Roberta Bonacina

Semantics for Homotopy Type Theory

PhD Dissertation

Advisor: Dr Marco Benini

Supervisor: Dr Marco Benini

Dottorato di Ricerca in Informatica e Matematica del Calcolo
Dipartimento di Scienze e Alta Tecnologia
Università degli Studi dell'Insubria

2019/2020

Contents

1 Introduction **2**
1.1 Technical background 2
1.2 Philosophical background 3
1.3 Plan 5

2 Background **7**
2.1 Martin-Löf type theory 10
2.2 Homotopy type theory 14

3 Syntax **18**
3.1 Basic system 18
3.2 General syntax for inductive types 21
3.3 Basic proof theory 25
3.4 Parallel substitution 36
3.5 Canonical inductive types 39

4 Normalisation **45**
4.1 Reductions 46
4.2 Strongly normalisable judgements 50
4.3 The normalisation theorem 52

5 Semantics **64**
5.1 Categorical preliminaries 64
5.2 Interpretation 74
5.3 Classifying model and completeness 82

6 Homotopic features **101**
6.1 Function extensionality and univalence 101
6.2 Syntax of higher inductive types 102
6.3 Canonical higher inductive types 105
6.4 Proof theory 110
6.5 Semantics 112

7 Conclusions **121**

Bibliography **123**

Acknowledgements

I am extremely grateful to

Marco Benini for guiding me on this path. He is not only my supervisor, but also and above all a friend.
My parents, and family. Without them nothing would have been possible.
Delia, my travelling companion. Challenges are easier and fun when faced together.
Peter Schuster, Daniel Wessel, and Giulio Fellin for their support.
Lucia, Federico, Federica, Eleonora, Fabrizio, Francesco, Stefano, Alessia, Sara, Chiara, Giulia, and my office mates for the time spent together during those years.
Finally, my four-legged family. Their presence in my life is fundamental for me.

Thanks to all of you
Roberta

Chapter 1

Introduction

The aim of this work is to answer one single question:

Is the ∞-groupoid theory needed to describe homotopy type theory?

It is evident by the strong interest and the amount of results, see, e.g, [14, 50, 11, 60, 59, 13, 12, 6, 8, 9, 16, 98, 96, 97, 7], that the homotopical interpretation is deep and useful. This is not to discuss: it is established beyond any doubt. However, it may sound odd that the full power of higher-order category theory is needed to describe what is, at the end of the day, a piece of recursive mathematics strictly contained in the class of Turing-computable functions. As Prof. Cardone was so kind to observe: "This is not especially odd if one considers the categorical interpretation of polymorphic (2nd order) lambda-calculus, which needs universes and a very strong version of product over a universe. Similarly for the calculus of constructions. And yet the functions representable in these systems are total and computable. The reason of this is that a categorical model of a typed calculus must reflect the syntactical structure of the system, which is the source of intensional phenomena of representability of *algorithms*, as opposed to the extensional expressiveness of systems measured in terms of classes of representable *functions*. This is a very interesting distinction which is often overlooked in the literature, and has to do with the distinction between function and algorithm. To my knowledge, it was brought to the surface by Colson's ICALP 1989 result [34] about the expressiveness of the system of primitive recursive definitions, which cannot represent a straightforward parallel algorithm of order $O(min(n, m))$ for computing the minimum of two natural numbers n, m (actually, no such algorithm)."

In short, the present dissertation answers this question in a negative way: homotopy type theory can be described in ordinary categorical terms, with elementary tools. In fact, in Chapter 5 a sound and complete semantics is given for Martin-Löf type theory, which is extended to a significant part of homotopy type theory in Chapter 6 covering the fundamental axioms of univalence and function extensionality and a relevant subset of higher inductive types.

Although the ultimate goal is clear, to achieve it many problems have been encountered and eventually solved. The major obstacle was a lack of rigour in the proof theory, from which the semantics moves on.

1.1 Technical background

Martin-Löf type theory, also called *dependent type theory* or *intuitionistic type theory*, is a typed lambda calculus mainly due to Martin-Löf. Dependent type theory has been presented in many different versions, between 1972 and 1984 [68, 69, 70, 71]. Aside, a large number of studies have been conducted inside this formal system and its applications. For example, it has been analysed as a programming language [77], and used (with some changes) as a base for some proof assistants, such as Agda [5] and Coq [36].

Recently it has been studied and deepened in [95], where equality types play a central role, together with their groupoid interpretation introduced for the first time in [49]. In this dissertation we go through the first part of the Book[1], called “Foundations”, focusing in particular on Chapters 1, 2, 5 and 6, and we use part of the Appendix A for the inference rules. Indeed, we are mainly interested in the system itself, rather than its many applications in mathematics studied in the second part.

This dissertation is concerned with the specific variant of Martin-Löf type theory used in homotopy type theory. Its proof theoretical aspects are analysed according to *structural proof theory* [76, 93]. In particular, the notion of normalisation is studied, following the Tait and Girard’s approach, mostly following the exposition in [43].

The semantic aspects require a categorical approach. In fact, category theory [45, 63, 64] has been used several times as a language to define semantics for Martin-Löf type theory. In particular, Seely [90] and other authors [32, 52, 39] interpret it in locally Cartesian closed categories, eventually with some variations, but focus only on a fragment of the system, with a syntax adapted to the constraints forced by the chosen categorical framework. Category theory and homotopy theory, with the common construction of ∞-groupoids, are used to interpret homotopy type theory, following respectively Streicher’s [49] and Voevodsky’s [95] ideas, as we will sketch in Section 2.2. This structure well-behaves for the whole homotopy type theory, but needs a very complex mathematical background, which is not fully formalised: as remarked in [95, page 68], some “complex combinatory” is needed to make precise all the notions; since those notions are unnecessary in the rest of the Book, the authors did not analyse them.

The above remark reveals that one of the difficulties this dissertation had to face was the lack of rigour in the literature: some fundamental results are *folklore*, like normalisation, but their proofs are missing, and, at a closer look, the gap is by far not obvious to fill: also, fundamental definitions are sketchy, like what is an higher inductive type. Therefore, there are syntactical and semantic aspects of homotopy type theory which are *understood* but not defined in the rigorous mathematical sense.

1.2 Philosophical background

Martin-Löf type theory was defined to formalise mathematical reasoning in the framework of intuitionistic logic and, more generally, of mathematical constructivism [94]. The first idea of a logic alternative to classical logic is due to Brouwer [30], who thought of proofs as *constructions*, in the sense that

- an object *exists* when it can be found explicitly;
- a statement is *true* when there is a proof for it.

[1]Since it is central in this work, we refer to [95] also as *the Book*.

Different approaches to this concept led to some schools [17]; the main ones are Brouwer's intuitionism, the Russian constructivism [66] mainly focused on recursion theory, and Bishop's "practical" constructive mathematics. Bishop showed [21] that a huge amount of results of classical mathematics can be developed constructively; following his ideas, a constructive approach has been applied, among the others, to analysis [22], algebra [61] and set theory [4, 3].

What relates intuitionistic logic and the type theories is the *Curry-Howard isomorphism*. As described in [31], it was introduced at first only for propositions and types [40], and then extended to terms and proofs [51]. This paved the way to Martin-Löf to define his type theory [68], based on the idea of proof-relevant mathematics. In Chapter 2 we study the evolution of Martin-Löf type theory during the years.

A feature of Martin-Löf type theory is that it is a *predicative* theory; indeed, as remarked in [73], a consequence of Girard's paradox [42] is that the Curry-Howard isomorphism is incompatible with impredicativity, at least in the strong sense of Martin-Löf. As written in [38, page 3], the concept of predicativity arose from some discussions between Poincaré and Russell: "a definition is impredicative if it defines an entity by reference to (e.g. generalisation over) a totality to which the entity itself belongs, and is predicative otherwise". Poincaré's study of predicativity [82, 83] arise from the fact that quantification over infinite domains may be tricky; indeed, when an entity X is defined depending on all the elements of a collection to which X itself belongs, this can cause instability: is X really defined? Could the collection exist before X? Martin-Löf type theory avoids this instability thanks to a subtle treatment of universes, and to the structure of Π and Σ types, which are the type theoretical correspondents of universal and existential quantification.

Point-free mathematics

During the XX century, as remarked at the end of [28, 56], a growing attention was put on *abstraction*, and on the study of general *structures*. One of the clearest examples of this trend is the work by Bourbaki [27]; it focuses on the logical structure of theories, with the aim to make the ideas, and not the calculations, the main object of interest. Grothendieck followed this path, generalising the idea of space through the concept of *topos* [47, 67].

Indeed the classical idea of topology [80, 81] was abstracted, using category theory and the notion of *locale* [54, 53], and generalised. In Sambin's *formal topology* [87, 86, 88, 85], which is strictly related to Martin-Löf type theory, the primitive notions are open subsets and coverings, while points are defined in terms of opens. This attitude, common to all the above mentioned non-classical approaches to spaces and topology [47, 54, 53], is called *point-free*. The idea is that the fundamental notion to define a space is its structure, not the single objects, i.e., the points in the space. Thus, for example, analysis can be done without real numbers: the algebraic structure of the propositions about real numbers is enough to assign a sound and complete meaning to mathematical analysis, as shown in [18], even in the extreme philosophical position to deny that real numbers exist. An example of this approach in algebra can be found in [89], which makes clear that the notion of point is not essential to define the notion of space, providing a natural counter-example.

This idea can be applied to logic; in particular, to semantics. Classical semantics for logic are based on algebraic structures [25, 46, 54, 65]. Focusing on first-order intuitionistic logic, a point-free semantics is defined in [18], where the universal and existential quantifiers are interpreted through a substitution functor. We tried to apply those ideas to Martin-Löf type theory in [24], obtaining a partial result. In this dissertation we present the full one, and extend it to a part of homotopy type theory.

1.3 Plan

The dissertation contains an extensive study of Martin-Löf type theory, which comprehends the definition of a general syntax, a proposal for a novel semantics and some proof theoretic results, including a normalisation theorem. Those results are extended to a large subset of homotopy type theory, i.e., the 1-HoTT theories. In details

Chapter 2 introduces the systems studied in the dissertation: Martin-Löf type theory and homotopy type theory. It begins with an informal presentation of the type theory, which describes the main notions and terminology. Then it is studied how the system by Martin-Löf evolved in the different papers, and the main novelties introduced by [95] — namely, the higher inductive types and the axioms of function extensionality and univalence — are described. The chapter ends with a focus on Voevodsky's interpretation of homotopy type theory, based on paths and homotopy theory. The purpose of this chapter is to position the thesis in the development of Martin-Löf type theory.

Chapter 3 is devoted to establish the syntax, and obtain some basic proof theoretic results. Our syntax for Martin-Löf type theory is divided into the basic system, which consists in the structural rules and dependent product, and the inductive types; for them we define a general syntax, which has all the canonical types (i.e., dependent sum, coproduct, natural numbers, propositional equality, etc) as instances.

A particular attention is given to some notions and results, which are usually considered folklore. For example, substitution is extensively studied, introducing the notion of parallel substitution. We focus also on contexts, proving a weakening result: whenever a judgement can be derived from a set of assumptions, it can also be derived from a larger set of assumptions.

Chapter 4 contains a proof of a strong normalisation theorem, in which Girard's ideas [43] are extended to our system. To deal with dependent types, the technique based on reducibility candidates is enriched with a notion of evaluation.

We begin defining the notions of reduction and conversion, and deriving their relation with judgemental equality. Then the notions of strongly normalisable judgements and reducibility candidates are defined and studied; their interplay allows to prove a strong normalisation theorem.

Chapter 5 is devoted to propose a sound semantics, together with a classifying model and a completeness result, which answers in a negative way the initial question motivating this dissertation.

The semantics interprets terms-in-context as objects in a slice category, and judgements as arrows, seeing the inductive types as transformations. The chapter ends with a discussion of the overall idea which motivated the semantics, explaining the reasons behind all the apparently ad-hoc definitions.

Chapter 6 extends the results obtained so far to some of the novelties of [95]. It begins describing the syntax of function extensionality and univalence. Then, a general syntax for a subset of the higher inductive types introduced in [95], called 1-higher inductive types, is defined.

The proof theoretic results of Section 3.3 and Chapter 4 are extended in a natural way to those constructions, and to truncation. The same is done for the semantics, interpreting the higher inductive types and the axioms as transformations. Thus, a wide fragment of homotopy type theory, 1-HoTT theories, satisfies a strong normalisation theorem and can be interpreted in our semantics in a sound and complete way.

The chapter terminates with a conjecture: if the syntax of some higher inductive type is given in fully formal terms, our proof theoretic results and semantics can easily be extended to that type.

Chapter 7 concludes the dissertation, summarising the achieved results.

Chapter 2

Background

In this chapter the the concept of type in Martin-Löf's theory is introduced, along with the canonical types. Then, we illustrate how this structure changed over the years.

The writing $a : A$ (or, sometimes, $a \in A$) is read as "a is a term of type A". This phrase can be understood in different ways: thinking of set theory it may be interpreted as "a is an element of the set A"; from a logical point of view, it can be read as "a is a proof of the proposition A", shifting the focus from the statement of a theorem to its proof; also, targeting computer programming, it can be thought of as "a is a program satisfying specification A".

Types	Logic	Sets	Programming
A	proposition	set	specification
$a : A$	proof	element	program

It is worth remarking that, although types can be seen as sets or propositions, they are not; in this sense, dependent type theory can be seen as a foundational theory, from which set theory, logic, and computing can be constructed by imposing a suitable point of view. As explained in [70], dependent type theory was defined with the aim to represent intuitionistic mathematics, but it turned out to be also a functional programming language. In fact, the computational aspect of dependent type theory is the feature enabling the logical and the set-theoretical interpretations. Interpreting types as propositions and inhabitants of a type as proofs forces a *proof-relevant* reading of propositions, thus capturing the core of intuitionistic logic. In turn, proof relevance imposes a constructive interpretation of the set theoretic point of view on types: the elements in a set on which we can really operate are the ones we can construct, i.e., the ones for which there is an algorithm[1] a satisfying specification A, thus $a \in A$. Therefore, it is of primary importance to fix which constructions are allowed so that the alternative points of view (logical, set-theoretical, etc.) keep their coherence in a constructive sense. In other words, a choice has to be made on which types are fundamental: these types are hereafter called *canonical*.

The canonical types, used as primitive notions to represent mathematical reasoning (or programs, from another point of view), are as follows.

If A and B are types, $A \to B$ is the type of constructible functions from A to B; a canonical[2] term of type $A \to B$ is $\lambda x : A.\, b$, where b is a term of type B which may have $x : A$ as a free variable. A generic element $f : A \to B$ can be

[1]Although beyond what is discussed here, this can be traced down to require a strictly constructive and predicative axiom of comprehension.

[2]We use the word "canonical" with two different meanings: the canonical types are the fundamental ones, while the canonical terms of a type are the ones obtained through an introduction rule for that type.

applied to a term $a{:}A$ to obtain $f\,a{:}B$, the evaluation of f in a, or the application of a to f. Function types are usually equipped with a type-theoretic version of β and η reduction, see [15], governing application when f is a canonical term. If $x : A$ is a free variable in B, function types can be generalised to the product types $\Pi x : A.\,B$; the canonical elements of a Π-type are λ-expressions as above, in which x may be free in B, but applying $f : \Pi x : A.\,B$ to $a : A$ yields $f\,a : B[a/x]$. Through the propositions-as-types interpretation, Π-types can be seen as the type-theoretic version of the universal quantifier; indeed, a proof of a proposition of the form $\forall x \in A.\,B$ consists in a function assigning to each $a \in A$ a proof $b[a/x]$ of $B[a/x]$ and, by β-reduction, $b[a/x]$ is equivalent to $(\lambda x : A.\,b)\,a$. From a set-theoretic perspective, a dependent product models the sections of the fibred space B over A, naturally connecting Π-types to stalk spaces (espace étalé) [45, Section 4.5].

Formally, the described reductions, as well as the construction of canonical terms, are presented as inference rules[3]. For function types, application provides a way to "use" a term of that type[4]. Rules like the construction of an applied term, called *elimination* rules, are coupled with *computation* rules analogous to β-reduction, which show how a canonical term of a type can be "used", and sometimes with a rule analogous to η reduction, the *uniqueness* rule, which shows that canonical terms provide enough elements to fully determine the extent of a type. The rules showing how a new type or a canonical term of a type can be defined are, respectively, the *formation* and *introduction* rules. The word "canonical" has thus a double meaning: it denotes the primitive types, and it denotes the standard form a term of a type may assume.

The notion of Cartesian product is captured by $A \times B$, with A and B independent types. The canonical terms are the pairs (a, b), where $a : A$ and $b : B$. When $x : A$ is free in B, Cartesian product can be generalised to the dependent sum $\Sigma x : A.\,B$; in this case the canonical element is (a, b), with $a : A$ and $b : B[a/x]$. The Σ-types are the type-theoretic version of the existential quantifier from a logical perspective.

The Π and Σ-types are the main ones which justify the name *dependent* type theory: indeed, they are called *dependent types*, because their definition depends on a term. For example, "the pairs of natural numbers where the second element is greater than the first" is a dependent sum. From a logical point of view they model the usual quantifiers; set-theoretically they model the relevant aspects of fibrations over sets; in a programming view they model a sub-collection of computable functions, and variant records [79]. As a rule of thumb, most (or, all the "natural" ones) functional programs that always terminate can be coded by means of these and the other canonical types. It is worth mentioning at this point that terminology varies: we adopt the one in Martin-Löf's papers, while the tradition in typed λ-calculi is different, and homotopy type theory uses other words to denote the same types.

Disjoint union, or the logical connective "or", can been obtained from the coproduct types $A + B$, with A and B types. Their canonical terms are $\mathsf{inl}(a)$,

[3]Formal systems presented by rules are usually preferred over axiomatic systems for at least two reasons: they are easier to interpret as programming languages, and they have better proof theoretic behaviours [76].

[4]There is an analogous rule also for the other canonical types, but it is more complicated: it shows how a term of a canonical type can be substituted in a generic type, and we will see it in details in Section 3.2.

with $a : A$, and $\mathsf{inr}(b)$, with $b : B$. It is worth remarking that each canonical term of a coproduct type comes with a label stating the type it comes from. It is what distinguishes disjoint union $\sqcup$ from union $\cup$ but, also, intuitionistic logic from classical logic: whenever we have a proof of $A + B$ (read: $A \vee B$), we know which is the provable disjunct. The same happens for existential statements: every term (a, b) of a Σ-types comes with a witness (a) of the existential proposition.

The trivial type $\mathbf{1}$ whose only canonical element is $* : \mathbf{1}$ and the empty type $\mathbf{0}$ are a limit case, respectively, of product and coproduct types, and can be logically interpreted as true and false. Other finite types are $\mathbb{N}_n$, whose canonical elements are $1, 2, \dots, n$; $\mathbb{N}_0$ is $\mathbf{0}$, $\mathbb{N}_1$ is $\mathbf{1}$, and $\mathbb{N}_n$ can be defined as the iterated coproduct $\mathbf{1} + \cdots + \mathbf{1}$, with $\mathbf{1}$ appearing n times.

An infinite type is needed to reproduce mathematical reasoning; thus it has been introduced the type $\mathbb{N}$ of natural numbers, whose canonical terms are $0 : \mathbb{N}$ and $\mathsf{succ}(n)$, with $n : \mathbb{N}$. They are the first example of an *inductive* type, i.e., a type whose terms are defined in dependence of other terms of that type. Another example of inductive type are the lists: if A is a type, $\mathsf{List}(A)$ is a type whose canonical terms are the empty list $\mathsf{nil} : \mathsf{List}(A)$ and $a.l : \mathsf{List}(A)$, with $a : A$ and $l : \mathsf{List}(A)$. Another important example are the W-types, which are a very general type able to represent well-founded trees: if A is a type and B is a type with $x : A$ free variable, then $Wx : A. B$ is a type whose canonical terms are $\mathsf{sup}(a, v)$, with $a : A$ and $v : B[a/x] \to Wx : A. B$.

Finally, and fundamental to characterise the set-theoretic point of view in a strong constructive way, fixed a type A, there is a family of types $=_A$ stating when two terms of type A are equal, called equality types or propositional equality. Thus, $a =_A b$ is a type whenever $a, b : A$, and the only canonical terms are $\mathsf{refl}(a) : a =_A a$. The equality types play a special role in [95], as will be explained in Section 2.2. Imposing propositional equality has a number of consequences: in general, a set does not provide an *algorithm* to decide whether two of its elements are equal (think to non-recursively enumerable sets [35], for example). However, equality is fundamental to express the properties of mathematical statements, and thus it deserves a special attention, particularly in constructive mathematics, see, e.g., [92], or the comprehensive discussion in [17][5]. Also, (in)equality plays a fundamental role in constructive analysis, see [22], whose tradition and influence cannot be underestimated in the development of constructive type theory.

Propositional equality is not the only concept of equality present in dependent type theory; there is judgemental equality (also called definitional equality), which is not a type, but exists at the same level of *judgements*, the formal assertions in type theory [71, 79]: as we write $a : A$, we can write $a \equiv b : A$ when $a : A$ and $b : A$ are judgementally equal. Propositional equality is a type, thus it can be treated as a proposition: it can be proved, assumed as an hypothesis and so on. Conversely, judgemental equality is an equality "by definition"; it is the notion of equality which allows *computation*, playing a significant role in the concept of *reduction*, see Chapter 4. Informally, propositional equality is the internal notion of being equal provided by each type, while judgemental equality is the computational engine allowing to show when two judgements

[5]For a somehow dated, but still very useful introduction of the history of constructive mathematics see [94, Chapter 1]; a concise and more recent account of the major trends in constructive mathematics in the late XX century can be found in [29].

can be decided to compute the same value.

In order to soundly talk about terms and types, two other concepts are needed: contexts and universes. Contexts are the set of hypotheses which allow to deduce that an expression is a term of a certain type. Sometimes mathematical arguments need hypothetical reasoning, thus it may be useful to suppose that a certain type is inhabited; this is the aim of contexts: a context $\Gamma = x_1 : A_1, \ldots, x_n : A_n$ says that the types A_j are inhabited by a certain element whose name is x_j. Computationally, this corresponds to variable declaration [79], while this corresponds to assign names to assumptions in a logical proof for the propositions-as-types perspective. Finally, universes are a way to say that a certain term is a type: if $\mathcal{U}$ is a universe and $a : \mathcal{U}$, i.e., a is a term of a universe, then it is a type. The structure of universes deeply evolved during the years in order to avoid paradoxes, as explained in Section 2.1, since a universe stands for the type of types, which is not a well-founded concept. Hence, the evolution of the idea has needed to coherently model the notion of a type of *small* types, analogously to small categories [63].

2.1 Martin-Löf type theory

This Section is devoted to study how Martin-Löf developed his type theory through the years. We will analyse only the papers by Martin-Löf mainly focused on his system's syntax [68, 69, 70, 71], discussing the novelties introduced in the different papers. When relevant, Σ-types will be used as a prototypical example to study how the syntax evolved.

An intuitionistic theory of types

In [68], which later has been revised and published as [72], are defined the formation, introduction, elimination and computation rules for types Π, Σ, $+$, $\mathbb{N}_n$ and $\mathbb{N}$; there are no uniqueness rules, even for the Π-types (i.e., the system lacks η-reduction).

First, it is introduced a universe V, the *type of small types*, with a number of introduction rules stating that $\mathbb{N}_n$ and $\mathbb{N}$ are terms of type V and, if the types in the premises of the formation rules for Π, Σ and $+$ are terms of type V, then also the type in the conclusion of the rules is. It is worth remarking that V is not a term of type V; this would lead to a paradox; indeed, [42] found an unintended effect of the assumption $\mathsf{Type} : \mathsf{Type}$ in the original formulation, which led to the modified version [68].

There are two forms of judgements: *A type* and $a \in A$, respectively saying that A is a type and a is a term of type A. The prototypical example of inference rules for a type is

$$\frac{A\ \mathit{type} \quad B[x]\ \mathit{type}}{(\Sigma x \in A)B[x]\ \mathit{type}}\ \Sigma{-}\mathsf{form} \qquad \frac{a \in A \quad b \in B[a]}{(a,b) \in (\Sigma x \in A)B[x]}\ \Sigma{-}\mathsf{intro}$$

$$\frac{c \in (\Sigma x \in A)B[x] \quad \begin{matrix}(x \in A \quad y \in B) \\ d[x,y] \in C[(x,y)]\end{matrix}}{E(c, (\lambda x)(\lambda y)d[x,y]) \in C[c]}\ \Sigma{-}\mathsf{elim}$$

also, seen as a V-introduction,

$$\frac{A \in V \quad \begin{matrix}(x \in A)\\ B[x] \in V\end{matrix}}{(\Sigma x \in A)B[x] \in V}$$

The work introduces the concepts of *contraction*, *reduction* and *conversion.* Contraction consists, substantially, in the computation rules: in the case of Σ,

$$E((a,b),(\lambda x)(\lambda y)d[x,y]) \textit{ contr } d[a,b] \quad .$$

It is said that

- an expression a reduces to an expression b, or a *red* b, if b can be obtained by repeated contractions of parts of a[6];
- a is normal if it can not be reduced;
- a converts in b, or a *conv* b, if there is an expression c such that a *red* c and b *red* c.

Definitional equality is introduced starting from conversion: two types A and B are definitionally equal if A *conv* B, and two terms $a : A$ and $b : B$ are definitionally equal if a *conv* b and A *conv* B. It is proved a Normalisation Theorem for this system, stating that every term reduces to a normal term.

Finally, a map to reduce intuitionistic formal theories to Martin-Löf type theory is described.

In this first version, propositional equality is absent and the universe has a minor role. However, the philosophical basis for a constructive and predicative theory of types is fully laid down.

An intuitionistic theory of types: Predicative part

In [69] propositional equality, denoted as $I(a,b)$, is added to the list of canonical types. The notion of free variables is stressed specifying them in the terms: $a[x_1,\dots,x_k]$ means that $x_1,\dots,x_k$ are the free variables in a. Another important difference is the introduction of an infinite hierarchy of universes $V = V_0 \in V_1 \in V_2 \dots$, where V_0 is the type of small types and the terms of the other universes are introduced saying that $V_n \in V_{n+1}$ and through the formation (here, called reflection) rules for the canonical types. As illustrated in [71], types in higher universes are needed to prove some results in concrete applications: as confirmed by G. Sambin, this was the reason to extend the system along this line.

Universes allow to say that a is a term if the judgement $a \in A$ can be derived for some A, and that A is a type if there is n such that $A \in V_n$; thus each type is also a term, avoiding the need of the two different kinds of judgement $a \in A$ and A *type*. The other possible form of judgement is a *conv* b, where the notion of conversion is the one introduced by the computation rules, but

[6]However, λ-expressions are considered as atoms, i.e., contractions are not allowed in their body.

with the additional requirements that it is an equivalence relation (i.e. reflexive, symmetric and transitive), that

$$\frac{a_1 \; conv \; c_1 \quad \ldots \quad a_k \; conv \; c_k}{b[a_1 \ldots a_k] \; conv \; b[c_1 \ldots c_k]}$$

whenever all the above terms are correctly typed, and that

$$\frac{a \in A \quad A \; conv \; B}{a \in B}$$

that is, *conv* is a *congruence* [15] conservative over types. Reduction is governed by the same rules of conversion, but without symmetry. Finally, it is said that c is a *closed normal term* with type symbol C, which is called a closed normal type symbol, if $c \in C$ can be derived applying repeatedly the following rule:

$$\frac{\{x_j \in A_j[x_1, \ldots, x_{j-1}]\}_{j=1}^{k} \quad f(x_1, \ldots, x_k) \in F(x_1, \ldots, x_k)}{f(c_1, \ldots, c_k) \in F(c_1, \ldots, c_k)}$$

with the additional requirements that F is a constant denoting one of the basic types, $c_j \in C_j$ are closed normal terms and $A_j[c_1, \ldots, c_{j-1}]$ *red* C_j for $1 \leqslant j \leqslant k$. This definition implies that a closed normal term must be a universe or a term with no free variables appearing in the conclusion of a formation or introduction rule. The work ends with the proof of a Normalisation Theorem, stating that each closed term reduces to a normal form. It is worth remarking that such a theorem uses the normalisation-by-evaluation technique and that, as before, λ-terms are atoms.

Constructive mathematics and computer programming

The connection of Martin-Löf type theory to programming languages is emphasised in [70], where a particular attention is devoted to the concept of *execution* of a term or type. Accordingly to this point of view, an expression is called *canonical* or *normal* if it is already fully evaluated, i.e., it has no free variables and it is irreducible.

Four different kinds of judgement are used:

- $A \; type \, (x_1 : A_1, \ldots, x_k : A_k)$,
- $A = B \, (x_1 : A_1, \ldots, x_k : A_k)$,
- $a \in A \, (x_1 : A_1, \ldots, x_k : A_k)$,
- $a = b \in A \, (x_1 : A_1, \ldots, x_k : A_k)$.

Accordingly to the computational point of view of [70], the expression A *type* means that A has a canonical type as value; similarly, $a \in A$ means that a has as value a canonical term, whose type is the value of the type A. Thus, for example, $A \; type \, (x_1 : A_1, \ldots, x_n : A_n)$ means $A(a_1, \ldots, a_n / x_1, \ldots, x_n) \; type$, with $a_1, \ldots, a_n$ values of type $A_1, \ldots, A_n$, respectively. This latter expression is governed by the substitution rules, which are introduced here for the first time, together with the W-types and η-reduction for the Π-types.

A second elimination rule for $I(a, b)$ is added, stating that the only inhabited equality types are the ones whose defining terms are equal according to $=$:

$$\frac{c \in I(A, a, b)}{a = b \in A}$$

this type theory is called *extensional type theory*, in opposition to the *intensional type theory* in which the latter rule does not appear.

Universes, which are denoted as U_n, have some particular formation, introduction and elimination rules: the formation rules are U_n *type* and $U_n = U_n$; for each type formation rule for a canonical type there is a corresponding introduction rule for universes, which introduces the type as a term of the proper universes; finally, the elimination rules are

$$\frac{A \in U_n}{A \ type} \qquad \frac{A \in U_n}{A \in U_{n+1}}$$

both with the corresponding equality rules.

Intuitionistic type theory

In [71] a lot of space is devoted to explain the meaning and the philosophy of dependent type theory, both in general and focusing on the idea behind each type, with a particular attention to the propositions-as-types correspondence.

The syntax is very similar to the one of [70]; the main differences with the previous article are the introduction of the types List and the treatment of equality and universes. Here there are three different types of equality: $=$, which is the one used above to denote the syntactical equality between objects, definitional equality $\equiv$ and propositional equality $I(A, a, b)$. Definitional equality is used when two expressions are equal by definition; for example, if $x : A$ is not free in B, $A \to B \equiv (\Pi x : A)B(x)$. Propositional equality is extensional as in [70], but the only elimination rule is the one stating that if $I(A, a, b)$ is inhabited then $a = b \in A$; the corresponding computation rule states that, if $I(A, a, b)$ is inhabited, all its terms are equal. This notion of equality allows to prove a generalisation of η-reduction for the other types; for example, in the case of Σ, it can be shown that $(p(c), q(c)) = c \in (\Sigma x \in A)B(x)$, with $p(c) \equiv E(c, (x, y)x)$ and $q(c) \equiv E(c, (x, y)y)$ the two projections.

Finally, two possible ways to define universes are explained: a la Tarski and a la Russell. In the former construction, the formation rules are

$$\frac{}{U \ set} \qquad \frac{a \in U}{T(a) \ set}$$

where T is a map associating to each element of the first universe U a set (i.e., a type); then, for each canonical type, there is a couple of introduction rules for universes like

$$\frac{a \in U \quad \begin{matrix}(x \in T(a)) \\ b(x) \in U\end{matrix}}{\sigma(a, (x)b(x)) \in U}$$

$$\frac{a \in U \quad \begin{matrix}(x \in T(a)) \\ b(x) \in U\end{matrix}}{T(\sigma(a, (x)b(x) \in U) = (\Sigma x \in T(a))T(b(x))}$$

where σ is a new symbol mirroring Σ, used to construct the canonical elements of U; finally, the infinite hierarchy of universes is constructed iterating the rules

$$\frac{}{u \in U'} \qquad \frac{}{T'(u) = U}$$

$$\frac{a \in U}{T(a) \in U'} \qquad \frac{a \in U}{T'(t(a)) = T(a)}$$

where T' associates a set to each element of the second universe U', and the map $t : U \to U'$ lifts the terms of the first universe to the second one, with the coherence condition $T'(t(a)) = T(a)$. Conversely, the construction a la Russell builds only one universe which is not a term of itself, through the formation rules

$$\frac{}{U\ set} \qquad \frac{A \in U}{A\ set}$$

and, for each canonical type, the introduction rule

$$\frac{a \in U \quad \begin{array}{c}(x \in A)\\ b(x) \in U\end{array}}{(\Sigma x \in A)B(x) \in U}$$

In the preface to this paper, Martin-Löf describes a few improvements of its system which have been presented during a series of lectures in Munich in 1980. The main ones are a different notation and the introduction of alternative elimination and computation rules for Π, which follow the same pattern of the ones for the other types. Those rules are equivalent to the canonical ones: application can be derived posing $f\,a = E(f, (y)y(a))$ with $y(x) \in B(x)$, η-reduction can be derived from the rules for I as seen for the other types and, conversely, the alternative elimination rule can be derived from application defining $E(c, d) = d((x), c.x)$.

2.2 Homotopy type theory

The community behind [95] deeply extended Martin-Löf type theory with the aim to use it as a more flexible foundation of mathematics, able for example to represent as primitive notions some geometric objects, or to interpret also classical logic.

The system described in the Book is based on the interpretation of the terms $p : a =_A b$ of an equality type as *paths* between the *points* a and b. The main novelties introduced are the higher inductive types and the axioms of function extensionality and univalence, which will be informally explained in this Section.

The syntax used in the Book is similar to the one adopted in this thesis; in the case of Σ-types, it is

$$\frac{\Gamma \vdash A : \mathcal{U}_i \quad \Gamma, x : A \vdash B : \mathcal{U}_i}{\Gamma \vdash \Sigma x : A.\, B : \mathcal{U}_i}\ \Sigma\text{−form}$$

$$\frac{\Gamma \vdash A \equiv C : \mathcal{U}_i \quad \Gamma, x : A \vdash B \equiv D : \mathcal{U}_i}{\Gamma \vdash (\Sigma x : A.\, B) \equiv (\Sigma x : C.\, D) : \mathcal{U}_i}\ \Sigma\text{−form−eq}$$

$$\frac{\Gamma, x : A \vdash B : \mathcal{U}_i \quad \Gamma \vdash a : A \quad \Gamma \vdash b : B[a/x]}{\Gamma \vdash (a,b) : (\Sigma x : A.\, B)}\ \Sigma\text{-intro}$$

$$\frac{\Gamma, x : A \vdash B : \mathcal{U}_i \quad \Gamma \vdash a \equiv c : A \quad \Gamma \vdash b \equiv d : B[a/x]}{\Gamma \vdash (a,b) \equiv (c,d) : (\Sigma x : A.\, B)}\ \Sigma\text{-intro-eq}$$

$$\frac{\begin{array}{c}\Gamma, z : \Sigma x : A.\, B \vdash C : \mathcal{U}_i \\ \Gamma, x : A, y : B \vdash g : C\,[(x,y)/z] \quad \Gamma \vdash e : \Sigma x : A.\, B\end{array}}{\Gamma \vdash \mathsf{ind}_{\Sigma x:A.\,B}\,(z.C, x.y.g, e) : C[e/x]}\ \Sigma\text{-elim}$$

$$\frac{\begin{array}{c}\Gamma, z : \Sigma x : A.\, B \vdash C \equiv D : \mathcal{U}_i \\ \Gamma, x : A, y : B \vdash g \equiv h : C\,[(x,y)/z] \quad \Gamma \vdash e \equiv f : \Sigma x : A.\, B\end{array}}{\Gamma \vdash \mathsf{ind}_{\Sigma x:A.\,B}\,(z.C, x.y.g, e) \equiv \mathsf{ind}_{\Sigma x:A.\,B}\,(z.D, x.y.h, f) : C[e/x]}\ \Sigma\text{-elim-eq}$$

$$\frac{\begin{array}{c}\Gamma, z : \Sigma x : A.\, B \vdash C : \mathcal{U}_i \qquad \Gamma \vdash b : B[a/x] \\ \Gamma, a : A, y : B \vdash g : C\,[(x,y)/z] \quad \Gamma \vdash a : A\end{array}}{\Gamma \vdash \mathsf{ind}_{\Sigma x:A.\,B}\,(z.C, x.y.g, (a,b)) \equiv g[a/x, b/y] : C\,[(a,b)/z]}\ \Sigma\text{-comp}$$

As suggested by the above rules, the system is characterised by an infinite hierarchy of universes a la Russell, such that $\mathcal{U}_i : \mathcal{U}_{i+1}$ and $A : \mathcal{U}_i$ implies $A : \mathcal{U}_{i+1}$, and by the notion of *context*: the free variables of a term are not specified as arguments of the term, as done by Martin-Löf, but they are listed in the context Γ. The rules governing the formation of new contexts and the introduction of variables from existing contexts will be described in Section 3.1. It is worth remarking that in [95, Chapter 5] the canonical terms of a type are seen as the application of a term of a function type to some terms of the proper types; this idea will be systematically developed in the thesis, see Section 3.2.

The canonical inductive types considered in [95] are Π, Σ, $+$, 1, 0, $\mathbb{N}$, $=$ (which is intensional) and W, but the construction of new types is allowed as long as they are "reasonable", as the Authors explained in Section 5.6.

