notin X$, by induction we have $\Sigma_X\Gamma \vdash$ and $\Sigma_X\Gamma \vdash A$ so the rule (CE) applies and shows $\Sigma_X((\Gamma, x : A)) \vdash$.
- For a context of the form $(\Gamma, x : A) \vdash$ with $x \in X$, we have by induction $\Sigma_X\Gamma \vdash$ and $\Sigma_X\Gamma \vdash A$, hence rule (CE) gives $(\Sigma_X\Gamma, x : A) \vdash$. Moreover, the induction gives a derivation of $\Sigma_X\Gamma \vdash A^+$, by weakening and applying the rule (CE) this provides a derivation of $(\Sigma_X\Gamma, x : A, x^+ : A^+) \vdash$. So it suffices to prove that we have a derivation of $\Sigma_X\Gamma \vdash \Sigma^{x,x^+} A$ and applying (CE) then gives a derivation of $\Sigma_X(\Gamma, x : A) \vdash$. We prove this by case analysis on A:
  - For the type $A = \star$, we have $(\Gamma, x : A, x^+ : A^+) = (\Gamma, x : \star, x^+ : \star)$, hence we have a derivation of $(\Gamma, x : \star, x^+ : \star) \vdash x : \star$ and of $(\Gamma, x : \star, x^+ : \star) \vdash x^+ : \star$. These give a derivation of
    $$(\Gamma, x : \star, x^+ : \star) \vdash \mathsf{Path}^i\ x\ x^+$$
    which is exactly the type $\Sigma^{x,x^+}\star$.
  - For the type $A = \mathsf{Path}^i\ B\ t\ u$, we have a derivation of $\Gamma \vdash t : B[0/i]$ and of $\Gamma \vdash u : B[1/i]$, which by the mutual induction give derivation of the following judgments
    $$\Sigma_X\Gamma \vdash \Sigma t : \Sigma^{t,t^+}(B[[0/i]]) \qquad \Sigma_X\Gamma \vdash \Sigma u : \Sigma^{u,u^+}(B[[1/i]])$$
    Since moreover $t \equiv x\ 0$, $t^+ \equiv x^+\ 0$, $u \equiv x\ 1$ and $u^+ \equiv x^+\ 1$, substituting these equations into the previous judgments and factoring the variable replacements shows that we have in fact derivations of
    $$\Sigma_X\Gamma \vdash \Sigma t : (\Sigma^{(x\ i),(x^+\ i)}B)[0/i] \qquad \Sigma_X\Gamma \vdash \Sigma u : (\Sigma^{(x\ i),(x^+\ i)}B)[1/i]$$
    So this lets us construct a derivation of
    $$\Sigma_X\Gamma \vdash \mathsf{Path}^i\ \Sigma^{(x\ i),(x^+\ i)}B\ \Sigma t\ \Sigma u$$
    which is exactly the type $\Sigma^{x,x^+}A$

Induction for $\Gamma \vdash_I A$:

- For the type $\Gamma \vdash_I \star$, we necessarily have $\Gamma \vdash$ hence by induction $\Sigma_X\Gamma \vdash$, so we have $\Sigma_X\Gamma \vdash \star$.
- For the type $\Gamma \vdash_I \mathsf{Path}^i\ A\ t\ u$, we necessarily have the judgments $\Gamma \vdash_{I,i} A$, $\Gamma \vdash_I t : A[0/i]$ and $\Gamma \vdash_I u : A[1/i]$, so the induction gives the judgments $\Sigma_X\Gamma \vdash_{I,i} A$, $\Sigma_X\Gamma \vdash_I t : A[0/i]$ and $\Sigma_X\Gamma \vdash_I u : A[1/i]$, which prove
  $$\Sigma_X\Gamma \vdash_I \mathsf{Path}^i\ A\ t\ u$$

Induction for $\Gamma \vdash_I t : A$:

- For a variable term $\Gamma \vdash x : A$, we necessarily have $\Gamma \vdash$ and hence by induction $\Sigma_X\Gamma \vdash$. Since moreover we always have $(x : A) \in \Sigma_X\Gamma$, this proves $\Sigma_X\Gamma \vdash x : A$.

– For a term of the form $\Gamma \vdash_{I,j} t\ j : A[j/i]$, we necessarily have $\Gamma \vdash_I t : \mathsf{Path}^i\ A\ u_0\ u_1$, which by induction provides a derivation of $\Sigma_X\Gamma \vdash_I t : \mathsf{Path}^i\ A\ u_0\ u_1$, hence we have a derivation of $\Sigma_X\Gamma \vdash_{I,j} t\ j : A[j/i]$.

Induction for $\Gamma \vdash_I A$ *with* $\mathrm{Var}(A) \subseteq X$:

– For the type $\Gamma \vdash_I \star$, we have already proved that we have $\Sigma_X\Gamma \vdash \star$, which rewrites as $\Sigma_X\Gamma \vdash \star^+$.

– For the type $\Gamma \vdash_I \mathsf{Path}^i\ A\ t\ u$, we necessarily have $\Gamma \vdash_{I,i} A$, $\Gamma \vdash_I t : A[0/i]$ and $\Gamma \vdash_I u : A[1/i]$, so the induction gives the judgments $\Sigma_X\Gamma \vdash_{I,i} A^+$, $\Sigma_X\Gamma \vdash_I t^+ : A[0/i]^+$ and $\Sigma_X\Gamma \vdash_I u^+ : A[1/i]^+$. Since moreover $A[r/i]^+ = A^+[r/i]$, this gives a derivation of

$$\Sigma_X\Gamma \vdash_I \mathsf{Path}^i\ A^+\ t^+\ u^+$$

Induction for $\Gamma \vdash_I t : A$ *with* $\mathrm{Var}(t) \subseteq X$:

– For a variable term $\Gamma \vdash x : A$, we necessarily have $\Gamma \vdash$ and hence by induction $\Sigma_X\Gamma \vdash$. The variables condition then implies that $x \in X$ and thus we have $(x^+ : A^+) \in \Sigma_X\Gamma$, this proves

$$\Sigma_X\Gamma \vdash x^+ : A^+$$

Moreover, we also have $(\Sigma x : \Sigma^{x,x^+} A) \in \Sigma_X\Gamma$, hence this gives a derivation of

$$\Sigma_X\Gamma \vdash \Sigma x : \Sigma^{x,x^+} A$$

– For a term of the form $\Gamma \vdash_{I,j} t\ j : A[j/i]$, we necessarily have $\Gamma \vdash_I t : \mathsf{Path}^i\ A\ u_0\ u_1$, which by induction provides a derivation of $\Sigma_X\Gamma \vdash_I t^+ : \mathsf{Path}^i\ A^+\ u_0^+\ u_1^+$, hence we have a derivation of

$$\Sigma_X\Gamma \vdash_{I,j} t^+\ j : A^+[j/i]$$

Moreover, the induction also provides a derivation of $\Sigma_X\Gamma \vdash_I \Sigma t : \Sigma^{t,t^+}(\mathsf{Path}^i\ A\ u_0\ u_1)$, which by definition of the extrusion gives $\Sigma_X\Gamma \vdash_I \Sigma t : \mathsf{Path}^i\ \Sigma^{(t\ i),(t^+\ i)} A\ \Sigma(u_0)\ \Sigma(u_1)$. Hence, we have constructed a derivation of

$$\Sigma_X\Gamma \vdash_{I,j} \Sigma t\ j : \Sigma^{(t\ i),(t\ ^+i)}(A[j/i])$$

□

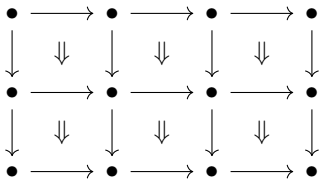
6.2 Type theory for cubical ω-categories

We now define a type theory that we call $\mathsf{CaTT}_\square$ to describe a structure of weak ω-categories in the cubical setting described by the theory $T_\square$, similar to our definition of the type theory CaTT.

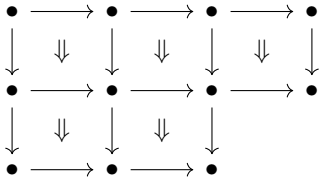
6.2.1 Ps-contexts

We start by defining a notion of ps-context adapted to the cubical framework, analogous to the notion of ps-contexts for globular categories. Intuitively this notion classifies contexts that form

a complete grid, as in the following examples



and rejects the context that form a grid where a square is missing, even if it is a square in a corner, as in the following example



Judgments. We introduce two new judgments to describe these ps-contexts, that we denote and interpret as follows

$\Gamma \vdash_{\mathsf{ps}}$	The context Γ is a ps-context
$\Gamma \vdash_{\mathsf{ps}} X$	The context Γ is a ps-context whose dangling border is X

The judgment $\Gamma \vdash_{\mathsf{ps}} X$ has a set of variables X on the right, intuitively, we build ps-contexts dimension by dimension, and we accumulate the information about the border of the context Γ in the set X, in order to always extend the context along a complete border.

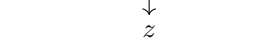
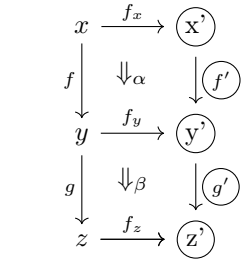
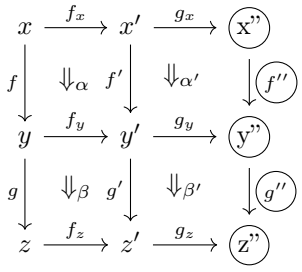
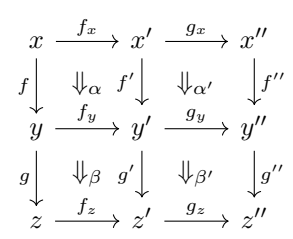
Inference rules. We now define the rules to check that a given context is a ps-context

$$\frac{}{x : \star \vdash_{\mathsf{ps}}}(\text{PSS}) \qquad \frac{\Gamma \vdash_{\mathsf{ps}} X}{\Sigma_X \Gamma \vdash_{\mathsf{ps}} X^+}(\text{PSE})$$

$$\frac{\Gamma \vdash_{\mathsf{ps}} X}{\Gamma \vdash_{\mathsf{ps}}}(\text{PS}) \qquad \frac{\Gamma \vdash_{\mathsf{ps}}}{\Sigma\Gamma \vdash_{\mathsf{ps}} \mathrm{Var}(\Gamma)^+}(\text{PS}+)$$

where the set X^+ is the set $\{x^+ : x \in X\}$, denoting x^+ the new copy of X coming from the extrusion on the left of the rule. Note that we denote x^+ a new variable that does not need to be literally named x^+. This is important because we might compute the extrusion with respect to the same variable x several times, producing a new copy of x every time, all of them having different names, and thus the notation x^+ is always given relatively to a particular extrusion. We now give an example of a derivation for this judgment, where in the graphical representation,

we circle all the variables that appear in the set X of the judgment $\Gamma \vdash_{\mathsf{ps}} X$

graphical representation	judgment
	$(x : \star) \vdash_{\mathsf{ps}}$
	$(x : \star, y : \star, f : \mathsf{Path}^i\ \star\ x\ y) \vdash_{\mathsf{ps}} \{y\}$
	$(x : \star, y : \star, f : \mathsf{Path}^i\ \star\ x\ y, z : \star, g : \mathsf{Path}^i\ \star\ y\ z) \vdash_{\mathsf{ps}} \{z\}$
	$(x : \star, y : \star, f : \mathsf{Path}^i\ \star\ x\ y, z : \star, g : \mathsf{Path}^i\ \star\ y\ z) \vdash_{\mathsf{ps}}$
	$(x : \star, y : \star, f : \mathsf{Path}^i\ \star\ x\ y, z : \star, g : \mathsf{Path}^i\ \star\ y\ z,$ $x' : \star, y' : \star, f' : \mathsf{Path}^i\ \star\ x'\ y', z' \star, g' : \mathsf{Path}^i\ \star\ y'\ z',$ $f_x : \mathsf{Path}^i\ \star\ x\ x',$ $f_y : \mathsf{Path}^i\ \star\ y\ y', \alpha : \mathsf{Path}^i\ \mathsf{Path}^j\ \star\ (f\ i)\ (f'\ i)\ f_x\ f_y,$ $f_z : \mathsf{Path}^i\ \star\ z\ z', \beta : \mathsf{Path}^i\ \mathsf{Path}^j\ \star\ (g\ i)\ (g'\ i)\ f_y\ f_z)$ $\vdash_{\mathsf{ps}} \{x', y', f', z', g'\}$
	$(x : \star, y : \star, f : \mathsf{Path}^i\ \star\ x\ y, z : \star, g : \mathsf{Path}^i\ \star\ y\ z,$ $x' : \star, y' : \star, f' : \mathsf{Path}^i\ \star\ x'\ y', z' \star, g' : \mathsf{Path}^i\ \star\ y'\ z',$ $f_x : \mathsf{Path}^i\ \star\ x\ x',$ $f_y : \mathsf{Path}^i\ \star\ y\ y', \alpha : \mathsf{Path}^i\ \mathsf{Path}^j\ \star\ (f\ i)\ (f'\ i)\ f_x\ f_y,$ $f_z : \mathsf{Path}^i\ \star\ z\ z', \beta : \mathsf{Path}^i\ \mathsf{Path}^j\ \star\ (g\ i)\ (g'\ i)\ f_y\ f_z,$ $x'' : \star, y'' : \star, f'' : \mathsf{Path}^i\ \star\ x''\ y'', z'' \star, g'' : \mathsf{Path}^i\ \star\ y''\ z'',$ $g_x : \mathsf{Path}^i\ \star\ x'\ x'',$ $g_y : \mathsf{Path}^i\ \star\ y'\ y'', \alpha' : \mathsf{Path}^i\ \mathsf{Path}^j\ \star\ (f'\ i)\ (f''\ i)\ g_x\ g_y,$ $g_z : \mathsf{Path}^i\ \star\ z'\ z'', \beta' : \mathsf{Path}^i\ \mathsf{Path}^j\ \star\ (g'\ i)\ (g''\ i)\ g_y\ g_z)$ $\vdash_{\mathsf{ps}} \{x'', y'', f'', z'', g''\}$
	$(x : \star, y : \star, f : \mathsf{Path}^i\ \star\ x\ y, z : \star, g : \mathsf{Path}^i\ \star\ y\ z,$ $x' : \star, y' : \star, f' : \mathsf{Path}^i\ \star\ x'\ y', z' \star, g' : \mathsf{Path}^i\ \star\ y'\ z',$ $f_x : \mathsf{Path}^i\ \star\ x\ x',$ $f_y : \mathsf{Path}^i\ \star\ y\ y', \alpha : \mathsf{Path}^i\ Pathj \star (f\ i)(f'\ i)\ f_x\ f_y,$ $f_z : \mathsf{Path}^i\ \star\ z\ z', \beta : \mathsf{Path}^i\ \mathsf{Path}^j\ \star\ (g\ i)\ (g'\ i)\ f_y\ f_z,$ $x'' : \star, y'' : \star, f'' : \mathsf{Path}^i\ \star\ x''\ y'', z'' \star, g'' : \mathsf{Path}^i\ \star\ y''\ z'',$ $g_x : \mathsf{Path}^i\ \star\ x'\ x'',$ $g_y : \mathsf{Path}^i\ \star\ y'\ y'', \alpha' : \mathsf{Path}^i\ \mathsf{Path}^j\ \star\ (f'\ i)\ (f''\ i)\ g_x\ g_y,$ $g_z : \mathsf{Path}^i\ \star\ z'\ z'', \beta' : \mathsf{Path}^i\ \mathsf{Path}^j\ \star\ (g'\ i)\ (g''\ i)\ g_y\ g_z) \vdash_{\mathsf{ps}}$

Sources and targets of a pasting scheme. Similarly to the case of CaTT, we define notions of sources and targets of a ps-context, but since we are now working in a cubical framework,

a ps-context may have more than one source and one target. More specifically, we introduce the judgment $\Gamma \vdash_{\mathsf{ps},n}$ to be equivalent to $\Gamma \vdash_{\mathsf{ps}}$ and $\dim \Gamma = n$. We can then define n source and targets for any ps-context Γ such that $\Gamma \vdash_{\mathsf{ps},n}$, we denote $\partial_0^-(\Gamma), \dots, \partial_{n-1}^-(\Gamma)$ for the n sources and $\partial_0^+(\Gamma), \dots, \partial_{n-1}^+(\Gamma)$ for the n targets. We define these operations by mutual induction on the derivation of a judgment of the form $\Gamma \vdash_{\mathsf{ps}} X$, and on the derivation of a judgment of the form $\Gamma \vdash_{\mathsf{ps}}$. We first define the sources, with the condition that $i < \dim \Gamma$

For the judgment $\Gamma \vdash_{\mathsf{ps}} X$:

- For a derivation obtained by application of the rule (PSE), the judgment is necessarily of the form $\Sigma_X \Gamma \vdash_{\mathsf{ps}} X^+$ and we have a derivation of $\Gamma \vdash_{\mathsf{ps}} X$, and we define

$$\partial_i^-(\Sigma_X \Gamma \vdash_{\mathsf{ps}} X^+) = \begin{cases} \partial_0^-(\Gamma \vdash_{\mathsf{ps}} X) & \text{If } i = 0 \\ \Sigma_{X \cap \mathrm{Var}(\partial_i^-(\Gamma))}(\partial_i^-(\Gamma \vdash_{\mathsf{ps}} X)) & \text{Otherwise} \end{cases}$$

- For a derivation obtained by the rule (PS+), the judgment is necessarily of the form $\Sigma\Gamma \vdash_{\mathsf{ps}} \mathrm{Var}(\Gamma)^+$ and we have a derivation of $\Gamma \vdash_{\mathsf{ps}}$, and we then pose

$$\partial_0^-(\Gamma \vdash_{\mathsf{ps}} \mathrm{Var}(\Gamma)) = \Gamma \qquad \partial_{i+1}^-(\Gamma \vdash_{\mathsf{ps}} \mathrm{Var}(\Gamma)) = \partial_i^-(\Gamma \vdash_{\mathsf{ps}})$$

For the judgment $\Gamma \vdash_{\mathsf{ps}}$:

- For a derivation obtained as a single application of the rule (PSS), the context is necessarily $(x : \star)$, which is of dimension 0, and the requirement that $i < 0$ implies that there is no source to define in this case.
- For a derivation obtained by application of the rule (PS), we necessarily have a derivation of $\Gamma \vdash_{\mathsf{ps}} X$, and we pose

$$\partial_i^-(\Gamma \vdash_{\mathsf{ps}}) = \partial_i^-(\Gamma \vdash_{\mathsf{ps}} X)$$

Similarly, we define the notion of target of a ps-context by induction mutual on the derivation of a judgment $\Gamma \vdash_{\mathsf{ps}} X$ and on the derivation of the judgment $\Gamma \vdash_{\mathsf{ps}}$, for all $i < \dim \Gamma$.