The novel notion of higher inductive types is introduced: they are types whose introduction rules define not only the canonical terms $a : A$ of the type, but also the canonical terms $p : a =_A b$ of the equality type of that type, or of the equality between equalities, and so on; through the homotopy interpretation, we can say that what can be defined are not only the points of a type, but also the paths, the paths between paths, etc. Those new types allow to define some geometric spaces such as the n-dimensional sphere and the torus, and also some concepts as "being a type with just one term", which can be seen as "all the proof of that proposition are equivalent", allowing to interpret classical logic. For example, the circle $\mathbb{S}^1$ is defined as the type whose canonical terms are a point $\mathsf{base} : \mathbb{S}^1$ and a path $\mathsf{loop} : \mathsf{base} =_{\mathbb{S}^1} \mathsf{base}$ which is not identified with refl. It is pretty clear why this type theory refers to homotopy theory in its name: in fact, the type $\mathbb{S}^1$ is nothing but the classical Poincaré's homotopy group of that topological space. The intuition, due by V. Voevodsky, is that, beside logic, set theory, and computation, another interpretation of type theory is possible: types as homotopy spaces, and terms as their points; then, equality types $a =_A b$ model the space of paths between a and b.

It can be seen that, in this theory, the equality types become more important than before; this is strengthened by the axioms of function extensionality and univalence. To explain the idea behind them, let us informally introduce some notions: a function $f : A \to B$ is an *isomorphism* if it is invertible[7], and two types A and B are *equivalent*, write $A \simeq B$, if there is an isomorphism $f : A \to B$.

Notice that, for $\times$ types, it can be shown that $x =_{A\times B} y$ is equivalent to $(\mathsf{pr}_1(x) =_A \mathsf{pr}_1(y)) \times (\mathsf{pr}_2(x) =_A \mathsf{pr}_2(y))$, and a similar result holds for Σ-types; thus, the equality of two terms of a $\times$ or Σ type depends on the behaviour of their projections. We would expect something analogous to hold for Π-types, i.e., that two functions are equal exactly when they are equal at each point, but this is not the case in Martin-Löf type theory; the function extensionality axiom, introduced to fill this gap, asserts that $f =_{\Pi x:A.\,B} g$ is equivalent to $\Pi x : A.\,(f(x) =_B g(x))$ for each $f, g \in \Pi x : A.\,B$, which is the type-theoretic version of the idea explained above. The univalence axiom, which was introduced by Voevodsky and implies function extensionality, states that two types are equal exactly when they are equivalent, i.e., $A =_{\mathcal{U}} B$ is equivalent to $A \simeq B$; thus, an equality type whose type is a universe is inhabited if and only if the two chosen terms of type $\mathcal{U}$ (which must be types) are equivalent.

In [33], it is introduced a system, called Cubical type theory, in which both function extensionality and univalence are provable. It is an extension of Martin-Löf type theory which allows to directly represent the elements of the unit interval $[0, 1]$ and, thus, the n-dimensional cubes.

Both the systems in [95] and [33] had a huge success, allowing to open new perspectives on foundation of mathematics currently under intense investigation.

Interpretation

The semantics proposed in the Book is due to Voevodsky, and based on the idea that types can be seen as spaces in homotopy theory. As already mentioned, it interprets terms $p : a =_A b$ as paths with the interpretations of $a : A$ as start point and the one of $b : B$ as end point. The iteration of this process allows to interpret the whole theory in an ∞-groupoid, which is a collection of objects, paths between objects, paths between paths and so on, equipped with a suitable algebraic structure.

$$\begin{tikzcd}
a \arrow[r, bend left, "p"{name=P}] \arrow[r, bend right, "q"'{name=Q}] & b
\arrow[from=P, to=Q, Rightarrow, "x"]
\end{tikzcd}$$

As far as we have been able to track, the original idea of interpreting dependent type theory in ∞-groupoids is found in [49]; here, the terms a and b are seen as objects in a category, and $p : a =_A b$ is a morphism between them.

For each type A, the identity path exists because propositional equality is reflexive, i.e., $\mathsf{refl}_a : a =_A a$. It is proved that paths can be reversed and concatenated (which correspond to symmetry and transitivity of propositional equality), and that those operations are well-behaved accordingly to the groupoid laws; for example, $\left(p^{-1}\right)^{-1} = p$. We remind that a 1-groupoid is a category whose arrows are all isomorphisms: an ∞-groupoid is an ∞-category whose n-morphisms are isos with respect to the $(n+1)$-level, see [63], [26] or [62, 1.2.5].

[7]For the sake of precision, one should speak of left and right invertibility. However, it is easier to grasp the idea with the more usual notion of invertible function. See Chapter 4 of [95] for the reasons why simple invertibility is inadequate.

The general notion of an ∞-category is not straightforward, and probably not unique[8].

Functions $f : A \to B$ behave functorially on paths, i.e., they respect equality: if $p : a =_A b$, then it can be constructed $q : f(a) =_B f(b)$; the generalisation of this result, especially to dependent types, leads to a number of constructions broadly used throughout the Book, too complex to summarise here.

Dependent types $B : A \to \mathcal{U}_i$, also called type families, can be seen as fibrations with A as base space; in particular $\Sigma x : A.\, B$ is the total space of the fibration, and $B[a/x]$ is the fiber over a. Indeed, for each $p{:}a =_A b$ and $u{:}B[a/x]$, it can be constructed $p_*{:}B[a/x] \to B[b/x]$ and $\mathsf{lift}(u,p){:}(x,u) =_{\Sigma x:A.\, B} (y, p_*(u))$. The term lift is the type-theoretic version of the lifting of paths in homotopy theory, which characterises fibrations.

In the light of this interpretation, which is strongly based on propositional equality, the relevance of the novel axioms becomes clear, and the introduction of the higher inductive structure for types appears natural.

[8]I thank Prof. Cardone for pointing out this aspect, which is not immediate from the literature.

Chapter 3

Syntax

This chapter is devoted to illustrate the syntax adopted in the thesis, and to show some basic results about proof theory which will be useful to develop the semantics. The proof theory will be extended in Chapter 4, proving a normalisation result.

We divide Martin-Löf type theory into a basic system coupled with inductive types; the basic system contains the rules for contexts, variables, judgemental equality, universes and Π-types. The canonical inductive types considered are the same of [95], i.e., Σ, $+$, $\mathbf{1}$, $\mathbf{0}$, $\mathbb{N}$, $=$ and W; we call them inductive even if the only types which are properly inductive[1] are $\mathbb{N}$ and W. To make the system more manageable, we introduce a generic syntax for inductive types, comprehending all the canonical ones; it includes only "reasonable" inductive types, see [95, Section 5.6], although it could be extended to cover also mutually recursive types and similar, and is based on the idea that the formation, introduction and elimination rules for an inductive type can be rewritten as repeated applications computed over a constant of a proper type.

3.1 Basic system

The rules for the basic system are the ones presented in [95, Appendix A.2]. As usual, types are identified with those terms having a universe as type. In general, when not specified otherwise, we adopt the standard terminology of [95].

Context rules

Context rules govern the introduction of new contexts: the empty context $\bullet$ ctx can always be introduced, and if a context Γ can prove that A is a type, then $\Gamma, x : A$ is a context, provided x is new in Γ and A.

$$\frac{}{\bullet\ \mathsf{ctx}}\ \mathsf{ctx{-}EMP} \qquad \frac{\Gamma \vdash A : \mathcal{U}_i}{\Gamma, x : A\ \mathsf{ctx}}\ \mathsf{ctx{-}EXT}$$

Contexts are a finite set of typed variables, representing the assumptions of a judgement. The assumption $x : A$ ctx means that the type A is inhabited by, at least, x. Properly speaking, a context is a list of declarations: a declaration $x : A$ may contain variables in its type A which must have been declared before, i.e., which appear before in the context. However, as proved later in this chapter, a context may be seen as a set of declarations in which dependency is made explicit by variables.

[1]In this, we follow Chapter 5 of [95], although there are a few differences, remarked in due time.

Structural rules

Structural rules show when a variable can be introduced from a context

$$\frac{x_1 : A_1, \dots, x_n : A_n \ \mathsf{ctx}}{x_1 : A_1, \dots, x_n : A_n \vdash x_i : A_i}\ \mathsf{Vble}$$

and govern judgemental equality: it is an equivalence relation, and equal types have the same terms.

$$\frac{\Gamma \vdash a : A}{\Gamma \vdash a \equiv a : A}\ {\equiv}{-}\mathsf{refl} \qquad \frac{\Gamma \vdash a \equiv b : A}{\Gamma \vdash b \equiv a : A}\ {\equiv}{-}\mathsf{sym}$$

$$\frac{\Gamma \vdash a \equiv b : A \quad \Gamma \vdash b \equiv c : A}{\Gamma \vdash a \equiv c : A}\ {\equiv}{-}\mathsf{trans}$$

$$\frac{\Gamma \vdash a : A \quad \Gamma \vdash A \equiv B : \mathcal{U}_i}{\Gamma \vdash a : B}\ {\equiv}{-}\mathsf{subst}$$

$$\frac{\Gamma \vdash a \equiv b : A \quad \Gamma \vdash A \equiv B : \mathcal{U}_i}{\Gamma \vdash a \equiv b : B}\ {\equiv}{-}\mathsf{subst}{-}\mathsf{eq}$$

Notice that the rules ≡−sym and ≡−trans could be derived from[2]

$$\frac{\Gamma \vdash a \equiv b : A \quad \Gamma \vdash c \equiv b : A}{\Gamma \vdash c \equiv a : A}\ {\equiv}{-}\mathsf{symtrans}$$

It is important to remark that judgemental equality does not form new terms: $a \equiv b$ is definitely not a term of type A. Rather, judgemental equality is the core of the computational engine: in fact, it acts as $\beta\eta$-equality in the pure λ-calculus, see, e.g., [15].

Universes

Those rules govern the infinite hierarchy of universes: each universe is a term of the next one, and all the terms of a universe are terms also of the next one. Those rules formalise Russell's interpretation of universes, in the sense of [71].

$$\frac{\Gamma \ \mathsf{ctx}}{\Gamma \vdash \mathcal{U}_i : \mathcal{U}_{i+1}}\ \mathcal{U}{-}\mathsf{intro}$$

$$\frac{\Gamma \vdash A : \mathcal{U}_i}{\Gamma \vdash A : \mathcal{U}_{i+1}}\ \mathcal{U}{-}\mathsf{cumul} \qquad \frac{\Gamma \vdash A \equiv B : \mathcal{U}_i}{\Gamma \vdash A \equiv B : \mathcal{U}_{i+1}}\ \mathcal{U}{-}\mathsf{cumul}{-}\mathsf{eq}$$

Universes allow to define the notion of type:

Definition 3.1.1. A term a is a *type* if it is a term of a universe, i.e., if a judgement $\Gamma \vdash a : \mathcal{U}_i$ is derivable for some context Γ and some $i \in \mathbb{N}$.

Dependent product

If B is a type and $b : B$, possibly having $x : A$ as a free variable[3], then the *dependent product type* $\Pi x : A.\, B$ and the *abstraction* $\lambda x : A.\, b$ can be introduced,

[2]To conform to the syntax of [95], the ≡−symtrans rule is not used, but it may be useful to automated reasoning, if one likes to pursue this kind of applications.

[3]See Section 3.3 for a precise definition.

as shown by the following rules. Fixed $f : \Pi x : A.\, B$ and $a : A$, the *application* $f\, a$ is a term of type $B[a/x]$; if f is a canonical term $\lambda x : A.\, b$, then the application $(\lambda x : A.\, b)\, a$ is judgementally equal to $b[a/x]$. Finally, computing abstraction after application leads to the identity, i.e., $f \equiv \lambda x : A.\, f x$. To each formation, introduction and elimination rule is associated a corresponding equality rule which performs the same action on judgemental equalities.

$$\frac{\Gamma \vdash A : \mathcal{U}_i \quad \Gamma, x : A \vdash B : \mathcal{U}_i}{\Gamma \vdash \Pi x : A.\, B : \mathcal{U}_i}\ \Pi\mathsf{-form}$$

$$\frac{\Gamma \vdash A \equiv C : \mathcal{U}_i \quad \Gamma, x : A \vdash B \equiv D : \mathcal{U}_i}{\Gamma \vdash (\Pi x : A.\, B) \equiv (\Pi x : C.\, D) : \mathcal{U}_i}\ \Pi\mathsf{-form-eq}$$

$$\frac{\Gamma, x : A \vdash b : B}{\Gamma \vdash (\lambda x : A.\, b) : \Pi x : A.\, B}\ \Pi\mathsf{-intro}$$

$$\frac{\Gamma, x : A \vdash b \equiv c : B \quad \Gamma \vdash A \equiv C : \mathcal{U}_i}{\Gamma \vdash (\lambda x : A.\, b) \equiv (\lambda x : C.\, c) : \Pi x : A.\, B}\ \Pi\mathsf{-intro-eq}$$

$$\frac{\Gamma \vdash f : \Pi x : A.\, B \quad \Gamma \vdash a : A}{\Gamma \vdash f\, a : B[a/x]}\ \Pi\mathsf{-elim}$$

$$\frac{\Gamma \vdash f \equiv g : \Pi x : A.\, B \quad \Gamma \vdash a \equiv b : A}{\Gamma \vdash f\, a \equiv g\, b : B[a/x]}\ \Pi\mathsf{-elim-eq}$$

$$\frac{\Gamma, x : A \vdash b : B \quad \Gamma \vdash a : A}{\Gamma \vdash (\lambda x : A.\, b)\ a \equiv b[a/x] : B[a/x]}\ \Pi\mathsf{-comp}$$

$$\frac{\Gamma \vdash f : \Pi x : A., B}{\Gamma \vdash f \equiv (\lambda x : A.\, f\, x) : \Pi x : A., B}\ \Pi\mathsf{-uniq}$$

The notation $A \to B$ is an abbreviation of $\Pi x : A.\, B$ when $x \notin \mathsf{FV}(B)$.

The terms of a product type are the dependent functions from A to B, i.e., the functions whose codomain depends on the choice of the term in the domain. As already remarked, the type of (non-dependent) functions $A \to B$ corresponds to the product type $\Pi x : A.\, B$ when $x : A$ is not free in B.

Constants

Extending the basic system of [95], we allow for the introduction of constants of already existing types using the schema

$$\frac{\Gamma \vdash A : \mathcal{U}_i}{\Gamma \vdash k : A}\ k\mathsf{-intro}$$

The purpose of having constants is to uniformly treat the inductive types. In fact, allowing for arbitrary constants is easily seen to lead to inconsistent type theories, e.g., consider when A is $\mathbf{0}$, and, in general, the Normalisation Theorem fails for them. However, the advantages to consider the formation, introduction, and elimination rules of inductive types as instances of the constant schema cleans up many proofs, so we retain it until we will tackle the normalisation proof. Note that, until that time, we assume that a prefixed number of constants obeying to the schema is given, not that arbitrary constants could be freely introduced for any type.

There are numerous reasons to consider the system illustrated so far as basic. Following Chapter 2, the structural part (contexts, judgemental equality, and universes) is clearly fundamental to express and to construct the kind of judgements which are the subject of the type theory. Moreover, the dependent product allows to define, together with suitable constants, all the inductive types, see the next section, and, in particular, the canonical types. It should be clear that the axioms of function extensionality and univalence state that the dependent product and universes occupy a special position in the system, as identity over them cannot be completely generated without imposing additional property, as discussed in Section 2.2 and in Chapter 2 and 4 of [95]. One may object about the exclusion of identity types from the basic system, as this type is central in the homotopic interpretation. The reasons to consider identity types as an extension of the basic system are both technical and philosophical: technically, they are an instance of a general schema, and recognising this fact makes more uniform and simplifies the development of the theory; philosophically, the homotopic interpretation makes sense because of *path induction*, which is the homotopic reading of the inductive principle about identity types that descends from being an instance of inductive types. In other words, what allows the homotopic interpretation of equality is the inductive nature of identity types. Thus, we believe that syntactically identity types are better thought to as an instance of a general schema, while homotopically they deserve special attention. But these are different levels at which the theory can be studied.

3.2 General syntax for inductive types

The structure of the inductive types is similar to the one of the product types: formation and introduction rules work in an analogous way, but the elimination, computation, and uniqueness rules are more general. Given

- a type C depending on a variable z of the inductive type,
- for each introduction rule for the type, a term c whose type is C with z substituted by a canonical term of that introduction rule,
- a term e of the inductive type,

the elimination rule says that the type C with z substituted by e is inhabited; the inhabitant is called *inductive term* and denoted with ind. The meaning of the elimination rules is that, to prove that a statement holds for each term of an inductive type, it is enough to prove it for its canonical terms. The computation rules, one for each introduction rule, give a way to reduce the inductive terms when e is a canonical term of the inductive type.

The elimination rules, together with the computation rules, behave accordingly to the *inversion principle* [76]:

whatever follows from the direct grounds for deriving a proposition
must follow from that proposition.

Indeed, when the term e is a constructor (i.e., when elimination is computed after an introduction rule), the term obtained is equivalent to a suitable instantiation of the related term c.

For the inductive types we considered rules very similar to the ones of [95, Appendix A.2], but following the philosophy presented in Section 5.1 of the Book: types and terms are seen as a constant term of a dependent product type. Indeed, each formation, introduction and elimination rule is seen as a rule introducing a constant term of a certain already existing type, i.e., an instance of $k-\mathsf{intro}$.

We also introduce, for each inductive type, a uniqueness rule stating how induction is performed in the trivial case, i.e., when C coincides with the inductive type considered. The uniqueness rules for the inductive types are not present in [95]; there is only $\Pi-\mathsf{uniq}$[4].

Syntactically, an inductive type is a finite collection of symbols τ, ind_τ, $\mathbb{K}_1$, …, $\mathbb{K}_k$ together with a collection of inference rules, defining how judgements could be constructed, and how to reason and calculate. These inference rules have a strict format, illustrated in the following.

Formation rule

The *formation rule* $\tau-\mathsf{form}$ is

$$\frac{\Gamma \vdash (\Pi\,(x:F)_n\,.\,\mathcal{U}_i) : \mathcal{U}_{i+1}}{\Gamma \vdash \tau : (\Pi\,(x:F)_n\,.\,\mathcal{U}_i)}\ \tau-\mathsf{form}$$

where $\Pi\,(x:A)_n\,.\,B$ abbreviates $\Pi x_1 : A_1 . \cdots (\Pi x_n : A_n . B)$ (similar notations will be used from now on without further explanations). Sometimes, the abbreviated writing $\Pi x_1 : A_1, \ldots, x_n : A_n . B$ is used, too.

The purpose of the formation rule is to allow the construction of a new inductive type τ, using the syntax of the basic system. In fact, $\tau-\mathsf{form}$ is an instance of $k-\mathsf{intro}$. Usually, a specific inductive type is named after the τ symbol. Also, the sequence $(x:F)_n$ denotes the *parameters* of the τ type.

It is possible to have $n = 0$, so that the conclusion is $\Gamma \vdash \tau : \mathcal{U}_i$. It happens[5], for example, for the type $\mathbb{N}$ of natural numbers, as shown in Section 3.5.

Introduction rule

The canonical terms of an inductive type τ are defined by the *introduction rules*. The introduction rule for the $\mathbb{K}$ *constructor* is

$$\frac{\Gamma \vdash (\Pi\,(x:F)_{n'}\,.\,\Pi\,(y:I)_m\,.\,\tau\,x'_1\,\cdots\,x'_n) : \mathcal{U}_i}{\Gamma \vdash \mathbb{K} : (\Pi\,(x:F)_{n'}\,.\,\Pi\,(y:I)_m\,.\,\tau\,x'_1\,\cdots\,x'_n)}\ \tau-\mathsf{intro}$$

where $n' \leqslant n$, $(x:F)_{n'}$ is a subsequence of $(x:F)_n$, the parameters of τ, and for every $1 \leqslant i \leqslant n$, x'_i lies in $(x:F)_{n'}$, with a possibly different index, and $x'_i : F_i$ is the i-th element in $(x:F)_n$.

A constructor $\mathbb{K}$ is *total* when $(x:F)_{n'} = (x:F)_n$, and it is *partial* otherwise, meaning it is defined for every instance of a type, or just for some instances, respectively. An inductive type is *partial* if it has a partial constructor, and it is *total* otherwise. The sequence $(y:I)_m$ denotes the *parameters* of the $\mathbb{K}$ constructor. As before, note how the $\tau-\mathsf{intro}$ rule is an instance of $k-\mathsf{intro}$.

As shown in Section 3.5, dependent sum and coproduct are total, while the identity types are partial. The adjective “partial” means that the canonical constructors apply to a subset of the instances of the type. For example, refl applies only on $a =_A a$.

[4]The Book says that uniqueness rules trivialise the homotopic structure. Although this is right, it is convenient to consider them as optional.

[5]We deviate from the Book here since the premise becomes $\Gamma \vdash \mathcal{U}_i : \mathcal{U}_{i+1}$, which is different although equivalently derivable from $\Gamma\ \mathsf{ctx}$, see Corollary 3.3.5.

Elimination rule

The *simple elimination rule* for the τ type is

$$\dfrac{\begin{array}{l}\Gamma \vdash (\Pi\,(x:F)_n\,.\,(\Pi C : (\Pi\,(x:F)_n\,.\,\tau\,x_1\,\cdots\,x_n \to \mathcal{U}_h).\\ \qquad (\Pi_{i=1}^{k} c_i : (\Pi\,(x:F)_{n'_i}\,.\,\Pi\,(y:I)_{m_i}\,.\\ \qquad\quad C\,x'_1\,\cdots\,x'_{n_i}\,(\mathbb{K}\,x'_1\,\cdots\,x'_{n_i}\,y_1\,\cdots\,y_{m_i})).\\ \qquad (\Pi e : \tau\,x_1\,\cdots\,x_n.\,C\,x_1\,\cdots\,x_n\,e)))) : \mathcal{U}_{h+1}\end{array}}{\begin{array}{l}\Gamma \vdash \mathsf{ind}_\tau : (\Pi\,(x:F)_n\,.\,(\Pi C : (\Pi\,(x:F)_n\,.\,\tau\,x_1\,\cdots\,x_n \to \mathcal{U}_h).\\ \qquad (\Pi_{i=1}^{k} c_i : (\Pi\,(x:F)_{n'_i}\,.\,\Pi\,(y:I)_{m_i}\,.\\ \qquad\quad C\,x'_1\,\cdots\,x'_{n_i}\,(\mathbb{K}\,x'_1\,\cdots\,x'_{n_i}\,y_1\,\cdots\,y_{m_i})).\\ \qquad (\Pi e : \tau\,x_1\,\cdots\,x_n.\,C\,x_1\,\cdots\,x_n\,e))))\end{array}}\ \tau\text{—}\mathsf{elim}$$

where $(x:F)_n$ are the parameters of τ, $(y:I)_{m_i}$ are the parameters of $\mathbb{K}_i$, and for each $1 \leqslant j \leqslant k$, $(x:F)_{n'_j}$ and $\{x'\}_{n_j}$ are as in the $\tau\text{—}\mathsf{intro}_j$ rule.

The rationale is that ind_τ encodes the structural induction principle for the τ type: if C is a proposition depending on a generic instance of τ, if c_i proves C to hold when C is applied to a generic instance of the $\mathbb{K}_i$ constructor, and if e is an element of a specific instance of τ, then C holds on e.

A constructor parameter $y_i : I_i$ in $(y:I)_m$ is *recursive* when I is an instance of τ. In turn, an inductive type is *recursive* when some of its constructors have at least one recursive parameter. When τ is a recursive type, it is normally[6] used the *full elimination rule* as follows:

$$\dfrac{\begin{array}{l}\Gamma \vdash (\Pi\,(x:F)_n\,.\,(\Pi C : (\Pi\,(x:F)_n\,.\,\tau\,x_1\,\cdots\,x_n \to \mathcal{U}_h).\\ \qquad (\Pi_{i=1}^{k} c_i : (\Pi\,(x:F)_{n'_i}\,.\,\Pi\,(y:I)_{m'_i}\,.\\ \qquad\quad C\,x'_1\,\cdots\,x'_{n_i}\,(\mathbb{K}\,x'_1\,\cdots\,x'_{n_i}\,y_1\,\cdots\,y_{m_i})).\\ \qquad (\Pi e : \tau\,x_1\,\cdots\,x_n.\,C\,x_1\,\cdots\,x_n\,e)))) : \mathcal{U}_{h+1}\end{array}}{\begin{array}{l}\Gamma \vdash \mathsf{ind}_\tau : (\Pi\,(x:F)_n\,.\,(\Pi C : (\Pi\,(x:F)_n\,.\,\tau\,x_1\,\cdots\,x_n \to \mathcal{U}_h).\\ \qquad (\Pi_{i=1}^{k} c_i : (\Pi\,(x:F)_{n'_i}\,.\,\Pi\,(y:I)_{m'_i}\,.\\ \qquad\quad C\,x'_1\,\cdots\,x'_{n_i}\,(\mathbb{K}\,x'_1\,\cdots\,x'_{n_i}\,y_1\,\cdots\,y_{m_i})).\\ \qquad (\Pi e : \tau\,x_1\,\cdots\,x_n.\,C\,x_1\,\cdots\,x_n\,e))))\end{array}}\ \tau\text{—}\mathsf{elim}$$

where the notation is the same as in the simple elimination rule except for $(y:I)_{m'_i}$, the sequence of parameters of $\mathbb{K}$, in which, if $y : \Pi(w:L)_s.\,\tau\,a_1\,\cdots\,a_n$ is recursive, then it is immediately followed by

$$r : \Pi(w:L)_s.\,C\,a_1\,\cdots\,a_n\,(y\,w_1\,\cdots\,w_s)\ .$$

This additional variable, which is the only difference between the simple and the full rules, allows to keep track of the evidence for y when unfolding the induction. It is worth noticing how the full and the simple elimination rules coincide when τ is not recursive.

As an example, natural numbers are a recursive type, while dependent sum is non recursive, see Section 3.5.

Note how the usual introduction schema follows the simple elimination rule in the presentation of most mathematical books not dealing with type theory. The full elimination rule allows to deal with more advanced programming

[6]However, especially when dependent type theory is used as a programming language, there may be exceptions.

constructions, like *continuations* [91], but this issue is outside the scope of this thesis. Also, in the light of [76], note how the elimination rule is constructed from the formation and the introduction rules via the inversion principle.

Computation rule

For every $1 \leqslant i \leqslant k$, a *computation rule* is associated to the $\mathbb{K}_i$ constructor:

$$\frac{\begin{array}{l} \Gamma \vdash C : \left(\Pi\,(x:F)_n\,.\,\tau\,x_1\,\cdots\,x_n \to \mathcal{U}_h\right) \\ \Gamma \vdash c_1 : \Big(\Pi\,(x:F)_{n'_1}\,.\,\Pi\,(y:I)_{m'_1}\,. \\ \qquad\qquad C\,x'_1\,\cdots x'_{n_1}\,\left(\mathbb{K}_1\,x'_1\,\cdots x'_{n_1}\,y_1\,\cdots\,y_{m_1}\right)\Big) \\ \cdots \\ \Gamma \vdash c_k : \Big(\Pi\,(x:F)_{n'_k}\,.\,\Pi\,(y:I)_{m'_k}\,. \\ \qquad\qquad C\,x'_1\,\cdots x'_{n_k}\,\left(\mathbb{K}_k\,x'_1\,\cdots x'_{n_k}\,y_1\,\cdots\,y_{m_k}\right)\Big) \\ \Gamma \vdash \mathbb{K}_i\,T_1\,\cdots\,T_n\,p_1\,\cdots\,p_m : \tau\,T_1\,\cdots\,T_n \end{array}}{\begin{array}{l} \Gamma \vdash \mathsf{ind}_\tau\,T_1\,\cdots\,T_n\,C\,c_1\,\cdots\,c_k\,(\mathbb{K}_i\,T_1\,\cdots\,T_n\,p_1\,\cdots\,p_m) \equiv \\ \qquad c_i\,T_1\,\cdots\,T_n\,p'_1\,\cdots\,p'_m : C\,T_1\,\cdots\,T_n\,(\mathbb{K}_i\,T_1\,\cdots\,T_n\,p_1\,\cdots\,p_m) \end{array}}\ \tau\text{−}\mathsf{comp}_i$$

where $p'_j = p_j$ if the j-th parameter is normal, and p'_j is p_j immediately followed by $(\mathsf{ind}_\tau\,T_1\,\cdots\,T_n\,C\,c_1\,\cdots\,c_k\,p_j)$ if the j-th parameter is recursive.

The computation rule says that the induction, when the witness e is an instance of some constructor, can be simplified to the instance of the corresponding c case.

Uniqueness rule

The *uniqueness rule* is

$$\frac{\Gamma \vdash e : \tau\,T_1\,\cdots\,T_n}{\begin{array}{l} \Gamma \vdash \mathsf{ind}_\tau\,T_1, \cdots\,T_n \\ \qquad (\lambda\,(x:F)_n\,.\,(\lambda z : \tau\,x_1\,\cdots\,x_n . \tau\,x_1\,\cdots\,x_n)) \\ \qquad \Big(\lambda\,(x:F)_{n_1}\,.\,\Big(\lambda\,(y:I)_{m'_1}\,.\,\mathbb{K}_1\,x_1\,\cdots\,x_{n_1}\,y_1\,\cdots\,y_{m_1}\Big)\Big) \\ \qquad \cdots \\ \qquad \Big(\lambda\,(x:F)_{n_k}\,.\,\Big(\lambda\,(y:I)_{m'_k}\,.\,\mathbb{K}_k\,x_1\,\cdots\,x_{n_k}\,y_1\,\cdots\,y_{m_k}\Big)\Big) \\ \qquad e \equiv e : \tau\,T_1\,\cdots\,T_n \end{array}}\ \tau\text{−}\mathsf{uniq}$$

using the same notation as before. The uniqueness rule tells that performing an induction on the τ predicate in which the constructors are interpreted as themselves is redundant. As already said, uniqueness rules are optional and homotopy type theory does not include them.

Inductive ordering

The inductive nature of the previous syntax for general inductive types stems since an ordering $<_\tau$ can be defined on the collection of terms of type τ. Precisely, if $\Gamma \vdash \tau\,f_1 \ldots f_n : \mathcal{U}_i$ then $\Gamma \vdash t : \tau\,f_1 \ldots f_n <_\tau \Gamma \vdash t' : \tau\,f_1 \ldots f_n$ exactly when the term t' is generated from t using the introduction rules, or, the other way around, when t is inductively generated before t'. For example,

$\Gamma \vdash n : \mathbb{N} <_{\mathbb{N}} \Gamma \vdash \mathsf{succ}\, n : \mathbb{N}$ and $\Gamma \vdash v\, c : W\, A\, B <_W \Gamma \vdash \mathsf{sup} A\, B\, a\, v : W\, A\, B$ whenever $\Gamma \vdash c : B\, a$.

A precise definition of $<_\tau$ is beyond the scope of this dissertation, and it can be reconstructed through the initial algebra associated to τ, see Chapter 5 in [95], and [75, 99] for a detailed development. The fundamental facts to remark are: $<_\tau$ is a well ordering; in recursive types, the left-hand side of the computation rule is $\mathsf{ind}_\tau \ldots e$, and every occurrence of $e' : \tau\, f_1 \ldots f_n$ in the right-hand side is such that $e' <_\tau e$.

3.3 Basic proof theory

In this section will be shown the first, basic results of the proof theory of dependent types. In particular, all the properties needed in Chapter 5 can be found in the following. We will adopt the standard notation and terminology, mostly conforming to [95]. Also, we will use $\pi \colon \gamma$ to indicate that π is a derivation of the judgement γ.

Once described the syntax, summarised in Figures 3.1 and 3.2 for easiness of reference, we need to specify when a variable is free in a term and how substitution is computed in our system.

Definition 3.3.1. The sets of the *free variables* and *depending variables* in a term are inductively defined by:

1. if x is a variable, $\mathsf{FV}(x) = \mathsf{DV}(x) = \{x\}$;
2. if $\mathcal{U}_i$ is a universe, $\mathsf{FV}(\mathcal{U}_i) = \mathsf{DV}(\mathcal{U}_i) = \varnothing$;
3. if k is a constant of type A, $\mathsf{FV}(k) = \varnothing$ and $\mathsf{DV}(k) = \mathsf{DV}(A)$;
4. for an application, $\mathsf{FV}(t\, s) = \mathsf{FV}(t) \cup \mathsf{FV}(s)$ and $\mathsf{DV}(t\, s) = \mathsf{DV}(t) \cup \mathsf{DV}(s)$;
5. for an abstraction, $\mathsf{FV}(\lambda x : A.\, b) = (\mathsf{FV}(A) \cup \mathsf{FV}(b)) \setminus \{x\}$ and $\mathsf{DV}(\lambda x : A.\, b) = (\mathsf{DV}(A) \cup \mathsf{DV}(b)) \setminus \{x\}$;
6. for a function space type, $\mathsf{FV}(\Pi x : A.\, B) = (\mathsf{FV}(A) \cup \mathsf{FV}(B)) \setminus \{x\}$ and $\mathsf{DV}(\Pi x : A.\, B) = (\mathsf{DV}(A) \cup \mathsf{DV}(B)) \setminus \{x\}$.

Given $\Gamma \vdash a : A$, the set of *active variables* in a is defined as

$$\begin{aligned} \mathsf{AV}_0(a) &= \mathsf{DV}(a) \ , \\ \mathsf{AV}_{i+1}(a) &= \mathsf{AV}_i(a) \cup \bigcup_{\{B \colon x \in \mathsf{AV}_i(a) \wedge x : B \in \Gamma\}} DV(B) \\ \mathsf{AV}(a) &= \bigcup_{i \in \mathbb{N}} \mathsf{AV}_i(a) \ . \end{aligned}$$

The *free variables* are a subset of the *depending variables*: the latter set defines the variables a term depends on while the former set identifies the variables a term directly depends on. A close inspection of the definition reveals that the difference between FV and DV matters just when dealing with constants. Usually, abusing terminology but conforming to the standard usage, we speak of free variables even if DV should be intended.