For the judgment $\Gamma \vdash_{\mathsf{ps}} X$:

- For a derivation obtained by application of the rule (PSE), the judgment is necessarily of the form $\Sigma_X \Gamma \vdash_{\mathsf{ps}} X^+$ and we have a derivation of $\Gamma \vdash_{\mathsf{ps}} X$, and we define

$$\partial_i^+(\Sigma_X \Gamma \vdash_{\mathsf{ps}} X^+) = \begin{cases} X^+ & \text{If } i = 0 \\ \Sigma_{X \cap \mathrm{Var}(\partial_i^+(\Gamma))}(\partial_i^+(\Gamma \vdash_{\mathsf{ps}} X)) & \text{Otherwise} \end{cases}$$

- For a derivation obtained by the rule (PSF), the judgment is necessarily of the form $\Sigma\Gamma \vdash_{\mathsf{ps}} \mathrm{Var}(\Gamma)^+$ and we have a derivation of $\Gamma \vdash_{\mathsf{ps}}$, and we then pose

$$\partial_0^+(\Gamma \vdash_{\mathsf{ps}} \mathrm{Var}(\Gamma)) = \Gamma \qquad \partial_{i+1}^+(\Gamma \vdash_{\mathsf{ps}} \mathrm{Var}(\Gamma)) = \partial_i^+(\Gamma \vdash_{\mathsf{ps}})$$

For the judgment $\Gamma \vdash_{\mathsf{ps}}$:

- For a derivation obtained as a single application of the rule (PSS), the context is necessarily $(x : \star)$, which is of dimension 0, and the requirement that $i < 0$ implies again that there is no target to define in this case.
- For a derivation obtained by application of the rule (PS), we necessarily have a derivation of $\Gamma \vdash_{\mathsf{ps}} X$, and we pose

$$\partial_i^+(\Gamma \vdash_{\mathsf{ps}}) = \partial_i^+(\Gamma \vdash_{\mathsf{ps}} X)$$

Uniqueness of derivation. The judgments $\Gamma \vdash_{\mathsf{ps}}$ and $\Gamma \vdash_{\mathsf{ps}} X$ enjoy uniqueness of derivations. Indeed, suppose that a context Γ is of length l, and that the set X is a subset of variables in Γ downwards closed of cardinal k, then the context $\Sigma_X \Gamma$ is of length $l + 2k$. Moreover, in the process of constructing a ps-context, each consecutive extrusion, so counting the number n of variables of maximal dimension gives the number of consecutive extrusion. The length of the context is then given by $n\ell(\Gamma')$, where Γ' is the context to which the rule (PSF) is applied. This shows that there is only one possible way to obtain a derivation of $\Gamma \vdash_{\mathsf{ps}}$ from a derivation of $\Gamma' \vdash_{\mathsf{ps}}$. Iterating this construction shows that there is a unique derivation for $\Gamma \vdash_{\mathsf{ps}}$. Moreover, this argument gives an algorithm to check whether a context is a ps-context or not, by testing if the only possible derivation is valid or not.

Typing rules for term constructors. We introduce two terms constructors that we again call op and coh, and whose introduction are, for the constructor op:

$$\frac{\Gamma \vdash_{\mathsf{ps},n} \qquad \partial_i^- \Gamma \vdash t_i : A_i \qquad \partial_i^+ \Gamma \vdash u_i : B_i \qquad \Gamma \vdash t_i \to u_i \qquad \Delta \vdash \gamma : \Gamma}{\Delta \vdash \mathsf{op}_{\Gamma,(t_i)\to(u_i)}[\gamma] : ((t_i) \to (u_i))\,[\gamma]}$$

where we denote $(t_i) \to (u_i)$ as a shorthand for $(t_0, \dots, t_{n-1}) \to (u_0, \dots, u_{n-1})$. This rule applies under the additional side conditions that for all i the following equalities hold on the variables, $\mathrm{Var}(t_i) \cup \mathrm{Var}(A_i) = \mathrm{Var}(\partial_i^- \Gamma)$ and $\mathrm{Var}(u_i) \cup \mathrm{Var}(B_i) = \mathrm{Var}(\partial_i^+ \Gamma)$, we denote this side condition $(\mathrm{C}_{\mathrm{cohop},\square})$. For the constructor coh, we give the rule

$$\frac{\Gamma \vdash_{\mathsf{ps}} \qquad \Gamma \vdash A \qquad \Delta \vdash \gamma : \Gamma}{\Delta \vdash \mathsf{coh}_{\Gamma,A}[\gamma] : A\,[\gamma]}$$

and this rule again applies under the condition that for all i, the equalities hold on the variables: $\mathrm{Var}(A) = \mathrm{Var}(\Gamma)$, we denote this side condition $(\mathrm{C}_{\mathrm{coh},\square})$. We refer the reader to Appendix A.6 for a standalone presentation of all the rules of this theory.

6.2.2 Examples of derivation

We present in this section various coherences that are derivable in this theory. We have a prototypical implementation of this type theory available at [13], and the examples we provide here are all formalized and computer-checked. Since the rules of the theory involve more computation, and in particular one needs to compute all the sources and targets of every ps-contexts, which is computationally significantly heavier than computing the source and target of a ps-context in the globular setting, an implementation is a valuable tool to ensure that all expected operations and coherences are indeed derivable. Note that our implementation does not feature yet the implicit arguments, which are by far the most efficient way to reduce the size of the terms, and as a result, the terms that we present here tend to be very long.

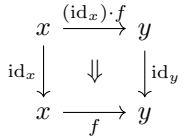
Identity and composition. The first few examples that we give are similar to the examples in CaTT, as they apply only in low dimension, where CaTT and $\mathsf{CaTT}_\square$ are the same. For instance, we can define the identity 1-cell of a 0-cell as follows

```
coh id (x : *) : Path i * x x
```

and similarly we can define the composition of two composable 1-cells by

```
coh comp (x : *) (y : *) (f : Path i * x y)
                 (z : *) (g : Path i * y z) :
   Path i * x z
```

From now on, we denote $f \cdot g$ for the composition of f and g and id_x for the identity of x when we present the terms that we derive in the theory. Of course in the core definition, we still use the names `id` and `comp` that we have introduced, in order to produce valid expressions of the theory. Showing the unitality of composition is a bit different, it is witnessed by a 2-cell, which then needs to be a square. In order to generate this square, we insert identity 1-cells for the 0-source and the 0-target, thus describing, for instance the left unitality as the following diagram

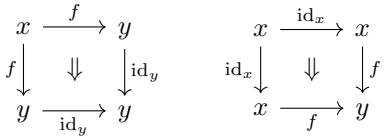


This diagram can be encoded as the following expression in the theory $\mathsf{CaTT}_{\square}$.

```
coh unit-l (x : *) (y : *) (f : Path i * x y) :
   Path i (Path j * ((id x) i) ((id y) i)) (comp x x (id x) y f) f
```

and we can do the same to derive the right unitality.

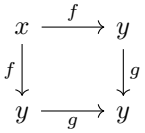
Connections and symmetry. In addition to the usual composition, there are some specific phenomena that appear in the cubical theory and that do not have an equivalent in globular. An instance of such coherences are the *connections*, which given a 1-cell f from x to y produces the two following 2-cells



These can both be defined in the theory as follows

```
coh connexion1 (x : *) (y : *) (f : Path i * x y) :
    Path i (Path j * (f i) ((id y) i)) f (id y)
coh connexion2 (x : *) (y : *) (f : Path i * x y) :
    Path i (Path j * ((id x) i) (f i)) (id x) f
```

Another instance is the symmetry, which given two composable cells f and g, produces a 2-cell as in the following diagram

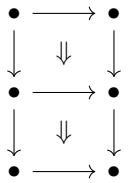


This can be thought of as a form of identity, and we can define it in the theory as follows

```
coh id-sym (x : *) (y : *) (f : Path i * x y) (z : *) (g : Path i * y z) :
    Path i (Path j * (f i) (g i)) f g
```

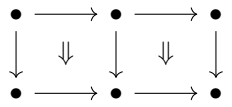
Composition of 2-cells. We now define the composition of the 2-cells, that are mostly analogous to the compositions of the 2-cells in CaTT. First we consider the *vertical composition*,

which acts on two 2-cells in the following configuration



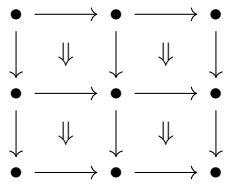
```
coh vcomp (x : *) (x' : *) (hx : Path i * x x')
                  (x'' : *) (hx' : Path i * x' x'')
         (y : *) (y' : *) (hy : Path i * y y')
                 (y'' : *) (hy' : Path i * y' y'')
                (f : Path i * x y)
                  (f' : Path i * x' y')
                     (a : Path i (Path j * (hx i) (hy i)) f f')
                (f'' : Path i * x'' y'')
                     (b : Path i (Path j * (hx' i) (hy' i)) f' f'') :
  Path i
    (Path j * ((comp x x' hx x'' hx') i) ((comp y y' hy y'' hy') i))
    f f''
```

Similarly, we also define the horizontal composition, that gives a result for composing two 2-cells in the following configuration



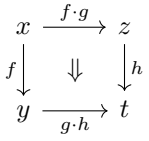
```
coh hcomp (x : *) (x' : *) (hx : Path i * x x')
          (y : *) (y' : *) (hy : Path i * y y')
  (f : Path i * x y) (f' : Path i * x' y')
          (a : Path i (Path j * (hx i) (hy i)) f f')
  (z : *) (z': * ) (hz : Path i * z z')
  (g : Path i * y z) (g' : Path i * y' z')
          (b : Path i (Path j * (hy i) (hz i)) g g') :
Path i
  (Path j * (hx i) (hz i))
  (comp x y f z g) (comp x' y' f' z' g')
```

We also define a term that we call `hvcomp` consisting in composing four 2-cell in the configuration



We use this composition for later definitions, but we do not give its full definition here, as it is very long and similar to the one we have already given. The interested reader look the definition up in our implemented formalization [13].

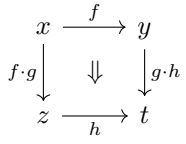
Associativity. We can define an associativity witness for the composition of 1-cells: Given three composable 1-cells f, g and h, we always have a 2-cell as represented in the following diagram



We can define this as a term in our theory

```
coh assoc (x : *) (y : *) (f : Path i * x y)
                  (z : *) (g : Path i * y z)
                  (w : *) (h : Path i * z w):
  Path i (Path j * (f i) (h i)) (comp x y f z g) (comp y z g w h)
```

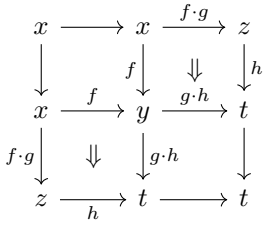
In the case of globular categories, the associativity is invertible, and we can directly define its inverse. In the case of $\mathsf{CaTT}_{\square}$ there is an analogue to this phenomenon, that is given by the symmetric of the associativity. Diagrammatically, we describe it as follows



and we define it as follows

```
coh assoc-sym (x : *) (y : *) (f : Path i * x y)
                      (z : *) (g : Path i * y z)
                      (w : *) (h : Path i * z w):
  Path i (Path j * ((comp x y f z g) i) ((comp y z g w h) i)) f h
```

We can define a witness that the associativity and its symmetric cancel each other, but it is a little involved. Indeed, in order to compose the associativity and its symmetric, we fit them in the following diagram



where the two additional 2-cells are connections. We can then relate this composition to an identity, using a 3-cell, that is definable as follows

```
coh assoc-can
    (x : *) (y : *) (f : Path i * x y)
            (z : *) (g : Path i * y z)
            (w : *) (h : Path i * z w) :
    Path i
      (Path j
       (Path k *
```

```
      (((unit-l x z (comp x y f z g)) i) j) (((unit-r z w h) i) j))
     ((unit-l x z (comp x y f z g)) i) ((unit-r z w h) i))
    (hvcomp x x (id x) z (comp x y f z g)
        x y f w (comp y z g w h)
     (id x) f (connexion2 x y f) h (assoc-sym x y f z g w h)
    z w h w (id w)
     (comp x y f z g) (comp y z g w h) (assoc x y f z g w h)
     (id w) (connexion1 y w (comp y z g w h)))
    (id-sym x z (comp x y f z g) w h)
```

Second associativity. We can also define a second 2-cell that witnesses the associativity, which given three composable 1-cell can be described with the the diagram

$$\begin{array}{ccc} x & \xrightarrow{(f\cdot g)\cdot h} & t \\ \downarrow{\scriptstyle \mathrm{id}_x} & \Downarrow & \downarrow{\scriptstyle \mathrm{id}_t} \\ x & \xrightarrow[f\cdot(g\cdot h)]{} & t \end{array}$$

we define the corresponding term as follows

```
coh assoc-bis (x : *) (y : *) (f : Path i * x y)
                      (z : *) (g : Path i * y z)
                      (w : *) (h : Path i * z w) :
    Path i (Path j * ((id x) i) ((id w) i))
    (comp x z (comp x y f z g) w h)
    (comp x y f w (comp y z g w h))
```

We can also define a witness that relates the two different cells that we have defined for the associativity, as follows (where `hcomp3` is the ternary horizontal composition of 2-cells, that we have not introduced, but have defined in our implementation).

```
coh equiv-assoc (x : *) (y : *) (f : Path i * x y)
                        (z : *) (g : Path i * y z)
                        (w : *) (h : Path i * z w) :
    Path i
        (Path j
              (Path k * (((id2 x x (id x)) i) j)
                         (((id2 w w (id w)) i) j))
        ((unit-l3 x z (comp x y f z g) w h) i)
         ((unit-r3 x y f w (comp y z g w h)) i))
    (hcomp3 x x (id x) x y f (id x) f
                (connexion2 x y f) z w h
                (comp x y f z g) (comp y z g w h)
                (assoc x y f z g w h)
                w w (id w) h (id w) (connexion1 z w h))
    (assoc-bis x y f z g w h)
```

In our implementation, we have also formalized the exchange rule, that we do not present here, as the corresponding term is very long.

6.2.3 Comparison with cubical type theory

Although the theory that we have introduced here is directly inspired by cubical type theory [29] in particular for the type Path , it does not compare immediately to it. The first difference is that in cubical type theory, the dimension variables enjoy an extra structure, corresponding to the negation, the meet and the join, whereas in the version we propose, these operations do not exist. Instead, the cells that are computed using the meet and the join in cubical type theory can be derived using the coherence rules in our version, hence our version is weaker. Moreover, cubical type theory is not directed, the negation witnesses this fact, where our theory is directed. A valuable further work would be to relate the theory $\mathsf{CaTT}_{\square}$ with cubical type theory in a precise way, accounting for the divergence between the strictness and the directedness of the theory. This would establish a connection between our term constructors for operations and coherences, and the term constructors for the gluing and Kan filling operations introduced in the cubical type theory.

6.3 Pre-cubical weak ω category

Our goal is now to give a full description of the category $\Theta_{\square,\infty}$ as a universal property construction the same way we have given a description of the category Θ_∞ in the Grothendieck-Maltsiniotis definition of globular weak ω-categories. We achieve this description by applying our framework for type theories, and under a conjecture for characterizing the syntactiic category as the opposite of the category $\Theta_{\square,\infty}$.

6.3.1 Monoidal structure of pre-cubical Sets

In order to understand the category $\Theta_{\square,\infty}$, we first exhibit a description of the cubical ps-contexts in the category of finite pre-cubical sets. We describe a monoidal structure on the category of pre-cubical sets which is helpful to achieve such a description.

The monoidal structure of $\square$. The category $\square$ is equipped with a monoidal structure $\otimes : \square \times \square \to \square$, defined on objects by

$$n \otimes m = n + m$$

On morphisms, the monoidal product is generated, for $\sigma_i\tau_i : n \to n+1$, by

$$\sigma_i \otimes \mathrm{id}_m = \sigma_i : n+m \to n+m+1 \qquad \tau_i \otimes \mathrm{id}_m = \tau_i : n+m \to n+m+1$$
$$\mathrm{id}_m \otimes \sigma_i = \sigma_i : m+n \to m+n+1 \qquad \mathrm{id}_m \otimes \tau_i = \tau_i : m+n \to m+n+1$$

Day convolution. The monoidal structure on $\square$ induces a monoidal structure on **CSet** given by the *Day convolution*, defined by

$$(X \otimes Y) = \int^{p,q:\square} X_p \times Y_q \times \square(\_, p+q)$$

In the case of the pre-cubical sets, this operation simplifies to give a very explicit description of the resulting presheaf. Indeed, the coequalizer characterization of the coend, along with the pointwise computations in the presheaves give

$$(X \otimes Y)_n = \mathrm{coeq}\left(\coprod_{\substack{g:p\to p' \\ h:q\to q'}} X_{p'} \times Y_{q'} \times \square(n, p+q) \rightrightarrows \coprod_{\substack{p \\ q}} X_p \times Y_q \times \square(n, p+q) \right)$$

In the category **Set**, the coequalizer can be computed as a quotient set, which gives the following expression

$$(X \otimes Y)_n = \bigsqcup_{p,q} X_p \times Y_q \times \square(n, p+q) / \simeq$$

where $\simeq$ is the equivalence relation defined, for two tuples $(x, y, f) \in X_p \times Y_q \times \square(n, p+q)$ and $(x', y', f') \in X_{p'} \times Y_{q'} \times \square(n, p'+q')$, by $(x, y, f) \simeq (x', y', f')$ if and only if there exists two maps $g : p \to p', h : q \to q'$ such that $X(g)(x') = x$, $Y(h)(y') = y$ and $(g \otimes h) \circ f = f'$. Notice that the set $\square(n, p+q)$ is non empty exactly when $p + q \geq n$. Suppose that $p + q > n$, let (x, y, f) be an element of $X_p \times Y_q \times \square(n, p+q)$. The morphisms of $\square$ being generated by σ_i, τ_i, there exists a factorization of f as $f = \sigma_i f'$ or $f = \tau_i f'$.