Substitution is defined as a purely syntactical operation which replaces a variable with an expression inside another expression:

$$\frac{}{\bullet\ \mathsf{ctx}}\ \mathsf{ctx{-}EMP} \qquad \frac{\Gamma \vdash A : \mathcal{U}_i}{\Gamma, x : A\ \mathsf{ctx}}\ \mathsf{ctx{-}EXT}$$

$$\frac{x_1 : A_1, \ldots, x_n : A_n\ \mathsf{ctx}}{x_1 : A_1, \ldots, x_n : A_n \vdash x_i : A_i}\ \mathsf{Vble}$$

$$\frac{\Gamma \vdash a : A}{\Gamma \vdash a \equiv a : A}\ {\equiv}{-}\mathsf{refl} \qquad \frac{\Gamma \vdash a \equiv b : A}{\Gamma \vdash b \equiv a : A}\ {\equiv}{-}\mathsf{sym}$$

$$\frac{\Gamma \vdash a \equiv b : A \quad \Gamma \vdash b \equiv c : A}{\Gamma \vdash a \equiv c : A}\ {\equiv}{-}\mathsf{trans} \qquad \frac{\Gamma \vdash a : A \quad \Gamma \vdash A \equiv B : \mathcal{U}_i}{\Gamma \vdash a : B}\ {\equiv}{-}\mathsf{subst}$$

$$\frac{\Gamma \vdash a \equiv b : A \quad \Gamma \vdash A \equiv B : \mathcal{U}_i}{\Gamma \vdash a \equiv b : B}\ {\equiv}{-}\mathsf{subst{-}eq}$$

$$\frac{\Gamma\ \mathsf{ctx}}{\Gamma \vdash \mathcal{U}_i : \mathcal{U}_{i+1}}\ \mathcal{U}{-}\mathsf{intro} \qquad \frac{\Gamma \vdash A : \mathcal{U}_i}{\Gamma \vdash A : \mathcal{U}_{i+1}}\ \mathcal{U}{-}\mathsf{cumul} \qquad \frac{\Gamma \vdash A \equiv B : \mathcal{U}_i}{\Gamma \vdash A \equiv B : \mathcal{U}_{i+1}}\ \mathcal{U}{-}\mathsf{cumul{-}eq}$$

$$\frac{\Gamma \vdash A : \mathcal{U}_i \quad \Gamma, x : A \vdash B : \mathcal{U}_i}{\Gamma \vdash \Pi x : A.\, B : \mathcal{U}_i}\ \Pi{-}\mathsf{form}$$

$$\frac{\Gamma \vdash A \equiv C : \mathcal{U}_i \quad \Gamma, x : A \vdash B \equiv D : \mathcal{U}_i}{\Gamma \vdash (\Pi x : A.\, B) \equiv (\Pi x : C.\, D) : \mathcal{U}_i}\ \Pi{-}\mathsf{form{-}eq}$$

$$\frac{\Gamma, x : A \vdash b : B}{\Gamma \vdash (\lambda x : A.\, b) : \Pi x : A.\, B}\ \Pi{-}\mathsf{intro}$$

$$\frac{\Gamma, x : A \vdash b \equiv d : B \quad \Gamma \vdash A \equiv C : \mathcal{U}_i}{\Gamma \vdash (\lambda x : A.\, b) \equiv (\lambda x : C.\, d) : \Pi x : A.\, B}\ \Pi{-}\mathsf{intro{-}eq}$$

$$\frac{\Gamma \vdash f : \Pi x : A.\, B \quad \Gamma \vdash a : A}{\Gamma \vdash f\, a : B[a/x]}\ \Pi{-}\mathsf{elim}$$

$$\frac{\Gamma \vdash f \equiv g : \Pi x : A.\, B \quad \Gamma \vdash a \equiv c : A}{\Gamma \vdash f\, a \equiv g\, c : B[a/x]}\ \Pi{-}\mathsf{elim{-}eq}$$

$$\frac{\Gamma, x : A \vdash b : B \quad \Gamma \vdash a : A}{\Gamma \vdash (\lambda x : A.\, b)\, a \equiv b[a/x] : B[a/x]}\ \Pi{-}\mathsf{comp}$$

$$\frac{\Gamma \vdash f : \Pi x : A.\, B}{\Gamma \vdash f \equiv (\lambda x : A.\, f\, x) : \Pi x : A.\, B}\ \Pi{-}\mathsf{uniq}$$

Figure 3.1: The basic system.

$$\frac{\Gamma \vdash (\Pi\,(x:F)_n\,.\,\mathcal{U}_i) : \mathcal{U}_{i+1}}{\Gamma \vdash \tau : (\Pi\,(x:F)_n\,.\,\mathcal{U}_i)}\;\tau\text{−}\mathsf{form}$$

$$\frac{\Gamma \vdash (\Pi\,(x:F)_{n'}\,.\,\Pi\,(y:I)_m\,.\,\tau\,x'_1\,\cdots\,x'_n) : \mathcal{U}_i}{\Gamma \vdash \mathbb{K} : (\Pi\,(x:F)_{n'}\,.\,\Pi\,(y:I)_m\,.\,\tau\,x'_1\,\cdots\,x'_n)}\;\tau\text{−}\mathsf{intro}$$

$$\frac{\begin{array}{l}\Gamma \vdash (\Pi\,(x:F)_n\,.\,(\Pi C : (\Pi\,(x:F)_n\,.\,\tau\,x_1\,\cdots\,x_n \to \mathcal{U}_h).\\ \qquad\quad (\Pi_{i=1}^{k} c_i : (\Pi\,(x:F)_{n'_i}\,.\,\Pi\,(y:I)_{m'_i}\,.\\ \qquad\qquad C\,x'_1\,\cdots\,x'_{n_i}\,(\mathbb{K}\,x'_1\,\cdots\,x'_{n_i}\,y_1\,\cdots\,y_{m_i})).\\ \qquad\quad (\Pi e : \tau\,x_1\,\cdots\,x_n.\,C\,x_1\,\cdots\,x_n\,e)))) : \mathcal{U}_{h+1}\end{array}}{\begin{array}{l}\Gamma \vdash \mathsf{ind}_\tau : (\Pi\,(x:F)_n\,.\,(\Pi C : (\Pi\,(x:F)_n\,.\,\tau\,x_1\,\cdots\,x_n \to \mathcal{U}_h).\\ \qquad\quad (\Pi_{i=1}^{k} c_i : (\Pi\,(x:F)_{n'_i}\,.\,\Pi\,(y:I)_{m'_i}\,.\\ \qquad\qquad C\,x'_1\,\cdots\,x'_{n_i}\,(\mathbb{K}\,x'_1\,\cdots\,x'_{n_i}\,y_1\,\cdots\,y_{m_i})).\\ \qquad\quad (\Pi e : \tau\,x_1\,\cdots\,x_n.\,C\,x_1\,\cdots\,x_n\,e))))\end{array}}\;\tau\text{−}\mathsf{elim}$$

$$\frac{\begin{array}{l}\Gamma \vdash C : (\Pi\,(x:F)_n\,.\,\tau\,x_1\,\cdots\,x_n \to \mathcal{U}_h)\\ \Gamma \vdash c_1 : \Big(\Pi\,(x:F)_{n'_1}\,.\,\Pi\,(y:I)_{m'_1}\,.\\ \qquad\qquad C\,x'_1\,\cdots x'_{n_1}\,\big(\mathbb{K}_1\,x'_1\,\cdots x'_{n_1}\,y_1\,\cdots\,y_{m_1}\big)\Big)\\ \cdots\\ \Gamma \vdash c_k : \Big(\Pi\,(x:F)_{n'_k}\,.\,\Pi\,(y:I)_{m'_k}\,.\\ \qquad\qquad C\,x'_1\,\cdots x'_{n_k}\,\big(\mathbb{K}_k\,x'_1\,\cdots x'_{n_k}\,y_1\,\cdots\,y_{m_k}\big)\Big)\\ \Gamma \vdash \mathbb{K}_i\,T_1\,\cdots\,T_n\,p_1\,\cdots\,p_m : \tau\,T_1\,\cdots\,T_n\end{array}}{\begin{array}{l}\Gamma \vdash \mathsf{ind}_\tau\,T_1\,\cdots\,T_n\,C\,c_1\,\cdots\,c_k\,(\mathbb{K}_i\,T_1\,\cdots\,T_n\,p_1\,\cdots\,p_m) \equiv\\ \qquad c_i\,T_1\,\cdots\,T_n\,p'_1\,\cdots\,p'_m : C\,T_1\,\cdots\,T_n\,(\mathbb{K}_i\,T_1\,\cdots\,T_n\,p_1\,\cdots\,p_m)\end{array}}\;\tau\text{−}\mathsf{comp}_i$$

$$\frac{\Gamma \vdash e : \tau\,T_1\,\cdots\,T_n}{\begin{array}{l}\Gamma \vdash \mathsf{ind}_\tau\,T_1, \cdots\,T_n\\ \qquad\qquad (\lambda\,(x:F)_n\,.\,(\lambda z : \tau\,x_1\,\cdots\,x_n.\,\tau\,x_1\,\cdots\,x_n))\\ \qquad\qquad \Big(\lambda\,(x:F)_{n_1}\,.\,\Big(\lambda\,(y:I)_{m'_1}\,.\,\mathbb{K}_1\,x_1\,\cdots\,x_{n_1}\,y_1\,\cdots\,y_{m_1}\Big)\Big)\\ \qquad\qquad \cdots\\ \qquad\qquad \Big(\lambda\,(x:F)_{n_k}\,.\,\Big(\lambda\,(y:I)_{m'_k}\,.\,\mathbb{K}_k\,x_1\,\cdots\,x_{n_k}\,y_1\,\cdots\,y_{m_k}\Big)\Big)\\ \qquad\qquad e \equiv e : \tau\,T_1\,\cdots\,T_n\end{array}}\;\tau\text{−}\mathsf{uniq}$$

Figure 3.2: Inductive types.

Definition 3.3.2. If a and b are terms, and x is a variable, then $a[b/x]$ is inductively defined as:

- if $x \notin \mathsf{DV}(a)$, then $a[b/x] = a$;
- if $a = k$ constant of type A with $x \in \mathsf{DV}(k)$, then $k[b/x]$ is another constant of type $A[b/x]$;
- if $a = x$ then $a[b/x] = b$;
- if $a = f\, c$ then $a[b/x] = f[b/x]\, c[b/x]$;
- if $a = \lambda y : C.\, d$ then $a[b/x] = \lambda y : C[b/x].\, d[b/x]$ provided $y \notin \mathsf{DV}(b)$;
- if $a = \Pi y : C.\, d$ then $a[b/x] = \Pi y : C[b/x].\, d[b/x]$ provided $y \notin \mathsf{DV}(b)$.

The assumption in the last two cases can always be enforced[7] by renaming the *bound variable* y, i.e. substituting y in b by $z \notin \mathsf{AV}(a) \cup \mathsf{AV}(b)$. Substitution in a typed term is defined as $(a : A)[b/x] = (a[b/x] : A[b/x])$. For contexts, there are two cases:

- if $x_i \notin \mathsf{FV}(b)$,
$$\begin{aligned}&(x_1 : A_1, \ldots, x_n : A_n \ \mathsf{ctx})[b/x_i]\\ &= (x_1 : A_1, \ldots, x_{i-1} : A_{i-1}, x_{i+1} : A_{i+1}[b/x_i], \ldots, x_n : A_n[b/x_i] \ \mathsf{ctx}) \quad ;\end{aligned}$$
- otherwise,
$$\begin{aligned}&(x_1 : A_1, \ldots, x_n : A_n \ \mathsf{ctx})[b/x_i]\\ &= (x_1 : A_1, \ldots, x_i : A_i, x_{i+1} : A_{i+1}[b/x_i], \ldots, x_n : A_n[b/x_i] \ \mathsf{ctx}) \quad ;\end{aligned}$$

and in the other kinds of judgements:

$$\begin{aligned}(\Gamma \vdash a : A)[b/x] &= (\Gamma[b/x] \vdash a[b/x] : A[b/x]) \quad ;\\ (\Gamma \vdash a \equiv a' : A)[b/x] &= (\Gamma[b/x] \vdash a[b/x] \equiv a'[b/x] : A[b/x]) \quad .\end{aligned}$$

From now on, we will tacitly identify terms, types, and judgements up to *α-conversion*, i.e., renaming of bound variables. So, $\lambda x : A.\, b$ equals $\lambda y : A.\, b[y/x]$, $\Pi x : A.\, B$ equals $\Pi y : A.\, B[y/x]$, and $x : A \vdash b : B$ equals $y : A \vdash b[y/x] : B[y/x]$. Here, equality means syntactically equal, i.e., the two expressions are considered indistinguishable except for a different writing, which does not matter. Therefore, $=$ indicates syntactical equality, $\equiv$ is judgemental equality, and $=_A$ the identity type over A.

The notion of *subproof*, denoted by $\subseteq$, is easily defined by induction in the usual way. Essentially, $\pi_1 \subseteq \pi_2$ when the π_1 derivation is a subtree in the π_2 derivation.

Fact 3.3.3. *If $\pi\colon \Gamma, x : A, \Delta \vdash b : B$, or $\pi\colon \Gamma, x : A, \Delta \vdash b \equiv c : B$, or $\pi\colon \Gamma, x : A, \Delta$ ctx, then there is $\pi'\colon \Gamma \vdash A : \mathcal{U}_i$ such that $\pi' \subset \pi$ and there is $\pi''\colon \Gamma$ ctx such that $\pi'' \subset \pi'$.*

[7]In fact, by using de Bruijn notation [41] one could avoid altogether the issues about free and bound variables, and renaming. However, the human-friendly notation adopted here simplifies understanding.

Proof. A straightforward induction on the structure of the π derivation. $\square$

Remark 3.3.4. This fact shows how the first premise in the $\Pi{-}\mathsf{form}$ rule can be safely omitted provided A and B lie in the same universe, considering the equivalent

$$\dfrac{\Gamma, x : A \vdash B : \mathcal{U}_i}{\Gamma \vdash (\Pi x : A.\, B) : \mathcal{U}_i}\;\Pi{-}\mathsf{form}\quad .$$

The requirement that A and B are in the same universe is essential to avoid the formation of impredicative types.

Corollary 3.3.5. *If* $\pi\colon \Gamma \vdash a{:}A$ *or* $\pi\colon \Gamma \vdash a \equiv b{:}A$*, then exists* $(\pi_1 : \Gamma\ \mathsf{ctx}) \subset \pi$*.*

The following completes the definition of substitution in Martin-Löf type theory. It anticipates an independent result proved in the following, so no circularity is present; however, it is cleaner to present the definition here.

Definition 3.3.6 (Substitution in proofs)**.** Let $\pi\colon \Gamma, x : A, \Delta\ \mathsf{ctx}$, or $\pi\colon \Gamma, x : A, \Delta \vdash b : B$, or $\pi\colon \Gamma, x : A, \Delta \vdash b \equiv c : B$, and let $\pi'\colon \Gamma \vdash a : A$. Also, let $\pi''\colon \Gamma\ \mathsf{ctx}$ be such that $\pi'' \subset \pi$ as for Fact 3.3.3[8]. Then $\pi[a/x]$ is inductively defined as:

- if the last step in π is not an instance of $\mathsf{ctx{-}EXT}$ or Vble, or is an instance of $\mathsf{ctx{-}EXT}$ in which the conclusion is $\Gamma, x : A, \Delta, y : B\ \mathsf{ctx}$ with $x \neq y$, or is an instance of Vble in which the conclusion is $\Gamma, x : A, \Delta \vdash y : B$ with $x \neq y$, and π has premises $\pi_1, \ldots, \pi_n$, then $\pi[a/x]$ is the application of the same rule to $\pi_1[a/x], \ldots, \pi_n[a/x]$;
- if π is an instance of $\mathsf{ctx{-}EXT}$ with conclusion $\Gamma, x : A\ \mathsf{ctx}$, then $\pi[a/x] = \pi''$;
- if π is an instance of Vble with conclusion $\Gamma, x{:}A \vdash x{:}A$, then $\pi[a/x] = \pi'$;
- if π is an instance of Vble with conclusion $\Gamma, x : A, \Delta, y : B \vdash x : A$, then $\pi[a/x]$ is inductively constructed as in Proposition 3.3.12 from π'.

Some further properties about derivations are obtained starting from a counterexample which shows that an apparently natural property is false: if $\pi\colon \Gamma \vdash \mathcal{U}_i \equiv A : B$ or $\pi\colon \Gamma \vdash A \equiv \mathcal{U}_i : B$, then $A = \mathcal{U}_i$. In fact, considering for the sake of simplicity the basic system plus the natural numbers,

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\bullet\ \mathsf{ctx}}\,\mathsf{ctx{-}EMP}}{\vdash \mathcal{U}_i : \mathcal{U}_{i+1}}\,\mathcal{U}{-}\mathsf{intro}}{\vdash \mathbb{N} : \mathcal{U}_i}\,k{-}\mathsf{intro}}{\vdash 0 : \mathbb{N}}\,k{-}\mathsf{intro} \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\bullet\ \mathsf{ctx}}\,\mathsf{ctx{-}EMP}}{\vdash \mathcal{U}_i : \mathcal{U}_{i+1}}\,\mathcal{U}{-}\mathsf{intro}}{\vdash \mathbb{N} : \mathcal{U}_i}\,k{-}\mathsf{intro}}{x : \mathbb{N}\ \mathsf{ctx}}\,\mathsf{ctx{-}EXT}}{x : \mathbb{N} \vdash \mathcal{U}_i : \mathcal{U}_{i+1}}\,\mathcal{U}{-}\mathsf{intro}}{\vdash (\lambda x : \mathbb{N}.\mathcal{U}_i)\ 0 \equiv \mathcal{U}_i[0/x] : \mathcal{U}_{i+1}[0/x]}\,\Pi{-}\mathsf{comp}$$

However, $\vdash (\lambda x : \mathbb{N}.\mathcal{U}_i)\ 0 \equiv \mathcal{U}_i[0/x] : \mathcal{U}_{i+1}[0/x]$ is equal to $\vdash (\lambda x : \mathbb{N}.\mathcal{U}_i)\ 0 \equiv \mathcal{U}_i : \mathcal{U}_{i+1}$.

The following lemma tells how a regular judgement can be derived.

[8]The proof of Fact 3.3.3 effectively constructs π''

Lemma 3.3.7. *The following facts hold:*

1. *if* $\pi\colon \Gamma \vdash x : A$ *with* x *a variable, then* π *ends with a possibly empty sequence of instances of* $\equiv-$subst *and* $\mathcal{U}-$cumul *after an application of* Vble *whose conclusion is* $\Gamma \vdash x : B$ *with* $x : B \in \Gamma$*, and* $\Gamma \vdash A \equiv B : \mathcal{U}_i$*,* $i \in \mathbb{N}$ *or, if there are* $k > 0$ *applications of* $\mathcal{U}-$cumul*,* $\Gamma \vdash B \equiv \mathcal{U}_i : \mathcal{U}_{i+1}$ *and* $\Gamma \vdash A \equiv \mathcal{U}_{i+k} : \mathcal{U}_{i+k+1}$*;*

2. *if* $\pi\colon \Gamma \vdash \mathcal{U}_i : A$*, then* π *ends with a possibly empty sequence of instances of* $\equiv-$subst *and* $\mathcal{U}-$cumul *after an application of* $\mathcal{U}-$intro *whose conclusion is* $\Gamma \vdash \mathcal{U}_i : \mathcal{U}_{i+1}$*, and* $\Gamma \vdash A \equiv \mathcal{U}_{i+k+1} : \mathcal{U}_{i+k+2}$*, with* $k \in \mathbb{N}$*;*

3. *if* $\pi\colon \Gamma \vdash k : A$ *with* k *a constant, then* π *ends with a possibly empty sequence of instances of* $\equiv-$subst *and* $\mathcal{U}-$cumul *after an application of* $k-$intro *whose conclusion is* $\Gamma \vdash k : B$*, and* $\Gamma \vdash A \equiv B : \mathcal{U}_i$*,* $i \in \mathbb{N}$ *or, if there are* $k > 0$ *applications of* $\mathcal{U}-$cumul*,* $\Gamma \vdash B \equiv \mathcal{U}_i : \mathcal{U}_{i+1}$ *and* $\Gamma \vdash A \equiv \mathcal{U}_{i+k} : \mathcal{U}_{i+k+1}$*;*

4. *if* $\pi\colon \Gamma \vdash (\Pi x : A.\, B) : C$ *then* π *ends with a possibly empty sequence of instances of* $\equiv-$subst *and* $\mathcal{U}-$cumul *after an application of* $\Pi-$form*, whose conclusion is* $\pi\colon \Gamma \vdash (\Pi x : A.\, B) : \mathcal{U}_i$*, and* $\Gamma \vdash C \equiv \mathcal{U}_{i+k} : \mathcal{U}_{i+k+1}$*;*

5. *if* $\pi\colon \Gamma \vdash (\lambda x : A.\, b) : C$ *then* π *ends with a possibly empty sequence of applications of* $\equiv-$subst *and* $\mathcal{U}-$cumul *after an instance of* $\Pi-$intro *whose conclusion is* $\Gamma \vdash (\lambda x : A.\, b) : (\Pi x : A.\, B)$*, and* $\Gamma \vdash C \equiv (\Pi x : A.\, B) : \mathcal{U}_i$ *or, if there are* $k > 0$ *applications of* $\mathcal{U}-$cumul*,* $\Gamma \vdash \Pi x : A.\, B \equiv \mathcal{U}_i : \mathcal{U}_{i+1}$ *and* $\Gamma \vdash C \equiv \mathcal{U}_{i+k} : \mathcal{U}_{i+k+1}$*;*

6. *if* $\pi\colon \Gamma \vdash f\, a : C$ *then* π *ends with a possibly empty sequence of applications of* $\equiv-$subst *and* $\mathcal{U}-$cumul *after an instance of* $\Pi-$elim *whose conclusion is* $\Gamma \vdash f\, a : B[a/x]$*, and* $\Gamma \vdash C \equiv B[a/x] : \mathcal{U}_i$ *or, if there are* $k > 0$ *applications of* $\mathcal{U}-$cumul*,* $\Gamma \vdash B[a/x] \equiv \mathcal{U}_i : \mathcal{U}_{i+1}$ *and* $\Gamma \vdash C \equiv \mathcal{U}_{i+k} : \mathcal{U}_{i+k+1}$*.*

Proof. All the cases are proved following the same pattern: the only rules allowing to derive a judgement in each of the cases are $\equiv-$subst, $\mathcal{U}-$cumul, or the mentioned rule. The conclusion follows since equivalences can be composed by $\equiv-$trans. □

The following two properties show that types in judgements are well given and, in particular, that an inhabited term is a type.

Proposition 3.3.8. *If* $\pi\colon \Gamma \vdash a \equiv b : A$ *then* $\Gamma \vdash a : A$ *and* $\Gamma \vdash b : A$*.*

Proof. By induction on π:

- if the last step of π is an instance of $\equiv-$refl, $a = b$ and the premise $\Gamma \vdash a : A$ is the statement;
- if the last step of π is an instance of $\equiv-$sym or $\equiv-$trans, the induction hypothesis on the premise(s) yields the result;
- if the last step of π is an application of $\equiv-$subst$-$eq, $\mathcal{U}-$cumul$-$eq, or $\Pi-$elim$-$eq, the corresponding plain rule applied to the induction hypothesis on the premise(s) yields the result;

- if the last step of π is an instance of Π−form−eq or Π−intro−eq, the result follows as in the previous case eventually with an additional application of $\equiv$−subst on the second premise;

- if the last step of π is an instance of Π−comp, calling its premises $\pi_1 : \Gamma, x : D \vdash e{:}E$ and $\pi_2 : \Gamma \vdash d{:}D$, with $a = (\lambda x{:}D.\, e)$, $b = e[d/x]$, and $A = E[d/x]$, $\pi_1[d/x]$ shows $\Gamma \vdash b : A$, while $\Gamma \vdash a : A$ because

$$\dfrac{\dfrac{\pi_1 : \Gamma, x : D \vdash e : E}{\Gamma \vdash (\lambda x : D.\, e) : (\Pi x : D.\, E)}\,\Pi\text{−intro} \quad \pi_2 : \Gamma \vdash d : D}{\Gamma \vdash (\lambda x : D.\, e)\, d : A}\,\Pi\text{−elim}$$

- if the last step of π is an instance of Π−uniq, the premise $\pi_1 : \Gamma \vdash a : (\Pi x : D.\, E)$ with $A = (\Pi x : D.\, E)$ and $b = (\lambda x : D.\, a\, x)$ shows that $\Gamma \vdash a : A$. Then $\Gamma \vdash b : A$ holds because

$$\dfrac{\dfrac{\pi_1' : \Gamma, x : D \vdash a : (\Pi x : D.\, E) \quad \pi_2 : \Gamma, x : D \vdash x : D}{\Gamma, x : D \vdash a\, x : E}\,\Pi\text{−elim}}{\Gamma \vdash (\lambda x : D.\, a\, x) : (\Pi x : D.\, E)}\,\Pi\text{−intro}$$

  where π_2 is easily derived by Vble and Fact 3.3.3, while π_1' is obtained by π_1 through Proposition 3.3.12;

- if the last step is an instance of τ−comp$_i$,

$$\begin{aligned}\Gamma \vdash \mathsf{ind}_\tau\, T_1 \cdots T_n\, C\, c_1 \cdots c_k\, (\mathbb{K}_i\, T_1 \cdots T_n\, p_1 \cdots p_m) :& \\ C\, T_1 \cdots T_n\, (\mathbb{K}_i\, T_1 \cdots T_n\, p_1 \cdots p_m)&\end{aligned}$$

  by applying Π−elim to

$$\begin{aligned}\Gamma \vdash \mathsf{ind}_\tau : (\Pi\, (x : F)_n\,.\, (\Pi C : (\Pi\, (x : F)_n\,.\, \tau\, x_1 \cdots x_n \to \mathcal{U}_h).& \\ (\Pi_{i=1}^{k} c_i : (\Pi\, (x : F)_{n_i'}\,.\, \Pi\, (y : I)_{m_i'}\,.& \\ C\, x_1' \cdots x_{n_i}'\, (\mathbb{K}\, x_1' \cdots x_{n_i}'\, y_1 \cdots y_{m_i})).& \\ (\Pi e : \tau\, x_1 \cdots x_n.\, C\, x_1 \cdots x_n\, e))))&\end{aligned}$$

  and, in order, to

$$\begin{aligned}&\Gamma \vdash C : (\Pi\, (x : F)_n\,.\, \tau\, x_1 \cdots x_n \to \mathcal{U}_h) \\ &\Gamma \vdash c_1 : \big(\Pi\, (x : F)_{n_1'}\,.\, \Pi\, (y : I)_{m_1'}\,. \\ &\qquad C\, x_1' \cdots x_{n_1}'\, \big(\mathbb{K}_1\, x_1' \cdots x_{n_1}'\, y_1 \cdots y_{m_1}\big)\big) \\ &\cdots \\ &\Gamma \vdash c_k : \big(\Pi\, (x : F)_{n_k'}\,.\, \Pi\, (y : I)_{m_k'}\,. \\ &\qquad C\, x_1' \cdots x_{n_k}'\, \big(\mathbb{K}_k\, x_1' \cdots x_{n_k}'\, y_1 \cdots y_{m_k}\big)\big) \\ &\Gamma \vdash \mathbb{K}_i\, T_1 \cdots T_n\, p_1 \cdots p_m : \tau\, T_1 \cdots T_n \;\;;\end{aligned}$$

  also $\Gamma \vdash c_i\, T_1 \cdots T_n\, p_1' \cdots p_m' : C\, T_1 \cdots T_n\, (\mathbb{K}_i\, T_1 \cdots T_n\, p_1 \cdots p_m)$ can be derived appropriately applying Π−elim to

$$\Gamma \vdash c_i : \Pi\, (x : F)_{n_i'}\,.\, \Pi\, (y : I)_{m_i'}\,.\, C\, x_1' \cdots x_{n_i}'\, \big(\mathbb{K}_i\, x_1' \cdots x_{n_i}'\, y_1 \cdots y_{m_i}\big) \;\;;$$

- if the last step is an instance of $\tau-\mathsf{uniq}$, $\Gamma \vdash e : \tau\, T_1 \cdots T_n$ by the premise, and $\Gamma \vdash \mathsf{ind}_\tau\, T_1, \cdots T_n \ldots e : \tau\, T_1 \cdots T_n$ follows from

$$\begin{gathered}\Gamma \vdash \mathsf{ind}_\tau : (\Pi\,(x:F)_n\,.\,(\Pi C : (\Pi\,(x:F)_n\,.\,\tau\, x_1 \cdots x_n \to \mathcal{U}_h).\\ (\Pi_{i=1}^{k} c_i : (\Pi\,(x:F)_{n'_i}\,.\,\Pi\,(y:I)_{m'_i}\,.\\ C\, x'_1 \cdots x'_{n_i}\,(\mathbb{K}\, x'_1 \cdots x'_{n_i}\, y_1 \cdots y_{m_i})).\\ (\Pi e : \tau\, x_1 \cdots x_n . C\, x_1 \cdots x_n\, e))))\end{gathered}$$

  by repeated applications of $\Pi-\mathsf{elim}$. □

Proposition 3.3.9. *If $\pi : \Gamma \vdash a : A$ then $\Gamma \vdash A : \mathcal{U}_i$ for some $i \in \mathbb{N}$.*

Proof. By induction on π:

- if the last step of π is an instance of Vble, the conclusion follows by Fact 3.3.3;
- if the last step of π is an instance of $\equiv-\mathsf{subst}$ with premises $\Gamma \vdash a : B$ and $\pi_1 : \Gamma \vdash B \equiv A : \mathcal{U}_i$ then the conclusion follows by Proposition 3.3.8 on π_1;
- if the last step of π is an instance of $k-\mathsf{intro}$, in particular, if it is an instance of $\tau-\mathsf{form}$, $\tau-\mathsf{intro}$ or $\tau-\mathsf{elim}$, then the premise immediately yields the conclusion;
- if the last step of π is an instance of $\mathcal{U}-\mathsf{intro}$, $\mathcal{U}-\mathsf{cumul}$ or $\Pi-\mathsf{form}$ the result is obtained observing that $A = \mathcal{U}_j$, and thus by $\mathcal{U}-\mathsf{intro}$, $\Gamma \vdash \mathcal{U}_j : \mathcal{U}_{j+1}$;
- if the last step of π is an instance of $\Pi-\mathsf{intro}$, by induction hypothesis on the only premise, $\pi_1 : \Gamma, x : A \vdash B : \mathcal{U}_i$. By Fact 3.3.3 on π_1, $\pi_2 : \Gamma \vdash A : \mathcal{U}_i$, so $\Gamma \vdash (\Pi x : A.\, B) : \mathcal{U}_i$ by $\Pi-\mathsf{form}$ on π_1 and π_2;
- if the last step of π is an instance of $\Pi-\mathsf{elim}$, then by induction hypothesis on its premises, $\pi_1 : \Gamma \vdash A : \mathcal{U}_i$ and $\pi_2 : \Gamma \vdash (\Pi x : A.\, B) : \mathcal{U}_i$. Thus, by Lemma 3.3.7 on π_2, $\pi_3 : \Gamma, x : A \vdash B : \mathcal{U}_i$, hence $\pi_3[c/x] : \Gamma \vdash B[c/x] : \mathcal{U}_i$. □

It is important to remark that Propositions 3.3.8 and 3.3.9 provide an algorithm to effectively construct the proofs in their conclusions. Also, Proposition 3.3.8 can be redundantly rephrased as

Corollary 3.3.10 (Substitution rule)**.** *If $\Gamma \vdash a \equiv b : A$ and $\Gamma \vdash a : A$, then $\Gamma \vdash b : A$.*

Contexts

A strong property we need to prove is *weakening*: if a conclusion can be derived from a set of assumptions, then the same conclusion can be derived also from a larger set of assumptions. The result is not immediate because contexts are ordered, and assumptions may be permuted as far as their free variables appear before.

Proposition 3.3.11. *Let $x_1 : A_1, \ldots, x_n : A_n$ ctx and $x_{\pi(1)} : A_{\pi(1)}, \ldots, x_{\pi(n)} : A_{\pi(n)}$ ctx be contexts, and let π be a permutation of $\{1, \ldots, n\}$. Then*

- *if* $x_1 : A_1, \ldots, x_n : A_n \vdash b : B$ *then* $x_{\pi(1)} : A_{\pi(1)}, \ldots, x_{\pi(n)} : A_{\pi(n)} \vdash b : B$*;*
- *if* $x_1 : A_1, \ldots, x_n : A_n \vdash b \equiv c : B$ *then* $x_{\pi(1)} : A_{\pi(1)}, \ldots, x_{\pi(n)} : A_{\pi(n)} \vdash b \equiv c : B$*.*

Proof. By induction on the structure of derivations:

- if $x_1 : A_1, \ldots, x_n : A_n \vdash x_i : A_i$ by the Vble rule, then $x_{\pi(1)} : A_{\pi(1)}, \ldots, x_{\pi(n)} : A_{\pi(n)} \vdash x_{\pi(i)} : A_{\pi(i)}$ by the same rule. Notice how α-conversion has been tacitly applied.
- in all the other cases, if γ is deduced by an instance of the r rule, then the conclusion follows by the same rule applied to the result of the induction hypothesis on the premises. □

The previous proposition may be understood as contexts are sets with dependency. The fundamental weakening consists of adding a new variable to the context.

Proposition 3.3.12. *Let* $\pi_1 : \Gamma, x : A\ \mathsf{ctx}$*. Then*

- *if* $\pi : \Gamma \vdash b : B$ *then* $\pi' : \Gamma, x : A \vdash b : B$*;*
- *if* $\pi : \Gamma \vdash b \equiv c : B$ *then* $\pi' : \Gamma, x : A \vdash b \equiv c : B$*.*

Proof. By induction on the structure of derivations,

- if the last step in π is an instance of Vble or $\mathcal{U}-\mathsf{intro}$, π' is the application of the same rule to π_1;
- otherwise, π' is obtained applying the same rule as in the last step of π to the result of the inductive hypothesis on the premises. □

Theorem 3.3.13 (Weakening)**.** *Let* $\Gamma\ \mathsf{ctx}$ *and* $\Delta\ \mathsf{ctx}$ *such that* $\Gamma \subseteq \Delta$*. Then*

- *if* $\Gamma \vdash b : B$ *then* $\Delta \vdash b : B$*;*
- *if* $\Gamma \vdash b \equiv c : B$ *then* $\Delta \vdash b \equiv c : B$*.*

Proof. Let $\Xi = \Delta \setminus \Gamma$, in which subtraction preserves the relative ordering of elements. By induction on the length of Ξ:

- if Ξ is empty, then the statement follows by Proposition 3.3.11;
- if $\Xi = \Xi', x : A$ then, by induction hypothesis $\Gamma, \Xi' \vdash b : B$ and $\Gamma, \Xi' \vdash b \equiv c : B$ whenever $\Gamma \vdash b : B$ and $\Gamma \vdash b \equiv c : B$, respectively. Thus by Proposition 3.3.12, $\Gamma, \Xi \vdash b : B$ and $\Gamma, \Xi \vdash b \equiv c : B$, respectively. Hence, by Proposition 3.3.11 the conclusion follows. □

The weakening property says that, if something can be derived from a given context, it can be derived from every super-context. Oppositely, the following proposition shows what is the minimal context, up to valid permutations, which allows to derive a given conclusion.

Proposition 3.3.14. *If* $\Gamma \vdash a : A$ *and* $\Gamma \vdash A : \mathcal{U}_i$ *in the basic theory, then* $\mathsf{FV}(a) \subseteq \mathsf{DV}(a) \subseteq \mathsf{AV}(a)$ *and* $\mathsf{AV}(A) \subseteq \mathsf{AV}(a)$*. Moreover,* $\mathsf{AV}(a)$ ctx *is a judgement by ordering the declarations according to dependencies,* $\mathsf{AV}(a) \vdash a : A$*, and it is the minimal subcontext of* Γ *for which this happens.*

Proof. Long but obvious inductions allow to prove all the statements. □

The last property we need says that if A and A' are equivalent types, then what can be derived from A, can be proved from A', too. This property complements weakening since it shows that replacing declarations with equivalent ones does not change what can be derived.

Proposition 3.3.15. *Let* $\Gamma \vdash A \equiv A' : \mathcal{U}_i$*. Then*

1. *if* $\Gamma, x : A, \Delta \vdash b : B$ *then* $\Gamma, x : A', \Delta \vdash b : B$*;*
2. *if* $\Gamma, x : A, \Delta \vdash b \equiv c : B$ *then* $\Gamma, x : A', \Delta \vdash b \equiv c : B$*.*

Proof. By induction on the structure of derivations, we prove the first statement; the latter is analogous.

- if the last step is an instance of Vble, the conclusion $\Gamma, x : A, \Delta \vdash y : B$ is such that either $x \neq y$ so $\mathsf{AV}(y) \vdash y : B$, hence $\Gamma, x : A', \Delta \vdash y : B$ by Theorem 3.3.13; or $x = y$ so $\Gamma, x : A', \Delta \vdash x : A'$ by Vble. Thus, $\Gamma, x : A', \Delta \vdash x : A$ by $\equiv\!-\mathsf{subst}$.
- if the last step is $\mathcal{U}-\mathsf{intro}$, then $\Gamma, x : A', \Delta \vdash \mathcal{U}_i : \mathcal{U}_{i+1}$ by $\mathcal{U}-\mathsf{intro}$.
- in all the other cases, the result follows from the induction hypothesis. □

The above results, in particular Propositions 3.3.11 and 3.3.15, show that, as long as they are derivable, contexts can be identified up to permutations and equivalence of the types appearing in them. Thus, we define two concepts of equivalence between contexts, respectively catching only the first and both the notions.

Definition 3.3.16. If Γ ctx and Δ ctx are derivable, then $\Gamma \approx \Delta$, i.e., the Γ and Δ are *permutation equivalent*, if and only if Δ is a rearrangement of Γ.

Fact 3.3.17. $\approx$ *is an equivalence relation.*

Proof. Immediate since permutations form a group. □

Definition 3.3.18. If $\Gamma = x_1 : A_1, \dots, x_n : A_n$ and $\Delta = x_1 : B_1, \dots, x_n : B_n$ are derivable, then $\Gamma \sim \Delta$, i.e., the Γ and Δ are *equivalent*, if and only if $\Delta \approx \Gamma' = x_1 : C_1, \dots x_n : C_n$ and $x_1 : A_1, \dots, x_j : A_j \vdash A_{j+1} \equiv C_{j+1} : \mathcal{U}_i$ for every $0 \leqslant j < n$.

Fact 3.3.19. $\sim$ *is an equivalence relation.*

Definition 3.3.20. Two judgements $\Gamma \vdash a : A$ and $\Delta \vdash b : B$ are called *equivalent* when $\Gamma \sim \Delta$, and the judgements $\Gamma \vdash a \equiv b : A$ and $\Gamma \vdash A \equiv B : \mathcal{U}_i$ are derivable.

The abbreviated notation $a \equiv b$, used when the missing pieces are clear, means that there are a context Γ and a type A such that $\Gamma \vdash a \equiv b : A$.

The previous results about contexts allow to obtain two structural properties of the calculus, illustrated in the following.

Universes a la Russell lead to possible ambiguities: it may happen, in some particular cases, that a term c belongs to two types C and D such that a judgement of the form $\Gamma \vdash C \equiv D : T$ cannot be derived. The trivial example is when c is a type and $c : \mathcal{U}_i$, $c : \mathcal{U}_{i+1}$. The following result states when this situation may arise.

Proposition 3.3.21. *Let $\Gamma \vdash c : C$ and $\Gamma \vdash c : D$ be two derivable judgements in the basic system. Then, it holds one of the following*

1. $\Gamma \vdash C \equiv D : \mathcal{U}_i$;
2. $\Gamma \vdash C \equiv \Pi(x : G)_l . \mathcal{U}_j : \mathcal{U}_{j+1}$ *and* $\Gamma \vdash D \equiv \Pi(x : G)_l . \mathcal{U}_k : \mathcal{U}_{k+1}$. *Notice that, if $l = 0$, then $C \equiv \mathcal{U}_j$ and $D \equiv \mathcal{U}_k$.*

Proof. By induction on the structure of derivation of $\Gamma \vdash c : C$. By Lemma 3.3.7, $\Gamma \vdash c : D$ has to be similar to $\Gamma \vdash c : C$. In particular

- If the last rule applied is Vble or $k-$intro, then the only possibilities to obtain $\Gamma \vdash c : D$ are:
  - when $\Gamma \vdash C \equiv D : \mathcal{U}_i$, applying $\equiv-$subst and falling in case (1);
  - when $C \equiv U_j$ and $D \equiv U_k$, by repeated applications of $\mathcal{U}-$cumul and $\equiv-$subst, obtaining case (2) with $l = 0$.
- If the last rule applied is $\equiv-$subst, then by the premises $\Gamma \vdash c : A$ and $\Gamma \vdash A \equiv C : \mathcal{U}_i$. If $\Gamma \vdash c : D$ then, by induction hypothesis on $\Gamma \vdash c : A$, there are two possibilities:
  - if $\Gamma \vdash A \equiv D : \mathcal{U}_i$ then, by $\equiv-$sym and $\equiv-$trans, $\Gamma \vdash C \equiv D : \mathcal{U}_i$, which is case (1);
  - if $A \equiv \Pi(x : G)_l . \mathcal{U}_j$ and $D \equiv \Pi(x : G)_l . \mathcal{U}_k$ then, by $\equiv-$sym and $\equiv-$trans, $C \equiv \Pi(x : G)_l . \mathcal{U}_j$, which is case (2).
- If the last rule applied is $\mathcal{U}-$intro, $\mathcal{U}-$cumul, or $\Pi-$form, then $C = \mathcal{U}_{i+1}$, and the only possibilities to obtain $\Gamma \vdash c : D$ are
  - by $\equiv-$subst if $\Gamma \vdash \mathcal{U}_{i+1} \equiv D : \mathcal{U}_{i+2}$, leading to case (1);
  - by repeated applications of $\mathcal{U}-$cumul and $\equiv-$subst since $D \equiv U_k$ by Lemma 3.3.7; this is case (2).
- If the last rule applied is $\Pi-$intro then $C = \Pi x : A . B$, and the only possibilities to have $\Gamma \vdash c : D$ are:
  - when $\Gamma, x : A \vdash B \equiv B' : \mathcal{U}_i$ and $D \equiv \Pi x : A . B'$ applying $\Pi-$form$-$eq and $\equiv-$subst, falling in case (1);
  - when $B \equiv \Pi(x : G)_l . \mathcal{U}_j$, $B' = \Pi(x : G)_l . \mathcal{U}_k$, $D \equiv \Pi x : A . B'$ and, by induction hypothesis, $\Gamma, x : A \vdash b : B$ and $\Gamma, x : A \vdash b : B'$. Then, applying $\equiv-$subst and $\Pi-$form, $\Gamma \vdash c : \Pi x : A . \Pi(x : G)_l . \mathcal{U}_j$ and $\Gamma \vdash c : \Pi x : A . \Pi(x : G)_l . \mathcal{U}_k$, which is case (2).

- If the last rule applied is Π−elim, then $C = B[a/x]$, and the only possibilities to have $\Gamma \vdash c : D$ are when, by induction hypothesis, $\Gamma \vdash f : \Pi x : A.\, B$, $\Gamma \vdash f : \Pi x : A.\, B'$ and $D \equiv B'[a/x]$. This can lead to two different cases:
  - $\Gamma, x : A \vdash B \equiv B' : \mathcal{U}_i$. Then by Π−intro−eq, Π−comp and $\equiv$−trans we obtain $B[a/x] \equiv D$, i.e., case (1);
  - $B \equiv \Pi(x : G)_l.\mathcal{U}_j$ and $B' \equiv \Pi(x : G)_l.\mathcal{U}_k$. Then, by Π−elim, $\Gamma \vdash f\, a : (\Pi(x : G)_l.\mathcal{U}_j)[a/x]$ and $\Gamma \vdash f\, a : (\Pi(x : G)_l.\mathcal{U}_k)[a/x]$. Since $(\Pi(x : G)_l.\mathcal{U}_j)[a/x] = \Pi(x : G[a/x])_l.\mathcal{U}_j$, this is case (2). □

The following result shows that every descending chain such that $\Gamma \vdash a_{j+1} : a_j$, $j \in \mathbb{N}$, is finite. Of course, it does not hold with ascending chains: universes form an infinite ascending chain such that $\Gamma \vdash a_j : a_{j+1}$.

Proposition 3.3.22. *Every sequence $\{a_j\}_j$ such that $\Gamma \vdash a_{j+1} : a_j$ for every j is finite.*

Proof. By contradiction, let the sequence be infinite. Then, $\Gamma \vdash a_j : \mathcal{U}_{i_j}$ for every $j \in \mathbb{N}$ by Proposition 3.3.8. By Proposition 3.3.21, $\Gamma \vdash \mathcal{U}_{i_{j+1}} \equiv \Pi(x : G_{j+1})_{l_{j+1}}.\mathcal{U}_{k_{j+1}} : \mathcal{U}_{k_{j+1}+1}$ and $\Gamma \vdash a_j \equiv \Pi(x : G_{j+1})_{l_{j+1}}.\mathcal{U}_{h_{j+1}} : \mathcal{U}_{h_{j+1}+1}$, possibly with $l_{j+1} = 0$ and $k_{j+1} = h_{j+1}$, for every $j \in \mathbb{N}$. Hence, by $\equiv$−subst and Corollary 3.3.10 (and omitting the first term), the sequence $\{a_j\}_{j\in\mathbb{N}}$ can be rewritten as $\{\Pi(x : G_j)_{l_j}.\mathcal{U}_{h_j}\}_{j\in\mathbb{N}}$ with

$$\Gamma \vdash \Pi(x : G_{j+1})_{l_{j+1}}.\mathcal{U}_{h_{j+1}} : \Pi(x : G_j)_{l_j}.\mathcal{U}_{h_j} \quad . \tag{3.1}$$

By Lemma 3.3.7:

- if $l_{j+1} = 0$, then (3.1) is obtained by $\mathcal{U}$−intro after a sequence of $\equiv$−subst and $\mathcal{U}$−cumul, so $\Gamma \vdash \Pi(x : G_j)_{l_j}.\mathcal{U}_{h_j} \equiv \mathcal{U}_{\delta_j} : \mathcal{U}_{\delta_j+1}$;
- otherwise (3.1) is derived by Π−form after a sequence of $\equiv$−subst and $\mathcal{U}$−cumul, so $\Gamma, (x : G_{j+1})_{l_{j+1}} \vdash \Pi(x : G_j)_{l_j}.\mathcal{U}_{h_j} \equiv \mathcal{U}_{\delta_j} : \mathcal{U}_{\delta_j+1}$. By Theorem 3.3.13 and Proposition 3.3.14, $\Gamma \vdash \Pi(x : G_j)_{l_j}.\mathcal{U}_{h_j} \equiv \mathcal{U}_{\delta_j} : \mathcal{U}_{\delta_j+1}$.

Hence, by $\equiv$−subst and Corollary 3.3.10, the sequence simplifies to $\{\mathcal{U}_{\delta_j}\}_{j\in\mathbb{N}}$ with $\Gamma \vdash \mathcal{U}_{\delta_{j+1}} : \mathcal{U}_{\delta_j}$. Thus, by Lemma 3.3.7, $\{\delta_j\}_{j\in\mathbb{N}}$ is an infinite descending chain in $\mathbb{N}$, impossible. □

Corollary 3.3.23. *If Γ is a derivable context, then there is a derivable judgement $\Gamma \vdash a : A$ such that a is not inhabited.*

Proof. Immediate, by Proposition 3.3.22. □

3.4 Parallel substitution

The notion of substitution in Definition 3.3.2 is the standard on a single variable. However, when we consider multiple substitutions, which substitute many variables at once, there are many different ways to define them. If $[a_1/x_1, \ldots, a_n/x_n]$ is such a substitution, we may define it on terms as:

$$t[a_1/x_1, \ldots, a_n/x_n] = (\cdots((t[a_1/x_1])[a_2/x_2])\cdots)[a_n/x_n]$$

performing the substitution in a sequence. This kind of multiple substitution is therefore called *sequential.*

However, sequential substitution poses a number of problems when doing proof theoretical analysis: in fact, if $x_2 \in \mathsf{FV}(a_1)$ then x_2 gets substituted in a_1, while x_1 does not get substituted in a_2 even if $x_1 \in \mathsf{FV}(a_2)$. This asymmetrical behaviour causes the above mentioned difficulties. Hence, a different notion of multiple substitution is required: *parallel* substitution. Although technically more complex, its behaviour is symmetric: it operates only on the free variables in t leaving untouched the a_i's.

As said the definition of parallel substitution is more complex, requiring to mark the substituting terms so that they do not get substituted. The marking naturally generates a notion of *delayed* substitution: the result of a substitution leaves marked terms in the result, and we are required to cancel the marking, or to *reify* them before getting a proper term. Hence, a delayed substitution is completed when reified, and this action could take place not immediately after the substitution occurs.

Definition 3.4.1 (Delayed substitution)**.** If a and b are terms, and x is a variable, then $a[b/x]_{\mathsf{d}}$ is inductively defined as:

- if $x \notin \mathsf{DV}(a)$, then $a[b/x]_{\mathsf{d}} = a$;
- if $a = k$ constant of type A with $x \in \mathsf{DV}(k)$, then $k[b/x]_{\mathsf{d}}$ is another constant of type $A[b/x]$ (note that the second substitution is not delayed);
- if $a = x$ then $a[b/x]_{\mathsf{d}} = \sharp b$;
- if $a = f\,c$ then $a[b/x]_{\mathsf{d}} = f[b/x]_{\mathsf{d}}\, c[b/x]_{\mathsf{d}}$;
- if $a = \lambda y : C.\, d$ then $a[b/x]_{\mathsf{d}} = \lambda y : C[b/x]_{\mathsf{d}}.\, d[b/x]_{\mathsf{d}}$ provided $y \notin \mathsf{DV}(b)$;
- if $a = \Pi y : C.\, d$ then $a[b/x]_{\mathsf{d}} = \Pi y : C[b/x]_{\mathsf{d}}.\, d[b/x]_{\mathsf{d}}$ provided $y \notin \mathsf{DV}(b)$;
- if $a = \sharp c$ then $a[b/x]_{\mathsf{d}} = a$.

The same remarks of Definition 3.3.2 apply, and delayed substitution is extended to typed terms and judgements accordingly.

The $\sharp$ operator marks a term, and it lies outside the syntax, so the result of a delayed substitution is not a term, properly speaking. Also, it is worth noticing that a delayed substitution naturally applies to already marked terms.

Definition 3.4.2 (Reification)**.** If t is a marked term, then $\Lambda(t)$ is inductively defined as the operation removing the $\sharp$ marks from t:

- if t is a variable or a constant, $\Lambda(t) = t$;
- if $t = a\,b$, $\Lambda(t) = \Lambda(a)\,\Lambda(b)$;
- if $t = \lambda x : A.\, b$, $\Lambda(t) = \lambda x : \Lambda(A).\, \Lambda(b)$;
- if $t = \Pi x : A.\, b$, $\Lambda(t) = \Pi x : \Lambda(A).\, \Lambda(b)$;
- if $t = \sharp a$ then $\Lambda(t) = a$.

Reification is extended to typed terms and judgements in the obvious way.

On a single-variable delayed substitution coincides with the usual notion.

Fact 3.4.3. *$\Lambda(a[b/x]_{\mathsf{d}}) = a[b/x]$.*

Proof. Straightforward induction on delayed substitutions. □

At this point, parallel substitution can be easily defined. It is convenient to keep separate two notions: parallel and delayed parallel substitutions, which are related as in Fact 3.4.3.

Definition 3.4.4 (Delayed parallel substitution)**.** Let $a_1, \ldots, a_n, t$ be terms, and let $x_1, \ldots, x_n$ be distinct variables. Then

$$t[a_1/x_1, \ldots, a_n/x_n]_{\mathsf{d}} = (\cdots(t[a_1/x_1]_{\mathsf{d}})[a_2/x_2]_{\mathsf{d}}\cdots)[a_n/x_n]_{\mathsf{d}} \enspace .$$

Extensions to typed terms and judgements are obvious.

Definition 3.4.5 (Parallel substitution)**.** Let $a_1, \ldots, a_n, t$ be terms, and let $x_1, \ldots, x_n$ be distinct variables. Then

$$t[a_1/x_1, \ldots, a_n/x_n] = \Lambda\left(t[a_1/x_1, \ldots, a_n/x_n]_{\mathsf{d}}\right) \enspace .$$

Extensions to typed terms and judgements are as before.

It is worth remarking how multiple substitutions are finite: in Martin-Löf type theory the only variables which may be affected by substitutions are those appearing in the context part of the substituted object. This observation leads to equip substitutions with a *reference context.* If Γ is a context, a substitution $[a_1/x_1, \ldots, a_n/x_n]$, delayed or not, is such that $x_1, \ldots, x_n$ occur in Γ in the *same* order.

We need to perform two operations on multiple substitutions: *composition* ($\circ$) and *subtraction* ($\setminus$). Although they can be defined more in general, we impose two constraints: the two operands are both substitutions on the same reference context, and the second operand is a substitution of a single variable. These constraints are what we use in the rest of the dissertation, so we leave the straightforward generalisations to the reader.

Definition 3.4.6 (Composition)**.** If $[a_1/x_1, \ldots, a_n/x_n]_{\mathsf{d}}$ and $[b/x_k]_{\mathsf{d}}$ are delayed parallel substitutions on $y_1 : A_1, \ldots, y_m : A_m$, with $x_k \notin \{x_1, \ldots, x_n\}$, then $[a_1/x_1, \ldots, a_n/x_n]_{\mathsf{d}} \circ [b/x_k]_{\mathsf{d}} = [a_1/x_1, \ldots, a_j/x_j, b/x_k, a_{j+1}/x_{j+1}, \ldots, a_n/x_n]_{\mathsf{d}}$, with x_k inserted in the right position according to the order given by $y_1, \ldots, y_m$.

Definition 3.4.7 (Subtraction)**.** If $[a_1/x_1, \ldots, a_n/x_n]_{\mathsf{d}}$ and $[b/x_k]_{\mathsf{d}}$ are delayed parallel substitutions on $y_1 : A_1, \ldots, y_m : A_m$, with $b = a_i$ if $1 \leqslant k \leqslant n$, then $[a_1/x_1, \ldots, a_n/x_n]_{\mathsf{d}} \setminus [b/x_k]_{\mathsf{d}} = [a_1/x_1, \ldots, a_{k-1}/x_{k-1}, a_{k+1}/x_{k+1}, \ldots, a_n/x_n]_{\mathsf{d}}$ if $1 \leqslant k \leqslant n$ and $[a_1/x_1, \ldots, a_n/x_n]_{\mathsf{d}}$ otherwise.

Since composition and subtraction have been defined just in the delayed case, we usually drop the subscript when the operations are used, since the nature of substitutions will be clear. Also, we will use "x is in the domain of σ" when $\sigma(x) \neq x$.

Proposition 3.4.8. *If σ is a delayed parallel substitution on Γ, then $(\sigma \setminus [\sigma(x)/x]) \circ [\sigma(x)/x] = \sigma$.*

Proof. Immediate by unfolding Definitions 3.4.6 and 3.4.7. □

Fact 3.4.9. *If σ is a delayed parallel substitution on Γ and x is in the domain of σ, then*

$$(\Gamma \vdash t : A)\sigma = (\Gamma[\sigma(x)/x] \vdash t[\sigma(x)/x] : A[\sigma(x)/x])(\sigma \setminus [\sigma(x)/x]) \ .$$

Proof. Obvious consequence of Proposition 3.4.8. □

Proposition 3.4.10. *If σ and $[a/x]$ are delayed parallel substitutions on Γ, and x in not in the domain of σ, then*

$$(\sigma \circ [\sigma(x)/x]) \setminus [\sigma(y)/y] = (\sigma \setminus [\sigma(y)/y]) \circ [\sigma(x)/x]$$

for every y in the domain of σ.

Proof. Immediate by unfolding Definitions 3.4.6 and 3.4.7. □

Proposition 3.4.11. *If σ is a delayed parallel substitution on Γ, and x, y are distinct variables in the domain of σ, then*

$$(\sigma \setminus [\sigma(x)/x]) \setminus [\sigma(y)/y] = (\sigma \setminus [\sigma(y)/y]) \setminus [\sigma(x)/x] \ .$$

Proof. Again, immediate by unfolding Definitions 3.4.6 and 3.4.7. □

In the following of the dissertation, we drop the "parallel" adjective when speaking about substitutions. However, when delayed (parallel) substitutions are used, this will be made explicit either in the discourse or in the notation. When ambiguity in this respect helps understanding, we will be ambiguous, principally to alleviate notation, but in the fundamental definitions and in the crucial passages, ambiguities will always be resolved.

3.5 Canonical inductive types

The end of this chapter is devoted to show how the canonical inductive types can be translated in our syntax, see Section 3.2. For the canonical rules, we recall the Appendix of [95]. Except for W-types, the translation is immediate, and it is evident that our rules are equivalent to the usual ones.

Dependent sum

The usual syntax does not use constants, so $\Sigma x : A.\, B$ is an abbreviation for $\Sigma\, A\ (\lambda x : A.\, B)$, and (a, b) stands for $\mathsf{pair}\, A\, B\, a\, b$. Here, we will use the unabbreviated syntax to make explicit the inductive nature of Σ-types. The dependent sum inductive type is total and non-recursive.

$$\frac{\Gamma \vdash (\Pi A : \mathcal{U}_i.\, \Pi B : (A \to \mathcal{U}_i)\, .\, \mathcal{U}_i) : \mathcal{U}_{i+1}}{\Gamma \vdash \Sigma : (\Pi A : \mathcal{U}_i.\, \Pi B : (A \to \mathcal{U}_i)\, .\, \mathcal{U}_i)}\ \Sigma\text{−form}$$

The conclusion could be rewritten as $\Gamma \vdash \Sigma : (\Pi A : \mathcal{U}_i.\, (A \to \mathcal{U}_i) \to \mathcal{U}_i)$.

$$\frac{\Gamma \vdash (\Pi A : \mathcal{U}_i.\, \Pi B : (A \to \mathcal{U}_i)\, .\, \Pi a : A.\, \Pi b : B\, a.\, \Sigma\, A\, B) : \mathcal{U}_{i+1}}{\Gamma \vdash \mathsf{pair} : (\Pi A : \mathcal{U}_i.\, \Pi B : (A \to \mathcal{U}_i)\, .\, \Pi a : A.\, \Pi b : B\, a.\, \Sigma\, A\, B)}\ \Sigma\text{−intro}$$

The conclusion could be rewritten as

$$\Gamma \vdash \mathsf{pair} : (\Pi A : \mathcal{U}_i .\, \Pi B : (A \to \mathcal{U}_i)\,.\, \Pi a : A .\, B\, a \to \Sigma\, A\, B) \quad .$$

$$\frac{\begin{array}{l}\Gamma \vdash \big(\Pi A : \mathcal{U}_i .\, \Pi B : (A \to \mathcal{U}_i). \\ \quad \Pi C : (\Pi A : \mathcal{U}_i .\, \Pi B : (A \to \mathcal{U}_i).\, \Sigma\, A\, B \to \mathcal{U}_i)\,. \\ \quad \Pi c_1 : (\Pi A : \mathcal{U}_i .\, \Pi B : (A \to \mathcal{U}_i).\, \Pi a : A .\, \Pi b : B\, a . \\ \quad\quad C\, A\, B\, (\mathsf{pair}\, A\, B\, a\, b)). \\ \quad \Pi e : \Sigma\, A\, B .\, C\, A\, B\, e\big) : \mathcal{U}_{i+1}\end{array}}{\begin{array}{l}\Gamma \vdash \mathsf{ind}_\Sigma : \big(\Pi A : \mathcal{U}_i .\, \Pi B : (A \to \mathcal{U}_i). \\ \quad \Pi C : (\Pi A : \mathcal{U}_i .\, \Pi B : (A \to \mathcal{U}_i).\, \Sigma\, A\, B \to \mathcal{U}_i)\,. \\ \quad \Pi c_1 : (\Pi A : \mathcal{U}_i .\, \Pi B : (A \to \mathcal{U}_i).\, \Pi a : A .\, \Pi b : B\, a . \\ \quad\quad C\, A\, B\, (\mathsf{pair}\, A\, B\, a\, b)). \\ \quad \Pi e : \Sigma\, A\, B .\, C\, A\, B\, e\big)\end{array}}\ \Sigma\mathsf{-elim}$$

$$\frac{\begin{array}{l}\Gamma \vdash C : (\Pi A : \mathcal{U}_i .\, \Pi B : (A \to \mathcal{U}_i).\, \Sigma\, A\, B \to \mathcal{U}_i) \\ \Gamma \vdash c_1 : (\Pi A : \mathcal{U}_i .\, \Pi B : (A \to \mathcal{U}_i).\, \Pi a : A .\, \Pi b : B\, a .\, C\, A\, B\ (\mathsf{pair}\, A\, B\, a\, b)) \\ \Gamma \vdash \mathsf{pair}\, A'\, B'\, a'\, b' : \Sigma\, A'\, B'\end{array}}{\begin{array}{l}\Gamma \vdash \mathsf{ind}_\Sigma\, A'\, B'\, C\, c_1\ (\mathsf{pair}\, A'\, B'\, a'\, b') \equiv \\ \quad c_1\, A'\, B'\, a'\, b' : C\, A'\ B'\ (\mathsf{pair}\, A'\, B'\, a'\, b')\end{array}}\ \Sigma\mathsf{-comp}$$

$$\frac{\Gamma \vdash e : \Sigma\, A'\, B'}{\begin{array}{l}\Gamma \vdash \mathsf{ind}_\Sigma\, A'\, B'\ (\lambda A : \mathcal{U}_i .\, \lambda B : (A \to \mathcal{U}_i).\, \lambda x : \Sigma\, A\, B .\, \Sigma\, A\, B) \\ \quad (\lambda A : \mathcal{U}_i .\, \lambda B : (A \to \mathcal{U}_i).\, \lambda a : A .\, \lambda b : B\, a .\, \mathsf{pair}\, A\, B\, a\, b) \\ \quad e \equiv e : \Sigma\, A'\, B'\end{array}}\ \Sigma\mathsf{-uniq}$$

Coproduct

The usual syntax is $A+B$, which abbreviates $+\, A\, B$, and $\mathsf{inl}\, a$, $\mathsf{inr}\, b$, abbreviating $\mathsf{inl}\, A\, B\, a$ and $\mathsf{inr}\, A\, B\, b$, respectively. The coproduct inductive type is total and non-recursive.

$$\frac{\Gamma \vdash (\Pi A : \mathcal{U}_i .\, \Pi B : \mathcal{U}_i . \mathcal{U}_i) : \mathcal{U}_{i+1}}{\Gamma \vdash + : (\Pi A : \mathcal{U}_i .\, \Pi B : \mathcal{U}_i . \mathcal{U}_i)}\ +\mathsf{-form}$$

The conclusion can be rewritten as $\Gamma \vdash + : \mathcal{U}_i \to (\mathcal{U}_i \to \mathcal{U}_i)$.

$$\frac{\Gamma \vdash (\Pi A : \mathcal{U}_i .\, \Pi B : \mathcal{U}_i .\, \Pi a : A .\, +\, A\, B) : \mathcal{U}_{i+1}}{\Gamma \vdash \mathsf{inl} : (\Pi A : \mathcal{U}_i .\, \Pi B : \mathcal{U}_i .\, \Pi a : A .\, +\, A\, B)}\ +\mathsf{-intro}_1$$

$$\frac{\Gamma \vdash (\Pi A : \mathcal{U}_i .\, \Pi B : \mathcal{U}_i .\, \Pi b : B .\, +\, A\, B) : \mathcal{U}_{i+1}}{\Gamma \vdash \mathsf{inr} : (\Pi A : \mathcal{U}_i .\, \Pi B : \mathcal{U}_i .\, \Pi b : B .\, +\, A\, B)}\ +\mathsf{-intro}_2$$

The conclusions can be rewritten, respectively, as $\Gamma \vdash \mathsf{inl} : \Pi A : \mathcal{U}_i .\, \Pi B : \mathcal{U}_i .\, A \to +\, A\, B$ and $\Gamma \vdash \mathsf{inr} : \Pi A : \mathcal{U}_i .\, \Pi B : \mathcal{U}_i .\, B \to +\, A\, B$.

$$\frac{\begin{array}{l}\Gamma \vdash \big(\Pi A : \mathcal{U}_i .\, \Pi B : \mathcal{U}_i . \\ \quad \Pi C : (\Pi A : \mathcal{U}_i .\, \Pi B : \mathcal{U}_i .\, + A\, B \to \mathcal{U}_i) \,. \\ \quad \Pi c_1 : (\Pi A : \mathcal{U}_i .\, \Pi B : \mathcal{U}_i .\, \Pi a : A .\, C\, A\, B\ (\mathsf{inl}\, A\, B\, a)) \,. \\ \quad \Pi c_2 : (\Pi A : \mathcal{U}_i .\, \Pi B : \mathcal{U}_i .\, \Pi b : B .\, C\, A\, B\ (\mathsf{inr}\, A\, B\, b)) \,. \\ \quad \Pi e : + A\, B .\, C\, A\, B\ e\big) : \mathcal{U}_{i+1}\end{array}}{\begin{array}{l}\Gamma \vdash \mathsf{ind}_+ : \big(\Pi A : \mathcal{U}_i .\, \Pi B : \mathcal{U}_i . \\ \quad \Pi C : (\Pi A : \mathcal{U}_i .\, \Pi B : \mathcal{U}_i .\, + A\, B \to \mathcal{U}_i) \,. \\ \quad \Pi c_1 : (\Pi A : \mathcal{U}_i .\, \Pi B : \mathcal{U}_i .\, \Pi a : A .\, C\, A\, B\ (\mathsf{inl}\, A\, B\, a)) \,. \\ \quad \Pi c_2 : (\Pi A : \mathcal{U}_i .\, \Pi B : \mathcal{U}_i .\, \Pi b : B .\, C\, A\, B\ (\mathsf{inr}\, A\, B\, b)) \,. \\ \quad \Pi e : + A\, B .\, C\, A\, B\ e\big)\end{array}}\ +\!-\mathsf{elim}$$

$$\frac{\begin{array}{l}\Gamma \vdash C : (\Pi A : \mathcal{U}_i .\, \Pi B : \mathcal{U}_i .\, + A\, B \to \mathcal{U}_i) \\ \Gamma \vdash c_1 : (\Pi A : \mathcal{U}_i .\, \Pi B : \mathcal{U}_i .\, \Pi a : A .\, C\, A\, B\ (\mathsf{inl}\, A\, B\, a)) \\ \Gamma \vdash c_2 : (\Pi A : \mathcal{U}_i .\, \Pi B : \mathcal{U}_i .\, \Pi b : B .\, C\, A\, B\ (\mathsf{inr}\, A\, B\, b)) \\ \Gamma \vdash \mathsf{inl}\, A'\, B'\, a' : + A'\, B'\end{array}}{\begin{array}{l}\Gamma \vdash \mathsf{ind}_+\, A'\, B'\, C\, c_1\, c_2\ (\mathsf{inl}\, A'\, B'\, a') \equiv \\ \quad c_1\, A'\, B'\, a' : C\, A'\, B'\ (\mathsf{inl}\, A'\, B'\, a')\end{array}}\ +\!-\mathsf{comp}_1$$

$$\frac{\begin{array}{l}\Gamma \vdash C : (\Pi A : \mathcal{U}_i .\, \Pi B : \mathcal{U}_i .\, + A\, B \to \mathcal{U}_i) \\ \Gamma \vdash c_1 : (\Pi A : \mathcal{U}_i .\, \Pi B : \mathcal{U}_i .\, \Pi a : A .\, C\, A\, B\ (\mathsf{inl}\, A\, B\, a)) \\ \Gamma \vdash c_2 : (\Pi A : \mathcal{U}_i .\, \Pi B : \mathcal{U}_i .\, \Pi b : B .\, C\, A\, B\ (\mathsf{inr}\, A\, B\, b)) \\ \Gamma \vdash \mathsf{inr}\, A'\, B'\, b' : + A'\, B'\end{array}}{\begin{array}{l}\Gamma \vdash \mathsf{ind}_+\, A'\, B'\, C\, c_1\, c_2\ (\mathsf{inr}\, A'\, B'\, b') \equiv \\ \quad c_2\, A'\, B'\, b' : C\, A'\, B'\ (\mathsf{inr}\, A'\, B'\, b')\end{array}}\ +\!-\mathsf{comp}_2$$

$$\frac{\Gamma \vdash e : + A'\, B'}{\begin{array}{l}\Gamma \vdash \mathsf{ind}_+\, A'\, B' \\ \quad (\lambda A : \mathcal{U}_i .\, \lambda B : \mathcal{U}_i .\, \lambda x : + A\, B .\, + A\, B) \\ \quad (\lambda A : \mathcal{U}_i .\, \lambda B : \mathcal{U}_i .\, \lambda a : A .\, \mathsf{inl}\, A\, B\, a) \\ \quad (\lambda A : \mathcal{U}_i .\, \lambda B : \mathcal{U}_i .\, \lambda b : B .\, \mathsf{inr}\, A\, B\, b)\ e \equiv e : + A'\, B'\end{array}}\ +\!-\mathsf{uniq}$$

Empty type

The empty type is total and non-recursive, and shows how the construction of inductive types behaves on the boundaries of the definition.

$$\frac{\Gamma \vdash \mathcal{U}_i : \mathcal{U}_{i+1}}{\Gamma \vdash \mathbf{0} : \mathcal{U}_i}\ \mathbf{0}\!-\mathsf{form}$$

$$\frac{\Gamma \vdash (\Pi C : (\mathbf{0} \to \mathcal{U}_i) \,.\, \Pi e : \mathbf{0} .\, C\, e) : \mathcal{U}_{i+1}}{\Gamma \vdash \mathsf{ind}_{\mathbf{0}} : (\Pi C : (\mathbf{0} \to \mathcal{U}_i) \,.\, \Pi e : \mathbf{0} .\, C\, e)}\ \mathbf{0}\!-\mathsf{elim}$$

$$\frac{\Gamma \vdash e : \mathbf{0}}{\Gamma \vdash \mathsf{ind}_{\mathbf{0}}\ (\lambda x : \mathbf{0} .\, \mathbf{0})\ e \equiv e : \mathbf{0}}\ \mathbf{0}\!-\mathsf{uniq}$$

Unit type

The unit type is total and non-recursive. It is the prototypical example of enumerations.

$$\frac{\Gamma \vdash \mathcal{U}_i : \mathcal{U}_{i+1}}{\Gamma \vdash \mathbf{1} : \mathcal{U}_i}\,\mathbf{1}\text{−form} \qquad \frac{\Gamma \vdash \mathbf{1} : \mathcal{U}_i}{\Gamma \vdash \star : \mathbf{1}}\,\mathbf{1}\text{−intro}$$

$$\frac{\Gamma \vdash (\Pi C : (\mathbf{1} \to \mathcal{U}_i) \,.\, \Pi c_1 : C\,\star .\, \Pi e : \mathbf{1} .\, C\,e) : \mathcal{U}_{i+1}}{\Gamma \vdash \mathsf{ind}_{\mathbf{1}} : (\Pi C : (\mathbf{1} \to \mathcal{U}_i) \,.\, \Pi c_1 : C\,\star .\, \Pi e : \mathbf{1} .\, C\,e)}\,\mathbf{1}\text{−elim}$$

$$\frac{\Gamma \vdash C : \mathbf{1} \to \mathcal{U}_i \quad \Gamma \vdash c_1 : C\,\star \quad \Gamma \vdash \star : \mathbf{1}}{\Gamma \vdash \mathsf{ind}_{\mathbf{1}}\, C\, c_1\, \star \equiv c_1 : C\,\star}\,\mathbf{1}\text{−comp}$$

$$\frac{\Gamma \vdash e : \mathbf{1}}{\Gamma \vdash \mathsf{ind}_{\mathbf{1}}\ (\lambda x : \mathbf{1} .\, \mathbf{1})\ \star\, e \equiv e : \mathbf{1}}\,\mathbf{1}\text{−uniq}$$

Natural numbers

The inductive type of natural numbers is total and recursive. In fact, it is the prototypical example of recursive types.

$$\frac{\Gamma \vdash \mathcal{U}_i : \mathcal{U}_{i+1}}{\Gamma \vdash \mathbb{N} : \mathcal{U}_i}\,\mathbb{N}\text{−form}$$

$$\frac{\Gamma \vdash \mathbb{N} : \mathcal{U}_i}{\Gamma \vdash 0 : \mathbb{N}}\,\mathbb{N}\text{−intro}_1 \qquad \frac{\Gamma \vdash (\Pi n : \mathbb{N} .\, \mathbb{N}) : \mathcal{U}_{i+1}}{\Gamma \vdash \mathsf{succ} : (\Pi n : \mathbb{N} .\, \mathbb{N})}\,\mathbb{N}\text{−intro}_2$$

$$\frac{\begin{array}{l}\Gamma \vdash \big(\Pi C : (\mathbb{N} \to \mathcal{U}_i) \,.\, \Pi c_1 : C\,0 .\\ \qquad \Pi c_2 : (\Pi x : \mathbb{N} .\, \Pi y : C\,x .\, C\,(\mathsf{succ}\,x)) \,.\, \Pi e : \mathbb{N} .\, C\,e\big) : \mathcal{U}_{i+1}\end{array}}{\begin{array}{l}\Gamma \vdash \mathsf{ind}_{\mathbb{N}} : \big(\Pi C : (\mathbb{N} \to \mathcal{U}_i) \,.\, \Pi c_1 : C\,0 .\\ \qquad\qquad \Pi c_2 : (\Pi x : \mathbb{N} .\, \Pi y : C\,x .\, C\,(\mathsf{succ}\,x)) \,.\, \Pi e : \mathbb{N} .\, C\,e\big)\end{array}}\,\mathbb{N}\text{−elim}$$

$$\frac{\begin{array}{ll}\Gamma \vdash C : \mathbb{N} \to \mathcal{U}_i & \Gamma \vdash 0 : \mathbb{N}\\ \Gamma \vdash c_1 : C\,0 & \Gamma \vdash c_2 : (\Pi x : \mathbb{N} .\, \Pi y : C\,x .\, C\,(\mathsf{succ}\,x))\end{array}}{\Gamma \vdash \mathsf{ind}_{\mathbb{N}}\, C\, c_1\, c_2\, 0 \equiv c_1 : C\,0}\,\mathbb{N}\text{−comp}_1$$

$$\frac{\begin{array}{ll}\Gamma \vdash C : \mathbb{N} \to \mathcal{U}_i & \Gamma \vdash \mathsf{succ}\,n : \mathbb{N}\\ \Gamma \vdash c_1 : C\,0 & \Gamma \vdash c_2 : (\Pi x : \mathbb{N} .\, \Pi y : C\,x .\, C\,(\mathsf{succ}\,x))\end{array}}{\Gamma \vdash \mathsf{ind}_{\mathbb{N}}\, C\, c_1\, c_2\, (\mathsf{succ}\,n) \equiv c_2\, n\ (\mathsf{ind}_{\mathbb{N}}\, C\, c_1\, c_2\, n) : C\,(\mathsf{succ}\,n)}\,\mathbb{N}\text{−comp}_2$$

$$\frac{\Gamma \vdash e : \mathbb{N}}{\Gamma \vdash \mathsf{ind}_{\mathbb{N}}\ (\lambda x : \mathbb{N} .\, \mathbb{N})\ 0\ (\lambda x : \mathbb{N} .\, \lambda y : \mathbb{N} .\, \mathsf{succ}\,x)\ e \equiv e : \mathbb{N}}\,\mathbb{N}\text{−uniq}$$

Identity types

The usual notation $a =_A b$ is formally rendered as $= A\,a\,b$, emphasising that identity acts as a functional. The identity inductive type is partial and non-recursive, providing the prototypical example of partial inductive types.

$$\frac{\Gamma \vdash (\Pi A : \mathcal{U}_i .\, \Pi a : A .\, \Pi b : A . \mathcal{U}_i) : \mathcal{U}_{i+1}}{\Gamma \vdash = : (\Pi A : \mathcal{U}_i .\, \Pi a : A .\, \Pi b : A . \mathcal{U}_i)}\; {=}\mathsf{-form}$$

$$\frac{\Gamma \vdash (\Pi A : \mathcal{U}_i .\, \Pi a : A . = A\,a\,a) : \mathcal{U}_{i+1}}{\Gamma \vdash \mathsf{refl} : (\Pi A : \mathcal{U}_i .\, \Pi a : A . = A\,a\,a)}\; {=}\mathsf{-intro}$$

$$\frac{\begin{array}{l}\Gamma \vdash \big(\Pi A : \mathcal{U}_i .\, \Pi a : A .\, \Pi b : A . \\ \qquad \Pi C : (\Pi A : \mathcal{U}_i .\, \Pi a : A .\, \Pi b : A . = A\,a\,b \to \mathcal{U}_i) \,. \\ \qquad \Pi c_1 : (\Pi A : \mathcal{U}_i .\, \Pi a : A .\, C\,A\,a\,a\ (\mathsf{refl}\,A\,a)) \,. \\ \qquad \Pi e : = A\,a\,b .\, C\,A\,a\,b\,e\big) : \mathcal{U}_{i+1}\end{array}}{\begin{array}{l}\Gamma \vdash \mathsf{ind}_= : \big(\Pi A : \mathcal{U}_i .\, \Pi a : A .\, \Pi b : A . \\ \qquad \Pi C : (\Pi A : \mathcal{U}_i .\, \Pi a : A .\, \Pi b : A . = A\,a\,b \to \mathcal{U}_i) \,. \\ \qquad \Pi c_1 : (\Pi A : \mathcal{U}_i .\, \Pi a : A .\, C\,A\,a\,a\ (\mathsf{refl}\,A\,a)) \,. \\ \qquad \Pi e : = A\,a\,b .\, C\,A\,a\,b\,e\big)\end{array}}\; {=}\mathsf{-elim}$$

$$\frac{\begin{array}{l}\Gamma \vdash C : (\Pi A : \mathcal{U}_i .\, \Pi a : A .\, \Pi b : A . = A\,a\,b \to \mathcal{U}_i) \\ \Gamma \vdash c_1 : (\Pi A : \mathcal{U}_i .\, \Pi a : A .\, C\,A\,a\,a\ (\mathsf{refl}\,A\,a)) \\ \Gamma \vdash \mathsf{refl}\,A'\,a' : = A'\,a'\,a'\end{array}}{\Gamma \vdash \mathsf{ind}_=\,A'\,a'\,a'\,C\,c_1\ (\mathsf{refl}\,A'\,a') \equiv c_1\,A'\,a' : C\,A'\,a'\,a'\ (\mathsf{refl}\,A'\,a')}\; {=}\mathsf{-comp}$$

$$\frac{\Gamma \vdash e : = A'\,a'\,b'}{\begin{array}{l}\Gamma \vdash \mathsf{ind}_=\,A'\,a'\,b'\ (\lambda A : \mathcal{U}_i .\, \lambda a : A .\, \lambda b : A .\, \lambda x : = A\,a\,b . = A\,a\,b) \\ \qquad (\lambda A : \mathcal{U}_i .\, \lambda a : A .\, \mathsf{refl}\,A\,a)\ e \equiv e : = A\,a\,b\end{array}}\; {=}\mathsf{-uniq}$$