- If $f = \sigma_i f'$ with $i < p$, then $(x, y, f) \simeq (\partial_i^- x, y, f')$.
- If $f = \tau_i f'$ with $i < p$, then $(x, y, f) \simeq (\partial_i^+ x, y, f')$.
- If $f = \sigma_i f'$ with $p \leq i < q$, then $(x, y, f) \simeq (x, \partial_{i-p}^- y, f')$.
- If $f = \tau_i f'$ with $p \leq i < q$, then $(x, y, f) \simeq (x, \partial_{i-p}^+ y, f')$.

So in all the cases, there exists $(x', y', f') \in X_{p'} \times Y_{q'} \times \square(n, p'+q')$, such that $(x, y, f) \simeq (x', y', f')$, and with $p' + q' < p + q$. By iterating this construction, we prove that any equivalence class of $(X \otimes Y)_n$ contains at least an element in $X_p \times Y_q \times \square(n, p+q)$, with $p + q = n$. Moreover, since $\square(n, n) = \{\mathrm{id}\}$, there is an equality $X_p \times Y_q \times \square(n, p+q) = X_p \times Y_q$ whenever $p + q = n$. If $(x, y) \in X_p \times Y_q$ and $(x', y') \in X_{p'} \times Y_{q'}$, with $p + q = p' + q' = n$, then $(x, y) \simeq (x', y')$ if there exists a pair of maps $g : p \to p'$ and $h : q \to q'$ satisfying the previous equalities. Such maps can only exist if $p \leq p'$ and $q \leq q'$, but the equality $p + q = p' + q'$ then implies $p = p'$ and $q = q'$, and so $f = \mathrm{id}_p$ and $g = \mathrm{id}_q$, which implies $(x, y) = (x', y')$. This proves the equality

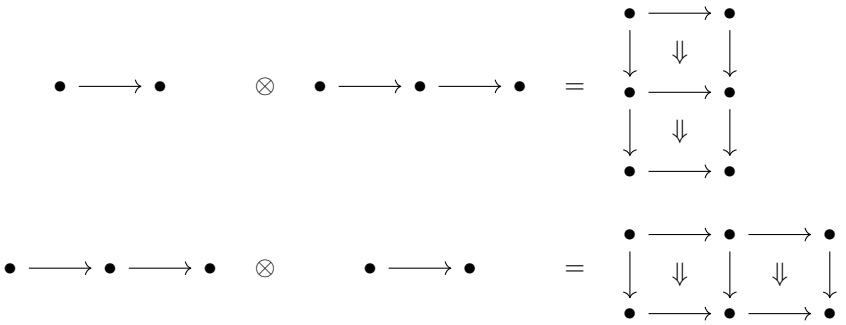
$$(X \otimes Y)_n = \bigsqcup_{p+q=n} X_p \times Y_q$$

The sources and targets maps are given by, for $z \in (X \otimes Y)_n$, let p, q and $x_p \in X_p, y_q \in Y_q$ such that $z = (x_p, y_q) \in X_p \times Y_q$, then

$$\partial_i^- z = \begin{cases} (\partial_i^-(x_p), y_q) & \text{for } 0 \leq i < p \\ (x_p, \partial_{p+i}^-(y_p)) & \text{for } 0 \leq i < q \end{cases}$$

$$\partial_i^+ z = \begin{cases} (\partial_i^+(x_p), y_q) & \text{for } 0 \leq i < p \\ (x_p, \partial_{p+i}^+(y_p)) & \text{for } 0 \leq i < q \end{cases}$$

For instance, with the diagrammatic notation, we have

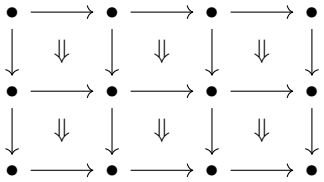


Pasting schemes. We denote Z_k the finite cubical set defined by $(Z_k)_0 = \{0, \dots, k\}$, and $(Z_k)_1 = \left\{\vec{1}, \dots, \vec{k}\right\}$ and $(Z_k)_n = \emptyset$ for $n > 1$, together with the applications $\partial_0^-(i) = i$ and $\partial_0^+(i) = i + 1$. Intuitively, they are cubical sets of dimension 1 constituted in k composable 1-cells. We can illustrate this definition with our diagrammatic notations as follows

$$\begin{array}{rl} Z_0 = \mathrm{Y}(0) & \bullet \\ Z_1 = \mathrm{Y}(1) & \bullet \longrightarrow \bullet \\ Z_2 & \bullet \longrightarrow \bullet \longrightarrow \bullet \\ Z_3 & \bullet \longrightarrow \bullet \longrightarrow \bullet \longrightarrow \bullet \end{array}$$

Definition 167. We define a *cubical pasting scheme* (or simply pasting scheme) to be either the cubical set Z_0, or a cubical set obtained as a finite Day convolution of the Z_k, with $k > 0$.

For instance, the expression $Z_2 \otimes Z_3$ defines a ps-context, which corresponds to the following diagram



Note that for $k = 0$, the cubical set Z_0 is the unit of the Day tensor product, and hence given a tensor containing the set Z_0, one can simply remove it to exhibit the resulting cubical set as a tensor which does not contain Z_0.

Sources and targets of pasting schemes. Given a pasting scheme X which is not Z_0, we write $X = Z_{k_1} \otimes \ldots \otimes Z_{k_n}$, we define its *$i$-th border* for $i \leq n$, denoted $\partial_{i,X}$, by the formula

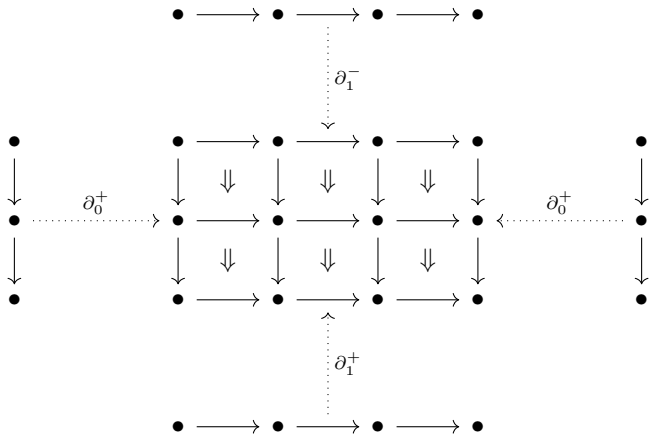
$$\partial_{i,X} = Z_{k_1} \otimes \ldots \otimes Z_{i-1} \otimes Z_0 \otimes Z_{i+1} \otimes \ldots \otimes Z_{k_n}$$

For $k > 0$, the cubical set Z_k comes naturally equipped with two maps $\partial^-_{0,Z_k} : Z_0 \to Z_k$ and $\partial^+_{0,Z_k} : Z_0 \to Z_k$, defined by $\partial^-_{0,Z_k}(0) = 0$ and $\partial^+_{0,Z_k}(0) = k$. We can give a diagrammatic definition as follows

These two maps induce two maps $\partial^-_{i,X} : \partial_{i,X} \to X$ and $\partial^+_{i,X} : \partial_{i,X} \to X$, that we call respectively the *i-th source* and the *i-th target* of the pasting scheme X. These maps are defined by

$$\partial_i^- = Z_{k_1} \otimes \ldots \otimes Z_{i-1} \otimes \partial^-_{0,Z_i} \otimes Z_{i+1} \otimes \ldots \otimes Z_{k_n}$$
$$\partial_i^+ = Z_{k_1} \otimes \ldots \otimes Z_{i-1} \otimes \partial^+_{0,Z_i} \otimes Z_{i+1} \otimes \ldots \otimes Z_{k_n}$$

We can again give a graphical intuition for the i-th source and the i-th target, with our previous example



Proposition 168. *For a pasting scheme X, and for all $i < \dim X$, the cubical set $\partial_i X$ is a pasting scheme.*

Proof. For the pasting scheme Z_0, we have $\dim Z_0 = 0$, thus there is no i such that $i < \dim Z_0$. For a pasting scheme of the form Z_k, with $k > 0$, we have $\dim Z_k = 1$, and by definition, $\partial_{0,Z_k} = Z_0$ is again a pasting scheme. For a pasting scheme of the form $X = Z_{k_1} \otimes \ldots \otimes Z_{k_n}$ with $k_1, \ldots, k_n \neq 0$, we have by definition $\partial_{i,X} = Z_{k_1} \otimes \ldots \otimes Z_{i-1} \otimes Z_0 \otimes Z_{i+1} \ldots \otimes Z_{k_n}$. Since Z_0 is the unit for the Day tensor product, this simplifies to $\partial_{i,X} = Z_{k_1} \otimes \ldots \otimes Z_{i-1} \otimes Z_{i+1} \ldots \otimes Z_{k_n}$, which exhibits $\partial_{i,X}$ as a pasting scheme. □

Nerve of an extrusion. We can use the Day convolution product in order to express the nerve of an extruded context in the theory $T_\square$. This is an important computation to establish the connection between pasting schemes and ps-contexts. Recall that there is a nerve functor $\nu : \mathcal{S}_{T_\square}^{\mathrm{op}} \to \mathbf{FinCSet}$, given by, for all context Γ, $\nu(\Gamma)_n = \mathcal{S}_{T_\square}(\Gamma, \mathrm{Y}(n))$. Consider a context Γ in the theory $T_\square$, and X a downwards closed set of elements of $\nu(\Gamma)$, then $\nu(\Gamma)$ can be written as the colimit of the diagram

$$\begin{array}{ccc} & \nu(\Gamma)_X & \\ \swarrow & & \searrow \\ \nu(\Gamma) & & \nu(\Gamma)_X \end{array}$$

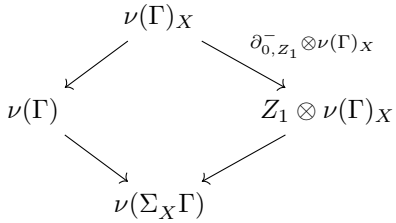
where $\nu(\Gamma)_X$ is the sub-cubical set $\nu(\Gamma)$ whose elements are in X.

Lemma 169. *The nerve of the extrusion $\nu(\Sigma_X \Gamma)$ is characterized as the following colimit*

$$\nu(\Sigma_X \Gamma) = \mathrm{colim} \left(\begin{array}{ccc} & \nu(\Gamma)_X & \\ & \swarrow \quad \searrow^{\partial^-_{0,Z_1} \otimes \nu(\Gamma)_X} & \\ \nu(\Gamma) & & Z_1 \otimes \nu(\Gamma)_X \end{array} \right)$$

Moreover, the set X^+ defines the sieve of all the elements of the sub-cubical set characterized by $\partial^+_{0,Z_1} \otimes \nu(\Gamma)_X : \nu(\Gamma)_X \to Z_1 \otimes \nu(\Gamma)_X \to \nu(\Sigma_X \Gamma)$

Proof. First note that we can construct a commutative diagram



by choosing the map $\nu(\Gamma) \to \nu(\Sigma_X \Gamma)$ to be the inclusion: to each variable x it associates x, and the map $Z_1 \otimes \nu(\Gamma)_X \to \nu(\Sigma_X \Gamma)$ is the map defined, for all $x \in X$ by the associations $(0, x) \mapsto x$, $(1, x) \mapsto x^+$ and $(\vec{1}, x) \mapsto \vec{x}$. We now check that this square is a pushout: consider a cubical set Y with two maps $f : \nu(\Gamma) \to Y$ and $g : Z_1 \otimes \nu(\Gamma)_X \to Y$ that make the diagram commute. Then we can define a map $h : \nu(\Sigma_X \Gamma)$ by setting, for each variable $x \in \Gamma$, $h(x) = f(x)$, and for each variable $x \in X$, $h(x) = g(0, x)$, $h(x^+) = g(1, x)$ and $h(\vec{x}) = g(\vec{1}, x)$. The fact that the square commute shows that these definitions agree on the variables $x \in X$, and it provides a unique map, hence proves the universal property of the pushout. □

In particular, when performing the extrusion with respect to all the variables in the context, we get the following result.

Lemma 170. *Given a context Γ in the theory $T_\square$, the nerve of its extrusion $\Sigma\Gamma$ is isomorphic to $\nu(\Sigma\Gamma) = Z_1 \otimes \nu(\Gamma)$. Moreover, the set of variable $\mathrm{Var}(\Gamma)^+$ correspond to the elements of the image of the sub-cubical set $\partial^+_{0,Z_1} \mathrm{Z}_0 \otimes \nu(\Gamma) \to Z_1 \otimes \nu(\Gamma)$ under this correspondence.*

Ps-contexts are pasting schemes. The category $\mathcal{S}_{T_\square}$ is equivalent to the opposite of the category of finite cubical sets. Among the contexts in $\mathcal{S}_{T_\square}$ we have exhibited a particular class of ps-contexts, characterized by the judgment $\vdash_{\mathsf{ps}}$. We now show that these particular contexts are pasting schemes.

Proposition 171. *Consider a context Γ in $T_\square$ such that $\Gamma \vdash_{\mathsf{ps}}$ is derivable, then the finite cubical set corresponding to Γ, $\nu(\Gamma)$ is a pasting scheme.*

Proof. We prove this proposition by mutual induction together with the fact that whenever the judgment Γ is derivable, the cubical set $\nu(\Gamma)$ is a pasting scheme such that the elements of the image of $\partial_{0,\nu(\Gamma)}$ are exactly the elements of X.

For the judgment $\Gamma \vdash_{\mathsf{ps}}$:

- For the judgment $(x : \star) \vdash_{\mathsf{ps}}$, obtained by the rule (PSS), we have $\nu(x : \star) = \mathrm{Y}(0) = Z_0$, hence $\nu(\times : \star)$ is a pasting scheme.
- For the judgment $\Gamma \vdash_{\mathsf{ps}}$ obtained by the rule (PSD), we necessarily have a derivation of the form $\Gamma \vdash_{\mathsf{ps}} X$, and by the induction case for this case, this proves in particular that $\nu(\Gamma)$ is a pasting scheme.

For the judgment $\Gamma \vdash_{\mathsf{ps}} X$:

- For the judgment $\Sigma\Gamma \vdash_{\mathsf{ps}} \mathrm{Var}(\Gamma)^+$ obtained by the rule (PS+), the judgment $\Gamma \vdash_{\mathsf{ps}}$ is necessarily derivable. By Lemma 170, the cubical set $\nu(\Sigma\Gamma)$ is obtained as the product $Z_1 \otimes \nu(\Gamma)$. By induction, that $\nu(\Gamma)$ is a pasting scheme, hence, either $\nu(\Gamma) = Z_0$, in

which case $Z_1 \otimes \nu(\Gamma) = Z_1$ is a ps-context, or $\nu(\Gamma) = Z_{k_1} \otimes \ldots \otimes Z_{k_n}$, in which case $Z_1 \otimes \nu(\Gamma) = Z_1 \otimes Z_{k_1} \otimes \ldots \otimes Z_{k_n}$ is also a pasting scheme. Moreover, Lemma 170 also shows that $\mathrm{Var}(\Gamma)^+$ characterizes the variables in the image of $\partial_{0,\nu(\Sigma\Gamma)}$.

– For the judgment $\Sigma\Gamma \vdash_{\mathsf{ps}} X^+$ obtained by the rule (PSE), the judgment $\Gamma \vdash_{\mathsf{ps}} X$ is necessarily derivable. Lemma 169 characterizes $\nu(\Sigma_X\Gamma)$ as the colimit

$$\nu(\Sigma_X\Gamma) = \mathrm{colim}\left(\begin{array}{ccc} & \nu(\Gamma)_X & \\ \swarrow & & \searrow \\ \nu(\Gamma) & & Z_1 \otimes \nu(\Gamma)_X \end{array}\right)$$

By induction, $\nu(\Gamma)$ is a pasting scheme of the form $Z_{k_1}\otimes, \ldots, Z_{k_n}$, and $\nu(\Gamma)_X = \partial_{0,\nu(\Gamma)}$, the inclusion $\nu(\Gamma)_X \to \nu(\Gamma)$ being the map $\partial^+_{0,\nu(\Gamma)}$. Rewriting this in the previous colimit and simplifying using the commutation of the Day convolution with the colimit yields

$$\nu(\Sigma_X\Gamma) = \mathrm{colim}\left(\begin{array}{ccc} & Z_0 & \\ {\scriptstyle \partial^+_{0,Z_{k_1}}}\swarrow & & \searrow{\scriptstyle \partial^-_{0,Z_1}} \\ Z_{k_1} & & Z_1 \end{array}\right) \otimes Z_{k_2} \otimes \ldots \otimes Z_{k_n}$$

The colimit now computes to Z_{k_1+1}, thus showing $\nu(\Sigma_X\Gamma) = Z_{k_1+1} \otimes Z_{k_2} \otimes \ldots \otimes Z_{k_n}$. Moreover, Lemma 169 also characterizes X^+ as the set of variables of the inclusion map $\partial^+_{0,Z,0} \otimes \mathrm{id} : Z_0 \otimes Z_{k_2} \otimes \ldots \otimes Z_{k_n} \to Z_1 \otimes Z_{k_2} \otimes \ldots \otimes Z_{k_n} \to \nu(\Sigma_X\Gamma)$, which is exactly the image of application $\partial^+_{0,\nu(\Sigma_X\Gamma)}$.