Well orderings

The W inductive type is total and recursive.

$$\frac{\Gamma \vdash (\Pi A : \mathcal{U}_i .\, \Pi B : (A \to \mathcal{U}_i) . \mathcal{U}_i) : \mathcal{U}_{i+1}}{\Gamma \vdash \mathsf{W} : (\Pi A : \mathcal{U}_i .\, \Pi B : (A \to \mathcal{U}_i) . \mathcal{U}_i)}\; \mathsf{W-form}$$

$$\frac{\begin{array}{l}\Gamma \vdash (\Pi A : \mathcal{U}_i .\, \Pi B : (A \to \mathcal{U}_i) .\, \Pi a : A . \\ \qquad \Pi v : (\Pi y : B\,a . \mathsf{W}\,A\,B) \,. \mathsf{W}\,A\,B) : \mathcal{U}_{i+1}\end{array}}{\begin{array}{l}\Gamma \vdash \mathsf{sup} : (\Pi A : \mathcal{U}_i .\, \Pi B : (A \to \mathcal{U}_i) .\, \Pi a : A . \\ \qquad \Pi v : (\Pi y : B\,a . \mathsf{W}\,A\,B) \,. \mathsf{W}\,A\,B)\end{array}}\; \mathsf{W-intro}$$

$$\frac{\begin{array}{l}\Gamma \vdash \big(\Pi A : \mathcal{U}_i.\, \Pi B : (A \to \mathcal{U}_i).\\ \quad \Pi C : (\Pi A : \mathcal{U}_i.\, \Pi B : (A \to \mathcal{U}_i).\, \mathsf{W}\, A\, B \to \mathcal{U}_i)\,.\\ \quad \Pi c_1 : (\Pi A : \mathcal{U}_i.\, \Pi B : (A \to \mathcal{U}_i).\, \Pi a : A.\\ \quad\quad \Pi v : (\Pi y : B\, a.\, \mathsf{W}\, A\, B)\,.\\ \quad\quad \Pi z : (\Pi y : B\, a.\, C\, A\, B\, (v\, y)).\, C\, A\, B\, (\mathsf{sup}\, A\, B\, a\, v)).\\ \quad \Pi e : \mathsf{W}\, A\, B.\, C\, A\, B\, e\big) : \mathcal{U}_{i+1}\end{array}}{\begin{array}{l}\Gamma \vdash \mathsf{ind}_{\mathsf{W}} : \big(\Pi A : \mathcal{U}_i.\, \Pi B : (A \to \mathcal{U}_i).\\ \quad \Pi C : (\Pi A : \mathcal{U}_i.\, \Pi B : (A \to \mathcal{U}_i).\, \mathsf{W}\, A\, B \to \mathcal{U}_i)\,.\\ \quad \Pi c_1 : (\Pi A : \mathcal{U}_i.\, \Pi B : (A \to \mathcal{U}_i).\, \Pi a : A.\\ \quad\quad \Pi v : (\Pi y : B\, a.\, \mathsf{W}\, A\, B)\,.\\ \quad\quad \Pi z : (\Pi y : B\, a.\, C\, A\, B\, (v\, y)).\, C\, A\, B\, (\mathsf{sup}\, A\, B\, a\, v)).\\ \quad \Pi e : \mathsf{W}\, A\, B.\, C\, A\, B\, e\big)\end{array}}\ \mathsf{W{-}elim}$$

$$\frac{\begin{array}{l}\Gamma \vdash C : (\Pi A : \mathcal{U}_i.\, \Pi B : (A \to \mathcal{U}_i).\, \mathsf{W}\, A\, B \to \mathcal{U}_i)\\ \Gamma \vdash c_1 : \big(\Pi A : \mathcal{U}_i.\, \Pi B : (A \to \mathcal{U}_i).\, \Pi a : A.\\ \quad\quad \Pi v : (\Pi y : B\, a.\, \mathsf{W}\, A\, B)\,.\\ \quad\quad \Pi z : (\Pi y : B\, a.\, C\, A\, B\, (v\, y)).\, C\, A\, B\, (\mathsf{sup}\, A\, B\, a\, v)\,\big)\\ \Gamma \vdash \mathsf{sup}\, A'\, B'\, a'\, v' : \mathsf{W}\, A'\, B'\end{array}}{\begin{array}{l}\Gamma \vdash \mathsf{ind}_{\mathsf{W}}\, A'\, B'\, C\, c_1\, (\mathsf{sup}\, A'\, B'\, a'\, v') \equiv\\ \quad\quad c_1\, A'\, B'\, a'\, v'\, (\lambda y : B' a'.\, \mathsf{ind}_{\mathsf{W}} A'\, B'\, C\, c_1\, (v'\, y)) :\\ \quad C\, A'\, B'\, (\mathsf{sup}\, A'\, B'\, a'\, v')\end{array}}\ \mathsf{W{-}comp}$$

$$\frac{\Gamma \vdash e : \mathsf{W}\, A'\, B'}{\begin{array}{l}\Gamma \vdash \mathsf{ind}_{\mathsf{W}}\, A'\, B'\, (\lambda A : \mathcal{U}_i.\, \lambda B : (A \to \mathcal{U}_i).\, \lambda x : \mathsf{W}\, A\, B.\, \mathsf{W}\, A\, B)\\ \quad \big(\lambda A : \mathcal{U}_i.\, \lambda B : (A \to \mathcal{U}_i).\, \lambda a : A.\, \lambda v : (\Pi y : B\, a.\, \mathsf{W}\, A\, B)\,.\\ \quad\quad \lambda z : (\Pi y : B\, a.\, W\, A\, B).\, \mathsf{sup}\, A\, B\, a\, v\big)\, e \equiv e : \mathsf{W}\, A'\, B'\end{array}}\ \mathsf{W{-}uniq}$$

To check that the present formalisation is equivalent to the usual one is easy by comparing with [77, Chapter 15]. As a side note, not further developed in this dissertation, W-types allow to predicatively simulate the construction of other data types along the lines of [23], but not all of them, as shown in [78].

Chapter 4

Normalisation

This chapter is devoted to prove a normalisation theorem for the system introduced in Chapter 3.

Although Martin-Löf type theory has been deeply and widely studied, a normalisation result for the system used in [95] or, equivalently, for the one used in this thesis is folklore as no full proof can be found in the current literature. Indeed, although Section A.4 of [95] discusses normalisation and its consequences, it does not prove the results, and the shape of the presentation strongly suggests that the normalisation theorem is derived from the one in [69], to which the main text of [95] refers to in the first chapter. The type system of [69] differs from our system (and the one used in [95]) in a few aspects. In particular, we use η-reduction, embodied in the uniqueness rule for the dependent product types; also, the notion of conversion in [69] is much weaker than ours, not admitting conversions under λ. The first aspect is delicate, but the second one is crucial, preventing a direct extension of the proof in [69] to a strong normalisation result for our system. These problems have been addressed in [1] and [2]. There, a normalisation result has been proved for a type system which has full β- and η-conversions, but it has a unique universe, which marks a crucial difference from our system. In fact, an irreducible type A, regarded as term, has the universe as a type; in a system with a cumulative hierarchy of universes a la Russell, A lies in $\mathcal{U}_j$ for every $j \geqslant i$ for some i, that is, A has an infinite number of distinct and irreducible types. This simple fact prevents to affirm that terms have a unique type up to conversions, see Proposition 3.3.21, a property that allows to separate the normalisation of types from the one of terms. Hence, the construction of a universe of all types, see section 3.2 of [2], is inadequate to cope with an infinite hierarchy of universes. We do not know whether the mentioned construction could be appropriately adapted to solve the problem.

The technique called *normalisation-by-evaluation* [20] is the tool to prove the strong normalisation results in [69], [1], and [2]. The great advantage of this technique, originally introduced by Peter Hancock [69], is that it does not require a separate uniqueness result showing that normal forms are unique up to α-conversion, or equivalently, are unique in the de Bruijn representation of terms [41]. The separate uniqueness result is usually an instance of Church-Rosser theorem [15], which poses a difficult combinatorial problem with a complex notion of conversion like the one of Martin-Löf type theory, a fact already remarked in [69].

Our proof uses Girard's technique [43] based on reducibility candidates, suitably enriched with a weak notion of evaluation to deal with dependent types. The normalisation theorem says that every reduction sequence starting from a judgement γ necessarily terminates after a finite number of steps, eventually yielding an irreducible judgement γ' such that $\gamma \equiv \gamma'$ is provable. While [69], [1], and [2] show that normal forms are unique, we prove the uniqueness only up conversions, which is a strictly weaker result with respect to the cited works, but perfectly matches with the semantics we propose in Chapter 5.

4.1 Reductions

The notion of reduction on terms is defined as one expects: the computation and uniqueness rules can be oriented, and the corresponding reductions are thus obtained; a reduction may be performed in the context, in the term(s), or in the type of a judgement, and within another term.

Definition 4.1.1. The one-step reduction $\rhd_1$ is a binary relation on judgements inductively defined as:

- β reduction, $\Gamma \vdash (\lambda x : A.\, b)\, a : T \rhd_1 \Gamma \vdash b[a/x] : T$;
- η reduction, $\Gamma \vdash (\lambda x : A.\, f\, x) : T \rhd_1 \Gamma \vdash f : T$;
- $\tau-\mathsf{comp}$ reduction,
$$\begin{aligned}&\Gamma \vdash \mathsf{ind}_\tau\, T_1 \cdots T_n\, C\, c_1 \cdots c_k\, (\mathbb{K}_i\, T_1 \cdots T_n\, p_1 \cdots p_m) : T \\ \rhd_1\, &\Gamma \vdash c_i\, T_1 \cdots T_n\, p'_1 \cdots p'_m : T \quad ;\end{aligned}$$
- $\tau-\mathsf{uniq}$ reduction, $\Gamma \vdash \mathsf{ind}_\tau\, T_1 \cdots T_n \ldots e : T \rhd_1 \Gamma \vdash e : T$;
- $\Gamma, x : A, \Delta\ \mathsf{ctx} \rhd_1 \Gamma, x : A', \Delta\ \mathsf{ctx}$ if $\Gamma \vdash A : \mathcal{U}_i \rhd_1 \Gamma \vdash A' : \mathcal{U}_i$;
- $\Gamma \vdash a : A \rhd_1 \Gamma' \vdash a : A$ if $\Gamma\ \mathsf{ctx} \rhd_1 \Gamma'\ \mathsf{ctx}$;
- $\Gamma \vdash a \equiv b : A \rhd_1 \Gamma' \vdash a \equiv b : A$ if $\Gamma\ \mathsf{ctx} \rhd_1 \Gamma'\ \mathsf{ctx}$;
- $\Gamma \vdash a : A \rhd_1 \Gamma \vdash a : A'$ if $\Gamma \vdash A : \mathcal{U}_i \rhd_1 \Gamma \vdash A' : \mathcal{U}_i$;
- $\Gamma \vdash a \equiv b : A \rhd_1 \Gamma \vdash a \equiv b : A'$ if $\Gamma \vdash A : \mathcal{U}_i \rhd_1 \Gamma \vdash A' : \mathcal{U}_i$;
- $\Gamma \vdash a \equiv b : A \rhd_1 \Gamma \vdash a' \equiv b : A$ if $\Gamma \vdash a : A \rhd_1 \Gamma \vdash a' : A$;
- $\Gamma \vdash a \equiv b : A \rhd_1 \Gamma \vdash a \equiv b' : A$ if $\Gamma \vdash b : A \rhd_1 \Gamma \vdash b' : A$;
- $\Gamma \vdash (\Pi x : A.\, B) : T \rhd_1 \Gamma \vdash (\Pi x : A'.\, B) : T$ if $\Gamma \vdash A : \mathcal{U}_i \rhd_1 \Gamma \vdash A' : \mathcal{U}_i$;
- $\Gamma \vdash (\Pi x : A.\, B) : T \rhd_1 \Gamma \vdash (\Pi x : A.\, B') : T$ when $\Gamma, x : A \vdash B : \mathcal{U}_i \rhd_1 \Gamma, x : A \vdash B' : \mathcal{U}_i$;
- $\Gamma \vdash (\lambda x : A.\, b) : T \rhd_1 \Gamma \vdash (\lambda x : A'.\, b) : T$ if $\Gamma \vdash A : \mathcal{U}_i \rhd_1 \Gamma \vdash A' : \mathcal{U}_i$;
- $\Gamma \vdash (\lambda x : A.\, b) : T \rhd_1 \Gamma \vdash (\lambda x : A.\, b') : T$ if $\Gamma, x : A \vdash b : B \rhd_1 \Gamma, x : A \vdash b' : B$;
- $\Gamma \vdash f\, a : T \rhd_1 \Gamma \vdash f'\, a : T$ if $\Gamma \vdash f : F \rhd_1 \Gamma \vdash f' : F$;
- $\Gamma \vdash f\, a : T \rhd_1 \Gamma \vdash f\, a' : T$ if $\Gamma \vdash a : A \rhd_1 \Gamma \vdash a' : A$.

The reflexive and transitive closure of $\rhd_1$ is the *reduction* relation $\rhd$, while the reflexive, symmetric, and transitive closure of $\rhd_1$ is the *congruence* or *equivalence* relation $\sim$, also called *conversion.* A judgement γ such that there is no δ for which $\gamma \rhd_1 \delta$ is said to be *irreducible* or *normal.*

Reductions can be localised: if $\Gamma \vdash a : A \rhd \Delta \vdash b : B$, there are $\Gamma \vdash a : A \rhd \Delta \vdash a : A$, $\Gamma \vdash a : A \rhd \Gamma \vdash b : A$, and $\Gamma \vdash a : A \rhd \Gamma \vdash a : B$. These reductions are often abbreviated as $\Gamma \rhd \Delta$, $a \rhd b$, and $A \rhd B$ when the missing pieces are clear.

The empty context $\bullet$ ctx is irreducible; also, if Γ ctx and $\Gamma \vdash A : \mathcal{U}_i$ are irreducible, $\Gamma \vdash x : A$ with x a variable, and $\Gamma \vdash k : A$ with k a constant are irreducible.

The link between $\equiv$ and $\sim$ is that $\Gamma \vdash a \equiv b : A$ if and only if $\Gamma \vdash a : A \sim \Gamma \vdash b : A$. Hence, $\equiv$ is the *internalisation* of $\sim$, thus justifying the claim that judgemental equality is the *computational engine* of Martin-Löf type theory. The proof of the relation between $\equiv$ and $\sim$ is presented in the following.

Proposition 4.1.2. *If* $\pi\colon \Gamma \vdash a \equiv b : A$ *then* $\Gamma \vdash a : A \sim \Gamma \vdash b : A$.

Proof. By Proposition 3.3.8, $\Gamma \vdash a : A$ and $\Gamma \vdash b : A$ are derivable. Then, the statement is proved by induction on π. Considering the last step of π, it must be an instance of one of the following rules:

- $\equiv-\mathsf{refl}$: since $\sim$ is reflexive, the conclusion is immediate;
- $\equiv-\mathsf{sym}$, $\equiv-\mathsf{trans}$: since $\sim$ is symmetric and transitive, the induction hypothesis yields the result;
- $\equiv-\mathsf{subst}-\mathsf{eq}$: by induction hypothesis, $\Gamma \vdash a : B \sim \Gamma \vdash b : B$ and $\Gamma \vdash A : \mathcal{U}_i \sim \Gamma \vdash B : \mathcal{U}_i$ so the result is obtained appropriately composing the reductions in these equivalences of judgements;
- $\Pi-\mathsf{comp}$, $\Pi-\mathsf{uniq}$, $\tau-\mathsf{comp}_i$, $\tau-\mathsf{uniq}$: immediate by Definition 4.1.1;
- in all the other cases, the result follows by induction hypothesis since $\sim$ is a congruence. □

In the opposite direction, the proof is more complex. It depends on two results which have been proved in the semantics: Π is one-to-one, and no product converts to a universe, which are Corollaries 5.3.24 and 5.3.25, respectively.

It is an open question to show that Π is injective without using semantic reasoning. Even with a single universe, see Corollary 15 in [2], this property is proved “outside” the system. Similarly, it is folklore that universes are distinct from other types, and it is perfectly reasonable to think so, as strange phenomena appear. However, we have been unable till now to prove such a result “inside” the system. It is worth remarking that Corollaries 5.3.24 and 5.3.25 are independent from the following propositions: the semantics in Chapter 5 relies on the uniqueness of normal forms up to convertibility, while the following results will show that, in a way, convertibility can be internalised as judgemental equality.

Proposition 4.1.3. *If* $\pi : \Gamma \vdash t : \Pi x : A.\, B$ *then* π *ends with a possibly empty sequence of instances of* $\equiv-\mathsf{subst}$ *after an instance of a rule introducing the main operation in* t.

Proof. By Lemma 3.3.7 and Corollary 5.3.25, such a sequence cannot contain an instance of $\mathcal{U}-\mathsf{cumul}$. □

Proposition 4.1.4. *If* $\Gamma \vdash \lambda x : A.\, b : C$ *then* $\Gamma \vdash \lambda x : A.\, b : \Pi x : A.\, B$ *and* $\Gamma \vdash \Pi x : A.\, B \equiv C : \mathcal{U}_i$.

Proof. Immediate consequence of Proposition 4.1.3. □

Proposition 4.1.5. *If* $\Gamma \vdash \mathsf{ind}_\tau : C$ *then* $\Gamma \vdash \mathsf{ind}_\tau : \Pi x : A.\, B$ *and* $\Gamma \vdash \Pi x : A.\, B \equiv C : \mathcal{U}_i$*. The same holds considering* $\Gamma \vdash \mathsf{ind}_\tau\, a_1 \ldots a_n : C$ *with* $a_1, \ldots, a_n$ *the first* n *parameters except the witness of* τ*, see Section 3.2.*

Proof. Again, immediate consequence of Proposition 4.1.3. □

Proposition 4.1.6. *If* $\Gamma \vdash a : A \rhd_1 \Gamma \vdash b : A$ *then* $\Gamma \vdash a \equiv b : A$*.*

Proof. By Lemma 3.3.7, the derivation of $\Gamma \vdash a : A$ terminates with a possibly empty sequence of instances of $\equiv-\mathsf{subst}$ and $\mathcal{U}-\mathsf{cumul}$ after $\pi : \Gamma \vdash a : T$ in which the last step introduces the main operator of t. Since the sequence affects only the type T, it can be replicated to derive $\Gamma \vdash a \equiv b : A$ from $\Gamma \vdash a \equiv b : T$ by $\equiv-\mathsf{subst}-\mathsf{eq}$ and $\mathcal{U}-\mathsf{cumul}-\mathsf{eq}$. Hence, it suffices to prove $\Gamma \vdash a \equiv b : T$: it follows by induction on the one-step reduction:

- if $\Gamma \vdash (\lambda x : C.\, d)\, e : A \rhd_1 \Gamma \vdash d[e/x] : A$, then π is an instance of $\Pi-\mathsf{elim}$ with premises $\pi_1 : \Gamma \vdash \lambda x : C.\, d : \Pi x : E.\, D$ and $\pi_2 : \Gamma \vdash e : E$, and $T = D[e/x]$. By Proposition 4.1.4, $\Gamma \vdash \lambda x : C.\, d : \Pi x : C.\, F$ and $\Gamma \vdash \Pi x : C.\, F \equiv \Pi x : E.\, D : \mathcal{U}_i$. By Corollary 5.3.24, $\Gamma \vdash C \equiv E : \mathcal{U}_i$ and $\Gamma, x : E \vdash F \equiv D : \mathcal{U}_i$. Also, by construction $\Gamma, x : C \vdash d : F$. By $\equiv-\mathsf{subst}$, $\Gamma \vdash e : C$ so $\Gamma \vdash (\lambda x : C.\, d)\, e \equiv d[e/x] : F[e/x]$ by $\Pi-\mathsf{comp}$. Also, substituting in the proof, $\Gamma \vdash F[e/x] \equiv D[e/x] : \mathcal{U}_i$, thus $\Gamma \vdash a \equiv b : T$ by $\equiv-\mathsf{subst}$.

- if $\Gamma \vdash \lambda x : C.\, f\, x : A \rhd_1 \Gamma \vdash f : A$ then π is an instance of $\Pi-\mathsf{intro}$ with premise $\Gamma, x : C \vdash f\, x : B$. By Lemma 3.3.7, this is an instance of $\Pi-\mathsf{elim}$ followed by a possibly empty sequence θ of instances of $\equiv-\mathsf{subst}$ and $\mathcal{U}-\mathsf{cumul}$. The premises are $\pi_1 : \Gamma, x : E \vdash f : \Pi x : E.\, F$ and $\Gamma, x : E \vdash x : E$. Since $\Gamma \vdash f : A$, $x \notin \mathsf{FV}(f)$ so $\pi_2 : \Gamma \vdash f : \Pi x : E.\, F$ by Proposition 3.3.14 and Theorem 3.3.13. Thus, $\Gamma \vdash \lambda x : E.\, f\, x \equiv f : \Pi x : E.\, F$ by $\equiv-\mathsf{sym}$ and $\Pi-\mathsf{uniq}$. Applying the sequence θ, the result follows.

- if the reduction in the statement is an instance of $\tau-\mathsf{uniq}$, it has the form $\pi_1 : \Gamma \vdash \mathsf{ind}_\tau\, a_1 \ldots a_n\, e : A \rhd_1 \Gamma \vdash e : A$. The last step of π is an instance of $\Pi-\mathsf{elim}$ from $\pi_1 : \Gamma \vdash \mathsf{ind}_\tau\, a_1 \ldots a_n\, e : \Pi x : E.\, D$ and $\Gamma \vdash e : E$. By recursively applying Lemma 3.3.7 and Proposition 4.1.5 to π_1, it follows $E \equiv T \equiv \tau\, a_1 \ldots a_k$, $a_1, \ldots, a_k$ being the formation parameters, and thus $\Gamma \vdash e : T$ by $\equiv-\mathsf{subst}$. Hence $\Gamma \vdash a \equiv b : T$ by $\tau-\mathsf{uniq}$.

- if the induction in the statement is an instance of $\tau-\mathsf{comp}$, the last step of π is an instance of $\Pi-\mathsf{elim}$ from $\pi_1 : \Gamma \vdash \mathsf{ind}_\tau\, a_1 \ldots a_n : \Pi x : E.\, F$ and $\pi_2 : \Gamma \vdash \mathbb{K}_i\, b_1 \ldots b_m : E$. By recursively applying Lemma 3.3.7 and Proposition 4.1.5 to π_1, it follows that $C_1, c_1, \ldots, c_k$ and $\mathbb{K}_i\, b_1 \ldots b_m$ all have the right types via $\equiv-\mathsf{subst}$. Hence $\Gamma \vdash a \equiv b : T$ by $\tau-\mathsf{comp}$.

  To concretely show how the process works, we illustrate the case of $\mathbf{1}-\mathsf{comp}$: $\Gamma \vdash \mathsf{ind}_1 C\, c_1\, * : A \rhd_1 \Gamma \vdash c_1 : A$. Considering π, it is an instance of $\Pi-\mathsf{elim}$ from $\pi_1 : \Gamma \vdash \mathsf{ind}_1\, C\, c_1 : \Pi x_0 : E_0.\, F_0$ and $\pi_2 : \Gamma \vdash * : E_0$. By Lemma 3.3.7 and Proposition 4.1.5 on π_1, $\pi_3 : \Gamma \vdash \mathsf{ind}_1\, C : \Pi x_1 : E_1.\, F_1$, $\pi_4 : \Gamma \vdash c_1 : E_1$ and $F_1[c_1/x_1] \equiv \Pi x_0 : E_0.\, F_0$. Repeating on π_3, $\pi_5 : \Gamma \vdash \mathsf{ind}_1 : \Pi x_2 : E_2.\, F_2$, $\pi_6 : \Gamma \vdash C : E_2$, and $F_2[C/x_1] \equiv \Pi x_1 : E_1.\, F_1$. By the same reasoning, $\pi_7 : \Gamma \vdash \mathsf{ind}_1 : \Pi\alpha : 1 \to \mathcal{U}_i, \beta : \alpha\,*, \gamma : 1.\, \alpha\gamma$, and $\Pi\alpha : 1 \to \mathcal{U}_i, \beta : \alpha\,*, \gamma : 1.\, \alpha\,\gamma \equiv \Pi x_2 : E_2.\, F_2$, so $E_2 \equiv 1 \to \mathcal{U}_i$ and $F_2[C/x_2] \equiv \Pi x_1 : E_1.\, F_1 \equiv \Pi\beta : C\,*, \gamma : 1.\, C\,\gamma$ by Corollary 5.3.24. Thus $\Gamma \vdash C : 1 \to \mathcal{U}_i$ by $\equiv-\mathsf{subst}$. Continuing, $E_1 \equiv C\,*$ and $F_1[c_1/x_1] \equiv \Pi x_0 : E_0.\, F_0 \equiv \Pi\gamma : 1.\, C\,\gamma$, thus $\Gamma \vdash c_1 : C\,*$ by $\equiv-\mathsf{subst}$. Finally, $E_0 \equiv 1$ and $F_0[*/x_0] = T \equiv C\,*$, thus $\Gamma \vdash * : 1$ by $\equiv-\mathsf{subst}$. Hence, $\mathsf{1}-\mathsf{comp}$ yields the result.

- if $\Gamma \vdash f\,c : A \rhd_1 \Gamma \vdash g\,d : A$ then π is an instance of $\Pi-\mathsf{elim}$ with premises $\pi_1 : \Gamma \vdash f : \Pi x : E.\, F$ and $\pi_2 : \Gamma \vdash c : E$, with $T = F[c/x]$. If the one-step reduction happens in f, by induction hypothesis $\pi_3 : \Gamma \vdash f \equiv g : \Pi x : E.\, F$ and $\pi_4 : \Gamma \vdash c \equiv d : E$ by $\equiv-\mathsf{refl}$. On the contrary, if the reduction happens in c, by induction hypothesis $\pi_4 : \Gamma \vdash c \equiv d : E$ and $\pi_3 : \Gamma \vdash f \equiv g : \Pi x : E.\, F$ by $\equiv-\mathsf{refl}$. In both cases, $\Pi-\mathsf{elim}-\mathsf{eq}$ on π_3 and π_4 yields the result.

- if $\Gamma \vdash \lambda x : E.\, e : A \rhd_1 \Gamma \vdash \lambda x : F.\, f : A$ then π is an instance of $\Pi-\mathsf{intro}$ from $\Gamma, x : E \vdash e : B$ and $T = \Pi x : E.\, B$. If the one-step reduction happens in E, by induction hypothesis $\pi_3 : \Gamma \vdash E \equiv F : \mathcal{U}_i$ and $\pi_4 : \Gamma, x : E \vdash e \equiv f : B$ by $\equiv-\mathsf{refl}$. Oppositely, if the reduction happens in e, by induction hypothesis $\pi_4 : \Gamma, x : E \vdash e \equiv f : B$ and $\pi_3 : \Gamma \vdash E \equiv F : \mathcal{U}_i$ by $\equiv-\mathsf{refl}$. In both cases, $\Pi-\mathsf{intro}-\mathsf{eq}$ on π_3 and π_4 yields the result.

- if $\Gamma \vdash \Pi x : C.\, D : A \rhd_1 \Gamma \vdash \Pi x : E.\, F : A$, π is an instance of $\Pi-\mathsf{form}$ from $\pi_1 : \Gamma \vdash C : \mathcal{U}_i$ and $\pi_2 : \Gamma, x : C \vdash D : \mathcal{U}_i$. If the one-step reduction happens in C, by induction hypothesis $\pi_3 : \Gamma \vdash C \equiv E : \mathcal{U}_i$ and $\pi_4 : \Gamma, x : C \vdash D \equiv F : \mathcal{U}_i$ by $\equiv-\mathsf{refl}$. Otherwise the reduction happens in D, thus by induction hypothesis $\pi_4 : \Gamma, x : C \vdash D \equiv F : \mathcal{U}_i$ and $\pi_3 : \Gamma \vdash C \equiv E : \mathcal{U}_i$ by $\equiv-\mathsf{refl}$. In both cases, $\Pi-\mathsf{form}-\mathsf{eq}$ on π_3 and π_4 yields the result.

□

The above result can be extended to arbitrary reductions as follows.

Corollary 4.1.7. *If* $\Gamma \vdash a : A \rhd \Gamma \vdash b : A$ *then* $\Gamma \vdash a \equiv b : A$.

Proof. Expanding the definition, $\Gamma \vdash a_1 : A \rhd_1 \cdots \rhd_1 \Gamma \vdash a_n : A$ with $a_1 = a$ and $a_n = b$. Then by induction on n:

- if $n = 0$ then necessarily $a = b$, so $\equiv-\mathsf{refl}$ applied to the derivation of $\Gamma \vdash a : A$ yields the result;

- if $n > 0$, then by induction hypothesis $\pi_1 : \Gamma \vdash a \equiv a_{n-1} : A$, and by Proposition 4.1.6 $\pi_2 : \Gamma \vdash a_{n-1} \equiv a_n : A$, so $\equiv-\mathsf{trans}$ applied to π_1 and π_2 yields the result. □

The same happens when the reduction is performed in the type.

Proposition 4.1.8. *If* $\Gamma \vdash a : A \rhd_1 \Gamma \vdash a : B$ *then* $\Gamma \vdash A \equiv B : \mathcal{U}_i$ *for some* $i \in \mathbb{N}$.

Proof. By Proposition 3.3.9, $\Gamma \vdash A : \mathcal{U}_i$ so $\Gamma \vdash A : \mathcal{U}_i \rhd_1 \Gamma \vdash B : \mathcal{U}_i$ by Definition 4.1.1. By Proposition 4.1.6, $\Gamma \vdash A \equiv B : \mathcal{U}_i$. □

Proposition 4.1.9. *If* $\Gamma \vdash a : A \rhd \Gamma \vdash a : B$ *then* $\Gamma \vdash A \equiv B : \mathcal{U}_i$ *for some* $i \in \mathbb{N}$.

Proof. Expanding the definition, $\Gamma \vdash a : A_1 \rhd_1 \cdots \rhd_1 \Gamma \vdash a : A_n$ with $A_1 = A$ and $A_n = B$. By Proposition 3.3.9, $\pi' \colon \Gamma \vdash A : \mathcal{U}_i$. Then by induction on n:

- if $n = 0$ then necessarily $A = B$, so $\equiv\!-\mathsf{refl}$ applied to π' yields the result;
- if $n > 0$, then by induction hypothesis $\pi_1 \colon \Gamma \vdash A_1 \equiv A_{n-1} : \mathcal{U}_i$, and by Proposition 4.1.8 $\pi_2 \colon \Gamma \vdash A_{n-1} \equiv A_n : \mathcal{U}_i$, so $\equiv\!-\mathsf{trans}$ applied to π_1 and π_2 yields the result. □

Thus, if the reduction involves both the term and the type, the above results lead to the following corollary.

Corollary 4.1.10. *If* $\Gamma \vdash a : A \rhd \Gamma \vdash b : B$ *then* $\Gamma \vdash a \equiv b : A$ *and* $\Gamma \vdash A \equiv B : \mathcal{U}_i$ *for some* $i \in \mathbb{N}$.

Proof. By Proposition 4.1.9, $\Gamma \vdash A \equiv B : \mathcal{U}_i$. Hence, by $\equiv\!-\mathsf{subst}$ and $\equiv\!-\mathsf{sym}$, $\Gamma \vdash b : A$. Since $\Gamma \vdash a : A \rhd \Gamma \vdash b : A$, by Corollary 4.1.7, $\Gamma \vdash a \equiv b : A$. □

Finally, the claimed relation between $\equiv$ and $\sim$ is obtained.

Corollary 4.1.11. *If* $\Gamma \vdash a : A \sim \Gamma \vdash b : B$ *then* $\Gamma \vdash a \equiv b : A$ *and* $\Gamma \vdash A \equiv B : \mathcal{U}_i$.

Proof. Since $\Gamma \vdash a : A \sim \Gamma \vdash b : B$ can be written as $(\Gamma \vdash a : A) = (\Gamma \vdash a_0 : A_0) \rhd \Gamma \vdash a_1 : A_1 \lhd \Gamma \vdash a_2 : A_2 \rhd \cdots \lhd (\Gamma \vdash a_n : A_n) = (\Gamma \vdash b : B)$, the statement is proved by induction on n:

- if $n = 0$ then $a = b$, so $\Gamma \vdash a \equiv b : A$ and $\Gamma \vdash A \equiv B : \mathcal{U}_i$ by $\equiv\!-\mathsf{refl}$ applied to $\Gamma \vdash a : A$ and $\Gamma \vdash A : \mathcal{U}_i$, as for Proposition 3.3.9;
- if $n = m + 1$ then $\Gamma \vdash a \equiv a_m : A$ and $\Gamma \vdash A \equiv A_m : \mathcal{U}_i$ by induction hypothesis. By Corollary 4.1.10 and possibly an application of $\equiv\!-\mathsf{sym}$, $\Gamma \vdash a_m \equiv a_{m+1} : A_m$ and $\Gamma \vdash A_m \equiv A_{m+1} : \mathcal{U}_i$, thus the result follows by $\equiv\!-\mathsf{subst}\!-\!\mathsf{eq}$ and $\equiv\!-\mathsf{trans}$. □

Corollary 4.1.12. $\Gamma \vdash a \equiv b : A$ *if and only if* $\Gamma \vdash a : A \sim \Gamma \vdash b : A$.

Proof. Evident by Corollary 4.1.11 and Proposition 4.1.2. □

4.2 Strongly normalisable judgements

A judgement is strongly normalisable when every reduction starting from it cannot be indefinitely extended. A formal system is strongly normalisable when every judgement is so.

Definition 4.2.1 (Strongly normalisable)**.** A judgement γ is strongly normalisable when every reduction sequence starting from it cannot be indefinitely extended.

Since the $\rhd$ relation has been defined as the reflexive and transitive closure of the one-step reduction, every reduction is finite. However, the strong normalisation property says that every reduction cannot be indefinitely extended: it is a property of the collection of all the reductions from a judgement, and not a property of a single reduction. This aspect justifies why the induction principle in the proof of the normalisation theorem must have an adequate proof theoretical strength, see also [43].

The following proposition from Girard's [43] characterises strongly normalisable judgements as those having a bound to the length of reductions starting from them:

Proposition 4.2.2. *The judgement γ_0 is strongly normalisable if and only if there is $\nu(\gamma_0) \in \mathbb{N}$ such that $n \leqslant \nu(\gamma_0)$ for every reduction sequence $\gamma_0 \rhd_1 \cdots \rhd_1 \gamma_n$.*

Proof. If $\nu(\gamma_0)$ exists, every reduction sequence from γ_0 is bounded by $\nu(\gamma_0)$, hence γ_0 is strongly normalisable.

Conversely, the reduction sequences from γ_0 can be organised in a directed tree whose nodes are judgements and whose edges are redexes, with γ_0 as root. Since each redex in a judgement δ is a subterm of δ, and the collection of all the subterms of δ is finite, the tree is finitely branching. Also, since γ_0 is strongly normalisable, each branch is finite. Hence the tree is finite by König's Lemma [57], and $\nu(\gamma_0)$ is its height. □

Occasionally, $\Gamma \vdash a : A$ may be said to be strongly normalisable on the context, the term, or the type, when reductions are localised in these pieces. The variants in the above definition and proposition are obvious.

In the following, Definition 4.2.1 and Proposition 4.2.2 are used indifferently to characterise the notion of being strongly normalisable.

We remind that

Fact 4.2.3. *If γ is irreducible, it is strongly normalisable.*

Fact 4.2.4. *If γ is strongly normalisable and every reduction from δ maps into a reduction from γ, then δ is strongly normalisable.*

More in general,

Fact 4.2.5. *If $\gamma_1, \ldots, \gamma_n$ are strongly normalisable*[1] *and every reduction from δ maps into reductions from $\gamma_1, \ldots, \gamma_n$, then δ is strongly normalisable.*

As illustrated in Definition 4.1.1, a reduction is always confined in a term, which may be the type part of a declaration in the context, or the term part of a regular judgement, or one of the terms in a judgemental equality, or the type in a judgement. When whatever term is applied to the term a, the resulting term does not reduce as a whole, a is called *neutral*.

Definition 4.2.6 (Neutral term)**.** A term a in a judgement $\Gamma \vdash a : A$ is *neutral* when $a\,b$ is not a redex for every term b such that $\Gamma \vdash a\,b : B$ for some B.

It is worth noticing how η reduction can be kept behind the scenes when proving strong normalisation. In fact $\Gamma \vdash (\lambda x : A.\, f\, x) : (\Pi x : A.\, B)$ is strongly normalisable if and only if $\Gamma \vdash f : (\Pi x : A.\, B)$ is strongly normalisable, provided $x \notin \mathsf{FV}(f)$, i.e., the second judgement is derivable. The argument is simple: since $\Gamma \vdash (\lambda x : A.\, f\, x) : (\Pi x : A.\, B) \rhd_1 \Gamma \vdash f : (\Pi x : A.\, B)$ via η, every reduction from one side can be mapped into a reduction from the other side. Thus, if one side is strongly normalisable, so is the other side.