□

Global computation. From the previous proof, we can extract the formulas allowing to exhibit $\nu(\Gamma)$ as a pasting scheme from a derivation of $\Gamma \vdash_{\mathsf{ps}}$. Suppose that the given derivation of $\Gamma \vdash_{\mathsf{ps}}$ is of the form

$$\Gamma \vdash_{\mathsf{ps}} = (\mathrm{PSS})(\mathrm{PS}+)(\mathrm{PSE})^{k_n}(\mathrm{PS})(\mathrm{PS}+)(\mathrm{PSE})^{k_{n-1}}(\mathrm{PS})(\mathrm{PS}+)(\mathrm{PSE})^{k_{n-2}} \ldots (\mathrm{PSE})^{k_1}(\mathrm{PS})$$

then the cubical set $\nu(\Gamma)$ can be expressed as a pasting scheme as the following expression

$$\nu(\Gamma) = Z^{k_1+1} \otimes \ldots \otimes Z^{k_{n-1}+1} \otimes Z^{k_n+1}$$

Converse. Conversely, we show that every pasting scheme can be represented in the category $\mathcal{S}_{T_\square}$ by a unique context Γ such that $\Gamma \vdash_{\mathsf{ps}}$ holds.

Proposition 172. *Given a pasting scheme X, there exists a unique ps-context Γ such that $\Gamma \vdash_{\mathsf{ps}}$ is derivable and $\nu(\Gamma) = X$.*

Proof. Given a pasting scheme X, if X is the pasting scheme Z_0, then $(x : \star)$ is the only ps-context such that $\nu(x : \star) = Z_0$. Otherwise, X is of the form $X = Z_{k_1} \otimes Z_{k_n}$, with $k_1, \ldots, k_n > 0$. Then there is a unique ps-context whose nerve is X, obtained by the derivation

$$(\mathrm{PSS})(\mathrm{PS}+)(\mathrm{PSE})^{k_n-1}(\mathrm{PS}) \ldots (\mathrm{PS}+)(\mathrm{PSE})^{k_1-1}(\mathrm{PS})$$

□

Source and target. Our definitions for sources and targets for the ps-contexts and for the pasting schemes are compatible. In particular, considering a ps-context Γ of dimension n and an integer $i < n$, we can compute its i-source $\partial_i^-(\Gamma)$ and its i-target $\partial_i^+(\Gamma)$. But we can also define the pasting scheme $\nu(\Gamma)$. Our computation shows that $\nu(\partial_i^-(\Gamma))$ and $\nu(\partial_i^+(\Gamma))$ both compute to $\partial_{i,\nu(\Gamma)}$, and moreover, under this equality, the image of the canonical substitution $\Gamma \to \partial_i^-(\Gamma)$ (resp. $\Gamma \to \partial_i^+(\Gamma)$) computes to the map $\partial^-_{i,\nu(\Gamma)} : \partial_{i,\nu(\Gamma)} \to \nu(\Gamma)$ (resp. the map $\partial^-_{i,\nu(\Gamma)} : \partial_{i,\nu(\Gamma)} \to \nu(\Gamma)$)

Equivalence. We have now proved that the map ν defines an equivalence of categories between the category of cubical pasting schemes and the category of cubical ps-contexts. Moreover, each ps-context corresponds to a given globular sum, and hence two different ps-contexts are two different globular sums, so are non-isomorphic. Hence it also defines a bijection between the pasting and the ps-contexts, and this bijection respects the notions of source and target.

6.3.2 Cubical extensions

In our general framework, we characterize the contexts indexing the introduction rules for term constructors, as corresponding to a class of colimits in the category of presheaves. Here we have made them explicit as freely generated via tensor product by a family of globular sets that we have called Z_k. In order to fit our framework, we provide another characterization of the pasting schemes as a class of colimits. We call a *cubical structure* on a category $\mathcal{C}$, a functor $\square \to \mathcal{C}$.

Cubical sums. Define the category Z_k, whose objects are the disjoint union of $\{0, \cdots, k\}$ and of $\left\{\vec{1}, \cdots, \vec{k}\right\}$, and whose morphisms are generated by $i^- : i - 1 \to \vec{i}$ and $i^+ : i \to \vec{i}$, for all $1 \leq i \leq k$.

The *dimension* in objects of Z_k is defined by $\dim(i) = 0$ and $\dim(\underline{i}) = 1$. For $(k_1, \cdots, k_n) \in \mathbb{N}$, consider the category $\mathrm{Z}_{k_1,\cdots,k_n} = \mathrm{Z}_{k_1} \times \cdots \times Z_{k_n}$. The dimension of an object of this category is defined to be the sum of the dimensions of its components. The arrows in the category $\mathrm{Z}_{k_1,\cdots,k_n}$ are generated by the arrows $(1, 1, \cdots, 1, \partial_i^\alpha, 1, \cdots, 1, 1)$, with the non identity in position $0 \leq l \leq n$ and $0 \leq i < k_l$. Such an arrow is called a *cosource* when $\alpha = -$ and a *cotarget* if $\alpha = +$. Let $\mathcal{C}$ a category with a cubical structure, and denote C_n the image of n. A *cubical sum* in $\mathcal{C}$ is the colimit of a diagram $D : \mathrm{Z}_{k_1,\cdots,k_n} \to \mathcal{C}$ sending objects of dimension n in $\mathrm{Z}_{k_1,\cdots,k_n}$ to C_n, and sending the cosources (resp. the cotargets) in $\mathrm{Z}_{k_1,\cdots,k_n}$ on cosources (resp. on cotargets) in $\mathcal{C}$.

Cubical extensions. A category $\mathcal{C}$ with a cubical structure is called a *cubical extension* if it has all cubical sums. The category **FinCSet** is equipped with a structure of a cubical extension for the Yoneda embedding $\square \to$ **FinCSet**, indeed, it has all finite colimits, so it has in particular the cubical sums.

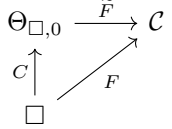
Proposition 173. *In the category* **FinCSet***, a cubical set is a cubical sum if and only if it is a pasting scheme.*

Proof. By definition, Z_k is the cubical sum indexed by the diagram Z_k. Consider two diagrams $\mathcal{D} : I \to \square$ and $\mathcal{D}' : J \to \square$, then they induce the diagram $\mathcal{D} \otimes \mathcal{D}' : I \times J \to \square$. Since the Day convolution product preserves the colimits in both variables, we have

$$\mathrm{colim}(\mathrm{Y} \circ \mathcal{D}) \otimes \mathrm{colim}(\mathrm{Y} \circ \mathcal{D}') \simeq \mathrm{colim}(\mathrm{Y} \circ (\mathcal{D} \circ \mathcal{D}'))$$

hence, if we denote X the cubical sum indexed by $\mathrm{Z}_{k_1,\dots,k_n}$ and Y the cubical sum indexed by $\mathrm{Z}_{l_1,\dots,l_m}$, then $X \otimes Y$ is the cubical sum indexed by $\mathrm{Z}_{k_1,\dots,k_n,l_1,\dots,l_m}$. This shows that $Z_{k_1} \otimes \ldots \otimes Z_{k_n}$ is the cubical sum indexed by $\mathrm{Z}_{k_1,\dots,k_n}$. □

Universal cubical extension. This shows that restricting the Yoneda embedding a functor $C : \square \to \Theta_{\square,0}$ exhibits $\Theta_{\square,0}$ as a cubical extension. This in fact realizes the initial cubical extension, in such a way that for any cubical extension $F : \square \to \mathcal{C}$, there is an essentially unique functor $\widetilde{F} : \Theta_{\square,0} \to \mathcal{C}$ preserving the cubical sums, such that the following triangle commutes



Indeed, suppose $\widetilde{F}$ exists, and consider X to be a cubical sum indexed by $\mathrm{Z}_{k_1,\dots,k_n}$ in $\Theta_{\square,0}$, then since $\widetilde{F}$ preserves the cubical sums, $\widetilde{F}(X)$ is a cubical sum indexed by $\mathrm{Z}_{k_1,\dots,k_n}$ for the cubical extension $F : \square \to \mathcal{C}$, this determines $\widetilde{F}$ up to isomorphism, hence $\widetilde{F}$ is essentially unique. Conversely, taking $\widetilde{F}$ to associate to X the chosen cubical sum indexed by $\mathrm{Z}_{k_1,\dots,k_n}$ in the cubical extension $F : \square \to \mathcal{C}$ yields a functor which preserves cubical sums.

6.3.3 Cubical Cat-coherator

Cubical theories. A *cubcial theory* is a cubical extension $\square \to \mathcal{C}$ such that the universal morphism of cubical extensions $\Theta_{\square,0} \to \mathcal{C}$ is faithful and induces a bijection of the set of isomorphism classes of objects. Similarly to [54], up to equivalence of category, we can always identify Θ_0 with a full subcategory of $\mathcal{C}$ whenever $\mathcal{C}$ is a globular theory, and arrow f of $\mathcal{C}$ will be said to be *cubical* if it is in Θ_0.

Compatible families of arrows. Given a cubical theory $\mathcal{C}$, we denote C_n the image of n,and σ_i, τ_i the respective images of σ_i and τ_i. A family of arrows $f_0, g_0, \dots, f_{n-1}, g_{n-1} : C_i \to X$ is said to be *compatible* if it satisfies the following equations, for all $0 \leq i < j \leq n-1$

$$f_j\sigma_i = f_{i+1}\sigma_j \qquad f_j\tau_i = g_{i+1}\sigma_j$$
$$g_j\sigma_i = f_{i+1}\tau_j \qquad g_j\tau_i = g_{i+1}\tau_j$$

Intuitively, saying that a family $(f_0, g_0, \dots, f_n, g_n)$ is compatible means that they fit into the border of a potential $(n+1)$-cell. In the category **FinCSet**, a compatible family of arrows $\mathrm{Y}([n])$ is equivalent to a morphism $\partial\,\mathrm{Y}([n+1]) \to X$, but in an arbitrary cubical theory, there might be enough limits to define $\partial\,\mathrm{Y}(n)$, thus we present it using explicit algebraic conditions. A *lifting* for a compatible family of arrows $(f_0, g_0, \dots, f_n, g_n) : C_n \to X$ is an arrow $h : C_{n+1} \to X$ such that for all $0 \leq i \leq n$, we have

$$f_i = h\sigma_i \qquad g_i = h\tau_i$$

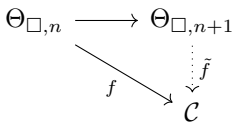
Intuitively, it corresponds to a filling of the $(n+1)$-cube in X whose border has been specified by the compatible family. An arrow f in a cubical theory is said to be *algebraic* if for all decompositions $f = gf'$ with g cubical, g is an identity. We also define a family of arrows $f_0, \dots, f_n$ to be *simultaneously algebraic* if for all simultaneous decomposition $f_0 = gf'_0, \dots, f_n = gf'_n$ with g cubical, g is an identity. A morphism of cubical theories is a morphism of the underlying cubical extensions.

Admissible families of arrows. Let $\mathcal{C}$ be a cubical theory. An *admissible family* of arrows is a compatible family of arrows $f_1, g_1, \dots, f_n, g_n : C_i \to X$ such that either the arrows $f_0, g_0, \dots, f_n, g_n$ are simultaneously algebraic, or for all k there exists decompositions $f_k = \partial^-_{k,X} f'_k$ and $g_k = \partial^+_{k,X} g'_k$, with f'_k and g'_k algebraic

Cubical Cat-coherator. We now introduce the *cubical Cat-coherator* to be the cubical theory $\square \to \Theta_{\square,\infty}$ defined as the colimit

$$\Theta_{\square,\infty} \simeq \mathrm{colim}(\Theta_{\square,0} \to \Theta_{\square,1} \to \Theta_{\square,2} \to \cdots \to \Theta_{\square,n} \to \cdots)$$

Where $\Theta_{\square,n}$ is given by induction on n. Define E_n to be the set of all pairs of admissible arrows of $\Theta_{\square,n}$ that are not in E'_n for any $n' < n$. Then we can define $\Theta_{\square,n+1}$ to be the universal cubical extension of $\Theta_{\square,n}$ obtained by formally adding a lift for each pairs in E_n. In other words, for each cubical extension $f : \Theta_{\square,n} \to \mathcal{C}$ such that the image by f of all pairs of arrows in E_n has a lift in $\mathcal{C}$, there is an essentially unique cubical extension $\tilde{f}$ preserving the chosen lifts, which makes the following triangle commute



Cubical weak ω-categories. We define the category of cubical weak ω-categories to be the full subcategory of $\widehat{\Theta_{\square,\infty}}$ whose objects are the presheaves that preserves the cubical products of $\Theta^{\mathrm{op}}_{\square,infty}$.

6.3.4 Coherator of the theory $\mathsf{CaTT}_\square$.

This definition of cubical weak ω-categories is the mathematical pendent of our type theory. Our general framework for the $\square$-type theories and our nerve theorem shows that for studying the models of this type theory, it suffices to study its coherator, which in this case is the subcategory of $\mathcal{S}_{\mathsf{CaTT}_\square}$ whose objects are the ps-contexts. We denote $\mathcal{S}_{\mathsf{PS}_\square,\infty}$ this category. As in the case of CaTT, there is a dualization happening, and we follow the same naming convention for the dual notions: We thus have coccubical structures, cocubical extension, coadmissible families of maps, and so on.

Coherence depth. As was the case with CaTT, we introduce the notion of *coherence depth* of a term, in order to reproduce the inductive definition of $\Theta_{\square,n}$. The coherence depth is defined

by induction as follows

$$\mathrm{cd}(x) = 0 \qquad \mathrm{cd}(\mathsf{op}_{\Gamma,A}[\gamma]) = \max(\mathrm{cd}(A)+1, \mathrm{cd}(\gamma))$$
$$\mathrm{cd}(\mathsf{coh}_{\Gamma,A}[\gamma]) = \max(\mathrm{cd}(A)+1, \mathrm{cd}(\gamma))$$
$$\mathrm{cd}(t\ i) = \mathrm{cd}(t)$$

$$\mathrm{cd}(\star) = 0 \qquad \mathrm{cd}(\mathsf{Path}^i\ A\ t\ u) = \max(\mathrm{cd}(A), \mathrm{cd}(t), \mathrm{cd}(u))$$

$$\mathrm{cd}(\langle\rangle) = 0 \qquad \mathrm{cd}(\langle\gamma, x \mapsto t\rangle) = \max(\mathrm{cd}(\gamma), \mathrm{cd}(t))$$

We consider the category $\mathcal{S}_{\mathsf{PS}_\square,n}$ which is the subcategory of $\mathcal{S}_{\mathsf{PS}_\square,\infty}$ whose substitution all use terms of coherence depth at most n.

Coadmissible families of arrows in $\mathcal{S}_{\mathsf{PS}_\square,\infty}$. The compatible families of arrows in $\mathcal{S}_{\mathsf{PS}_\square,\infty}$ are exactly the families satisfying the cubical relations, that define the type constructors in the theory $T_\square$. They are equivalent to the relations in our implementation with the types Path . Thus compatible families of arrows exactly correspond to types. We conjecture the two following results that are analogous to the case of CaTT. We believe that proving these results would require a similar analysis of the use of the variables allowed in a type and a term.

Conjecture 174. *for any compatible pair of arrows $(f_0, g_0 \dots, f_n, g_n) : C_n \to X$, in $\Theta_{\square,m}$, the family the arrows $f_0, g_0, \dots, f_n, g_n$ are simultaneously algebraic if and only if both the families of arrows $f_0, \dots, f_n$ and $g_0, \dots, g_n$ are simultaneously algebraic.*

Conjecture 175. *The compatible family of morphisms corresponding to a type is coadmissible if and only if the type satisfies either* $(\mathrm{C}_{\mathrm{cohop},\square})$ *or* $(\mathrm{C}_{\mathrm{coh},\square})$

Note that Conjecture 175 is the stronger one, as it implies Conjecture 174. From now on, we admit this conjecture for continuing the study of the type theory.

Tower of definition. Following our proof in the case of CaTT, we define the set E_n to be the set of all types $\Gamma \vdash A$ of coherence depth n in a ps-context Γ, satisfying $(\mathrm{C}_{\mathrm{cohop},\square})$ or $(\mathrm{C}_{\mathrm{coh},\square})$. Conjecture 175 the family E_n can be defined inductively as the set of all coadmissible families of maps in $\mathcal{S}_{\mathsf{PS},n}$ that do not belong to any $E_{n'}$ for $n < n'$.

Lemma 176. *The inclusion $\mathcal{S}_{\mathsf{PS}_\square,n} \to \mathcal{S}_{\mathsf{PS}_\square,n+1}$ exhibits $\mathcal{S}_{\mathsf{PS}_\square,n+1}$ as the universal cocubical extension of $\mathcal{S}_{\mathsf{PS}_\square,n}$ which has a lift for all pair of morphisms in E_n.*

Proof. As we have already proved, this map commutes with the cocubical structure, and both these categories have all the cubical products, hence it defines a cocubical extension. Moreover, consider a coadmissible pair (f, g) in E_n, then id corresponds to a type $\Gamma \vdash A$ in a ps-context Γ, which satisfies $(\mathrm{C}_{\mathsf{op}})$ or $(\mathrm{C}_{\mathsf{coh}})$ and which is of depth n. Hence we can derive a term t by $\Gamma \vdash \mathsf{op}_{\Gamma,A}[\mathrm{id}_\Gamma] : A$ if A satisfies $(\mathrm{C}_{\mathsf{op}})$, or $\Gamma \vdash \mathsf{coh}_{\Gamma,A}[\mathrm{id}_\Gamma] : A$ if A satisfies $(\mathrm{C}_{\mathsf{coh}})$, which is of depth $n+1$. This term defines a map χ_t in the category $\mathcal{S}_{\mathsf{PS}_\square,n+1}$, that fits in the following diagram

$$\begin{array}{ccc} & & \mathrm{Y}(n) \\ & \overset{\chi_t}{\nearrow} & \big\downarrow \pi \\ \Gamma & \xrightarrow[\chi_A]{} & \partial\,\mathrm{Y}(n) \end{array}$$

By definition of A corresponding to the coadmissible pair (f, g), this shows that χ_t defines exactly a lift for the pair (f, g). Hence $\mathcal{S}_{\mathsf{PS}_\square,n+1}$ is a coglobular extension which contains a lift for all pairs in E_n. We now show that this extension is universal: consider another extension $F : \mathcal{S}_{\mathsf{PS}_\square,n} \to \mathcal{C}$ that defines a lift for all the pairs in E_n, we show that there exists a unique $\tilde{F}$ that preserves the chosen lifts which makes the following diagram commute

$$\begin{array}{ccc} \mathcal{S}_{\mathsf{PS}_\square,n} & \longrightarrow & \mathcal{S}_{\mathsf{PS}_\square,n+1} \\ & \searrow^{F} & \downarrow{\scriptstyle \tilde{F}} \\ & & \mathcal{C} \end{array}$$

Indeed, the map $\tilde{F}$ is already defined on all objects of $\mathcal{S}_{\mathsf{PS}_\square,n+1}$, and all maps of coherence depth less than n, to coincide with F, so it suffices that there is a unique extension to the maps of coherence depth $n+1$. Since all the object in $\mathcal{S}_{\mathsf{PS}_\square,n+1}$ are cubical products, it suffices to show it for the maps of the form $\Gamma \to \mathrm{Y}(n)$. We can thus reformulate by saying that it suffices to show that there is a unique map $\tilde{F}$ on terms, with the condition that $\tilde{F}(t[\gamma]) = \tilde{F}t \circ \tilde{F}\gamma$. We proceed by induction on the depth, noticing that a term of coherence depth $n+1$ cannot be a variable, hence we have already defined a unique value for $\tilde{F}$ on terms of depth 0, by our previous condition, and thus the induction is already initialized

- For a term $\Delta \vdash \mathsf{op}_{\Gamma,A}[\gamma] : A[\gamma]$ of depth $d+1$, the value of F is uniquely determined by $\tilde{F}(\mathsf{op}_{\Gamma,A}[\gamma]) = \tilde{F}(\mathsf{op}_{\Gamma,A}[\mathrm{id}_\Gamma])\tilde{F}\gamma$, and since γ is of depth d, by induction $\tilde{F}(\gamma)$ is defined, and $\tilde{F}(\mathsf{op}_{\Gamma,A}[\mathrm{id}_\Gamma])$ is uniquely defined by the condition of preserving the lifts for the pairs in E_n
- Similarly for a term $\Delta \vdash \mathsf{coh}_{\Gamma,A}[\gamma] : A[\gamma]$ of depth $d+1$, the value of F is uniquely determined by $\tilde{F}(\mathsf{op}_{\Gamma,A}[\gamma]) = \tilde{F}(\mathsf{op}_{\Gamma,A}[\mathrm{id}_\Gamma])\tilde{F}\gamma$, and since γ is of depth d, by induction $\tilde{F}(\gamma)$ is defined, and $\tilde{F}(\mathsf{coh}_{\Gamma,A}[\mathrm{id}_\Gamma])$ is uniquely defined by the condition of preserving the lifts for the pairs in E_n

This proves that there exists a unique $\tilde{F}$ satisfying the condition, and hence $\mathcal{S}_{\mathsf{PS}_\square,n+1}$ is the universal globular extension obtained by adding a lift for all arrows in E_n to $\mathcal{S}_{\mathsf{PS}_\square,n}$ □

Theorem 177. *Under Conjecture 175, the category $\mathcal{S}_{\mathsf{PS}_\square,\infty}$ is equivalent to the category $\Theta^{\mathrm{op}}_{\square,\infty}$*

Proof. By induction, Lemma 176 shows that the categories $\mathcal{S}_{\mathsf{PS}_\square,n}$ and $\Theta^{\mathrm{op}}_{\square,n}$ satisfy the same universal property, hence they are equivalent, and taking the colimit yields an equivalence $\mathcal{S}_{\mathsf{PS}_\square,\infty} \simeq \Theta^{\mathrm{op}}_{\square,\infty}$ □

Models of the theory $\mathsf{CaTT}_\square$. Our framework for studying □-type theories then allows us to obtain immediately the following characterization of the models of the theory $\mathsf{CaTT}_\square$.

Theorem 178. *Under Conjecture 175, the models of the theory $\mathsf{CaTT}_\square$ are equivalent to the category of pre-cubical weak ω-categories.*

Proof. This is an application of Theorem 116: Since the arity limits are the cubical products, the models are equivalent to the functors $\mathcal{S}_{\mathsf{PS}_\square,\infty} \to \mathbf{Set}$ that preserve the cubical products. Theorem 177 then gives the result. □

Conclusion and further work

The approach we have presented for studying weak ω-categories and similar higher structures has many practical advantages. It allows for an implementation of a coherence checker for these structures, which gives a direct way to access and manipulate the operations and coherence, experiment with them and build up strong intuition in a safe and computer checked environment. Additionally it provides a syntax to do so, which is more convenient than diagram chasing. The presence of a syntax also gives a good setup for inductive reasoning, which is useful both for understanding the theory, and defining operations like the suspension (c.f. Section 3.2) and the functorialization (c.f. Section 3.4). Additionally, the framework more general framework of I-types theories for any direct finite branching category I, that we have introduced in Section 4.3 allows for defining various higher structures similar to globular weak ω-categories, and gives a convenient and modular way to directly characterize their models. The intuition gained from the practical use of a proof assistant can then be used to tweak a little the rules and define another type theory fit for different higher structures, and extract a mathematical definitions of the models of the theory, as we have done in Section 6 for cubical weak ω-categories. This has left several directions open for further research

Polygraphs and weak functors. As we briefly mentioned, the syntactic category of CaTT (or of any I-type theory) gives a natural notion of finite polygraphs for weak ω-categories (or for other higher structures). A natural follow up to this remark is to try to define all the polygraphs of such structures, with a theory where contexts may be infinitely long. This approach is currently being studied by Finster, and would have important consequences: We expect that for any weak ω-category $\mathcal{F} \in \mathbf{Mod}(\mathcal{S}_{\mathsf{CaTT}})$, there is a morphism $\Gamma \to \mathcal{F}$ in $\mathbf{Mod}(\mathcal{S}_{\mathsf{CaTT}})$, where Γ is a category freely generated by such a polygraph (not necessarily finite), and that this morphism corresponds to an expected notion of weak equivalence, providing an analogue to the cofibrant replacement. There may not be however a full-fledged model structure on the category of models of CaTT. This "cofibrant replacement" is expected to provide a notion of weak functors: A weak functor between $\mathcal{F}$ and $\mathcal{F}'$ is then a functor in the category $\mathbf{Mod}(\mathcal{S}_{\mathsf{CaTT}})$ between their cofibrant replacements. This is analogous to a known definition of weak functors for weak ω-groupoids.

Automation of the inverse. Working to prove Conjecture 80 is an important goal for two reasons. It first strengthen our claim that terms of strictly positive degree are invertible, thus providing a complete understanding of the invertibility of all the cells of the ω-categories freely generated by finite polygraphs. It would also provide with an explicit algorithm to compute the cancellation witness of a term of degree positive with its inverse. Further work along these lines would then be to implement the algorithm giving the inverse of a term of positive degree together with the aforementioned algorithm in order to automatically generate all the inverses and cancellation witnesses. This would be a significant improvement for the proof assistant, sparing the user to manually define all of these, as they are an important part of the development. An

approach that we consider for proving this conjecture is to work with an *unbiased* version of it. It would be more general, but it is likely to also yield a more natural algorithm for computing the inverses and cancellation witnesses, similarly to the fact that weak ω-categories are easier to define in their unbiased versions, even if in the end one wants to work in practice with the (biased) binary composition.

Furthering automation. The meta-operations of suspension and functorialization that we have defined respectively in Section 3.2 and Section 3.4 are important for the practical use of the proof assistant, and using them enables for shorter and simpler developments. Their use is however restricted to very specific cases, in particular for the functorialization which only applies to operation, and does not allow for generating new coherences. We believe that this operation can be extended to coherences and then used to generate automatically the coherences associated to a functorialized operation. Simple examples show that such a definition would have to generate complicated terms, and in particular it would have to encode all the cubical compositions, in a globular setting. We believe that defining this operation in an appropriate notion of cubical categories, and then defining a translation from this notion to the globular categories is a promising approach towards a definition of the functorialization in general.

Cubical categories. In Section 6 we have defined a type theory that describes a cubical notion of ω-categories, motivated by the example of functorialization. However the functorialization in this theory, that we have not presented, is very involved and yields to terms that are too long to be manageable in practice. We believe that the reason for this is because we rely on a framework that gives only cubes. Our intuition is that introducing more diverse shapes for composing, and in particular allowing for some of the low dimensional faces of the cubes to be contracted to a point, would provide a theory that encompasses both the globular approach and the cubical one. Such a theory can be obtained by allowing weakenings with respect to the dimension variables, which is not permitted in the theory we have presented, but the notion of ps-context and the term introduction rules need then to be adapted. We believe that this is the natural setting in which the functorialization lives, and expect such a definition of weak ω-categories to be computationally better behaved: It allows for both simpler expressions and heavier automation. Additionally, while working with CaTT, it happens often that we perform cubical reasoning and then manually encode it into globular shapes to fit the framework, this expected new framework could provide a better account for this. We expect that any term of this theory can be translated into the theory CaTT, requiring to automate the translation of a cubical cell into a globular one. Such a translation would then immediately provide a general definition of the functorialization in CaTT, and we expect it to be simpler than this general definition.

Monoidal categories. In Section 5 we have presented a type theory for describing monoidal weak ω-categories. As a generalization, we have briefly introduced a type theory for describing k-tuply monoidal ω-categories without studying it. In particular characterizing the models of this theory in relation with the models of CaTT as we have done for MCaTT is an potential research project, that could have deep consequences for the theory CaTT. We illustrate this fact with the doubly monoidal ω-categories: the theory 2-MCaTT describes a monoidal ω-category with a single object, whereas all the contexts in the theory MCaTT define infinitely many terms of dimension 0: They all define at least (using the coherence names introduced in Section 5.3) `e`, `prod e e`, `prod (prod e e) e`,... Hence a syntactic translation from 2-MCaTT to MCaTT cannot be defined by simply reindexing the term constructors: It has to use the compositions of the theory MCaTT to relate the borders of the terms to a reduced version of them where all the products of the monoidal unit have been simplified. Defining such a translation amounts

to a *strictification procedure* on the coherences that are generated by products of the monoidal unit. Since the Eckmann-Hilton morphism is just a single definition in the theory 2-MCaTT, this translation has to be powerful enough to generate our entire formalized definition of the Eckmann-Hilton morphism in CaTT. As this formalization makes heavy use of the suspension and the functorialization, and could also be simplified using automatic inverses, cancellation witness and our expected general formalization, we expect all the previously mentioned work to be very helpful for defining this translation. We do not know whether there is a general scheme for this strictification that could work for $k > 2$ and give translations from k-MCaTT to CaTT. Such a scheme would result in extremely heavy automation, making almost completely immediate a definition such as the syllepsis in triply monoidal categories, that is barely manageable in the current state of development. Another follow up would be to try and adapt the theory k-MCaTT to a theory ∞-MCaTT describing symmetric weak ω-categories.

Comparison with homotopy.io Homotopy.io[1] is another proof assistant for working with higher categories of a slightly different flavor. These are higher categories where the identities and the associativities are on the nose, but the other higher cells are weak. It would be very valuable to study the connection between homotopy.io and CaTT, and in particular defining a translation between these two would amount to generating automatically a lot of higher cells in CaTT with very few input commands by the user. Extensive work has already been pursued in this direction by Finster, Reutter and Vicary [34].

Flavors of type theory. All the theory that we have presented have exchange, contractions and weakening. This is usually the case for dependent type theories, and these rules force every context to be obtained as a series of pullbacks along display maps. For this reason we call these kind of type theories *cartesian*. For simple type theories many other flavors of type theory exist: One may present a theory that forbids some combinations of these rules. Not all combinations can be allowed or forbidden (c.f. [31]). These yields to other notions of simple type theories such as *linear type theories* (disallowing the contractions). Reconciling dependent types and linear type theory in a single framework is an active project. In this regard, frameworks analogous to the I-type theory that we have introduced could give a syntax for a notion of generalized algebraic theory in which the arities of the operation are given by a computation that is not a limit.

[1] https://homotopy.io/

Appendix A

Presentation of the type theories

A.1 The type theory GSeTT

Syntax.

Contexts: lists $(x_0 : A_0, \dots, x_n : A_n)$ with x_i variables and A_i types

Types: either $\star$ or of the form $t \underset{A}{\longrightarrow} u$ with A type and t and u terms

Terms: variables

Substitutions: lists $\langle x_0 \mapsto t_0, \dots, x_n \mapsto t_n \rangle$ with x_i variables and t_i terms

Inference rules.

For contexts:

$$\frac{}{\varnothing \vdash}(\text{EC}) \qquad \frac{\Gamma \vdash A}{\Gamma, x : A \vdash}(\text{CE}) \quad \text{Where } x \notin \text{Var}(\Gamma)$$

For types:

$$\frac{\Gamma \vdash}{\Gamma \vdash \star}(\star\text{-INTRO}) \qquad \frac{\Gamma \vdash t : A \qquad \Gamma \vdash u : A}{\Gamma \vdash t \underset{A}{\longrightarrow} u}(\to\text{-INTRO})$$

For terms:

$$\frac{\Gamma \vdash \qquad (x : A) \in \Gamma}{\Gamma \vdash x : A}(\text{VAR})$$

For substitutions:

$$\frac{\Delta \vdash}{\Delta \vdash \langle\rangle : \varnothing}(\text{ES}) \qquad \frac{\Delta \vdash \gamma : \Gamma \qquad \Gamma, x : A \vdash \qquad \Delta \vdash t : A[\gamma]}{\Delta \vdash \langle \gamma, x \mapsto t \rangle : (\Gamma, x : A)}(\text{SE})$$

Semantics.

$$\mathcal{S}_{\mathsf{GSeTT}} = \mathbf{FinGSet}^{\text{op}}$$
$$\mathbf{Mod}(\mathcal{S}_{\mathsf{GSeTT}}) = \mathbf{GSet}$$

A.2 The type theory CaTT

Syntax.

Contexts: lists $(x_0 : A_0, \dots, x_n : A_n)$ with x_i variables and A_i types

Types: either $\star$ or of the form $t \underset{A}{\longrightarrow} u$ with A type and t and u terms

Terms: either variables or of the form $\mathsf{op}_{\Gamma,A}[\gamma]$ or $\mathsf{coh}_{\Gamma,A}[\gamma]$ with Γ a ps-context, A a type and γ a substitution

Substitutions: lists $\langle x_0 \mapsto t_0, \dots, x_n \mapsto t_n \rangle$ with x_i variables and t_i terms

Rules for ps-contexts.

$$\frac{}{(x : \star) \vdash_{\mathsf{ps}} x : \star}(\text{PSS}) \qquad \frac{\Gamma \vdash_{\mathsf{ps}} x : A}{\Gamma, y : A, f x \underset{A}{\longrightarrow} y \vdash_{\mathsf{ps}} f : x \underset{A}{\longrightarrow} y}(\text{PSE})$$

$$\frac{\Gamma \vdash_{\mathsf{ps}} f : x \underset{A}{\longrightarrow} y}{\Gamma \vdash_{\mathsf{ps}} y : A}(\text{PSD}) \qquad \frac{\Gamma \vdash_{\mathsf{ps}} x : \star}{\Gamma \vdash_{\mathsf{ps}}}(\text{PS})$$

Source and target of a ps-context.

$$\partial_i^-(x : \star) = (x : \star) \qquad \partial_i^-(\Gamma, y : A, f : x \to y) = \begin{cases} \partial_i^-\Gamma & \text{if } \dim A \ \geq i-1 \\ (\partial_i^-\Gamma, y : A, f : x \to y) & \text{otherwise} \end{cases}$$

$$\partial_i^+(x : \star) = (x : \star) \qquad \partial_i^+(\Gamma, y : A, f : x \to y) = \begin{cases} \partial_i^+\Gamma & \text{if } \dim A \ \geq i \\ \mathrm{drop}(\partial_i^+\Gamma), y : A & \text{if } \dim A \ = i-1 \\ \partial_i^+\Gamma, y : A, f : x \to y & \text{otherwise} \end{cases}$$

$$\partial^-(\Gamma) = \partial^-_{\dim\Gamma-1}\Gamma \qquad \partial^+(\Gamma) = \partial^+_{\dim\Gamma-1}\Gamma$$

where $\mathrm{drop}(\Gamma)$ is the context Γ with its last variable removed.

Inference rules.

For contexts :

$$\frac{}{\varnothing \vdash}(\text{EC}) \qquad \frac{\Gamma \vdash A}{\Gamma, x : A \vdash}(\text{CE}) \quad \text{Where } x \notin \text{Var}(\Gamma)$$

For types :

$$\frac{\Gamma \vdash}{\Gamma \vdash \star}(\star\text{-INTRO}) \qquad \frac{\Gamma \vdash t : A \qquad \Gamma \vdash u : A}{\Gamma \vdash t \underset{A}{\longrightarrow} u}(\to\text{-INTRO})$$

For terms :

$$\frac{\Gamma \vdash \qquad (x : A) \in \Gamma}{\Gamma \vdash x : A}(\text{VAR}) \qquad \frac{\Gamma \vdash_{\mathsf{ps}} \qquad \Gamma \vdash A \qquad \Delta \vdash \gamma : \Gamma}{\Delta \vdash \mathsf{op}_{\Gamma,A}[\gamma] : A[\gamma]}(\text{OP})$$

$$\frac{\Gamma \vdash_{\mathsf{ps}} \qquad \Gamma \vdash A \qquad \Delta \vdash \gamma : \Gamma}{\Delta \vdash \mathsf{coh}_{\Gamma,A}[\gamma] : A[\gamma]}(\text{COH})$$

For substitutions :

$$\frac{\Delta \vdash}{\Delta \vdash \langle\rangle : \varnothing}(\text{ES}) \qquad \frac{\Delta \vdash \gamma : \Gamma \qquad \Gamma, x : A \vdash \qquad \Delta \vdash t : A[\gamma]}{\Delta \vdash \langle \gamma, x \mapsto t \rangle : (\Gamma, x : A)}(\text{SE})$$

where in the rule (OP), the type A is assumed to satisfy (C_{op}) and in the rule (COH), the type A is assumed to satisfy (C_{coh}).