[1]Here, the fact that $n \in \mathbb{N}$ is essential

4.3 The normalisation theorem

The normalisation theorem states that Martin-Löf type theory is strongly normalisable. To simplify the proof of the normalisation theorem, a slightly different calculus is used[2]. The only difference is that the $\equiv-\mathsf{subst}$ is replaced by

$$\frac{\Gamma \vdash a : A \quad \Gamma \vdash A \equiv B : \mathcal{U}_i \quad \Gamma \vdash A : \mathcal{U}_i \quad \Gamma \vdash B : \mathcal{U}_i}{\Gamma \vdash a : B}\ \equiv-\mathsf{subst}$$

Because of Proposition 3.3.8, every proof from one calculus can be mapped into an equivalent proof in the other calculus.

The proof of the normalisation theorem follows the guideline of [43], but a number of enhancements have to be made. The first enhancement is in the notion of reducibility candidates, which can be found in the next definition: in this respect, we adopt the notation $t \in R^{\sigma}_{\Gamma}(A)$, even if it is ambiguous, because of its convenience. In fact, in the light of Section 3.4, it should be $\Lambda(t) \in R^{\sigma}_{\Lambda(\Gamma)}(\Lambda(A))$, meaning that t, Γ, and A are reified with σ a delayed substitution. This is precise but cumbersome, and therefore avoided.

Definition 4.3.1 (Reducibility candidates)**.** The sets $R^{\sigma}_{\Gamma}(A)$, whose elements are called *reducibility candidates*, are defined when $\Gamma \vdash A : \mathcal{U}_i$ and σ is a delayed substitution on Γ such that $\sigma(x) \in R^{\sigma\backslash[\sigma(x)/x]}_{\Gamma}(A)$ whenever x is in the domain of σ. The definition is by induction:

- if $\Gamma \vdash A : \mathcal{U}_i \sim \Gamma \vdash x : \mathcal{U}_i$ with x a variable in the domain of σ, then $t\sigma \in R^{\sigma}_{\Gamma}(A)$ if and only if $\Lambda(\Gamma \vdash t : A)$ and
$$t\sigma \in R^{\sigma\backslash[\sigma(x)/x]}_{\Gamma[\sigma(x)/x]}(\sigma(x))\ ;$$
- if $\Gamma \vdash A : \mathcal{U}_i \sim \Gamma \vdash \Pi x : B.\, C : \mathcal{U}_i$ with $\Lambda((\Gamma \vdash \Pi x : B.\, C : \mathcal{U}_i)\sigma)$ strongly normalisable and A does not fall in the previous case, then $t\sigma \in R^{\sigma}_{\Gamma}(A)$ if and only if $\Lambda(\Gamma \vdash t : A)$, $\Lambda((\Gamma \vdash t : A)\sigma)$ is strongly normalisable, and for every B, C, b such that $\Gamma \vdash A : \mathcal{U}_i \sim \Gamma \vdash \Pi x : B.\, C : \mathcal{U}_i$ with $\Lambda((\Gamma \vdash \Pi x : B.\, C : \mathcal{U}_i)\sigma)$ strongly normalisable and $b\sigma \in R^{\sigma}_{\Gamma}(B)$, $(t\, b)\sigma \in R^{\sigma\circ[b/x]}_{\Gamma, x:B}(C)$;
- otherwise the type A is said to be *atomic* and $t\sigma \in R^{\sigma}_{\Gamma}(A)$ if and only if $\Lambda(\Gamma \vdash t : A)$ and $\Lambda((\Gamma \vdash t : A)\sigma)$ is strongly normalisable.

As in [43], product types have to obey the law that their reducibility candidates generate reducibility candidates when applied to reducibility candidates of an appropriate type. However, the induction principle is more complex than the one in [43]: viewing it as a construction process, first the collections R^{σ}_{Γ} for atomic types are generated; then the product types are considered, with B and C atomic; finally, the variable types generate their reducibility candidates from the already constructed sets. The second and the third steps of the process are iterated: in a finite amount of iterations each type A is necessarily considered and its reducibility candidates, if any, are generated.

[2]In principle, this is not necessary. However, it lowers the complexity of induction in Proposition 4.3.2.

Here, a number of subtleties enter the picture. First, a type variable which does not get substituted is atomic. Second, a reducibility candidate t is evaluated, i.e., it does not contain variables in t that σ is going to modify, except for those resulting in the evaluation: this is obtained collecting $\Lambda(t\sigma)$ instead of $\Lambda(t)$. Hence, in the second clause, $(t\,b)\sigma = (t\sigma)\,(b\sigma)$ since $b\sigma \in R^\sigma_\Gamma(B)$, and thus $x \notin \mathsf{FV}(b)$.

A glimpse through the definition shows that R^σ_Γ is empty if A or Γ are not strongly normalisable. The purpose of the normalisation theorem is to show that R^σ_Γ is always defined because Γ and A are always strongly normalisable, and thus R^σ_Γ is empty exactly when A is an empty type and no variable $x : A$ lies in the context. Here, we are considering Γ and A up to the *evaluation* σ: we should say "...if $\Lambda(A\sigma)$ and $\Lambda(\Gamma\sigma)$ are strongly normalisable ...". Informally, but not in the proofs and the definitions, we will often explain the results hiding the role of evaluation.

The first and the second clauses are given up to conversions: "if $A \sim x$ substituted variable ..." or "if $A \sim P$ strongly normalisable product type". Since $\sim$ is transitive, the definition is sound. However, at least in principle, the choice of P is relevant: it is not, ex ante, a quotient on the set of types; it is a partition without an explicit underlying equivalence relation. Of course, ex post, it is easy to prove that it is a quotient, but the proof relies on the normalisation theorem, the final result we are aiming to. This argument justifies why the second clause have to consider all the suitable B, C, and b.

A natural question is why A in R^σ_Γ is not evaluated on σ: this choice allows to reason on types before evaluation, and to consider terms before evaluation. Hence, the overall shape of the argument in [43] can be applied with essentially no changes. To ease notation, we will silently apply reification when needed.

Definition 4.3.1 suggests that $R^\sigma_{\Gamma(A)} = R^\sigma_{\Gamma(B)}$ when $A \sim B$. This property allows to lift (CR1), (CR2), and (CR3) of [43] to types: since $\Gamma \vdash t : A$ implies $\Gamma \vdash A : \mathcal{U}_i$ by Proposition 3.3.9, these properties on terms can be transferred to types. These informal statements are made precise in the following, starting with:

Proposition 4.3.2. *If $t\sigma \in R^\sigma_\Gamma(A)$, $\Gamma \vdash A \equiv B : \mathcal{U}_i$, and $(\Gamma \vdash A \equiv B : \mathcal{U}_i)\sigma$ is strongly normalisable, then $t\sigma \in R^\sigma_\Gamma(B)$.*

Proof. By induction on the type A:

- A atomic: by Corollary 4.1.12, $\Gamma \vdash A : \mathcal{U}_i \sim \Gamma \vdash B : \mathcal{U}_i$, so B is atomic, the other two cases of Definition 4.3.1 being excluded by transitivity of $\sim$. Since $t\sigma \in R^\sigma_\Gamma(A)$, $\Gamma \vdash t : A$ and $(\Gamma \vdash t : A)\sigma$ is strongly normalisable by Definition 4.3.1, so $\Gamma \vdash t : B$ by $\equiv\!-\mathsf{subst}$ and $(\Gamma \vdash t : B)\sigma$ is strongly normalisable since every reduction from it maps into a pair of reductions from $(\Gamma \vdash t : A)\sigma$ and $(\Gamma \vdash A \equiv B : \mathcal{U}_i)\sigma$. Thus $t\sigma \in R^\sigma_\Gamma(B)$ by Definition 4.3.1.

- $A \sim x$ substituted variable: then $t\sigma \in R^\sigma_\Gamma(A)$ implies $\Gamma \vdash t : A$ and $t\sigma \in R^{\sigma\setminus[\sigma(x)/x]}_{\Gamma[\sigma(x)/x]}(\sigma(x))$ by Definition 4.3.1. By Corollary 4.1.12, $B \sim A \sim x$ and $\Gamma \vdash t : B$ by $\equiv\!-\mathsf{subst}$ hence $t\sigma \in R^\sigma_\Gamma(B)$ by Definition 4.3.1.

- A converts to a strongly normalisable product type: as in the atomic case, $\Gamma \vdash t : B$ and $(\Gamma \vdash t : B)\sigma$ is strongly normalisable. For every C, D such that $B \sim (\Pi x : C.\, D)$ with $(\Gamma \vdash (\Pi x : C.\, D) : \mathcal{U}_i)\sigma$ strongly normalisable, and $c\sigma \in R^{\sigma}_{\Gamma}(C)$, since $A \sim (\Pi x : C.\, D)$ by transitivity of $\sim$, $(t\, c)\sigma \in R^{\sigma\circ[c/x]}_{\Gamma,x:C}(C)$ from $t\sigma \in R^{\sigma}_{\Gamma}(A)$ and Definition 4.3.1. Hence, $t\sigma \in R^{\sigma}_{\Gamma}(B)$ by Definition 4.3.1. □

The following Proposition 4.3.3 is (CR1) of Girard of [43], Proposition 4.3.4 is (CR2), and Proposition 4.3.5 is (CR3), slightly adapted to cope with evaluations.

Proposition 4.3.3. *If* $t\sigma \in R^{\sigma}_{\Gamma}(A)$ *then* $\Gamma \vdash t : A$ *and* $(\Gamma \vdash t : A)\sigma$ *is strongly normalisable.*

Proof. By induction[3] on the type A:

- A atomic or A product: immediate by Definition 4.3.1;
- $A \sim x$: from $t\sigma \in R^{\sigma}_{\Gamma}(A)$, $\Gamma \vdash t : A$ and $t\sigma \in R^{\sigma\setminus[\sigma(x)/x]}_{\Gamma[\sigma(x)/x]}(\sigma(x))$ by Definition 4.3.1. Since

$$\Lambda((\Gamma \vdash t : x)\sigma) = \Lambda\left((\Gamma[\sigma(x)/x] \vdash t[\sigma(x)/x] : \sigma(x))(\sigma \setminus [\sigma(x)/x])\right)$$

  by Fact 3.4.9, the right-hand side is strongly normalisable by induction hypothesis, so is the left-hand side. □

Proposition 4.3.4. *If* $t\sigma \in R^{\sigma}_{\Gamma}(A)$ *and* $\Gamma \vdash t : A \rhd \Gamma \vdash t' : A$ *then* $t'\sigma \in R^{\sigma}_{\Gamma}(A)$.

Proof. By induction on the type A:

- A atomic: by Proposition 4.3.3, $\Gamma \vdash t : A$ and $(\Gamma \vdash t : A)\sigma$ is strongly normalisable. By Corollary 4.1.10 and Proposition 3.3.8, $\Gamma \vdash t' : A$. Also $(\Gamma \vdash t' : A)\sigma$ is strongly normalisable since every reduction from it maps into a reduction from $(\Gamma \vdash t : A)\sigma$. Hence $t'\sigma \in R^{\sigma}_{\Gamma}(A)$ by Definition 4.3.1.
- $A \sim x$: by Definition 4.3.1, $\Gamma \vdash t : A$ and $t\sigma \in R^{\sigma'}_{\Gamma[\sigma(x)/x]}(\sigma(x))$. As in the atomic case, $\Gamma \vdash t' : A$. Clearly, $\Gamma[\sigma(x)/x] \vdash t[\sigma(x)/x] : \sigma(x) \rhd \Gamma[\sigma(x)/x] \vdash t'[\sigma(x)/x] : \sigma(x)$, so by Fact 3.4.9 $(t'[\sigma(x)/x])(\sigma \setminus [\sigma(x)/x]) = t'\sigma \in R^{\sigma\setminus[\sigma(x)/x]}_{\Gamma[\sigma(x)/x]}(\sigma(x))$ by induction hypothesis. Thus $t'\sigma \in R^{\sigma}_{\Gamma}(A)$ by Definition 4.3.1.
- A product: as in the atomic case, $\Gamma \vdash t' : A$ and $(\Gamma \vdash t' : A)\sigma$ is strongly normalisable. Let B, C be such that $A \sim \Pi x : B.\, C$ and $(\Gamma \vdash \Pi x : B.\, C : \mathcal{U}_i)\sigma$ is strongly normalisable, and let $b\sigma \in R^{\sigma}_{\Gamma}(B)$. Then the reduction $\Gamma \vdash t : A \rhd \Gamma \vdash t' : A$ maps into a reduction $\Gamma, x : B \vdash t\, b : C[b/x] \rhd \Gamma, x : B \vdash t'\, b : C[b/x]$, both judgements derived by $\Pi-$elim on the weakening of $\Gamma \vdash b : B$, which follows by Proposition 4.3.3. Since $(t\, b)\sigma \in R^{\sigma\circ[b/x]}_{\Gamma,x:B}(C)$ by Definition 4.3.1 on $t\sigma \in R^{\sigma}_{\Gamma}(A)$, $(t'\, b)\sigma \in R^{\sigma\circ[b/x]}_{\Gamma,x:B}(C)$ by induction hypothesis on C. Being B, C and b generic, $t'\sigma \in R^{\sigma}_{\Gamma}(A)$ by Definition 4.3.1. □

Proposition 4.3.5. *If* $(\Gamma \vdash t : A)\sigma$ *is neutral and* $t'\sigma \in R^{\sigma}_{\Gamma}(A)$ *for every* t' *such that* $\Gamma \vdash t : A \rhd_1 \Gamma \vdash t' : A$, *then* $t\sigma \in R^{\sigma}_{\Gamma}(A)$.

Proof. By induction on the type A:

[3]From now on, we abbreviate the specifications of the clauses of Definition 4.3.1 for the sake of conciseness.

- A atomic: by hypothesis, $\Gamma \vdash t : A$. If $(\Gamma \vdash t : A)\sigma$ is irreducible, it is strongly normalisable and so $t \in R^{\sigma}_{\Gamma}(A)$ by Definition 4.3.1. Otherwise, every reduction in the term from $(\Gamma \vdash t : A)\sigma$ passes through some t' as in the statement. By Proposition 4.3.3, $(\Gamma \vdash t' : A)\sigma$ is strongly normalisable. Hence, every reduction from $(\Gamma \vdash t : A)\sigma$ maps into a reduction from $(\Gamma \vdash t' : A)\sigma$ for some t'. Since $(\Gamma \vdash t : A)\sigma$ contains a finite number of redexes, the number of t' to consider is finite, thus $(\Gamma \vdash t : A)\sigma$ is strongly normalisable. Hence $t\sigma \in R^{\sigma}_{\Gamma}(A)$ by Definition 4.3.1.

- $A \sim x$: by hypothesis, $\Gamma \vdash t{:}A$. If $(\Gamma \vdash t{:}A)\sigma$ is irreducible, $t\sigma \in R^{\sigma}_{\Gamma}(A)$ as in the atomic case. Otherwise, by Definition 4.3.1, $t'\sigma \in R^{\sigma\backslash[\sigma(x)/x]}_{\Gamma[\sigma(x)/x]}(\sigma(x))$ for every t' as in the statement. Then, by induction hypothesis, $t\sigma \in R^{\sigma\backslash[\sigma(x)/x]}_{\Gamma[\sigma(x)/x]}(\sigma(x))$, hence $t\sigma \in R^{\sigma}_{\Gamma}(A)$ by Definition 4.3.1.

- A product: let B, C such that $A \sim \Pi x : B.\, C$ with $(\Gamma \vdash \Pi x : B.\, C : \mathcal{U}_i)\sigma$ strongly normalisable, and let $b\sigma \in R^{\sigma}_{\Gamma}(B)$. By Proposition 4.3.3, $\Gamma \vdash b{:}B$ and $(\Gamma \vdash b : B)\sigma$ is strongly normalisable.

  Considering the one-step reductions $\Gamma, x : B \vdash t\, b : C[b/x] \rhd_1 \Gamma, x : B \vdash t'' : C[b/x]$, it follows that $t''\sigma \in R^{\sigma\circ[b/x]}_{\Gamma,x:B}(C)$ by induction on $\nu\,((\Gamma \vdash b : B)\sigma)$:

  - if $\Gamma, x{:}B \vdash t\, b{:}C[b/x] \rhd_1 \Gamma, x{:}B \vdash t'\, b{:}C[b/x]$ then $(t'\, b)\sigma \in R^{\sigma\circ[b/x]}_{\Gamma,x:B}(C)$ by Definition 4.3.1 on $t'\sigma \in R^{\sigma}_{\Gamma}(A)$;
  - if $\Gamma, x{:}B \vdash t\, b{:}C[b/x] \rhd_1 \Gamma, x{:}B \vdash t\, b'{:}C[b'/x]$ and $\Gamma \vdash b{:}B \rhd_1 \Gamma \vdash b'{:}B$ then $b'\sigma \in R^{\sigma}_{\Gamma}(B)$ by Proposition 4.3.4 on $b\sigma \in R^{\sigma}_{\Gamma}(B)$. Since $\nu((\Gamma \vdash b' : B)\sigma) < \nu((\Gamma \vdash b : B)\sigma)$, $(t\, b')\sigma \in R^{\sigma\circ[b/x]}_{\Gamma,x:B}(C)$ by induction hypothesis;
  - being $(\Gamma \vdash t{:}A)\sigma$ neutral, $(t\, b)\sigma$ is not a redex, thus all the possibilities have already been considered.

  Therefore, $(t\, b)\sigma \in R^{\sigma\circ[b/x]}_{\Gamma,x:B}(C)$ for every $b\sigma \in R^{\sigma}_{\Gamma}(B)$ by induction hypothesis on C. Since B, C and b are generic, $\Gamma \vdash t{:}A$ and $(\Gamma \vdash t{:}A)\sigma$ is strongly normalisable as in the atomic case, $t\sigma \in R^{\sigma}_{\Gamma}(A)$ by Definition 4.3.1. □

Since, in R^{σ}_{Γ}, σ may only act on the variables in Γ, it may be the case that $\sigma(x) = x$. The next property says that unevaluated variables like x are reducibility candidates.

Proposition 4.3.6. *If $\Gamma \vdash x : A$ by* Vble*, $\sigma(x) = x$, $(\Gamma\ \mathsf{ctx})\sigma$ is strongly normalisable, and for every $y{:}B$ in Γ such that $\sigma(y) \neq y$, $\sigma(y) \in R^{\sigma\backslash[\sigma(y)/y]}_{\Gamma}(B)$, then $x \in R^{\sigma}_{\Gamma}(A)$.*

Proof. By induction on the type A:

- A atomic: $(\Gamma \vdash x : A)\sigma$ is strongly normalisable since every reduction from it maps into a pair of reductions from $(\Gamma\ \mathsf{ctx})\sigma$. Hence $x\sigma = x \in R^{\sigma}_{\Gamma}(A)$ by Definition 4.3.1.

- $A \sim y$: by induction hypothesis $x \in R^{\sigma\backslash[\sigma(y)/y]}_{\Gamma[\sigma(y)/y]}(\sigma(y))$ since $x \neq y$ because $\sigma(y) \neq y$ while $\sigma(x) = x$. Hence $x\sigma = x \in R^{\sigma}_{\Gamma}(A)$ by Definition 4.3.1.

- A product: as in the atomic case $(\Gamma \vdash x : A)\sigma$ is strongly normalisable. Let $B_1, \ldots, B_n, C$ be such that C is not a product, $A \sim \Pi y_1{:}B_1, \ldots, y_n{:}B_n.\, C$, and $(\Pi y_1 : B_1, \ldots, y_n : B_n.\, C)\sigma$ is strongly normalisable. Also, let $b_1\sigma \in R^{\sigma}_{\Gamma}(B_1)$, $\ldots$, $b_n(\sigma \circ [b_1/y_1, \ldots b_{n-1}/y_{n-1}]) \in R^{\sigma\circ[b_1/y_1,\ldots b_{n-1}/y_{n-1}]}_{\Gamma,y_1:B_1,\ldots,y_{n-1}:B_{n-1}}(B_n)$.

  If z is a new new variable, $z \in R^{\sigma\circ[b_1/y_1,\ldots,b_n/y_n]}_{\Gamma,y_1:B_1,\ldots,y_n:B_n}(C)$ by induction hypothesis on C. Thus $(\Gamma, y_1{:}B_1, \ldots, y_n{:}B_n, z{:}C \vdash z{:}C)(\sigma \circ [b_1/y_1, \ldots, b_n/y_n])$ is strongly normalisable by Proposition 4.3.3. Hence $(\Gamma, y_1{:}B_1, \ldots, y_n{:}B_n \vdash x\, b_1 \cdots b_n : C)(\sigma \circ [b_1/y_1, \ldots, b_n/y_n])$ is strongly normalisable since every reduction from it maps into a finite set of reductions from $(\Gamma \vdash b_1 : B_1)\sigma$, $\ldots$, $(\Gamma, y_1{:}B_1, \ldots, y_{n-1}{:}B_{n-1} \vdash b_n{:}B_n)(\sigma\circ[b_1/y_1, \ldots, b_{n-1}/y_{n-1}])$, which are strongly normalisable by Proposition 4.3.3, and $(\Gamma, y_1 : B_1, \ldots, y_n : B_n, z : C \vdash z : C)(\sigma \circ [b_1/y_1, \ldots, b_n/y_n])$.

  We prove $(x\, b_1 \cdots b_n)(\sigma \circ [b_1/y_1, \ldots, b_n/y_n]) \in R^{\sigma\circ[b_1/y_1,\ldots,b_n/y_n]}_{\Gamma,y_1:B_1,\ldots,y_n:B_n}(C)$ by induction on C: because of the hypotheses, we can reason directly on C, and not up to conversions:

  - C product: by hypothesis, this case does not occur;
  - C atomic: by Definition 4.3.1, the conclusion is immediate;
  - C variable such that $(\sigma \circ [b_1/y_1, \ldots, b_n/y_n])(C) \neq C$: by induction hypothesis,

    $$\begin{aligned}&(x\, b_1 \cdots b_n)(\sigma \circ [b_1/y_1, \ldots, b_n/y_n]) \in \\ &\in R^{(\sigma\circ[b_1/y_1,\ldots,b_n/y_n])\setminus[\sigma(C)/C]}_{(\Gamma,y_1:B_1,\ldots,y_n:B_n)[\sigma(C)/C]}(\sigma(C)) \ .\end{aligned}$$

    Hence, by Definition 4.3.1 the conclusion holds.

  Since b_n is generic, by Definition 4.3.1,

  $$\begin{aligned}&(x\, b_1 \cdots b_{n-1})(\sigma \circ [b_1/y_1, \ldots, b_n/y_n]) \in \\ &\in R^{\sigma\circ[b_1/y_1,\ldots,b_{n-1}/y_{n-1}]}_{\Gamma,y_1:B_1,\ldots,y_{n-1}:B_{n-1}}(\Pi y_n : B_n.\, C) \ .\end{aligned}$$

  Iterating the same argument $(n-1)$ more times,

  $$x(\sigma \circ [b_1/y_1, \ldots, b_n/y_n]) = x \in R^{\sigma}_{\Gamma}(\Pi y_1 : B_1, \ldots, y_n : B_n.\, C) \ .$$

  Hence, $x \in R^{\sigma}_{\Gamma}(A)$ by Proposition 4.3.2. □

The following proposition shows that if an abstraction λ is such that all its β-reductions are reducibility candidates, so is λ. It follows [43]'s presentation.

Proposition 4.3.7. *Let* $(\Gamma\ \mathsf{ctx})\sigma$ *be strongly normalisable. Also, let* $b(\sigma \circ [a/x]) \in R^{\sigma\circ[a/x]}_{\Gamma,x:A}(B)$ *for every* $a\sigma \in R^{\sigma}_{\Gamma}(A)$*. Then* $(\lambda x : A.\, b)\sigma \in R^{\sigma}_{\Gamma}(\Pi x : A.\, B)$.

Proof. Fix $a\sigma \in R^{\sigma}_{\Gamma,x:A}(A)$. First, one proves $((\lambda x : A.\, b)\, a)\sigma \in R^{\sigma\circ[a/x]}_{\Gamma,x:A}(B)$ by induction on

$$\nu\left((\Gamma, x : A \vdash b : B)(\sigma \circ [a/x])\right) + \nu\left((\Gamma \vdash a : A)\sigma\right) \ .$$

This sum is finite since $(\Gamma, x : A \vdash b : B)(\sigma \circ [a/x])$ and $(\Gamma \vdash a : A)\sigma$ are both strongly normalisable by Proposition 4.3.3. Consider the one-step reductions in the term from $(\Gamma \vdash (\lambda x : A.\, b)\, a : B[a/x])\sigma$:

- $(\Gamma \vdash b[a/x] : B[a/x])\sigma = (\Gamma, x : A \vdash b : B)(\sigma \circ [a/x])$ by Fact 3.4.9. Then $b[a/x]\sigma = b(\sigma \circ [a/x]) \in R^{\sigma\circ[a/x]}_{\Gamma,x:A}(B)$ by hypothesis.
- $(\Gamma, x : A \vdash (\lambda x : A.\, b')\, a : B[a/x])\sigma$ with $\Gamma, x : A \vdash b : B \rhd_1 \Gamma, x : A \vdash b' : B$. Then $b'(\sigma \circ [a/x]) \in R^{\sigma\circ[a/x]}_{\Gamma,x:A}(B)$ by Proposition 4.3.4 on $b(\sigma \circ [a/x]) \in R^{\sigma\circ[a/x]}_{\Gamma,x:A}(B)$. Since
$$\nu\left((\Gamma, x : A \vdash b' : B)(\sigma \circ [a/x])\right) < \nu\left((\Gamma, x : A \vdash b : B)(\sigma \circ [a/x])\right) \;,$$
by induction hypothesis $((\lambda x : A.\, b')\, a)\, \sigma \in R^{\sigma\circ[a/x]}_{\Gamma,x:A}(B)$.
- $(\Gamma, x : A \vdash (\lambda x : A.\, b)\, a' : B[a/x])\sigma$ with $\Gamma \vdash a : A \rhd_1 \Gamma \vdash a' : A$: then $a'\sigma \in R^{\sigma}_{\Gamma}(A)$ by Proposition 4.3.4 on $a\sigma \in R^{\sigma}_{\Gamma}(A)$. Since $\nu\left((\Gamma \vdash a' : A)\sigma\right) < \nu\left((\Gamma \vdash a : A)\sigma\right)$, $((\lambda x : A.\, b)\, a')\, \sigma \in R^{\sigma\circ[a/x]}_{\Gamma,x:A}(B)$ by induction hypothesis.
- $(\Gamma, x{:}A \vdash (\lambda x{:}A'.\, b)\, a{:}B[a/x])\sigma$ with $\Gamma \vdash A{:}\mathcal{U}_i \rhd_1 \Gamma \vdash A'{:}\mathcal{U}_i$. Since $a\sigma \in R^{\sigma}_{\Gamma}(A)$, $(\Gamma \vdash a{:}A)\sigma$ is strongly normalisable by Proposition 4.3.3, thus $(\Gamma \vdash A{:}\mathcal{U}_i)\sigma$ is strongly normalisable since every reduction from it maps into a reduction from $(\Gamma \vdash a : A)\sigma$. Also, $\Gamma \vdash A{:}\mathcal{U}_i$ by Proposition 3.3.9 so $A\sigma \in R^{\sigma}_{\Gamma}(\mathcal{U}_i)$ thus $A'\sigma \in R^{\sigma}_{\Gamma}(\mathcal{U}_i)$ by Proposition 4.3.4. Since $\nu\left((\Gamma \vdash a : A')\sigma\right) < \nu\left((\Gamma \vdash a : A)\sigma\right)$, $((\lambda x : A'.\, b)\, a)\sigma \in R^{\sigma\circ[a/x]}_{\Gamma,x:A}(B)$ by induction hypothesis.

Since a is generic and $(\Gamma, x : A \vdash (\lambda x : A.\, b)\, a : B[a/x])(\sigma \circ [a/x])$ is neutral, by Proposition 4.3.5, $((\lambda x : A.\, b)\, a)\sigma \in R^{\sigma\circ[a/x]}_{\Gamma,x:A}(B)$.

By Proposition 4.3.6, $x \in R^{\sigma}_{\Gamma,x:A}(A)$, so by hypothesis there is $b(\sigma \circ [a/x]) \in R^{\sigma\circ[a/x]}_{\Gamma,x:A}(B)$. Hence $\Gamma, x : A \vdash b : B$ and $(\Gamma, x : A \vdash b : B)(\sigma \circ [a/x])$ is strongly normalisable by Proposition 4.3.3. Thus $\Gamma \vdash (\lambda x : A.\, b) : (\Pi x : A.\, B)$ by $\Pi-$intro, and $(\Gamma \vdash (\lambda x : A.\, b) : (\Pi x : A.\, B))\sigma$ is strongly normalisable since every reduction from it maps[4] into a pair of reductions from $(\Gamma, x : A \vdash b : B)(\sigma \circ [a/x])$. Hence $(\lambda x : A.\, b)\sigma \in R^{\sigma}_{\Gamma}(\Pi x : A.\, B)$ by Definition 4.3.1. ☐

The core of the normalisation theorem is Proposition 4.3.10. Informally, given a derivation of $\Gamma \vdash b : B$, it says that when Γ is evaluated on reducibility candidates, the evaluation of b is a reducibility candidate of B.

Since variables, when not evaluated, are reducibility candidates by Proposition 4.3.6, b becomes a reducibility candidate of B under the identity evaluation. This is done in Proposition 4.3.11. Then, by Proposition 4.3.3, every regular judgement is strongly normalisable, hence every judgement is so, proving Theorem 4.3.12.

To lighten notation, $\underline{x} : \underline{A}$ abbreviates $x_1 : A_1, \ldots, x_n : A_n$, $[\underline{a}/\underline{x}]$ abbreviates $[a_1/x_1, \ldots, a_n/x_n]$, and R abbreviates $R^{[\underline{a}/\underline{x}]}_{\underline{x}:\underline{A}}$.

Before the long proof of Proposition 4.3.10, we need a result saying that evaluation preserves reducibility candidates:

Proposition 4.3.8. *Let $y : B$ be in Γ and let $t\sigma \in R^{\sigma}_{\Gamma}(A)$. Then $t\sigma \in R^{\sigma\setminus[\sigma(y)/y]}_{\Gamma[\sigma(y)/y]}(A[\sigma(y)/y])$.*

[4]This map is evident as soon as one disregards the reductions involving a, that is, if a is considered as a constant.

Proof. If $\sigma(y) = y$, the conclusion is the hypothesis. So, assume $\sigma(y) \neq y$. By Definition 4.3.1, $\sigma(y) \in R_{\Gamma}^{\sigma\setminus[\sigma(y)/y]}(B)$. The result follows by induction on the type $A[\sigma(y)/y]$:

- $A[\sigma(y)/y]$ atomic: by Fact 3.4.9, $\Lambda((\Gamma \vdash t : A)\sigma) = \Lambda((\Gamma[\sigma(y)/y] \vdash t[\sigma(y)/y] : A[\sigma(y)/y])(\sigma \setminus [\sigma(y)/y]))$, and it is strongly normalisable by Proposition 4.3.3 on $t\sigma \in R_{\Gamma}^{\sigma}(A)$. Hence $\Lambda(t\sigma) = \Lambda((t[\sigma(y)/y])(\sigma \setminus [\sigma(y)/y])) \in R_{\Gamma[\sigma(y)/y]}^{\sigma\setminus[\sigma(y)/y]}(A[\sigma(y)/y])$ by Definition 4.3.1.

- $A[\sigma(y)/y]$ product: as in the atomic case, $\Lambda((\Gamma[\sigma(y)/y] \vdash t[\sigma(y)/y] : A[\sigma(y)/y])(\sigma \setminus [\sigma(y)/y]))$ is strongly normalisable. Let $A[\sigma(y)/y]$ be equivalent to a strongly normalisable product. There are two cases: either $A \sim y$ and $A[\sigma(y)/y] \sim \sigma(y)$, or $A \sim \Pi x : A'.\, A''$ such that $\Pi x : A'[\sigma(y)/y].\, A''[\sigma(y)/y]$ is strongly normalisable.

  The former case implies $t\sigma \in R_{\Gamma[\sigma(y)/y]}^{\sigma\setminus[\sigma(y)/y]}(\sigma(y))$ by Definition 4.3.1, so $t\sigma \in R_{\Gamma[\sigma(y)/y]}^{\sigma\setminus[\sigma(y)/y]}(A[\sigma(y)/y])$ by Proposition 4.3.2.

  The latter case implies $(t\,a)\sigma \in R_{\Gamma,x:A'}^{\sigma\circ[a/x]}(A'')$ for every $a\sigma \in R_{\Gamma}^{\sigma}(A')$ by Definition 4.3.1. By induction hypothesis on A' and A'', for every $a\sigma \in R_{\Gamma[\sigma(y)/y]}^{\sigma\setminus[\sigma(y)/y]}(A'[\sigma(y)/y])$, $(t\,a)\sigma \in R_{(\Gamma,x:A')[\sigma(y)/y]}^{(\sigma\circ[a/x])\setminus[\sigma(y)/y]}(A''[\sigma(y)/y])$ or, equivalently, by Proposition 3.4.10, $(t\,a)\sigma \in R_{\Gamma[\sigma(y)/y],x:A'[\sigma(y)/y]}^{(\sigma\setminus[\sigma(y)/y])\circ[a/x]}(A''[\sigma(y)/y])$. Hence $t\sigma \in R_{\Gamma[\sigma(y)/y]}^{\sigma\setminus[\sigma(y)/y]}(A''[\sigma(y)/y])$ by Definition 4.3.1.

- $A[\sigma(y)/y]$ variable: there are two cases: either $A \sim y$ and $\sigma(y)$ is a variable, or $A \sim z$ for some $z : C$ occurring in Γ and in the domain of σ.

  In the former case $t\sigma \in R_{\Gamma[\sigma(y)/y]}^{\sigma\setminus[\sigma(y)/y]}(\sigma(y))$ by Definition 4.3.1 on $t\sigma \in R_{\Gamma}^{\sigma}(A)$. Hence $t\sigma \in R_{\Gamma[\sigma(y)/y]}^{\sigma\setminus[\sigma(y)/y]}(A[\sigma(y)/y])$ by Proposition 4.3.2 since $A[\sigma(y)/y] \sim y[\sigma(y)/y] = \sigma(y)$.

  In the latter case $t\sigma \in R_{\Gamma[\sigma(z)/z]}^{\sigma\setminus[\sigma(z)/z]}(\sigma(z))$ by Definition 4.3.1. By induction hypothesis on $\sigma(z)$,

  $$t\sigma \in R_{\Gamma[\sigma(z)/z,\sigma(y)/y]}^{(\sigma\setminus[\sigma(z)/z])\setminus[\sigma(y)/y]}(\sigma(z)[\sigma(y)/y]) \ .$$

  Equivalently by Proposition 3.4.11,

  $$t\sigma \in R_{\Gamma[\sigma(y)/y,\sigma(z)/z]}^{(\sigma\setminus[\sigma(y)/y])\setminus[\sigma(z)/z]}(\sigma(z)[\sigma(y)/y]) \ .$$

  Since $A[\sigma(y)/y] \sim z[\sigma(y)/y] = z$, $t\sigma \in R_{\Gamma[\sigma(y)/y]}^{\sigma\setminus[\sigma(y)/y]}(A[\sigma(y)/y])$ by Definition 4.3.1. □

Corollary 4.3.9. *Let $\Gamma, x : A \vdash t : B$ and $a\sigma \in R_{\Gamma}^{\sigma}(A)$. If $t(\sigma \circ [a/x]) \in R_{\Gamma,x:A}^{\sigma\circ[a/x]}(B)$, then $t(\sigma \circ [a/x]) \in R_{\Gamma}^{\sigma}(B[a/x])$.*

Proof. Immediate instance of Proposition 4.3.8. □

Proposition 4.3.10. *Let π: $\underline{x} : \underline{A} \vdash b : B$ and $a_j[\underline{a}/\underline{x}] \in R(A_j)$ for every $1 \leqslant j \leqslant n$. Then $b[\underline{a}/\underline{x}] \in R(B)$.*

Proof. First, $(\underline{x} : \underline{A})[\underline{a}/\underline{x}]$ is strongly normalisable since every reduction from it maps into a reduction from $(\underline{x} : \underline{A} \vdash a_j : A_j)[\underline{a}/\underline{x}]$ for any $1 \leqslant j \leqslant n$, which is strongly normalisable by Proposition 4.3.2 on the hypothesis. In the limit case $n = 0$, the context is empty, i.e., $\bullet$ ctx which is irreducible and thus strongly normalisable.

We remind that, if $\leqslant$ is a well-ordering, it holds the following induction principle: if $P(e)$ holds for every $e \leqslant e'$ then $P(e')$ holds implies that $P(x)$ holds for every x. In our case $\leqslant$ is given by $\sqcup_\tau <_\tau$ for every inductive type τ: $\leqslant$ is obviously a well-ordering, see Proposition 18 in [19]. The proof is by an external induction on $\leqslant$ and an internal induction on π. In essence, this double induction says that we first prove the statement for all the proofs not involving "difficult" instances of inductive terms, and then we move to more complex ones. The only point in which the external induction is needed is the $\tau-\mathsf{elim}$ case of the internal induction when τ is a recursive type.

- (Vble): $b = x_j$, $B = A_j$, $1 \leqslant j \leqslant n$ from the premise $\underline{x} : \underline{A}$ ctx. Thus $b[\underline{a}/\underline{x}] = a_j[\underline{a}/\underline{x}]$ and $B[\underline{a}/\underline{x}] = A_j[\underline{a}/\underline{x}]$. Hence $b[\underline{a}/\underline{x}] \in R(B)$ by hypothesis.