(C_{op}) A is of the form $t \underset{B}{\longrightarrow} u$ with $\begin{cases} \text{Var}(t) \cup \text{Var}(B) = \text{Var}(\partial^-(\Gamma)) \\ \text{Var}(u) \cup \text{Var}(B) = \text{Var}(\partial^+(\Gamma)) \end{cases}$

(C_{coh}) $\text{Var}(A) = \text{Var}(\Gamma)$ or equivalently
A is of the form $t \underset{B}{\longrightarrow} u$ with $\begin{cases} \text{Var}(t) \cup \text{Var}(B) = \text{Var}(\Gamma) \\ \text{Var}(u) \cup \text{Var}(B) = \text{Var}(\Gamma) \end{cases}$

Semantics.

$\mathcal{S}_{\mathsf{PS},\infty} = \Theta_\infty^{\text{op}}$
$\mathcal{S}_{\mathsf{CaTT}}$: free completion of $\mathcal{S}_{\mathsf{PS},\infty}$ by canonical limits preserving globular products
$\mathbf{Mod}(\mathcal{S}_{\mathsf{CaTT}})$: equivalent to the category of weak ω-categories

A.3 The theory MCaTT

Syntax.

Contexts: lists $(x_0 : A_0, \ldots, x_n : A_n)$ with x_i variables and A_i types

Types: either $\star$ or of the form $t \underset{A}{\longrightarrow} u$ with A type and t and u terms

Terms: either variables or of the form $\mathsf{mop}_{\Gamma,A}[\gamma]$ or $\mathsf{mcoh}_{\Gamma,A}[\gamma]$ with Γ a ps-context, A a type and γ a substitution

Substitutions: lists $\langle x_0 \mapsto t_0, \ldots, x_n \mapsto t_n \rangle$ with x_i variables and t_i terms

Desuspension.

$$\text{For the context } \varnothing \vdash \qquad \text{For the context } (\Gamma, x : A) \vdash$$
$$\downarrow\varnothing = \varnothing \qquad \downarrow(\Gamma, x : A) = \begin{cases} \downarrow\Gamma & \text{if } A = \star \\ \downarrow\Gamma, x : \downarrow A & \text{otherwise} \end{cases}$$

$$\text{For the type } \Gamma \vdash \star \qquad \text{For the type } \Gamma \vdash t \underset{A}{\longrightarrow} u$$
$$\downarrow\star = \star \qquad \downarrow(t \underset{A}{\longrightarrow} u) = \begin{cases} \star & \text{if } A = \star \\ \downarrow t \underset{\downarrow A}{\longrightarrow} \downarrow u & \text{otherwise} \end{cases}$$

$$\text{For a variable } \Gamma \vdash x : A \qquad \text{For the term } \Delta \vdash \mathsf{op}_{\Gamma,A}[\gamma] : A[\gamma]$$
$$\downarrow x = x \qquad \downarrow\mathsf{op}_{\Gamma,A}[\gamma] = \mathsf{mop}_{\Gamma,A}[\downarrow\gamma]$$

$$\text{For the term } \Delta \vdash \mathsf{coh}_{\Gamma,A}[\gamma] : A[\gamma]$$
$$\downarrow\mathsf{coh}_{\Gamma,A}[\gamma] = \mathsf{mcoh}_{\Gamma,A}[\downarrow\gamma]$$

$$\text{For } \Delta \vdash \langle\rangle : \varnothing \qquad \text{For } \Delta \vdash \langle\gamma, x \mapsto t\rangle : (\Gamma, x : A)$$
$$\downarrow\langle\rangle = \langle\rangle \qquad \downarrow\langle\gamma, x \mapsto t\rangle = \begin{cases} \downarrow\gamma & \text{if } A = \star \\ \langle\downarrow\gamma, x \mapsto \downarrow t\rangle & \text{otherwise} \end{cases}$$

Inference rules.

For contexts :

$$\frac{}{\varnothing \vdash}(\text{EC}) \qquad \frac{\Gamma \vdash A}{\Gamma, x : A \vdash}(\text{CE}) \quad \text{Where } x \notin \text{Var}(\Gamma)$$

For types :

$$\frac{\Gamma \vdash}{\Gamma \vdash \star}(\star\text{-INTRO}) \qquad \frac{\Gamma \vdash t : A \qquad \Gamma \vdash u : A}{\Gamma \vdash t \underset{A}{\longrightarrow} u}(\rightarrow\text{-INTRO})$$

For terms :

$$\frac{\Gamma \vdash \qquad (x : A) \in \Gamma}{\Gamma \vdash x : A}(\text{VAR}) \qquad \frac{\Gamma \vdash_{\mathsf{ps}} \qquad \Gamma \vdash_{\mathsf{CaTT}} A \qquad \Delta \vdash \gamma : \downarrow\Gamma}{\Delta \vdash \mathsf{mop}_{\Gamma,A}[\gamma] : (\downarrow A)[\gamma]}(\text{MOP})$$

$$\frac{\Gamma \vdash_{\mathsf{ps}} \qquad \Gamma \vdash_{\mathsf{CaTT}} A \qquad \Delta \vdash \gamma : \downarrow\Gamma}{\Delta \vdash \mathsf{coh}_{\Gamma,A}[\gamma] : (\downarrow A)[\gamma]}(\text{MCOH})$$

For substitutions :

$$\frac{\Delta \vdash}{\Delta \vdash \langle\rangle : \varnothing}(\text{ES}) \qquad \frac{\Delta \vdash \gamma : \Gamma \qquad \Gamma, x : A \vdash \qquad \Delta \vdash t : A[\gamma]}{\Delta \vdash \langle\gamma, x \mapsto t\rangle : (\Gamma, x : A)}(\text{SE})$$

where in the rule (MOP), the type A is assumed to satisfy (C_{op}) and in the rule (MCOH), the type A is assumed to satisfy (C_{coh}).

Semantics.

- $\mathcal{S}_{\mathsf{MCaTT}}$ is equivalent to the full subcategory of $\mathcal{S}_{\mathsf{CaTT}}$ whose objects are the contexts with a unique variable of type $\star$.
- $\mathbf{Mod}(\mathcal{S}_{\mathsf{MCaTT}})$ is equivalent to the full subcategory of $\mathbf{Mod}(\mathcal{S}_{\mathsf{CaTT}})$ that define a single 0-cell

A.4 The theory MCaTT′

Syntax.

Contexts: lists $(x_0 : A_0, \ldots, x_n : A_n)$ with x_i variables and A_i types

Types: either $\star$ or of the form $t \underset{A}{\longrightarrow} u$ with A type and t and u terms

Terms: either variables or of the form $\mathsf{mop}_{\Gamma,A}[\gamma]$ or $\mathsf{mcoh}_{\Gamma,A}[\gamma]$ with Γ a ps-context, A a type and γ a substitution

Substitutions: lists $\langle x_0 \mapsto t_0, \ldots, x_n \mapsto t_n \rangle$ with x_i variables and t_i terms

Monoidal contexts: lists of contexts $[\Gamma_0; \ldots; \Gamma_n]$

Monoidal substitutions: lists of substitutions $[\gamma_0; \ldots; \gamma_n]$

Monoidal ps-contexts.

$$\frac{}{[] \vdash_{\mathsf{ps}}} (\text{MPSNIL})$$

$$\frac{\overline{\Gamma} \vdash_{\mathsf{ps}}}{[\overline{\Gamma}; (x : \star)] \vdash_{\mathsf{ps}} x : \star} (\text{MPSS}) \qquad \frac{\overline{\Gamma} \vdash_{\mathsf{ps}} x : A}{(\overline{\Gamma}, y : A, f : x \underset{A}{\longrightarrow} y) \vdash_{\mathsf{ps}} f : x \underset{A}{\longrightarrow} y} (\text{MPSE})$$

$$\frac{\overline{\Gamma} \vdash_{\mathsf{ps}} f : x \underset{A}{\longrightarrow} y}{\overline{\Gamma} \vdash_{\mathsf{ps}} y : A} (\text{MPSD}) \qquad \frac{\overline{\Gamma} \vdash_{\mathsf{ps}} x : \star}{\overline{\Gamma} \vdash_{\mathsf{ps}}} (\text{MPS})$$

Source and target.

$$\partial_i^-([]) = [] \qquad \partial_i^-([\overline{\Gamma}; (x : \star)]) = [\partial_i^-(\overline{\Gamma}); (x : \star)]$$

$$\partial_i^-(\overline{\Gamma}, y : A, f : x \underset{A}{\longrightarrow} y) = \begin{cases} \partial_i^-(\overline{\Gamma}) & \text{if } \dim y \geq i \\ (\partial_i^-(\overline{\Gamma}), y : A, f : x \underset{A}{\longrightarrow} y) & \text{otherwise} \end{cases}$$

$$\partial_i^+([]) = [] \qquad \partial_i^+([\overline{\Gamma}; (x : \star)]) = [\partial_i^+(\overline{\Gamma}); (x : \star)]$$

$$\partial_i^+(\overline{\Gamma}, y : A, f : x \underset{A}{\longrightarrow} y) = \begin{cases} \partial_i^+(\overline{\Gamma}) & \text{if } \dim y > i \\ (\mathrm{drop}(\partial_i^+(\overline{\Gamma})), y : A) & \text{if } \dim y = i \\ (\partial_i^+(\overline{\Gamma}), y : A, f : x \underset{A}{\longrightarrow} y) & \text{otherwise} \end{cases}$$

$$\partial^-(\overline{\Gamma}) = \partial^-_{\dim \overline{\Gamma}}(\overline{\Gamma}) \qquad \partial^+(\overline{\Gamma}) = \partial^+_{\dim \overline{\Gamma}}(\overline{\Gamma})$$

where $\mathrm{drop}\,\overline{\Gamma}$ is the operator drop applied to the last context in $\overline{\Gamma}$

Inference rules.

For regular judgments:

For contexts:

$$\frac{}{\varnothing \vdash}(\text{EC}) \qquad \frac{\Gamma \vdash \qquad \Gamma \vdash A}{\Gamma, x : A}(\text{CE}) \quad \text{Where } x \notin \text{Var}(\Gamma)$$

For types:

$$\frac{\Gamma \vdash}{\Gamma \vdash \star}(\star\text{-INTRO}) \qquad \frac{\Gamma \vdash A \qquad \Gamma \vdash t : A \qquad \Gamma \vdash u : A}{\Gamma \vdash t \underset{A}{\longrightarrow} u}(\rightarrow\text{-INTRO})$$

For terms:

$$\frac{\Gamma \vdash \qquad x : A \in \Gamma}{\Gamma \vdash x : A}(\text{VAR}) \qquad \frac{\overline{\Gamma} \vdash_{\mathsf{ps}} \qquad \overline{\Gamma} \vdash A \qquad \Delta \vdash \overline{\gamma} : \overline{\Gamma}}{\Delta \vdash \mathsf{mop}'_{\overline{\Gamma},A}[\overline{\gamma}] : A[\overline{\gamma}]}(\text{MOP'})$$

$$\frac{\overline{\Gamma} \vdash_{\mathsf{ps}} \qquad \overline{\Gamma} \vdash A \qquad \Delta \vdash \overline{\gamma} : \overline{\Gamma}}{\Delta \vdash \mathsf{mcoh}'_{\overline{\Gamma},A}[\overline{\gamma}] : A[\overline{\gamma}]}(\text{MCOH'})$$

For substitutions:

$$\frac{\Delta \vdash}{\Delta \vdash \langle\rangle : \varnothing}(\text{ES}) \qquad \frac{\Delta \vdash \gamma : \Gamma \qquad \Gamma, x : A \vdash \qquad \Delta \vdash t : A[\gamma]}{\Delta \vdash \langle \gamma, x \mapsto t \rangle : (\Gamma, x : A)}(\text{SE})$$

For monoidal judgments:

For monoidal contexts:

$$\frac{}{[] \vdash}(\text{MEC}) \qquad \frac{\overline{\Gamma} \vdash}{[\overline{\Gamma}; \varnothing] \vdash}(\text{MC}+)$$

$$\frac{[\overline{\Gamma}; \Gamma] \vdash \qquad [\overline{\Gamma}; \Gamma] \vdash A}{[\overline{\Gamma}, (\Gamma, x : A)] \vdash}(\text{MCE}) \quad \text{Where } x \notin \text{Var}([\overline{\Gamma}; \Gamma])$$

For monoidal types:

$$\frac{\overline{\Gamma} \vdash}{\overline{\Gamma} \vdash \star} \qquad \frac{\overline{\Gamma} \vdash A \qquad \overline{\Gamma} \vdash t : A \qquad \overline{\Gamma} \vdash u : A}{\overline{\Gamma} \vdash t \underset{A}{\longrightarrow} u}$$

For monoidal terms:

$$\frac{\overline{\Gamma} \vdash \qquad (x : A) \in \overline{\Gamma}}{\overline{\Gamma} \vdash x : A} \qquad \frac{\overline{\Gamma} \vdash_{\mathsf{ps}} \qquad \overline{\Gamma} \vdash A \qquad \overline{\Delta} \vdash \overline{\gamma} : \overline{\Gamma}}{\overline{\Delta} \vdash \mathsf{mop}'_{\overline{\Gamma},A}[\overline{\gamma}] : A[\overline{\gamma}]}(\text{MOP'})_m$$

$$\frac{\overline{\Gamma} \vdash_{\mathsf{ps}} \qquad \overline{\Gamma} \vdash A \qquad \overline{\Delta} \vdash \overline{\gamma} : \overline{\Gamma}}{\overline{\Delta} \vdash \mathsf{mcoh}'_{\overline{\Gamma},A}[\overline{\gamma}] : A[\overline{\gamma}]}(\text{MCOH'})_m$$

For monoidal substitutions:

$$\frac{}{[] \vdash [] : []}(\text{EMS}) \qquad \frac{\overline{\Delta} \vdash \overline{\gamma} : \overline{\Gamma} \qquad \overline{\Delta}' \vdash}{\overline{\Delta} @ \overline{\Delta}' \vdash [\gamma ;_{\overline{\Delta}} \langle\rangle] : [\overline{\Gamma}; \varnothing]}(\text{MS}+)$$

$$\frac{\overline{\Delta} @ \overline{\Delta}' \vdash [\overline{\gamma} ;_{\overline{\Delta}} \gamma] : [\overline{\Gamma}; \Gamma] \qquad [\overline{\Gamma}'; (\Gamma, x : A)] \vdash \qquad \overline{\Delta}' \vdash t : A[\overline{\gamma} ;_{\overline{\Delta}} \gamma]}{\overline{\Delta} @ \overline{\Delta}' \vdash [\gamma' ;_{\overline{\Delta}} \langle \gamma, x \mapsto t \rangle] : [\overline{\Gamma}; (\Gamma, x : A)]}(\text{MSE})$$

For substitutions between contexts and monoidal contexts:

$$\frac{\Delta \vdash}{\Delta \vdash [] : []} \qquad \frac{\Delta \vdash \overline{\gamma} : \overline{\Gamma} \qquad \Delta \vdash \gamma : \Gamma}{\Delta \vdash [\overline{\gamma}; \gamma] : [\overline{\Gamma}; \Gamma]}$$

$$\frac{\overline{\Delta} \vdash}{\overline{\Delta} \vdash \langle\rangle : \varnothing} \qquad \frac{\overline{\Delta} \vdash \gamma : \Gamma \qquad \Gamma, x : A \vdash \qquad \overline{\Delta} \vdash t : A[\gamma]}{\overline{\Delta} \vdash \langle \gamma, x \mapsto t \rangle : (\Gamma, x : A)}$$

Where in the rules (MOP') and $(\text{MOP'})_m$ the type A is assumed to satisfy $(\mathrm{C}_{\mathsf{mop'}})$ and in the rules (MCOH') and $(\text{MCOH'})_m$ the type A is assumed to satisfy $(\mathrm{C}_{\mathsf{mcoh'}})$.

$$(\mathrm{C}_{\mathsf{mop'}}) \quad \begin{array}{l} \text{Either } A = \star \text{ and } \dim \overline{\Gamma} = 0 \\ \text{Or } A = t \underset{B}{\longrightarrow} u \text{ with } \begin{cases} \mathrm{Var}(t) \cup \mathrm{Var}(A) = \mathrm{Var}(\partial^-(\overline{\Gamma})) \\ \mathrm{Var}(u) \cup \mathrm{Var}(A) = \mathrm{Var}(\partial^+(\overline{\Gamma})) \end{cases} \end{array}$$

$$(\mathrm{C}_{\mathsf{mcoh'}}) \quad \begin{array}{l} \text{Either } A = \star \text{ and } \overline{\Gamma} = [] \\ \text{Or } A = t \underset{B}{\longrightarrow} u \text{ with } \begin{cases} \mathrm{Var}(t) \cup \mathrm{Var}(A) = \mathrm{Var}(\overline{\Gamma}) \\ \mathrm{Var}(u) \cup \mathrm{Var}(A) = \mathrm{Var}(\overline{\Gamma}) \end{cases} \end{array}$$

Semantics.

- $\mathcal{S}_{\mathsf{MCaTT'}}$ is equivalent to $\mathcal{S}_{\mathsf{MCaTT}}$
- $\mathbf{Mod}(\mathcal{S}_{\mathsf{MCaTT'}})$ is equivalent to $\mathbf{Mod}(\mathcal{S}_{\mathsf{MCaTT}})$

A.5 The theory $T_\square$

Syntax.

Contexts: lists $(x_0 : A_0, \ldots, x_n : A_n)$ with x_i variables and A_i types

Types: either $\star$ or of the form $\mathsf{Path}^i\ A\ t\ u$ with i a dimension variable, A type and t and u terms

Terms: either variables or of the form $t\ i$ where t is a term and i a dimension variable

Substitutions: lists $\langle x_0 \mapsto t_0, \ldots, x_n \mapsto t_n \rangle$ with x_i variables and t_i terms

The judgments are now indexed by a list of dimension variables, we denote it $\vdash_I$

Inference rules.

For contexts:

$$\frac{}{\varnothing \vdash} \qquad \frac{\Gamma \vdash A \qquad x \notin \mathrm{Var}(\Gamma)}{(\Gamma, x : A) \vdash}$$

For types:

$$\frac{\Gamma \vdash}{\Gamma \vdash_I \star} \qquad \frac{\Gamma \vdash_{I,i} A \qquad \Gamma \vdash_I t_0 : A[0/i] \qquad \Gamma \vdash_I t_1 : A[1/i]}{\Gamma \vdash_I \mathsf{Path}^i\ A\ t_0\ t_1}$$

For terms:

$$\frac{\Gamma \vdash \qquad (x : A) \in \Gamma}{(\Gamma, x : A) \vdash x : A} \qquad \frac{\Gamma \vdash_I t : \mathsf{Path}^i\ A\ u_0\ u_1 \qquad r \notin I}{\Gamma \vdash_{I,r} t\ r : A[r/i]}$$

For substitutions:

$$\frac{\Delta \vdash}{\Delta \vdash \langle\rangle : \varnothing} \qquad \frac{\Delta \vdash \sigma : \Gamma \qquad \Gamma \vdash A \qquad \Delta \vdash t : A\,[\sigma]}{\Delta \vdash \langle \sigma, x \mapsto t \rangle : (\Gamma, x : A)}$$

Rewriting:

$$\frac{\Gamma \vdash_I t : \mathsf{Path}^i\ A\ u_0\ u_1}{\Gamma \vdash_I t\ \epsilon \rightsquigarrow u_\epsilon : A[\epsilon/i]} (\epsilon = 0, 1)$$

Semantics.

$$\mathcal{S}_{T_\square} = \mathbf{FinCSet}^{\mathrm{op}}$$
$$\mathbf{Mod}(\mathcal{S}_{T_\square}) = \mathbf{CSet}$$

A.6 The theory $\mathsf{CaTT}_\square$

Contexts: lists $(x_0 : A_0, \ldots, x_n : A_n)$ with x_i variables and A_i types

Types: either $\star$ or of the form $\mathsf{Path}^i\ A\ t\ u$ with i a dimension variable, A type and t and u terms

Terms: either variables or of the form $\mathsf{op}_{\Gamma,A}[\gamma]$ or $\mathsf{coh}_{\Gamma,A}[\gamma]$ where Γ is a ps-context, A a type and γ a substitution, or of the form $t\ i$ where t is a term and i a dimension variable

Substitutions: lists $\langle x_0 \mapsto t_0, \ldots, x_n \mapsto t_n \rangle$ with x_i variables and t_i terms

Extrusion.

$$\Sigma_X \varnothing = \varnothing$$
$$\Sigma_X(\Gamma, x : A) = \begin{cases} (\Sigma_X \Gamma, x : A, x^+ : A^+, \vec{x} : \Sigma^{x,x^+} A) & \text{If } x \in X \\ (\Sigma_X \Gamma, x : A) & \text{Otherwise} \end{cases}$$

$$\Sigma^{a,b}\star = \mathsf{Path}^i\ \star\ a\ b \qquad \Sigma^{a,b}(\mathsf{Path}^i\ A\ t\ u) = \mathsf{Path}^i\ (\Sigma^{(a\ i),(b\ i)} A)\ \Sigma t\ \Sigma u$$
$$\Sigma x = \vec{x} \qquad \Sigma(t\ i) = (\Sigma t)\ i$$

where A^+ is defined by

$$\star^+ = \star \qquad (\mathsf{Path}^i\ A\ t\ u)^+ = \mathsf{Path}^i\ A^+\ t^+\ u^+$$
$$x^+ = x^+ \qquad (t\ i)^+ = t^+\ i$$

Where X is set that is downwards closed in Γ.

Ps-contexts.

$$\frac{}{x : \star \vdash_{\mathsf{ps}}}(\text{PSS}) \qquad \frac{\Gamma \vdash_{\mathsf{ps}} X}{\Sigma_X \Gamma \vdash_{\mathsf{ps}} X^+}(\text{PSE})$$

$$\frac{\Gamma \vdash_{\mathsf{ps}} X}{\Gamma \vdash_{\mathsf{ps}}}(\text{PS}) \qquad \frac{\Gamma \vdash_{\mathsf{ps}}}{\Sigma\Gamma \vdash_{\mathsf{ps}} \mathrm{Var}(\Gamma)^+}(\text{PS}+)$$

Inference rules.

For contexts:

$$\frac{}{\varnothing \vdash} \qquad \frac{\Gamma \vdash A}{(\Gamma, x : A) \vdash} \quad \text{where } x \notin \mathrm{Var}(\Gamma)$$

For types:

$$\frac{\Gamma \vdash}{\Gamma \vdash_I \star} \qquad \frac{\Gamma \vdash_{I,i} A \qquad \Gamma \vdash_I t_0 : A[0/i] \qquad \Gamma \vdash_I t_1 : A[1/i]}{\Gamma \vdash_I \mathsf{Path}^i\ A\ t_0\ t_1}$$

For terms:

$$\frac{\Gamma \vdash \qquad (x : A) \in \Gamma}{(\Gamma, x : A) \vdash x : A} \qquad \frac{\Gamma \vdash_I t : \mathsf{Path}^i\ A\ u_0\ u_1 \qquad r \notin I}{\Gamma \vdash_{I,r} t\ r : A[r/i]}$$

$$\frac{\Gamma \vdash_{\mathsf{ps}} \qquad \Gamma \vdash A\Delta \vdash \gamma : \Gamma}{\Delta \vdash \mathsf{op}_{\Gamma,A}[\gamma] : A\,[\gamma]}(\square\text{OP}) \qquad \frac{\Gamma \vdash_{\mathsf{ps}} \qquad \Gamma \vdash A \qquad \Delta \vdash \gamma : \Gamma}{\Delta \vdash \mathsf{coh}_{\Gamma,A}[\gamma] : A\,[\gamma]}(\square\text{COH})$$

For substitutions:

$$\frac{\Delta \vdash}{\Delta \vdash \langle\rangle : \varnothing} \qquad \frac{\Delta \vdash \sigma : \Gamma \qquad \Gamma \vdash A \qquad \Delta \vdash t : A\,[\sigma]}{\Delta \vdash \langle \sigma, x \mapsto t \rangle : (\Gamma, x : A)}$$

Rewriting:

$$\frac{\Gamma \vdash_I t : \mathsf{Path}^i\ A\ u_0\ u_1}{\Gamma \vdash_I t\ \epsilon \rightsquigarrow u_\epsilon : A[\epsilon/i]}(\epsilon = 0, 1)$$

Where in the rule ($\square$OP) we assume that A satisfies the condition ($\mathrm{C}_{\mathrm{cohop},\square}$) and in the rule ($\square$COH) we assume that A satisfies the condition ($\mathrm{C}_{\mathrm{coh},\square}$)

$$(\mathrm{C}_{\mathrm{cohop},\square}) \quad \begin{array}{l} \Gamma \text{ is of dimension } n \text{ and } A \text{ is of the form } (t_0, \ldots, t_{n-1}) \to (u_0, \ldots, u_{n-1}) \\ \text{with for all } i, \begin{cases} \mathrm{Var}(t_i) \cup \mathrm{Var}(B_i^-) = \mathrm{Var}(\partial_i^-(\Gamma)) \\ \mathrm{Var}(t_i) \cup \mathrm{Var}(B_i^+) = \mathrm{Var}(\partial_i^+(\Gamma)) \end{cases} \end{array}$$

$(\mathrm{C}_{\mathrm{coh},\square})$ $\mathrm{Var}(A) = \mathrm{Var}(\Gamma)$

A.7 Summary of the formalized results

A.7.1 Formalized results for the theory GSeTT

Definition of the syntax. The pre-syntax, where we define the expressions for contexts, types terms and substitutions is declared in the file `Syntax.agda` as follows

```
data Pre-Ty : Set
data Pre-Tm : Set

data Pre-Ty where
  ∗ : Pre-Ty
```

```
  ⇒ : Pre-Ty → Pre-Tm → Pre-Tm → Pre-Ty

data Pre-Tm where
  Var : ℕ → Pre-Tm

Pre-Ctx : Set₁
Pre-Ctx = list (ℕ × Pre-Ty)

Pre-Sub : Set₁
Pre-Sub = list (ℕ × Pre-Tm)
```

and we define the action of substitutions on types and terms

```
_[_]Pre-Ty : Pre-Ty → Pre-Sub → Pre-Ty
_[_]Pre-Tm : Pre-Tm → Pre-Sub → Pre-Tm

∗ [ γ ]Pre-Ty = ∗
⇒ A t u [ γ ]Pre-Ty = ⇒ (A [ γ ]Pre-Ty) (t [ γ ]Pre-Tm) (u [ γ ]Pre-Tm)
Var x [ nil ]Pre-Tm = Var x
Var x [ γ :: (v , t) ]Pre-Tm = if x ≡ v then t else ((Var x) [ γ ]Pre-Tm)

_∘_ : Pre-Sub → Pre-Sub → Pre-Sub
nil ∘ γ = nil
(γ :: (x , t)) ∘ δ = (γ ∘ δ) :: (x , (t [ δ ]Pre-Tm))
```

Judgments and rules of the theory. We then define the judgments in the file `Rules.agda`, as inductive inductive types, following, where the inference rules give the generators

```
data _⊢C : Pre-Ctx → Set
data _⊢T_ : Pre-Ctx → Pre-Ty → Set
data _⊢t_#_ : Pre-Ctx → Pre-Tm → Pre-Ty → Set
data _⊢S_>_ : Pre-Ctx → Pre-Sub → Pre-Ctx → Set

data _⊢C where
  ec : nil ⊢C
  cc : ∀ {Γ A} → Γ ⊢C → Γ ⊢T A → (Γ :: ((length Γ) , A)) ⊢C

data _⊢T_ where
  ob : ∀ {Γ} → Γ ⊢C → Γ ⊢T ∗
  ar : ∀ {Γ A t u} → Γ ⊢t t # A → Γ ⊢t u # A → Γ ⊢T ⇒ A t u

data _⊢t_#_ where
  var : ∀ {Γ x A} → Γ ⊢C → x # A ∈ Γ → Γ ⊢t (Var x) # A

data _⊢S_>_ where
  es : ∀ {Δ} → Δ ⊢C → Δ ⊢S nil > nil
  sc : ∀ {Δ Γ γ x A t} → Δ ⊢S γ > Γ → (Γ :: (x , A)) ⊢C
                       → Δ ⊢t t # (A [ γ ]Pre-Ty)
                       → Δ ⊢S (γ :: (x , t)) > (Γ :: (x , A))
```

Note that in this theory, the judgment _⊢S_>_ is not mutually inductive with the others and could be defined separately, however in more complicated theories, we define it by mutual induction and thus proceed the same way here. We then prove in this file the easier properties satisfied by these rules, and particular we prove cut admissibility

```
[]T : ∀ {Γ A Δ γ} → Γ ⊢T A
                  → Δ ⊢S γ > Γ
                  → Δ ⊢T (A [ γ ]Pre-Ty)
[]t : ∀ {Γ A t Δ γ} → Γ ⊢t t # A
                    → Δ ⊢S γ > Γ
                    → Δ ⊢t (t [ γ ]Pre-Tm) # (A [ γ ]Pre-Ty)
```

Structure of category with families. In the file `CwF-Structure.agda`, we proceed with showing all the equalities that define the syntactic category and endow it with a structure of category with families. We have already proved cut admissibility, we now prove in particular the functoriality of the application of substitutions and the associativity and unitality of the composition

```
[∘]T : ∀ {Γ Δ Θ A γ δ} → Γ ⊢T A
                       → Δ ⊢S γ > Γ
                       → Θ ⊢S δ > Δ
                       → ((A [ γ ]Pre-Ty) [ δ ]Pre-Ty) == (A [ γ ∘ δ ]Pre-Ty)
[∘]t : ∀ {Γ Δ Θ A t γ δ} → Γ ⊢t t # A
                         → Δ ⊢S γ > Γ
                         → Θ ⊢S δ > Δ
                        → ((t [ γ ]Pre-Tm) [ δ ]Pre-Tm) == (t [ γ ∘ δ ]Pre-Tm)
∘-admissibility : ∀ {Γ Δ Θ γ δ} → Δ ⊢S γ > Γ
                                → Θ ⊢S δ > Δ
                                → Θ ⊢S (γ ∘ δ) > Γ
∘-associativity : ∀ {Γ Δ Θ Ξ γ δ θ} → Δ ⊢S γ > Γ
                                     → Θ ⊢S δ > Δ
                                     → Ξ ⊢S θ > Θ
                                     → ((γ ∘ δ) ∘ θ) == (γ ∘ (δ ∘ θ))
∘-left-unit : ∀{Γ Δ γ} → Δ ⊢S γ > Γ
                       → (Pre-id Γ ∘ γ) == γ
∘-right-unit : ∀ {Δ γ} →  (γ ∘ Pre-id Δ) == γ
```

Uniqueness of derivations. In the file `Uniqueness-Derivations.agda` we prove by mutual induction that every derivable judgment is derivable in a unique way, by showing that the type of derivations of this judgment is contractible [63], using a terminology from homotopy type theory. We can simplify the statement even further by showing that every judgment defines a proposition, again in the sense of homotopy type theory

```
is-prop-⊢C : ∀ Γ → is-prop (Γ ⊢C)
is-prop-⊢T : ∀ Γ A → is-prop (Γ ⊢T A)
is-prop-⊢t : ∀ Γ A t → is-prop (Γ ⊢t t # A)
is-prop-⊢S : ∀ Δ Γ γ → is-prop (Δ ⊢S γ > Γ)
```

Decidability of type checking. We show that every judgment defines a decidable type in the file `Dec-Type-Checking.agda`

```
dec-⊢C : ∀ Γ → dec (Γ ⊢C)
dec-⊢T : ∀ Γ A → dec (Γ ⊢T A)
dec-⊢t : ∀ Γ A t → dec (Γ ⊢t t # A)
dec-⊢S : ∀ Δ Γ γ → dec (Δ ⊢S γ > Γ)
```

These are the most involved proofs and they in particular give a certified implementation of a type checker for this theory. Since the theory GSeTT is not very relevant in practice, this is not so important here, however for more complicated theories formally proving the decidability in Agda gives a certified implementation of the theory, of which one could extract a code that computes.

Disk and Sphere context. The Agda definition of the disks and sphere contexts are formalized as follows

```
𝕊 : ℕ → Pre-Ctx
𝔻 : ℕ → Pre-Ctx

𝕊 0 = nil
𝕊 (S n) = (𝔻 n) :: (length (𝔻 n) , n⇒ n)
𝔻 n = (𝕊 n) :: (length (𝕊 n) , n⇒ n)
```

we then prove that they define valid contexts

```
𝕊⊢ : ∀ n → 𝕊 n ⊢C
𝔻⊢ : ∀ n → 𝔻 n ⊢C
n⇒ : ℕ → Pre-Ty
```

and we show the familial representability of the type functor

```
Ty-n : ∀ {Γ} → Σ (ℕ × Pre-Sub) (λ {(n , γ) →  Γ ⊢S γ > 𝕊 n})
             → Σ Pre-Ty (λ A → (Γ ⊢T A))
Ty-classifier : ∀ Γ → is-equiv (Ty-n {Γ})
```

A.7.2 Formalized results for globular type theories

In order to study this general framework for globular type theories along with their properties, we have formalized it in Agda [12], using de Bruijn levels for variables. This construction is found in the directory `Globular-TT/` of the project and follows the same structure as the one of the folder `GSeTT` that we have presented in Section 2.2. The folder `CaTT` is dedicated to the work in progress of formalizing the type theory CaTT as a particular case of a globular type theory (which is slightly more difficult in a constructive and proof relevant setup, that what we have presented here). We define in particular the syntax of a globular type theory in the file `Syntax.agda` as follows, and define the action of substitutions on types and terms as well as their composition

```
module Globular-TT.Syntax {l} (index : Set l) where

  data Pre-Ty : Set (lsuc l)
  data Pre-Tm : Set (lsuc l)
  data Pre-Sub : Set (lsuc l)
  data Pre-Ctx : Set (lsuc l)
```

```
  data Pre-Ty where
    ∗ : Pre-Ty
    ⇒ : Pre-Ty → Pre-Tm → Pre-Tm → Pre-Ty

  data Pre-Tm where
    Var : ℕ → Pre-Tm
    Tm-constructor : ∀ (i : index) → Pre-Sub → Pre-Tm

  data Pre-Sub where
    <> : Pre-Sub
    <_,_↦_> : Pre-Sub → ℕ → Pre-Tm → Pre-Sub

  data Pre-Ctx where
    ∅ : Pre-Ctx
    _·_#_ : Pre-Ctx → ℕ → Pre-Ty → Pre-Ctx

  _[_]Pre-Ty : Pre-Ty → Pre-Sub → Pre-Ty
  _[_]Pre-Tm : Pre-Tm → Pre-Sub → Pre-Tm
  _∘_ : Pre-Sub → Pre-Sub → Pre-Sub
```

We then define the judgments as inductive inductive types whose generators are the inference rules of the theory, they are indexed over typed contexts in the theory GSeTT, which we include as an argument of the module.

```
module Globular-TT.Rules {l} (index : Set l)
  (rule : index → GSeTT.Typed-Syntax.Ctx × (Globular-TT.Syntax.Pre-Ty index))
where
  open import Globular-TT.Syntax index

  {- Notational shortcuts : the context corresponding to an index -}
  Ci : index → Pre-Ctx
  Ci i = GPre-Ctx (fst (fst (rule i)))

  Ti : index → Pre-Ty
  Ti i = snd (rule i)

  data _⊢C : Pre-Ctx → Set (lsuc l)
  data _⊢T_ : Pre-Ctx → Pre-Ty → Set (lsuc l)
  data _⊢t_#_ : Pre-Ctx → Pre-Tm → Pre-Ty → Set (lsuc l)
  data _⊢S_>_ : Pre-Ctx → Pre-Sub → Pre-Ctx → Set (lsuc l)

  data _⊢C where
    ec : ∅ ⊢C
    cc : ∀ {Γ A} → Γ ⊢C → Γ ⊢T A → (Γ · (C-length Γ) # A) ⊢C

  data _⊢T_ where
    ob : ∀ {Γ} → Γ ⊢C → Γ ⊢T ∗
    ar : ∀ {Γ A t u} → Γ ⊢T A → Γ ⊢t t # A → Γ ⊢t u # A → Γ ⊢T ⇒ A t u

  data _⊢t_#_ where
```

```
    var : ∀ {Γ x A} → Γ ⊢C → x # A ∈ Γ → Γ ⊢t (Var x) # A
    tm : ∀ {Δ γ} → (i : index)
                 → Ci i ⊢T Ti i
                 → Δ ⊢S γ > Ci i
                 → Δ ⊢t Tm-constructor i γ # (Ti i [ γ ]Pre-Ty)

  data _⊢S_>_ where
    es : ∀ {Δ} → Δ ⊢C → Δ ⊢S <> > ∅
    sc : ∀ {Δ Γ γ x A t} → Δ ⊢S γ > Γ
                         → (Γ · x # A) ⊢C
                         → (Δ ⊢t t # (A [ γ ]Pre-Ty))
                         → Δ ⊢S < γ , x ↦ t > > (Γ · x # A)
```

Syntactic properties. We can first check by mutual induction that all the properties of Proposition 2 hold in any globular type theory. Our formalization shows all these properties in the file `Rules.agda`. Note that in this framework, term constructors depend mutually inductively on substitution, which makes some of these properties harder to prove - If we are not careful, some of the inductions become ill-formed. For this reason, some of the properties stated here are proved in the file `CwF-Structure.agda`, although the corresponding properties for GSeTT are in the file `Rules.agda`. The statements corresponding to these properties are similar to the ones we have presented for the theory GSeTT.