- ($\equiv-\mathsf{subst}$): the premises are $\underline{x} : \underline{A} \vdash b : C$, $\underline{x} : \underline{A} \vdash C \equiv B : \mathcal{U}_i$, $\underline{x} : \underline{A} \vdash C : \mathcal{U}_i$, and $\underline{x}:\underline{A} \vdash B:\mathcal{U}_i$. By induction hypothesis $b[\underline{a}/\underline{x}] \in R(C)$, $C[\underline{a}/\underline{x}] \in R(\mathcal{U}_i)$, and $B[\underline{a}/\underline{x}] \in R(\mathcal{U}_i)$, so $(\underline{x} : \underline{A} \vdash C : \mathcal{U}_i)[\underline{a}/\underline{x}]$ and $(\underline{x} : \underline{A} \vdash B : \mathcal{U}_i)[\underline{a}/\underline{x}]$ are strongly normalisable by Proposition 4.3.3. Thus $(\underline{x} : \underline{A} \vdash C \equiv B : \mathcal{U}_i)[a/x]$ is strongly normalisable, hence $b[\underline{a}/\underline{x}] \in R(B)$ by Proposition 4.3.2.

- ($\mathcal{U}-\mathsf{intro}$): $b = \mathcal{U}_i$, $B = \mathcal{U}_{i+1}$ from the premise $\underline{x} : \underline{A}$ ctx. So $b[\underline{a}/\underline{x}] = \mathcal{U}_i$ and $B[\underline{a}/\underline{x}] = \mathcal{U}_{i+1}$. Since $(\underline{x} : \underline{A} \vdash \mathcal{U}_i : \mathcal{U}_{i+1})[\underline{a}/\underline{x}]$ is strongly normalisable when $(\underline{x} : \underline{A}$ $\mathsf{ctx})$ is so, see the beginning of this proof, $b[\underline{a}/\underline{x}] \in R(B)$ by Definition 4.3.1.

- ($\mathcal{U}-\mathsf{cumul}$): $B = \mathcal{U}_{i+1}$ from the premise $\underline{x} : \underline{A} \vdash B : \mathcal{U}_i$. By induction hypothesis $b[\underline{a}/\underline{x}] \in R(\mathcal{U}_i)$, thus $(\underline{x}:\underline{A} \vdash b:\mathcal{U}_i)[\underline{a}/\underline{x}]$ is strongly normalisable by Proposition 4.3.3. Then $(\underline{x} : \underline{A} \vdash b : \mathcal{U}_{i+1})[\underline{a}/\underline{x}]$ is strongly normalisable since every reduction from it maps into a reduction from $(\underline{x}:\underline{A} \vdash b:\mathcal{U}_i)[\underline{a}/\underline{x}]$, hence $b[\underline{a}/\underline{x}] \in R(\mathcal{U}_{i+1})$ by Definition 4.3.1.

- ($\Pi-\mathsf{form}$): $b = (\Pi x : C.\, D)$, $B = \mathcal{U}_i$ from the premises $\underline{x} : \underline{A} \vdash C : \mathcal{U}_i$ and $\underline{x} : \underline{A}, x : C \vdash D : \mathcal{U}_i$. By induction hypothesis, $C[\underline{a}/\underline{x}] \in R(\mathcal{U}_i)$ and $D[\underline{a}/\underline{x}, c/x] \in R^{[\underline{a}/\underline{x},c/x]}_{\underline{x}:\underline{A},x:C}(\mathcal{U}_i)$ for every $c[\underline{a}/\underline{x}] \in R(C)$, so $(\underline{x} : \underline{A} \vdash C : \mathcal{U}_i)[\underline{a}/\underline{x}]$ and $(\underline{x} : \underline{A}, x : C \vdash D : \mathcal{U}_i)[\underline{a}/\underline{x}, c/x]$ are strongly normalisable by Proposition 4.3.3. Thus $(\underline{x} : \underline{A} \vdash (\Pi x : C.\, D) : \mathcal{U}_i)[\underline{a}/\underline{x}]$ is strongly normalisable since every reduction from it maps into a pair of reductions from $(\underline{x} : \underline{A} \vdash C : \mathcal{U}_i)[\underline{a}/\underline{x}]$ and $(\underline{x} : \underline{A}, x : C \vdash D : \mathcal{U}_i)[\underline{a}/\underline{x}]$, the latter being strongly normalisable posing $c = x$ by Proposition 4.3.6. Hence $(\Pi x : C.\, D)[\underline{a}/\underline{x}] \in R(\mathcal{U}_i)$ by Definition 4.3.1.

- ($\Pi-\mathsf{intro}$): $b = (\lambda x : C.\, d)$, $B = (\Pi x : C.\, D)$ from the premise $\underline{x} : \underline{A}, x : C \vdash d : D$. By induction hypothesis $d[\underline{a}/\underline{x}, c/x] \in R^{[\underline{a}/\underline{x},c/x]}_{\underline{x}:\underline{A},x:C}(D)$ for every $c \in R(C)$. Hence $b[\underline{a}/\underline{x}] \in R(B)$ by Proposition 4.3.7.

- ($\Pi-\mathsf{elim}$): $b = f\, c$, $B = D[c/x]$ from the premises $\underline{x} : \underline{A} \vdash f : (\Pi x : C.\, D)$ and $\underline{x} : \underline{A} \vdash c : C$. By induction hypothesis, $f[\underline{a}/\underline{x}] \in R(\Pi x : C.\, D)$ and

$c[\underline{a}/\underline{x}] \in R(C)$. Hence $(f\,c)[\underline{a}/\underline{x}, c/x] \in R^{[\underline{a}/\underline{x},c/x]}_{\underline{x}:\underline{A},x:C}(D)$ by Definition 4.3.1 on f. Hence, by Corollary 4.3.9, $(f\,c)[\underline{a}/\underline{x}] \in R(D[c/x])$.

- (τ–form): b is a constant, $B = (\Pi(y:F)_m.\mathcal{U}_i)$ from the premise $\underline{x}:\underline{A} \vdash B : \mathcal{U}_{i+1}$. By induction hypothesis $B[\underline{a}/\underline{x}] \in R(\mathcal{U}_{i+1})$, so $(\underline{x}:\underline{A} \vdash B:\mathcal{U}_{i+1})[\underline{a}/\underline{x}]$ is strongly normalisable by Proposition 4.3.3 and thus $(\underline{x}:\underline{A} \vdash b:B)[\underline{a}/\underline{x}]$ is strongly normalisable since every reduction from it maps into a reduction from $(\underline{x}:\underline{A} \vdash B:\mathcal{U}_{i+1})[\underline{a}/\underline{x}]$.

  Let $f_j[\underline{a}/\underline{x}, f_1/y_1, \ldots, f_{j-1}/y_{j-1}] \in R^{[\underline{a}/\underline{x},f_1/y_1,\ldots,f_{j-1}/y_{j-1}]}_{\underline{x}:\underline{A},\underline{y}:\underline{F}}(F_j)$ for every $1 \leqslant j \leqslant m$. By Π–elim and Theorem 3.3.13, $\underline{x}:\underline{A}, \underline{y}:\underline{F} \vdash b\,f_1 \cdots f_m : \mathcal{U}_i$. Also $(\underline{x}:\underline{A}, \underline{y}:\underline{F} \vdash b\,f_1 \cdots f_m : \mathcal{U}_i)[\underline{a}/\underline{x}, \underline{f}/\underline{y}]$ is strongly normalisable since every reduction from it maps into a finite set of reductions from $(\underline{x}:\underline{A}, \underline{y}:\underline{F} \vdash f_j : F_j)[\underline{a}/\underline{x}, f_1/y_1, \ldots, f_{j-1}/y_{j-1}]$, $1 \leqslant j \leqslant m$, which are strongly normalisable by Proposition 4.3.3. Thus, by Definition 4.3.1,

$$(b\,f_1 \cdots f_m)[\underline{a}/\underline{x}, \underline{f}/\underline{y}] \in R^{[\underline{a}/\underline{x},\underline{f}/\underline{y}]}_{\underline{x}:\underline{A},\underline{y}:\underline{F}}(\mathcal{U}_i) \ .$$

  Being f_m generic,

$$\begin{aligned}&(b\,f_1 \cdots f_{m-1})[\underline{a}/\underline{x}, f_1/y_1, \ldots, f_{m-1}/y_{m-1}] \in \\ &\in R^{[\underline{a}/\underline{x},f_1/y_1,\ldots,f_{m-1}/y_{m-1}]}_{\underline{x}:\underline{A},y_1:F_1,\cdots,y_{m-1}:F_{m-1}}(\Pi y_m : F_m.\mathcal{U}_i) \ .\end{aligned}$$

  Iterating $m-1$ more times, $b[\underline{a}/\underline{x}] \in R(B)$.

- (τ–intro): b is a constant, $B = (\Pi(y:F)_k.\tau\,y_1 \ldots y_m)$, with $m \leqslant k$ and omitting the irrelevant details, from the premise $\underline{x}:\underline{A} \vdash B : \mathcal{U}_i$.

  By induction hypothesis $B[\underline{a}/\underline{x}] \in R(\mathcal{U}_i)$, so $(\underline{x}:\underline{A} \vdash b:B)[\underline{a}/\underline{x}]$ is strongly normalisable since every reduction from it maps into a reduction from $(\underline{x}:\underline{A} \vdash B:\mathcal{U}_i)[\underline{a}/\underline{x}]$, which is strongly normalisable by Proposition 4.3.3.

  Let $f_j[\underline{a}/\underline{x}, f_1/y_1, \ldots, f_{j-1}/y_{j-1}] \in R^{[\underline{a}/\underline{x},f_1/y_1,\ldots,f_{j-1}/y_{j-1}]}_{\underline{x}:\underline{A},\underline{y}:\underline{F}}$ for every $1 \leqslant j \leqslant k$. By Π–elim and Theorem 3.3.13, $\underline{x}:\underline{A}, \underline{y}:\underline{F} \vdash b\,f_1 \cdots f_k : \tau\,f_1 \ldots f_m$. Also, every reduction from

$$(\underline{x}:\underline{A}, \underline{y}:\underline{F} \vdash b\,f_1 \cdots f_k : \tau\,f_1 \ldots f_m)[\underline{a}/\underline{x}, \underline{f}/\underline{y}]$$

  is strongly normalisable since every reduction from it maps into a finite set of reductions from $(\underline{x}:\underline{A}, \underline{y}:\underline{F} \vdash f_j : F_j)[\underline{a}/\underline{x}, f_1/y_1, \ldots, f_{j-1}/y_{j-1}]$, $1 \leqslant j \leqslant k$, which are strongly normalisable by Proposition 4.3.3. Thus $(b\,f_1 \ldots f_k)[\underline{a}/\underline{x}, \underline{f}/\underline{y}] \in R^{[\underline{a}/\underline{x},\underline{f}/\underline{y}]}_{\underline{x}:\underline{A},\underline{y}:\underline{F}}(\tau\,f_1 \ldots f_m)$ by Definition 4.3.1. Iterating Definition 4.3.1 k times, as in the previous case, $b[\underline{a}/\underline{x}] \in R(B)$.

- (τ–elim): b is a constant from the premise

$$\begin{aligned}\underline{x}:\underline{A} \vdash (\Pi(y:F)_m, C &: (\Pi(y:F)_m.\tau\,y_1 \cdots y_m \to \mathcal{U}_i), \\ &c_1 : (\Pi(y:F)_{m_1}, (z:I)_{k_1}. \\ &\qquad C\,y_1 \cdots y_{m_1}\,(\mathbb{K}_1\,y_1 \cdots y_{m_1}\,z_1 \cdots z_{k_1})), \\ &\ldots, \\ &c_h : (\Pi(y:F)_{m_h}, (z:I)_{k_h}.\end{aligned}$$

$$C\,y_1\,\cdots\,y_{m_h}\,(\mathbb{K}_h\,y_1\,\cdots\,y_{m_g}\,z_1\,\cdots\,z_{k_h})),$$
$$e:\tau\,y_1\,\cdots\,y_m.\,C\,y_1\,\cdots\,y_m\,e):\mathcal{U}_{i+1}$$

with B the long term in the premise.

Since $B[\underline{a}/\underline{x}] \in R(\mathcal{U}_i)$ by induction hypothesis, $(\underline{x}:\underline{A} \vdash B:\mathcal{U}_i)[\underline{a}/\underline{x}]$ is strongly normalisable by Proposition 4.3.3. Thus $(\underline{x}:\underline{A} \vdash b:B)[\underline{a}/\underline{x}]$ is strongly normalisable since every reduction from it maps into a reduction from $(\underline{x}:\underline{A} \vdash B:\mathcal{U}_i)[\underline{a}/\underline{x}]$.

Let $\Gamma = \underline{x}:\underline{A}, \underline{y}:\underline{F}, C:T_C, c_1:T_{c_1},\ldots,c_h:T_{c_h}, e:T_e$, let

$$f_j[\underline{a}/\underline{x}, f_1/y_1,\ldots,f_{j-1}/y_{j-1}] \in R_\Gamma^{[\underline{a}/\underline{x},f_1/y_1,\ldots,f_{j-1}/y_{j-1}]}(F_j)$$

for every $1 \leqslant j \leqslant m$, let $C^*[\underline{a}/\underline{x}, \underline{f}/\underline{y}] \in R_\Gamma^{[\underline{a}/\underline{x},\underline{f}/\underline{y}]}(T_C)$, let

$$c_j^*[\underline{a}/\underline{x}, \underline{f}/\underline{y}, C^*/C, c_1^*/c_1,\ldots,c_{j-1}^*/c_{j-1}] \in$$
$$\in R_\Gamma^{[\underline{a}/\underline{x},\underline{f}/\underline{y},C^*/C,c_1^*/c_1,\ldots,c_{j-1}^*/c_{j-1}]}(T_{c_j})$$

for every $1 \leqslant j \leqslant h$, and let

$$e^*[\underline{a}/\underline{x}, \underline{f}/\underline{y}, C^*/C, \underline{c}^*/\underline{x}] \in R_\Gamma^{[\underline{a}/\underline{x},\underline{f}/\underline{y},C^*/C,\underline{c}^*/\underline{x}]}(T_e)\ .$$

By $\Pi-\mathsf{elim}$ and Theorem 3.3.13,

$$\underline{x}{:}\underline{A}, \underline{y}{:}\underline{F}, C{:}T_C, \underline{c}{:}\underline{T_c}, e{:}T_e \vdash b\,f_1\,\cdots\,f_m\,C^*\,c_1^*\,\ldots\,c_h^*\,e^*{:}C^*\,f_1\,\ldots\,f_m\,e^*\ .$$

Consider a reduction in the term from

$$(\underline{x}:\underline{A}, \underline{y}:\underline{F}, C:T_C, \underline{c}:\underline{T}_c, e:T_e \vdash b\,f_1\ldots f_m\,C^*\,c_1^*\ldots c_h^*\,e^*:$$
$$C^*\,f_1\ldots f_m\,e^*)[\underline{a}/\underline{x}, \underline{f}/\underline{y}, C^*/C, \underline{c}^*/c, e^*/e]\ .$$

If it does not reduce the term as a whole, i.e., if it does not perform a $\tau-\mathsf{comp}$ or $\tau-\mathsf{uniq}$ reduction spanning $b\,f_1\ldots f_n\,C^*\,c_1^*\ldots c_h^*\,e^*$, then the reduction can be mapped into $m+h+2$ reductions from $\underline{f}, C^*, \underline{c}^*, e^*$, which are strongly normalisable by Proposition 4.3.3. Thus the reduction cannot be indefinitely extended. If the reduction performs a $\tau-\mathsf{uniq}$ step involving the whole term, the reduction can be mapped as before, and the result of the $\tau-\mathsf{uniq}$ step gets mapped into a reduction from e^*, which is strongly normalisable. Hence, the reduction cannot be indefinitely extended. If the reduction performs a $\tau-\mathsf{comp}$ step on the whole term, the reduction can be mapped as above up to the step, yielding

$$(\underline{x}:\underline{A} \vdash c'\,f_1'\ldots f_m' p_1\ldots p_k : C'\,f_1'\ldots f_m'\,e')[\underline{a}/\underline{x}]\ .$$

If τ is non-recursive, since the terms $c', f_1',\ldots,f_m', p_1,\ldots,p_k$ are strongly normalisable because they come from reducibility candidates, an immediate induction on $m+k$ shows that this term is strongly normalisable since c' is a reducibility candidate of a non-atomic type, and by Definition 4.3.1 the result follows. If τ is a recursive type, the same reasoning applies since the occurrence of ind_τ is simpler with respect to $\leqslant$ and so it is a reducibility candidate by external induction hypothesis. To help understanding consider the $\mathbb{N}$ type: if $\Delta \vdash \mathsf{ind}_{\mathbb{N}}\, C^*\, c_1^*\, c_2^*\, (\mathsf{succ}\, n) : C^*(\mathsf{succ}\, n)$ is the judgement under analysis, it may reduce to $\Delta \vdash c_2^*\, n\, (\mathsf{ind}_{\mathbb{N}}\, C^*\, c_1^*\, c_2^*\, n) : C^*\, (\mathsf{succ}\, n)$. It is clear that C^*, c_1^*, c_2^*, $\mathsf{succ}\, n$, and so n are reducibility candidates. Since $n \leqslant \mathsf{succ}\, n$, the external induction hypothesis says that $\mathsf{ind}_{\mathbb{N}}\, C^*\, c_1^*\, c_2^*\, n$ is a reducibility candidate, showing that the reduced term is strongly normalisable. Hence, every reduction is bounded.

Since the number of redexes in the initial term is finite, the maximum of $\nu(\gamma)$, with γ ranging on the results of one-step reductions in the term, is defined. Thus, by Proposition 4.2.2, the judgement is strongly normalisable on the term. Since the type is strongly normalisable, too, and the context, once substituted, is empty, the whole judgement is strongly normalisable.

Since $(C^*\, f_1\, \dots\, f_m\, e^*)[\underline{a}/\underline{x}, \underline{f}/\underline{y}, C^*/C, \underline{c}^*/c, e^*/e]$ is atomic,

$$\begin{aligned}&(b\, f_1\, \cdots\, f_m\, C^*\, c_1^*\, \dots\, c_h^*\, e^*)[\underline{a}/\underline{x}, \underline{f}/\underline{y}, C^*/C, \underline{c}^*/c, e^*/e] \in \\ &\in R_{\Gamma}^{[\underline{a}/\underline{x}, \underline{f}/\underline{y}, C^*/C, \underline{c}^*/c, e^*/e]}(C^*\, f_1\, \dots\, f_m\, e^*)\end{aligned}$$

by Definition 4.3.1. Hence, by the same definition,

$$\begin{aligned}&(b\, f_1\, \cdots\, f_m\, C^*\, c_1^*\, \dots\, c_h^*)[\underline{a}/\underline{x}, \underline{f}/\underline{y}, C^*/C, \underline{c}^*/c] \in \\ &\in R_{\underline{x}:\underline{A}, \underline{y}:\underline{F}, C:T_C, c_1:T_{c_1}, \dots, c_h:T_{c_h}}^{[\underline{a}/\underline{x}, \underline{f}/\underline{y}, C^*/C, \underline{c}^*/c]}(\Pi e : T_e.\, C^*\, f_1\, \dots\, f_m\, e^*)\ .\end{aligned}$$

Iterating Definition 4.3.1 $h + m + 1$ more times, $b[\underline{a}/\underline{x}] \in R(B)$. □

Proposition 4.3.11. *If $\underline{x} : \underline{A} \vdash b : B$ then $b \in R_{\underline{x}:\underline{A}}^{\mathsf{id}}(B)$ with id the identity substitution, $\mathsf{id}(x) = x$ for every variable x.*

Proof. By induction on the length n of the context, to show it is strongly normalisable:

- $n = 0$: $\bullet$ ctx is irreducible, thus strongly normalisable;
- $n > 0$: then $x_1 : A_1, \dots, x_{n-1} : A_{n-1}$ ctx is strongly normalisable by induction hypothesis, and $x_1 : A_1, \dots, x_{n-1} : A_{n-1} \vdash A_n : \mathcal{U}_i$. By Proposition 4.3.6, $x_j \in R_{\underline{x}:\underline{A}}^{\mathsf{id}}(A_j)$ for every $1 \leqslant j < n$, so $A_n \in R_{\underline{x}:\underline{A}}^{\mathsf{id}}(\mathcal{U}_i)$ by Proposition 4.3.10. Hence $\underline{x} : \underline{A}$ ctx is strongly normalisable since every reduction from it maps into a pair of reductions from $(x_1 : A_1, \dots, x_{n-1} : A_{n-1}\ \mathsf{ctx})$ and $\underline{x} : \underline{A} \vdash A : \mathcal{U}_i$, the latter being strongly normalisable by Proposition 4.3.3.

Since $x_j \in R_{\underline{x}:\underline{A}}^{\mathsf{id}}(A_j)$ for every $1 \leqslant j \leqslant n$, by Proposition 4.3.10, $b \in R_{\underline{x}:\underline{A}}^{\mathsf{id}}(B)$. □

Theorem 4.3.12 (Normalisation)**.** *If $\Gamma \vdash b : B$ is derivable then it is strongly normalisable; if Γ ctx is derivable then it is strongly normalisable; if $\Gamma \vdash a \equiv b : A$ is derivable then it is strongly normalisable.*

Proof. The first statement is immediate by Proposition 4.3.3 and Proposition 4.3.11; the second statement follows because $\Gamma \vdash \mathcal{U}_i : \mathcal{U}_{i+1}$, derived by $\mathcal{U}-\mathsf{intro}$ from Γ ctx, is strongly normalisable by the first statement; the third statement follows because $\Gamma \vdash a : A$ and $\Gamma \vdash b : A$ by Proposition 3.3.8 thus the result is a consequence of the first statement. $\square$

In fact, by Corollary 4.1.12, irreducible judgements are unique up to conversion, as it is immediate to show.

Chapter 5

Semantics

In this chapter a semantics, loosely inspired by [18], is first defined for the basic system, and then extended to the inductive types. The soundness and completeness results are obtained, together with the existence of a classifying model.

The main idea consists in interpreting the contexts as objects in a suitable category, and the terms as objects of slice categories over the interpretation of their context. A term belongs to a type when there is a suitable arrow between their interpretations; weakening is formalised by the properties of slice categories. Finally, dependent product and the inductive types are seen as transformation on the interpretations of terms and formalised as functors.

At a first glance, this semantics resembles [90]: however, there are significant differences, especially on the interpretation of products and inductive types, allowing to avoid the problems [32, 52, 39] of locally Cartesian closed categories.

The semantics presented here can be seen as a category with suitable structure, plus a set of endofunctors. The endofunctors provide the meaning of inductive types and dependent products. In this sense, the semantics is *point-free*: meaning arises from transformation of the framework into itself. This idea, which is a novelty of this work, will be detailed in the following, and it provides a complementary view with respect to the homotopic interpretation of [95]. This view says that there is no need to consider ∞-groupoids to understand Martin-Löf type theory. However, this semantics does not intend to say that the homotopic interpretation is incorrect, in any way: positively, it says that the theory of types admits many ways in which one can understand it, and the intrinsic power of higher category theory is not necessary, without disputing its value.

5.1 Categorical preliminaries

To define the suitable categories we use to interpret Martin-Löf type theory, some preliminary definitions are needed.

Indecomposable morphisms

The semantics will be defined *up-to-isomorphisms.* Although this is the standard way to think in category theory, isomorphisms will play a special role, so a notion of factorisation, which is aware of how isomorphisms act in the semantics, is needed.

Definition 5.1.1. In a category $\mathbb{C}$ a morphism f has a *proper factorisation* if there are arrows g and h, which are not isomorphisms, such that $f = g \circ h$.

Definition 5.1.2. In a category $\mathbb{C}$ a morphism f is *weakly indecomposable* when it has no proper factorisation; f is *indecomposable* when it is also not an isomorphism.

Fact 5.1.3. *In a category $\mathbb{C}$, if f is a weakly indecomposable isomorphism, for every factorisation $f = g \circ h$ it holds that g and h are both isomorphisms.*

Proof. Let $f = g \circ h$. Being f weakly indecomposable, either g or h is an isomorphism. Being f an isomorphism, either g is an isomorphism and $h = g^{-1} \circ f$, or h is an isomorphism and $g = f \circ h^{-1}$. Since the composition of isomorphisms is an isomorphism, the result follows. $\square$

Iso-stable categories

Every object in a category is isomorphic to itself via its identity. We need to say that this is the only way in which an object is isomorphic to itself, and, moreover, that identity cannot be decomposed in a section and a retraction.

Definition 5.1.4. An object A in a category is *iso-stable* when the only arrow $A \to A$ is the identity id_A and id_A is weakly indecomposable. A category is *iso-stable* when all its objects are iso-stable.

Fact 5.1.5. *In an iso-stable category, there is at most one isomorphism between each pair of objects.*

Proof. Let $f, g \colon A \to B$ be isomorphisms. Then $f \circ g^{-1} \colon B \to B$ and thus $f \circ g^{-1} = \mathsf{id}_B$ by iso-stability. Hence $f = f \circ (g^{-1} \circ g) = (f \circ g^{-1}) \circ g = g$. $\square$

Fact 5.1.6. *In a category $\mathbb{C}$, if $f \colon A \to B$, $g \colon B \to A$, and A, B are iso-stable objects, then f and g are isomorphisms and each other's inverse.*

Proof. By iso-stability, $g \circ f = \mathsf{id}_A$ and $f \circ g = \mathsf{id}_B$. $\square$

Typical categories and universes

Typing is the fundamental feature of type theory. As obvious as it may sound, a semantics has to model this feature. This will be done by arrows from the type to the term. Typical categories are those allowing for typing. Since types are terms in a universe, universes have to come in the picture exactly now.

Definition 5.1.7. A category $\mathbb{C}$ is *typical* when for every $A \in \mathsf{Obj}\,\mathbb{C}$, there is an indecomposable arrow f such that $\mathsf{cod}(f) = A$.

Roughly, in a typical category every term has a type.

Definition 5.1.8. Given $\langle \mathbb{C}; u \rangle$ with $\mathbb{C}$ a typical category and $u \in \mathsf{Obj}\,\mathbb{C}$, an object v is a *weak universe* when there is an arrow $v \to u$ in $\mathbb{C}$. Also, $\langle \mathbb{C}; u \rangle$ *has weak universes* when for every $A \in \mathsf{Obj}\,\mathbb{C}$ there is a weak universe v such that exists $v \to A$.

A pointed and typical category $\langle \mathbb{C}; u \rangle$ has at least u as a weak universe. The conditions say that every universe contains u, and every term lies in some universe.

Proposition 5.1.9. *If $\langle \mathbb{C}; u\rangle$ is an iso-stable, typical category having weak universes, then it has infinitely many non isomorphic weak universes.*

Proof. Let $u_0 = u$: since $\mathsf{id}_u \colon u_0 = u \to u$, u_0 is a weak universe. If u_i, $i \in \mathbb{N}$ is a weak universe, then there is $f_i \colon v \to u_i$ indecomposable for some $v \in \mathsf{Obj}\,\mathbb{C}$ since $\mathbb{C}$ is typical. Let $u_{i+1} = v$. Since u_i is a weak universe, there is $g \colon u_i \to u$, thus $g \circ f_i \colon u_{i+1} \to u$ showing that u_{i+1} is a weak universe, too. Hence, by induction on $\mathbb{N}$, u_i is a weak universe for every $i \in \mathbb{N}$.

Suppose $\{|i-j| \colon i, j \in \mathbb{N}, i \neq j \text{ and } u_i \cong u_j\} \subseteq \mathbb{N}$ to be non empty. Then it contains a minimal element k and there is $m \in \mathbb{N}$ such that $h \colon u_m \to u_{m+k}$ is an isomorphism. Let $f = f_m \circ \cdots \circ f_{m+k-1} \colon u_{m+k} \to u_m$. Then $f \circ h \colon u_m \to u_m$ so $f \circ h = id_{\mathsf{cod}(f)}$ being $\mathbb{C}$ iso-stable. Hence $f = (f \circ h) \circ h^{-1} = h^{-1}$.

If $k = 1$ then $f = f_m$ is an isomorphism, contradicting f_m to be indecomposable.

If $k > 1$ then $f_{m+1} \circ \cdots \circ f_{m+k-1} \circ f^{-1} \circ f_m \colon u_{m+1} \to u_{m+1}$ thus, by iso-stability, $f_{m+1} \circ \cdots \circ f_{m+k-1} \circ f^{-1} \circ f_m = \mathsf{id}_{u_{m+1}}$ and it is weakly indecomposable. Then, either $f_{m+1} \circ \cdots \circ f_{m+k-1}$ is an isomorphism, or $f^{-1} \circ f_m$ is an isomorphism. In the former case, $u_m \cong u_{m+k} \cong u_{m+1}$; in the latter, $u_m \cong u_{m+1}$. Both cases contradict k being minimal. □

Corollary 5.1.10. *If $\langle \mathbb{C}; u\rangle$ is an iso-stable, typical category having weak universes, then there is a chain $\{u_i\}_{i\in\mathbb{N}}$ of non-isomorphic weak universes such that $u_{i+1} \to u_i$ exists and it is indecomposable. Therefore, the chain of universes is a family of objects together with a family of arrows, each one linking to successive objects in the first family.*

The previous corollary justifies to fix the universes in the chain, telling them apart and being able to refer them. This is done in the following definition. Then the usual terminology of type theory can be given inside the semantics.

Definition 5.1.11. Let $\langle \mathbb{C}; u\rangle$ be an iso-stable, typical category having weak universes and terminal object, and let $\{u_i\}_{i\in\mathbb{N}}$ be a chain of non-isomorphic weak universes such that $u_{i+1} \to u_i$ exists and it is indecomposable. Then $\langle \mathbb{C}; \{u_i\}_{i\in\mathbb{N}}\rangle$ is a *category with universes*, and each u_i is said to be a (*strong*) *universe*.

Definition 5.1.12. Let $\langle \mathbb{C}; \{u_i\}_{i\in\mathbb{N}}\rangle$ be a category with universes. Then $A \in \mathsf{Obj}\,\mathbb{C}$ is a *term* when A is not terminal and there is $i \in \mathbb{N}$ such that $u_i \to A$.

Definition 5.1.13. Let $\langle \mathbb{C}; \{u_i\}_{i\in\mathbb{N}}\rangle$ be a category having universes. Then a term A is a *type* when there is $i \in \mathbb{N}$ such that $u_i \to A$ is indecomposable.

Definition 5.1.14. Let $\langle \mathbb{C}; \{u_i\}_{i\in\mathbb{N}}\rangle$ be a category having universes, let A be a type and a be a term. Then, *a has type A* when there is $f \colon A \to a$ in $\mathbb{C}$ such that f is not an isomorphism and either f is indecomposable or a is a type and A a universe. Moreover, *a has type A through f* emphasises which morphism f links A to a.

Fact 5.1.15. *Let $\langle \mathbb{C}; \{u_i\}_{i\in\mathbb{N}}\rangle$ be a category having universes. Then*

1. *every u_i is a type;*

2. *u_i has type u_{i+1};*

3. *every type A has type u_i for some $i \in \mathbb{N}$ through an indecomposable f.*

Proof. By Definition 5.1.11, (2) is evident and (1) follows. By Definition 5.1.13, (3) is immediate, showing how the second clause of Definition 5.1.14 is needed only when a is a type and A is not the minimal universe in which it lies. □

Definition 5.1.16. Let A be a type in $\langle \mathbb{C}; \{u_i\}_{i\in\mathbb{N}} \rangle$, a category with universes. Then $A^U = i$ with $u_i \to A$ indecomposable. In this case u_{A^U} is said to be the *minimal universe* in which A lies.

Definition 5.1.17. Let $\langle \mathbb{C}; \{u_i\}_{i\in\mathbb{N}} \rangle$ be a category with universes, and let $d \in \mathsf{Obj}\,\mathbb{C}$. Then, $\mathbb{C} \uparrow d$, the *groupoid of terms of* d, is the subcategory of $\mathbb{C}$ whose objects are the terms of type d and whose arrows are the isomorphisms of $\mathbb{C}$ between objects.

Fact 5.1.18. *If $\langle \mathbb{C}; \{u_i\}_{i\in\mathbb{N}} \rangle$ is a category with universes such that $d, d' \in \mathsf{Obj}\,\mathbb{C}$ and $d \cong d'$, then $\mathbb{C} \uparrow d = \mathbb{C} \uparrow d'$.*

Proof. Immediate by noticing that the objects are in bijection via the identity function, and the isomorphisms are preserved. □

ML-categories

Finally, we introduce the notion of *ML-category*, which is used to interpret the basic system of Chapter 3. The definition of ML-category is complex; the intuition behind it is that $\mathfrak{M}$ contains all the elements deputed to interpret the expressions in dependent type theory:

- $\mathbb{C}$ models the contexts, with $\bullet$ being the empty context;
- $U_i = !_c \circ u_i$ is the i-th universe in the context c;
- ν^c_A models how a context c is extended with a variable of type A, and χ^c_A identifies the term corresponding to that variable;
- F_Π, I_Π, and E_Π model the action of the formation, introduction, and elimination rules, respectively;
- the auxiliary S functor stands for substitution: if a is a term of type A in the context c, and b is a term of type B in the context $c, x : A$, then $S(c, A, B)(b, a)$ is the term $b[a/x]$ of type $B[a/x]$ in the context c.

Definition 5.1.19. A *ML-category* (ML stands for Martin-Löf) is

$$\mathfrak{M} = \langle \mathbb{M}; \mathbb{C}, \{u_i\}_{i\in\mathbb{N}}, \nu, \chi, \bullet, F_\Pi, I_\Pi, E_\Pi \rangle$$

such that (super- and subscripts will be omitted when clear from the context)

1. $\mathbb{M}$ is a category and $\mathbb{C}$ is a full subcategory of $\mathbb{M}$ such that $\bullet$ is initial in $\mathbb{C}$: for every $c \in \mathsf{Obj}\,\mathbb{C}$, the unique arrow $\bullet \to c$ is denoted as $!_c$;

2. for every $i \in \mathbb{N}$, u_i is an arrow of $\mathbb{M}$ whose codomain is $\bullet$, and for every $c \in \mathsf{Obj}\,\mathbb{C}$, $\mathcal{M}_c = \langle \mathbb{M}/c; \{!_c \circ u_i\}_{i\in\mathbb{N}} \rangle$ is a category with universes; we write U_i for $!_c \circ u_i$ when c is clear from the context;

3. for every $c \in \mathsf{Obj}\,\mathbb{C}$, if A is a type in $\mathcal{M}_c$ then ν^c_A is an indecomposable arrow in $\mathbb{C}$ such that $\mathsf{dom}(\nu^c_A) = c$; moreover, for every $c \in \mathbb{C}$, there is $n \in \mathbb{N}$ and, for each $1 \leqslant i \leqslant n$, there is a type A_i in $\mathcal{M}_{\mathsf{dom}(\nu_{A_i})}$ such that $!_c = \nu_{A_1} \circ \cdots \circ \nu_{A_n}$;

4. ν *respects isomorphisms*: for every $c, d \in \mathsf{Obj}\,\mathbb{C}$ such that $c \cong d$, A is a type in $\mathcal{M}_c$, B is a type in $\mathcal{M}_d$ and $A \cong B$ it holds that $\mathsf{cod}(\nu^c_A) \cong \mathsf{cod}(\nu^d_B)$;

5. for every $c \in \mathsf{Obj}\,\mathbb{C}$, if A and B are types in $\mathcal{M}_c$ then $\nu_{\nu_A \circ B} \circ \nu_A = \nu_{\nu_B \circ A} \circ \nu_B$;

6. for every $c \in \mathsf{Obj}\,\mathbb{C}$, if A is a type in $\mathcal{M}_c$ then χ^c_A is a term of type A in $\mathcal{M}_{\mathsf{cod}(\nu^c_A)}$ such that, for every term a in $\mathcal{M}_c$, $\chi^c_A \neq \nu_A \circ a$ in $\mathbb{M}$;

7. F_Π is a family of functors indexed by $c \in \mathsf{Obj}\,\mathbb{C}$ and a type in $\mathcal{M}_c$ such that

   a) $F_\Pi(c, a)\colon \bigcup_{i \in \mathbb{N}} \left(\mathcal{M}_{\mathsf{cod}(\nu^c_a)} \uparrow U_i\right) \to \bigcup_{i \in \mathbb{N}} (\mathcal{M}_c \uparrow U_i)$;

   b) F_Π *preserves universes*: $F_\Pi(c, a)(e) \in \mathsf{Obj}\,\left(\mathcal{M}_c \uparrow U_{\max(i, a^u)}\right)$ whenever $e \in \mathsf{Obj}\,\left(\mathcal{M}_{\mathsf{cod}(\nu^c_a)} \uparrow U_i\right)$;

   c) F_Π *respects isomorphisms*: if $c \cong c'$ and $a \cong a'$ then $F_\Pi(c, a) \cong F_\Pi(c', a')$ in the corresponding functor category;

   d) F_Π is *stable with respect to context extensions*: $F_\Pi(\mathsf{cod}(\nu^c_b), \nu^c_b \circ a)(\nu^c_b \circ e) = \nu^c_b \circ F_\Pi(c, a)(e)$;

   e) $F_\Pi(c, a)$ is *conservative*, i.e., it reflects isomorphisms. Notice how this requirement implies that $F_\Pi(c, a)$ is injective on objects: just consider that the identity is reflected necessarily in the identity. As usual, the inverse on the image is denoted by $F_\Pi^{-1}(c, a)$.