Structure of category with families. We have defined all the structure of a cut-full category as we have presented in Section 1.1, and proved that all the defining equations of a cut-full type theory hold for globular type theories. The corresponding proofs are in the file `CwF-Structure.agda`. The fact that term constructors depend on substitution also makes this fact a little bit harder to prove, and most of the results are mutually inductive together. This makes the proof very hard to check by hand, and illustrates the use of having a proof-assistant as Agda to manipulate type theories in our case.

Decidability of type checking. The existence of a derivation for any judgment in a globular type theory is decidable, and we have proved this by induction. For this proof, we follow the structure described in Proposition 43, and it requires a subtle argument keeping track both of the depth and the dimension of the terms to ensure that the induction is well-formed. In particular, it is this property that motivates us to assume that $\dim A_i \geq \dim \Gamma_i - 1$. We have formalized this argument in the file `Dec-Type-Checking.agda`.

```
module Globular-TT.Dec-Type-Checking {l} (index : Set l)
   (rule : index → GSeTT.Typed-Syntax.Ctx × (Globular-TT.Syntax.Pre-Ty index))
   (assumption : Globular-TT.Rules.well-founded index rule)
   (eqdec-index : eqdec index)
where

  dec-G⊢T : ∀ (Γ : GSeTT.Typed-Syntax.Ctx) n A
            → dim A ≤ n
            →  dec (GPre-Ctx (fst Γ) ⊢T A)
  dec-G⊢t : ∀ (Γ : GSeTT.Typed-Syntax.Ctx) n d A t
            → dim A ≤ n → depth t ≤ d
            →  dec (GPre-Ctx (fst Γ) ⊢t t # A)
```

```
  dec-G⊢S : ∀ (Δ Γ : GSeTT.Typed-Syntax.Ctx) n d γ
            → dimC (GPre-Ctx (fst Γ)) ≤ n
            → depthS γ ≤ d
            → dec (GPre-Ctx (fst Δ) ⊢S γ > GPre-Ctx (fst Γ))

  dec-⊢C : ∀ Γ → dec (Γ ⊢C)
  dec-⊢T : ∀ Γ A →  dec (Γ ⊢T A)
  dec-⊢t : ∀ Γ A t → dec (Γ ⊢t t # A)
  dec-⊢S:G : ∀ Δ (Γ : GSeTT.Typed-Syntax.Ctx) γ
           → dec (Δ ⊢S γ > GPre-Ctx (fst Γ))

  dec-⊢S : ∀ Δ Γ γ → dec (Δ ⊢S γ > Γ)
```

The statements separate in three groups, corresponding to the three steps of the proof that we have sketch for CaTT (Proposition 43). We first prove the decidability for derivability of judgments in a context that is in the theory GSeTT. We then prove the derivability for the regular judgments, restricting the substitution to those whose target is in the theory GSeTT. We then can prove it for every substitution. The first part of this proof is a bit subtle, as it requires keeping track both of the dimension and the depth of the objects we manipulate, and relies on the assumption (which is denoted by the argument `assumption` of the module) that for all $j \in J$, we have the inequality $\dim \Gamma_j \leq \dim A_j + 1$, we have formalized this by the type well-founded defined as follows

```
  well-founded : Set (lsuc l)
  well-founded = ∀ (i : index) → Ci i ⊢T Ti i → dimC (Ci i) ≤ dim (Ti i)
```

We also rely on the fact that our indexing set for the term constructors have decidable equality.

Uniqueness of derivation. Every judgment is derivable in at most one way in a globular type theory: This is a straightforward mutual induction, and can be seen by the fact that the syntactic expression of a context, type, term or substitution completely encodes its possible derivation. We have proved this property in the file `Uniqueness-Derivations.agda` of our formalization. The formulation is similar to the one presented in Section 2.2

```
module Globular-TT.Uniqueness-Derivations {l} (index : Set l)
  (rule : index → GSeTT.Typed-Syntax.Ctx × (Globular-TT.Syntax.Pre-Ty index))
where

  is-prop-⊢C : ∀ Γ → is-prop (Γ ⊢C)
  is-prop-⊢T : ∀ Γ A → is-prop (Γ ⊢T A)
  is-prop-⊢t : ∀ Γ A t → is-prop (Γ ⊢t t # A)
  is-prop-⊢S : ∀ Δ Γ γ → is-prop (Δ ⊢S γ > Γ)
```

Familial representability of Ty**.** We have defined the disk and sphere contexts and proved their validity as well as the familial representability of the type functor in this case in the file `Disks.agda`.

```
  𝕊 : ℕ → Pre-Ctx
  𝔻 : ℕ → Pre-Ctx
  n⇒ : ℕ → Pre-Ty
```

```
𝕊⊢ : ∀ n → 𝕊 n ⊢C
𝔻⊢ : ∀ n → 𝔻 n ⊢C
𝕊⊢⇒ : ∀ n → 𝕊 n ⊢T n⇒ n

Ty-n : ∀ {Γ} → Σ (ℕ × Pre-Sub) (λ {(n , γ) →  Γ ⊢S γ > 𝕊 n})
             → Σ Pre-Ty (λ A → (Γ ⊢T A))
Ty-classifier : ∀ Γ → is-equiv (Ty-n {Γ})
```

A.7.3 Formalized results for the theory CaTT

The theory CaTT is an instance of a globular type theory, and thus in order to formalize, we can simply instantiate our previous formalization with the right indexes. This requires formalizing the algorithm for recognizing ps-contexts, that we have formalized in the file `Ps-contexts.agda` as well as the side conditions, that we have defined in the file `Fullness.agda`. However, due to the proof-relevance setting of Agda, it is not completely straightforward to translate these definitions directly, in particular we have to be careful about not having different witnesses to the equality $\mathrm{Var}(\Gamma) = \mathrm{Var}(A)$. There are various ways to circumvent this issue. One is to only work with variables of maximal dimension, another is to implement sets in Agda. In all the cases, this requires some substantial extra work, that we do not discuss here.

Bibliography

[1] Thorsten Altenkirch and Ambrus Kaposi. Type theory in type theory using quotient inductive types. *ACM SIGPLAN Notices*, 51(1):18–29, 2016.

[2] Thorsten Altenkirch and Ondrej Rypacek. A syntactical approach to weak omega-groupoids. In *Computer Science Logic (CSL'12)-26th International Workshop/21st Annual Conference of the EACSL*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2012.

[3] Mathieu Anel, Georg Biedermann, Eric Finster, and André Joyal. A generalized blakers-massey theorem. *arXiv preprint arXiv:1703.09050*, 2017.

[4] Dimitri Ara. *Sur les ∞-groupoïdes de Grothendieck et une variante ∞-catégorique*. PhD thesis, Ph. D. thesis, Université Paris 7, 2010.

[5] Multiple authors. The initiality project on nlab.

[6] Steve Awodey. Natural models of homotopy type theory. *Mathematical Structures in Computer Science*, 28(2):241–286, 2018.

[7] Steve Awodey, Álvaro Pelayo, and Michael A Warren. Voevodsky's univalence axiom in homotopy type theory. *Notices of the AMS*, 60(9):1164–1167, 2013.

[8] Steve Awodey and Michael A Warren. Homotopy theoretic models of identity types. *arXiv preprint arXiv:0709.0248*, 2007.

[9] John C Baez and Michael Shulman. Lectures on n-categories and cohomology. *arXiv preprint math/0608420*, 2006.

[10] Henk P Barendregt. Lambda calculi with types. 1992.

[11] Michael A Batanin. Monoidal globular categories as a natural environment for the theory of weakn-categories. *Advances in Mathematics*, 136(1):39–103, 1998.

[12] Thibaut Benjamin. Catt formalization. `https://github.com/ThiBen/catt-formalization/`.

[13] Thibaut Benjamin. CubiCaTT, 2019. `https://github.com/ThiBen/cubicatt`.

[14] Thibaut Benjamin. Monoidal weak ω-categories as models of a type theory, 2019. Preprint available at `http://www.lix.polytechnique.fr/~tbenjamin/articles/publications/publications.html`.

[15] Thibaut Benjamin, Eric Finster, and Samuel Mimram. The CaTT proof assistant, 2018. `https://github.com/ThiBen/catt`.

[16] Thibaut Benjamin, Eric Finster, and Samuel Mimram. Globular weak ω-categories as models of a type theory, 2020. Preprint available at `http://www.lix.polytechnique.fr/~tbenjamin/articles/publications/publications.html`.

[17] Thibaut Benjamin and Samuel Mimram. Suspension et Fonctorialité: Deux Opérations Implicites Utiles en CaTT. In *Journées Francophones des Langages Applicatifs*, 2019.

[18] Clemens Berger. A cellular nerve for higher categories. *Advances in Mathematics*, 169(1):118–175, 2002.

[19] Clemens Berger, Paul-André Mellies, and Mark Weber. Monads with arities and their associated theories. *Journal of Pure and Applied Algebra*, 216(8-9):2029–2048, 2012.

[20] Guillaume Brunerie. On the homotopy groups of spheres in homotopy type theory. *arXiv preprint arXiv:1606.05916*, 2016.

[21] Albert Burroni. Higher-dimensional word problems with applications to equational logic. *Theoretical computer science*, 115(1):43–62, 1993.

[22] Kevin Buzzard. Xena project. `https://xenaproject.wordpress.com/`.

[23] John Cartmell. Generalised algebraic theories and contextual categories. *Annals of pure and applied logic*, 32:209–243, 1986.

[24] Simon Castellan, Pierre Clairambault, and Peter Dybjer. Undecidability of equality in the free locally cartesian closed category (extended version). *arXiv preprint arXiv:1504.03995*, 2015.

[25] Eugenia Cheng and Aaron Lauda. Higher-dimensional categories: an illustrated guide book. *Preprint*, 2004.

[26] Alonzo Church. *The calculi of lambda-conversion*. Number 6. Princeton University Press, 1985.

[27] Pierre Clairambault and Peter Dybjer. The biequivalence of locally cartesian closed categories and martin-löf type theories. In *International Conference on Typed Lambda Calculi and Applications*, pages 91–106. Springer, 2011.

[28] Pierre Clairambault and Peter Dybjer. The biequivalence of locally cartesian closed categories and martin-löf type theories. 2014.

[29] Cyril Cohen, Thierry Coquand, Simon Huber, and Anders Mörtberg. Cubical type theory: a constructive interpretation of the univalence axiom. *arXiv preprint arXiv:1611.02108*, 2016.

[30] Thierry Coquand and Gérard Huet. *The calculus of constructions*. PhD thesis, INRIA, 1986.

[31] Pierre-Louis Curien. Operads, clones, and distributive laws. In *Operads and universal algebra*, pages 25–49. World Scientific, 2012.

[32] Peter Dybjer. Internal Type Theory. In *Types for Proofs and Programs. TYPES 1995*, pages 120–134. Springer, Berlin, Heidelberg, 1996.

[33] Eric Finster and Samuel Mimram. A Type-Theoretical Definition of Weak ω-Categories. In *2017 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–12, 2017.

[34] Eric Finster, David Reutter, and Jamie Vicary. A type theory for strictly unital ∞-categories. *arXiv preprint arXiv:2007.08307*, 2020.

[35] Peter Gabriel and Friedrich Ulmer. *Lokal präsentierbare kategorien*, volume 221. Springer-Verlag, 2006.

[36] Georges Gonthier. A computer-checked proof of the four colour theorem.

[37] Georges Gonthier, Andrea Asperti, Jeremy Avigad, Yves Bertot, Cyril Cohen, François Garillot, Stéphane Le Roux, Assia Mahboubi, Russell O'Connor, Sidi Ould Biha, et al. A machine-checked proof of the odd order theorem. In *International Conference on Interactive Theorem Proving*, pages 163–179. Springer, 2013.

[38] Marco Grandis. *Category Theory and Applications: A Textbook for Beginners*. World Scientific, 2018.

[39] Alexander Grothendieck. Pursuing stacks. Unpublished manuscript, 1983.

[40] Allen Hatcher. *Algebraic Topology*. Cambridge University Press, 2002.

[41] Martin Hofmann. Syntax and semantics of dependent types. In *Extensional Constructs in Intensional Type Theory*, pages 13–54. Springer, 1997.

[42] Bart Jacobs. Comprehension categories and the semantics of type dependency. *Theoretical Computer Science*, 107(2):169–207, 1993.

[43] André Joyal. Disks, duality and θ-categories. *preprint*, 1997.

[44] Ambroise Lafont. Weak Omega Agda, 2018. `https://github.com/amblafont/omegatt-agda`.

[45] F William Lawvere. Functorial semantics of algebraic theories. *Proceedings of the National Academy of Sciences of the United States of America*, 50(5):869, 1963.

[46] Chaitanya Leena-Subramaniam and Peter LeFanu Lumsdaine. Contextual categories as monoids in the category of collections, 2019. Presentation at HoTT 2019, `https://sites.google.com/view/chaitanyals`.

[47] Tom Leinster. A survey of definitions of n-category. *Theory and applications of Categories*, 10(1):1–70, 2002.

[48] Tom Leinster. *Higher operads, higher categories*, volume 298. Cambridge University Press, 2004.

[49] Xavier Leroy et al. The compcert verified compiler, 2012.

[50] Daniel R Licata and Robert Harper. 2-dimensional directed type theory. *Electronic Notes in Theoretical Computer Science*, 276:263–289, 2011.

[51] Daniel R Licata and Michael Shulman. Calculating the fundamental group of the circle in homotopy type theory. In *2013 28th Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 223–232. IEEE, 2013.

[52] Peter LeFanu Lumsdaine. Weak ω-categories from intensional type theory. In *International Conference on Typed Lambda Calculi and Applications*, pages 172–187. Springer, 2009.

[53] Saunders Mac Lane. *Categories for the working mathematician*, volume 5. Springer Science & Business Media, 2013.

[54] Georges Maltsiniotis. Grothendieck ∞-groupoids, and still another definition of ∞-categories. Preprint `arXiv:1009.2331`, 2010.

[55] Per Martin-Löf and Giovanni Sambin. *Intuitionistic type theory*, volume 9. Bibliopolis Naples, 1984.

[56] Paige Randall North. Towards a directed homotopy type theory. *Electronic Notes in Theoretical Computer Science*, 347:223–239, 2019.

[57] Emily Riehl. *Category theory in context*. Courier Dover Publications, 2017.

[58] Emily Riehl and Michael Shulman. A type theory for synthetic ∞-categories. *arXiv preprint arXiv:1705.07442*, 2017.

[59] Michael Shulman. Categorical logic from a categorical point of view. *Available on the web*, 2016.

[60] Ross Street. Limits indexed by category-valued 2-functors. *Journal of Pure and Applied Algebra*, 8(2):149–181, 1976.

[61] Thomas Streicher. Contextual categories and categorical semantics of dependent types. In *Semantics of Type Theory*, pages 43–111. Springer, 1991.

[62] Paul Taylor. *Practical foundations of mathematics*, volume 59. Cambridge University Press, 1999.

[63] The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. `https://homotopytypetheory.org/book`, Institute for Advanced Study, 2013.

[64] Benno Van Den Berg and Richard Garner. Types are weak ω-groupoids. *Proceedings of the London Mathematical Society*, 102(2):370–394, 2011.

[65] Vladimir Voevodsky. A c-system defined by a universe category. *Theory Appl. Categ*, 30(37):1181–1215, 2015.

[66] Vladimir Voevodsky. Mathematical theory of type theories and the initiality conjecture, april 2016. *Research proposal to the Templeton Foundation for*, 2019, 2016.