8. I_Π is a family of functors indexed by $c \in \mathsf{Obj}\,\mathbb{C}$, a type in $\mathcal{M}_c$, and b type in $\mathcal{M}_{\mathsf{cod}(\nu^c_a)}$ such that

   a) $I_\Pi(c, a, b)\colon \mathcal{M}_{\mathsf{cod}(\nu^c_a)} \uparrow b \to \mathcal{M}_c \uparrow F_\Pi(c, a)(b)$;

   b) I_Π *preserves universes*: when they are both defined, recalling Definition 5.1.20 for $\bar{F}_\Pi$,

   $$\begin{aligned} &I_\Pi(c, a, \bar{F}_\Pi(\mathsf{cod}(\nu^c_a), d_1, \ldots, d_l)(U_j)) \\ &= I_\Pi(c, a, \bar{F}_\Pi(\mathsf{cod}(\nu^c_a), d_1, \ldots, d_l)(U_k)) \ ; \end{aligned}$$

   c) I_Π *respects isomorphisms*: if $c \cong c'$, $a \cong a'$, and $b \cong b'$ then $I_\Pi(c, a, b) \cong I_\Pi(c', a', b')$ in the corresponding functor category;

   d) I_Π is *stable with respect to context extensions*: $I_\Pi(\mathsf{cod}(\nu^c_d), \nu^c_d \circ a, \nu^c_d \circ b)(\nu^c_d \circ e) = \nu^c_d \circ I_\Pi(c, a, b)(e)$;

   e) $I_\Pi(c, a, b)$ is a categorical equivalence.

9. E_Π is a family of functors indexed by $c \in \mathsf{Obj}\,\mathbb{C}$, a type in $\mathcal{M}_c$, and b type in $\mathcal{M}_{\mathsf{cod}(\nu^c_a)}$ such that

   a) a and b are in the same universe U_i;

b) $E_\Pi(c,a,b)(x,y) \in \mathsf{Obj}\ (\mathcal{M}_c \uparrow S(c,a,U_i)(b,y))$, with

$$\begin{aligned} E_\Pi(c,a,b) \colon \mathcal{M}_c \uparrow F_\Pi(c,a)(b) \times \mathcal{M}_c \uparrow a \to \\ \to \bigcup_{d \in \mathcal{M}_c \uparrow a} \mathcal{M}_c \uparrow S(c,a,U_i)(b,d) \ ; \end{aligned}$$

c) E_Π *preserves universes*: when they are both defined

$$\begin{aligned} & E_\Pi(c,a,\bar{F}_\Pi(\mathsf{cod}(\nu^c_a),d_1,\dots,d_l)(U_j)) \\ &= E_\Pi(c,a,\bar{F}_\Pi(\mathsf{cod}(\nu^c_a),d_1,\dots,d_l)(U_k)) \ , \end{aligned}$$

see Definition 5.1.20, and

$$\begin{aligned} & E_\Pi(c,\bar{F}_\Pi(c,d_1,\dots,d_l)(U_j),b) \\ &= E_\Pi(c,\bar{F}_\Pi(c,d_1,\dots,d_l)(U_k),b) \ ; \end{aligned}$$

d) E_Π *respects isomorphisms*: if $c \cong c'$, $a \cong a'$, and $b \cong b'$ then it holds that $E_\Pi(c,a,b) \cong E_\Pi(c',a',b')$ in the corresponding functor category;

e) E_Π is *stable with respect to context extensions*: $E_\Pi(\mathsf{cod}(\nu^c_d),\nu^c_d \circ a,\nu^c_d \circ b)(\nu^c_d \circ e,\nu^c_d \circ f) = \nu^c_d \circ E_\Pi(c,a,b)(e,f)$.

10. the auxiliary S is a family of functors indexed by $c \in \mathsf{Obj}\,\mathbb{C}$, a type in $\mathcal{M}_c$, and b type in $\mathcal{M}_{\mathsf{cod}(\nu^c_a)}$ such that

a) $S(c,a,b)(x,y) = E_\Pi(c,a,b)(I_\Pi(c,a,b)(x),y)$;

b) $S(c,a,\nu^c_a \circ a)(\chi^c_a,e) \cong e$;

c) $S(c,a,b)(\nu^c_a \circ p,e) \cong p$;

d) if $E_\Pi(\mathsf{cod}(\nu^c_a),Q,P)(p,q)$ is a term of type b in $\mathcal{M}_{\mathsf{cod}(\nu^c_a)}$ but not in $\mathcal{M}_{\mathsf{dom}(\nu^c_a)}$, then, calling $Q_e = S(c,a,U_i)(Q,e)$,

$$\begin{aligned} & S(c,a,S(\mathsf{cod}(\nu^c_a),Q,U_i)(P,q)) \\ & \quad (E_\Pi(\mathsf{cod}(\nu^c_a),Q,P)(p,q),e) \\ & \cong E_\Pi(c,Q_e,F_\Pi^{-1}(c,Q_e)(S(c,a,U_i)(F_\Pi(\mathsf{cod}(\nu^c_a),Q)(P),e))) \\ & \quad\quad (S(c,a,F_\Pi(\mathsf{cod}(\nu^c_a),Q)(P))(p,e),S(c,a,Q)(q,e)) \ ; \end{aligned}$$

e) if $I_\Pi(\mathsf{cod}(\nu^c_a),d,b)(p)$ is a term of type $F_\Pi(\mathsf{cod}(\nu^c_a),d)(b)$ in $\mathcal{M}_{\mathsf{cod}(\nu^c_a)}$ but not in $\mathcal{M}_{\mathsf{dom}(\nu^c_a)}$, and posing

$$\begin{aligned} d_e &= S(c,a,U_i)(d,e) \\ b_e &= F_\Pi^{-1}(c,d_e)(S(c,a,U_i)(F_\Pi(\mathsf{cod}(\nu^c_a),d)(b),e)) \\ p_e &= I_\Pi^{-1}(c,d_e,b_e)(S(c,a,U_i)(I_\Pi(\mathsf{cod}(\nu^c_a),d,b)(p),e)) \ , \end{aligned}$$

then

$$S(c,a,F_\Pi(\mathsf{cod}(\nu^c_a),d)(b))(I_\Pi(c,d,b)(p),e) \cong I_\Pi(c,d_e,b_e)(p_e) \ ;$$

f) if $F_\Pi(\mathsf{cod}(\nu^c_a), d)(p)$ is a type in $\mathcal{M}_{\mathsf{cod}(\nu^c_a)}$ but not in $\mathcal{M}_{\mathsf{dom}(\nu^c_a)}$, and posing

$$\begin{aligned} d_e &= S(c, a, U_i)(d, e) \\ p_e &= F_\Pi^{-1}(c, d_e)(S(c, a, U_i)(F_\Pi(\mathsf{cod}(\nu^c_a), d)(p), e)) \ , \end{aligned}$$

then

$$S(c, a, U_i)(F_\Pi(\mathsf{cod}(\nu^c_a), d)(p), e) \cong F_\Pi(c, d_e)(p_e) \ ;$$

g) Finally,

$$\begin{aligned} &S(c, A, S(\mathsf{cod}(\nu^c_A), \nu^c_A \circ B, U_i)(E, \nu^c_A \circ b)) \\ &\quad (S(\mathsf{cod}(\nu^c_A), \nu^c_A \circ B, E)(e, \nu^c_A \circ b), a) \\ \cong\ &S(c, B, S(\mathsf{cod}(\nu^c_B), \nu^c_B \circ A, U_i)(E, \nu^c_B \circ a)) \\ &\quad (S(\mathsf{cod}(\nu^c_B), \nu^c_B \circ A, E)(e, \nu^c_B \circ a), b) \ . \end{aligned}$$

We remark that conditions (8b) and (9c) are due to Proposition 3.3.21.

Sometimes, as already happened in Definition 5.1.19, there is the need to iterate the application of the functors F_Π, I_Π, E_Π and S. To simplify the notation, iterated functions are defined.

Definition 5.1.20. Let $a_1, \ldots, a_n$ such that $a_1 \in \mathsf{Obj}\,\mathcal{M}_c$, for $1 \leqslant j < n$ $a_{j+1} \in \mathsf{Obj}\,\mathcal{M}_{\mathsf{cod}(\nu_{a_j} \circ \cdots \circ \nu_{a_1})}$, and $b \in \mathsf{Obj}\,\mathcal{M}_{\mathsf{cod}(\nu_{a_n} \circ \cdots \circ \nu_{a_1})}$. Then, define $\bar{F}_\Pi(c, a_1, \ldots, a_n)(b)$ by induction on n as

- if $n = 0$, $\bar{F}_\Pi(c)(b) = b$;
- if $n = 1$, $\bar{F}_\Pi(c, a_1)(b) = F_\Pi(c, a_1)(b)$;
- if $n = j + 1$,

$$\begin{aligned} &\bar{F}_\Pi(c, a_1, \ldots, a_{j+1})(b) \\ =\ &\bar{F}_\Pi(c, a_1, \ldots, a_j) \\ &\quad \big(F_\Pi(\mathsf{cod}(\nu_{a_j} \circ \cdots \circ \nu_{a_1}), a_{j+1})(b)\big) \ . \end{aligned}$$

Definition 5.1.21. Let $a_1, \ldots, a_n$ such that $a_1 \in \mathsf{Obj}\,\mathcal{M}_c$, for $1 \leqslant j < n$ $a_{j+1} \in \mathsf{Obj}\,\mathcal{M}_{\mathsf{cod}(\nu_{a_j} \circ \cdots \circ \nu_{a_1})}$, $b \in \mathsf{Obj}\,\mathcal{M}_{\mathsf{cod}(\nu_{a_n} \circ \cdots \circ \nu_{a_1})}$, and $x \in \mathsf{Obj}\,\mathcal{M}_{\mathsf{cod}(\nu_{a_n} \circ \cdots \circ \nu_{a_1})} \uparrow b$. Then, define $\bar{I}_\Pi(c, a_1, \ldots, a_n, b)(x)$ by induction on n as

- if $n = 1$, $\bar{I}_\Pi(c, a_1, b)(x) = I_\Pi(c, a_1, b)(x)$;
- if $n = j + 1$,

$$\begin{aligned} &\bar{I}_\Pi(c, a_1, \ldots, a_{j+1}, b)(x) \\ =\ &\bar{I}_\Pi(c, a_1, \ldots, a_j, \big(F_\Pi(\mathsf{cod}(\nu_{a_j} \circ \cdots \circ \nu_{a_1}), a_{j+1})(b)\big) \\ &\quad \big(I_\Pi(\mathsf{cod}(\nu_{a_j} \circ \cdots \circ \nu_{a_1}), a_{j+1}, b)(x)\big) \ . \end{aligned}$$

Definition 5.1.22. Let $a_1, \ldots, a_n$ such that $a_1 \in \mathsf{Obj}\,\mathcal{M}_c$, for $1 \leqslant j < n$ $a_{j+1} \in \mathsf{Obj}\,\mathcal{M}_{\mathsf{cod}(\nu_{a_j} \circ \cdots \circ \nu_{a_1})}$, and $b \in \mathsf{Obj}\,\mathcal{M}_{\mathsf{cod}(\nu_{a_n} \circ \cdots \circ \nu_{a_1})}$. Also, let $x_1, \ldots, x_n$ such that $x_{j+1} \in \mathsf{Obj}\,\mathcal{M}_{\mathsf{cod}(\nu_{a_j} \circ \cdots \circ \nu_{a_1})} \uparrow a_{j+1}$ and $y \in \mathsf{Obj}\,\mathcal{M}_{\mathsf{cod}(\nu_{a_n} \circ \cdots \circ \nu_{a_1})} \uparrow b$. Then, define $\bar{S}(c, a_1, \ldots, a_n, b)(y, x_1, \ldots, x_n)$ by induction on n as

- if $n = 1$, $\bar{S}(c, a_1, b)(y, x_1) = S(c, a_1, b)(y, x_1)$;
- if $n = j + 1$,

$$\begin{aligned}&\bar{S}(c, a_1, \dots, a_{j+1}, b)(y, x_1, \dots, x_{j+1})\\ &= S(c, a_{j+1}, \bar{S}(c, a_1, \dots, a_j, U_i)(b, x_1, \dots, x_j))\\ &\quad \big(\bar{S}(c, a_1, \dots, a_j, b)(y, x_1, \dots, x_j), x_{j+1}\big) \quad .\end{aligned}$$

Definition 5.1.23. Let $a_1, \dots, a_n$ such that $a_1 \in \mathsf{Obj}\,\mathcal{M}_c$, for $1 \leqslant j < n$ $a_{j+1} \in \mathsf{Obj}\,\mathcal{M}_{\mathsf{cod}(\nu_{a_j} \circ \cdots \circ \nu_{a_1})}$, and $b \in \mathsf{Obj}\,\mathcal{M}_{\mathsf{cod}(\nu_{a_n} \circ \cdots \circ \nu_{a_1})}$. Also, let $x_1, \dots, x_n$ such that $x_{j+1} \in \mathsf{Obj}\,\mathcal{M}_{\mathsf{cod}(\nu_{a_j} \circ \cdots \circ \nu_{a_1})} \uparrow a_{j+1}$ and $y \in \mathsf{Obj}\,\mathcal{M}_{\mathsf{cod}(\nu_{a_n} \circ \cdots \circ \nu_{a_1})} \uparrow b$. Then, define $\bar{E}_\Pi(c, a_1, \dots, a_n, b)(y, x_1, \dots, x_n)$ by induction on n as

- if $n = 1$, $\bar{E}_\Pi(c, a_1, b)(y, x_1) = E_\Pi(c, a_1, b)(y, x_1)$;
- if $n = j + 1$,

$$\begin{aligned}&\bar{E}_\Pi(c, a_1, \dots, a_{j+1}, b)(y, x_1, \dots, x_{j+1})\\ &= E_\Pi(c, a_{j+1}, \bar{S}(c, a_1, \dots, a_j, U_i)(b, x_1, \dots, x_j))\\ &\quad \big(\bar{E}_\Pi(c, a_1, \dots, a_j, b)(y, x_1, \dots, x_j), x_{j+1}\big) \quad .\end{aligned}$$

An inductive theory is the theory generated by the rules for the basic system extended with a finite number of inductive types, see Chapter 3.

Definition 5.1.24. Let T be an inductive theory. A *T-ML-category* is

$$\mathfrak{M}^i = \langle \mathbb{M}; \mathbb{C}, \{u_i\}_{i \in \mathbb{N}}, \nu, \chi, \bullet, F_\Pi, I_\Pi, E_\Pi, \{F_\tau, \{I_{\tau,\mathbb{K}}\}_\mathbb{K}, E_\tau\}_\tau \rangle$$

with $\langle \mathbb{M}; \mathbb{C}, \{u_i\}_{i \in \mathbb{N}}, \nu, \chi, \bullet, F_\Pi, I_\Pi, E_\Pi \rangle$ an ML-category, see Definition 5.1.19, and the families of functors F_τ, $I_{\tau,\mathbb{K}}$ and E_τ defined as follows

1. each inductive type τ is uniquely identified by an object

$$p = \bar{F}_\Pi(c, a_1, \dots, a_n)(U_i) \in \mathsf{Obj}\,\mathcal{M}_c \uparrow U_{i+1} \quad .$$

   Then F_τ is a family of functors indexed by $c \in \mathsf{Obj}\,\mathbb{C}$ and p such that

   a) $F_\tau(c, p) \colon \{\cdot\} \to \mathcal{M}_c \uparrow p$, where $\{\cdot\}$ is the trivial category with just one object and the identity;
   b) F_τ *respects equivalent contexts*: if $c \cong c'$, then $F_\tau(c, p) \cong F_\tau(c', p)$ in the corresponding functor category;

2. each constructor $\mathbb{K}$ of an inductive type τ is uniquely identified by an object

$$\begin{aligned}q = {}&\bar{F}_\Pi(c, a'_1, \dots, a'_{n'}, d_1, \dots, d_m)\\ &\quad(\bar{E}_\Pi(c, a'_1, \dots, a'_n, U_{i+1})\\ &\qquad(F_\tau(c, p)(\cdot), \chi_{a'_1}, \dots, \chi_{a'_n})) \in \mathsf{Obj}\,\mathcal{M}_c \uparrow U_i \quad ,\end{aligned}$$

   where the $\{a'_j\}_{1 \leqslant j \leqslant n'}$ are a subsequence of the $\{a_j\}_{1 \leqslant j \leqslant n}$. Then $I_{\tau,\mathbb{K}}$ is a family of functors indexed by c and q such that

a) $I_{\tau,\mathbb{K}}(c,q)\colon \{\cdot\} \to \mathcal{M}_c \uparrow q$;

b) $I_{\tau,\mathbb{K}}$ *respects equivalent contexts*: if $c \cong c'$ then $I_{\tau,\mathbb{K}}(c,q) \cong I_{\tau,\mathbb{K}}(c',q)$ in the corresponding functor category;

3. in the non recursive case, each inductive type τ is uniquely associated to an object

$$\begin{aligned} r = \bar{F}_\Pi(&c, a_1, \dots, a_n, F_C, F_{c_1}, \dots, F_{c_k}, E_e, \\ &\bar{E}_\Pi(\mathsf{cod}(\nu_{E_e} \circ \nu_a), a_1, \dots, a_n, E_e, F_C)(\chi_{E_e}, \chi_{a_1}, \dots, \chi_{a_n})) \end{aligned}$$

with

$$\begin{aligned} F_C &= \bar{F}_\Pi(c, a_1, \dots, a_n, \\ &\qquad \bar{E}_\Pi(\mathsf{cod}(\nu_a), a_1, \dots, a_n, p) \\ &\qquad\quad (F_\tau(\mathsf{cod}(\nu_a), p)(\cdot), \chi_{a_1}, \dots, \chi_{a_n})))(U_h)\ , \\ F_{c_i} &= \bar{F}_\Pi(c, a'_1, \dots, a'_{n'_i}, d_1, \dots, d_m, \\ &\qquad \bar{E}_\Pi(\mathsf{cod}(\nu_{a,d}), a'_1, \dots, a'_{n_i}, q_i)) \\ &\qquad (\chi_{F_C}, \chi_{a_i}, \bar{E}_\Pi(\mathsf{cod}(\nu_{a,d}), a'_1, \dots, a'_{n_i}, d_1, \dots, d_m, q_i) \\ &\qquad\qquad (I_{\tau,\mathbb{K}}(\mathsf{cod}(\nu_{a,d}), q)(\cdot), \chi_{a_i}, \chi_{d_1}, \dots, \chi_{d_m})))\ , \\ E_e &= \bar{E}_\Pi(\mathsf{cod}(\nu_a), a_1, \dots, a_n, p)(F_\tau(\mathsf{cod}(\nu_a), p)(\cdot), \chi_{a_1}, \dots, \chi_{a_n})\ , \end{aligned}$$

and $\nu_a = \nu_{a_n} \circ \cdots \circ \nu_{a_1}$, $\nu_{a,d} = \nu_{d_m} \circ \cdots \circ \nu_{d_1} \circ \nu_{a_n} \circ \cdots \circ \nu_{a_1}$, $\chi_{a_i} = \chi_{a'_1}, \dots, \chi_{a'_{n'_i}}$. Then E_τ is a family of functors indexed by c and r such that

a) $E_\tau(c,r)\colon \{\cdot\} \to \mathcal{M}_c \uparrow r$;

b) E_τ *respects equivalent contexts*: if $c \cong c'$, then $E_\tau(c,r) \cong E_\tau(c',r)$ in the corresponding functor category;

c) given $C \in \mathsf{Obj}\, \mathcal{M}_c \uparrow F_C$, $c_i \in \mathcal{M}_c \uparrow F_{c_i}$ for $1 \leqslant i \leqslant k$, $T_i \in \mathcal{M}_c \uparrow a_i$ for $1 \leqslant i \leqslant n$, $p_i \in \mathcal{M}_c \uparrow d_i$ for $1 \leqslant i \leqslant m$,

$$\begin{aligned} &\bar{E}_\Pi(c, a_1, \dots, a_n, F_C, F_{c_1}, \dots, F_{c_k}, \\ &\qquad \bar{E}_\Pi(c, a_1, \dots, a_n, p)(T_1, \dots, T_n, F_\tau(c,p)(\cdot))) \\ &\quad (E_\tau(c,r)(\cdot), T_1, \dots, T_n, C, c_1, \dots, c_k, \\ &\qquad \bar{E}_\Pi(c, a_1, \dots, a_n, d_1, \dots, d_m, q) \\ &\qquad\quad (I_{\tau,\mathbb{K}}(c,q)(\cdot), T_1, \dots, T_k, p_1, \dots, p_m)) \\ \cong\ &\bar{E}_\Pi(c, a_1, \dots, a_n, d_1, \dots, d_m, F_{c_i}) \\ &\quad (c_i, T_1, \dots, T_n, p_1, \dots, p_n)\ ; \end{aligned}$$

d) given $T_1 \in \mathcal{M}_c \uparrow a_1, \dots, T_n \in \mathcal{M}_c \uparrow a_n$ and an object $e \in \mathcal{M}_c \uparrow \bar{E}_\Pi(c, a_1, \dots, a_n, p)(T_1, \dots, T_n, F_\tau(c,p)(\cdot))$,

$$\begin{aligned}
&\bar{E}_\Pi(c, a_1, \ldots, a_n,\\
&\quad \bar{F}_\Pi(c, a_1, \ldots, a_n, E_C)\\
&\qquad (\bar{S}(\mathsf{cod}(\nu_{E_C} \circ \nu_a), a_1, \ldots, a_n, p)(F_\tau(\mathsf{cod}(\nu_a), p)(\cdot), \chi_a)),\\
&\quad \{\bar{I}_\Pi(c, a'_1, \ldots, a'_{n_i}, d_1, \ldots, d_{m_i}, E_{c_i})\\
&\qquad (\bar{S}(\mathsf{cod}(\nu_{E_{c_i}} \circ \nu_{a,d}), a'_1, \ldots, a'_{n_i}, d_1, \ldots, d_{m_i}, q_i)\\
&\qquad\quad (I_{\tau,\mathbb{K}}(\mathsf{cod}(\nu_{a,d}), q)(\cdot), \chi_{a,d_i}))\}_{i=1}^k,\\
&\quad \bar{E}_\Pi(c, a_1, \ldots, a_n, p)(F_\tau(c, p)(\cdot), T_1, \ldots, T_n), r)\\
&\quad (E_\tau(\mathsf{cod}(\nu_{E_C} \circ \nu_{d_k} \circ \cdots \circ \nu_{d_1} \circ \nu_a), r)(\cdot),\\
&\quad T_1, \ldots, T_n, \bar{I}_\Pi(c, a_1, \ldots, a_n, E_C,\\
&\quad \bar{S}(\mathsf{cod}(\nu_{E_C} \circ \nu_a), a_1, \ldots, a_n, p)(F_\tau(\mathsf{cod}(\nu_a), p)(\cdot), \chi_a))(E_C),\\
&\quad \{\bar{I}_\Pi(c, a'_1, \ldots, a'_{n_i}, d_1, \ldots, d_{m_i}, E_{c_i},\\
&\qquad \bar{S}(\mathsf{cod}(\nu_{E_{c_i}} \circ \nu_{a,d}), a'_1, \ldots, a'_{n_i}, d_1, \ldots, d_{m_i}, q_i)\\
&\qquad\quad (I_{\tau,\mathbb{K}}(\mathsf{cod}(\nu_{a,d}), q_i)(\cdot), \chi_{a,d_i}))(E_{c_i})\}_{i=1}^k, e)\\
&\cong e \ ,
\end{aligned}$$

with

$$\begin{aligned}
E_C &= \bar{E}_\Pi(\mathsf{cod}(\nu_a), a_1, \ldots, a_n, p)(F_\tau(\mathsf{cod}(\nu_a), p)(\cdot), \chi_a) \ ,\\
E_{c_i} &= \bar{E}_\Pi(\mathsf{cod}(\nu_{a,d}), a'_1, \ldots, a'_{n_i}, d_1, \ldots, d_{m_i}, q_i)\\
&\qquad (I_{\tau,\mathbb{K}}(\mathsf{cod}(\nu_{a,d_i}), q_i)(\cdot), \chi_{a,d_i}) \ ,\\
\chi_a &= \chi_{a_1}, \ldots, \chi_{a_n},\\
\chi_{a,d_i} &= \chi_{a_i}, \chi_{d_1}, \ldots, \chi_{d_{m_i}},\\
\nu_{d_i} &= \nu_{d_{n_i}} \circ \cdots \circ \nu_{d_1} \ .
\end{aligned}$$

If a constructor $\mathbb{K}_i$ is recursive, that is, a certain d_j is an instance $\tau s_1, \ldots, s_n$ of τ, then F_{c_i} depends also on the object

$$\bar{E}_\Pi(c, a_1, \ldots, a_n, E_y, F_C)(C, s_1, \ldots, s_n, \chi_{E_y}) \ ,$$

with $E_y = \bar{E}_\Pi(c, a_1, \ldots, a_n, p)(F_\tau(c, p)(\cdot), s_1, \ldots, s_n)$. Also, in Point (3c), p_j is followed by

$$\begin{aligned}
&\bar{E}_\Pi(c, a_1, \ldots, a_n, F_C, F_{c_1}, \ldots, F_{c_k},\\
&\quad \bar{E}_\Pi(c, a_1, \ldots, a_n, p)\\
&\qquad (T_1, \ldots, T_n, F_\tau(c, p)(\cdot)))\\
&\quad (E_\tau(c, r)(\cdot), T_1, \ldots, T_n, C, c_1, \ldots, c_k) \ .
\end{aligned}$$

4. for each inductive type τ and constructor $\mathbb{K}$, the auxiliary functor S must satisfy the following conditions:
   a) if $F_\tau(\mathsf{cod}(\nu_a^c), p)(\cdot)$ is a type in $\mathcal{M}_{\mathsf{cod}(\nu_a^c)}$ but not in $\mathcal{M}_{\mathsf{dom}(\nu_a^c)}$, then $S(c, a, U_i)(F_\tau(\mathsf{cod}(\nu_a^c), p)(\cdot), e) \cong F_\tau(c, S(c, a, U_i)(p, e))(\cdot)$;
   b) if $I_{\tau,\mathbb{K}}(\mathsf{cod}(\nu_a^c), q)(\cdot)$ is a type in $\mathcal{M}_{\mathsf{cod}(\nu_a^c)}$ but not in $\mathcal{M}_{\mathsf{dom}(\nu_a^c)}$, then $S(c, a, U_i)(I_{\tau,\mathbb{K}}(\mathsf{cod}(\nu_a^c), q)(\cdot), e) \cong I_{\tau,\mathbb{K}}(c, S(c, a, U_i)(q, e))(\cdot)$;

c) if $E_\tau(\mathsf{cod}(\nu_a^c), r)(\cdot)$ is a type in $\mathcal{M}_{\mathsf{cod}(\nu_a^c)}$ but not in $\mathcal{M}_{\mathsf{dom}(\nu_a^c)}$, then $S(c, a, U_i)(E_\tau(\mathsf{cod}(\nu_a^c), r)(\cdot), e) \cong E_\tau(c, S(c, a, U_i)(r, e))(\cdot)$.

The very complex definition of T-ML-category takes care of numerous technical aspects, which will play a role in the following. However, abstracting away these technicalities, it can be understood in a simple way: there is a framework, which interprets the judgements - the contexts in $\mathbb{C}$, the regular judgements in the slices over $\mathbb{C}$, the judgemental equalities as isomorphisms in the slices; there is an internal transformation to interpret dependent products, and there are internal transformations to interpret inductive types. The complex conditions are needed to coordinate the framework and the term formations in a coherent way.

5.2 Interpretation

This section is devoted to interpret the basic system and the inductive theories in the framework introduced above, i.e., respectively in ML-categories and in T-ML-categories, and to prove that the interpretation is sound.

Basic system

Definition 5.2.1. Given an ML-category $\mathfrak{M}$, see Definition 5.1.19, an *interpretation* $[\![\cdot]\!]$ is a collection of functions mapping judgements in arrows of $\mathbb{M}$ and terms-in-context in objects of a slice category of $\mathbb{M}$. Also, a notion of *validity* applies, that selects which judgements and terms are subject to interpretation. Formally, an interpretation has to satisfy:

1. $\bullet\ \mathsf{ctx}$ is valid and $[\![\bullet\ \mathsf{ctx}]\!] = \mathsf{id}_\bullet$;
2. $\Gamma, x : A\ \mathsf{ctx}$ is valid if and only if $\Gamma\ \mathsf{ctx}$ is valid and $[\![\Gamma \vdash A]\!]$ is a type in $\mathcal{M}_{\mathsf{cod}([\![\Gamma\ \mathsf{ctx}]\!])}$; then $[\![\Gamma, x : A\ \mathsf{ctx}]\!] = \nu_{[\![\Gamma \vdash A]\!]} \circ [\![\Gamma\ \mathsf{ctx}]\!]$; from now on, to ease the notation, we write $\mathcal{M}_\Gamma$ for $\mathcal{M}_{\mathsf{cod}([\![\Gamma\ \mathsf{ctx}]\!])}$ and ν_A for $\nu_{[\![\Gamma \vdash A]\!]}$;
3. $\Gamma \vdash a : A$ is valid when $\Gamma\ \mathsf{ctx}$ is valid and $[\![\Gamma \vdash a]\!]$ is a term of type $[\![\Gamma \vdash A]\!]$ through f in $\mathcal{M}_\Gamma$; then $[\![\Gamma \vdash a : A]\!] = f$;
4. $\Gamma \vdash a \equiv b : A$ is valid when $\Gamma \vdash a : A$ is valid, $\Gamma \vdash b : A$ is valid, and $h \colon [\![\Gamma \vdash a]\!] \to [\![\Gamma \vdash b]\!]$ is a necessarily unique isomorphism in $\mathcal{M}_\Gamma$ such that $[\![\Gamma \vdash b : A]\!] = h \circ [\![\Gamma \vdash a : A]\!]$; then $[\![\Gamma \vdash a \equiv b : A]\!] = h$;
5. $\Gamma \vdash a$ is valid when there is A such that $\Gamma \vdash a : A$ is valid;
6. for every $i \in \mathbb{N}$, $\vdash \mathcal{U}_i$ is valid and $[\![\vdash \mathcal{U}_i]\!] = u_i$; moreover, $[\![\vdash \mathcal{U}_i : \mathcal{U}_{i+1}]\!]$ is the indecomposable arrow of Definition 5.1.11;
7. if $\Gamma, x : A \vdash x$ is valid then $[\![\Gamma, x : A \vdash x]\!] = \chi^{[\![\Gamma\ \mathsf{ctx}]\!]}_{[\![\Gamma \vdash A]\!]}$, abbreviated to χ^Γ_A;
8. if $\Gamma \vdash a$ and $\Gamma, x : A\ \mathsf{ctx}$ are valid, then $[\![\Gamma, x : A \vdash a]\!] = \nu_A \circ [\![\Gamma \vdash a]\!]$; also, if $\Gamma \vdash a : A$ and $\Gamma, x : B\ \mathsf{ctx}$ are valid, then $[\![\Gamma, x : B \vdash a : A]\!] = [\![\Gamma \vdash a : A]\!]$ in $\mathbb{M}$, see Figure 5.1;
9. if $\Gamma \vdash A : \mathcal{U}_i$ and $\Gamma, x : A \vdash B : \mathcal{U}_i$ are valid, then $\Gamma \vdash \Pi x : A.\, B : \mathcal{U}_i$ is valid, and $[\![\Gamma \vdash \Pi x : A.\, B]\!] = F_\Pi\left([\![\Gamma\ \mathsf{ctx}]\!], [\![\Gamma \vdash A]\!]\right)\left([\![\Gamma, x : A \vdash B]\!]\right)$;

10. if $\Gamma, x : A \vdash b : B$ is valid, then $\Gamma \vdash \lambda x : A.\, b : \Pi x : A.\, B$ is valid, and $[\![\Gamma \vdash \lambda x : A.b]\!] = I_\Pi([\![\Gamma\ \mathsf{ctx}]\!], [\![\Gamma \vdash A]\!], [\![\Gamma, x : A \vdash B]\!])([\![\Gamma, x : A \vdash b]\!])$;

11. if $\Gamma \vdash f : \Pi x : A.\, B$ and $\Gamma \vdash a : A$ are valid, then $\Gamma \vdash f\, a : B[a/x]$ is valid,and $[\![\Gamma \vdash f\, a]\!] = E_\Pi([\![\Gamma\ \mathsf{ctx}]\!], [\![\Gamma \vdash A]\!], [\![\Gamma, x : A \vdash B]\!])([\![\Gamma \vdash f]\!], [\![\Gamma \vdash a]\!])$.

In the following, to alleviate the notation $[\![\Gamma\ \mathsf{ctx}]\!]$ will be used to indicate both the arrow and its codomain.

Notice that:

- $\Gamma, x : A \vdash x$ is valid implies that $\Gamma\ \mathsf{ctx}$ and $\Gamma \vdash A$ are valid, thus Point (7) is well defined;
- since $[\![\Gamma \vdash \Pi x : A.\, B]\!] \in \mathsf{Obj}\ (\mathcal{M}_\Gamma \uparrow [\![\Gamma \vdash \mathcal{U}_i]\!])$, the arrow $[\![\Gamma \vdash \Pi x : A.\, B : \mathcal{U}_i]\!]$ is well defined, see (3);
- since $[\![\Gamma \vdash \lambda x : A.\, b]\!] \in \mathsf{Obj}\ (\mathcal{M}_\Gamma \uparrow [\![\Gamma \vdash \Pi x : A.\, B]\!])$, the arrow $[\![\Gamma \vdash (\lambda x : A.\, b) : \Pi x : A.\, B]\!]$ is well defined, see (3);
- if $\Gamma \vdash A \equiv A' : \mathcal{U}_i$ and $\Gamma, x : A \vdash B \equiv B' : \mathcal{U}_i$ are valid, then $[\![\Gamma, x : A \vdash B]\!] \cong [\![\Gamma, x : A \vdash B']\!]$ and, thus, $F_\Pi([\![\Gamma\ \mathsf{ctx}]\!], [\![\Gamma \vdash A]\!])([\![\Gamma, x : A \vdash B]\!]) \cong F_\Pi([\![\Gamma\ \mathsf{ctx}]\!], [\![\Gamma \vdash A']\!])([\![\Gamma, x : A \vdash B']\!])$; this isomorphism is $[\![\Gamma \vdash (\Pi x : A.\, B) \equiv (\Pi x : A'.\, B')]\!]$, see (4);
- if $\Gamma \vdash A \equiv A' : \mathcal{U}_i$ and $\Gamma, x : A \vdash b \equiv b' : B$ are valid, then $[\![\Gamma, x : A \vdash b]\!] \cong [\![\Gamma, x : A' \vdash b']\!]$ and thus $[\![\Gamma \vdash \lambda x : A.\, b]\!] \cong [\![\Gamma \vdash \lambda x : A'.\, b']\!]$, see (10); this isomorphism is $[\![\Gamma \vdash (\lambda x : A.\, b) \equiv (\lambda x : A'.\, b')]\!]$, see (4);
- if $\Gamma \vdash f \equiv f' : \Pi x : A.\, B$ and $\Gamma \vdash a \equiv a' : A$ are valid, $[\![\Gamma \vdash f]\!] \cong [\![\Gamma \vdash f']\!]$ and $[\![\Gamma \vdash a]\!] \cong [\![\Gamma \vdash a']\!]$, thus $[\![\Gamma \vdash f\, a]\!] \cong [\![\Gamma \vdash f'\, a']\!]$, see (11); this isomorphism is $[\![\Gamma \vdash f\, a \equiv f'\, a']\!]$, see (4);
- if $\Gamma \vdash f : \Pi x : A.\, B$ is valid, then $[\![\Gamma \vdash \lambda x : A.\, f\, x]\!] = I_\Pi([\![\Gamma\ \mathsf{ctx}]\!], [\![\Gamma \vdash A]\!], [\![\Gamma, x{:}A \vdash B]\!])([\![\Gamma, x{:}A \vdash f\, x]\!])$. But $[\![\Gamma, x{:}A \vdash f\, x]\!] = I_\Pi^{-1}([\![\Gamma\ \mathsf{ctx}]\!], [\![\Gamma \vdash A]\!], [\![\Gamma, x : A \vdash B]\!])([\![\Gamma \vdash f]\!])$, since each instance of I_Π is an equivalence of categories. Because the composition of the above instances of I_Π is isomorphic to the identity, it holds that $[\![\Gamma \vdash f]\!] \cong [\![\Gamma \vdash \lambda x : A.\, f\, x]\!]$. This isomorphism is $[\![\Gamma \vdash (\lambda x : A.\, f\, x) \equiv f]\!]$, see (4).

The definition of interpretation, illustrated in Figure 5.1, clarifies and justifies many aspects of the notion of ML-category: the terminology (context, term, type, etc.) previously introduced to indicate objects and arrows in the categories directly reflects the corresponding objects in the syntax via the interpretation $[\![\cdot]\!]$. A special mention is due to the notion of validity: only valid judgements are interpreted. In principle, there could be non-valid judgements in the syntax: in this respect, the soundness theorem says that every judgement that one can write is valid, thus interpretable. Here, as explained in Chapter 3, the judgements are exactly the derivable expressions. Moreover, in this respect, the semantics fixes the interpretation of inference rules as the collection of syntactically sound, validity preserving transformations over a ML-category.

To interpret $\Pi-\mathsf{comp}$, another result is needed. The proof is, essentially, calculation, but we will make explicit all the passages to let the reader get used to the style of the semantics.

Lemma 5.2.2 (Substitution)**.** *If* $\Gamma, x : A \vdash b : B$ *and* $\Gamma \vdash a : A$ *are valid, then* $[\![\Gamma \vdash (\lambda x : A.\, b)\, a]\!] \cong [\![\Gamma \vdash b[a/x]]\!]$.

Proof. By Definition 5.2.1, $\Gamma \vdash (\lambda x : A.\, b) : \Pi x : A.\, B$ is valid. Also, $[\![\Gamma\ \mathsf{ctx}]\!]$, $[\![\Gamma \vdash a]\!]$, $[\![\Gamma \vdash A]\!]$, $[\![\Gamma, x : A \vdash b]\!]$, $[\![\Gamma, x : A \vdash B]\!]$, and $[\![\Gamma \vdash (\lambda x : A.\, b)\, a]\!]$ are defined. Hence

$$\begin{aligned}
&[\![\Gamma \vdash (\lambda x : A.\, b)\, a]\!] \\
&= E_\Pi\left([\![\Gamma\ \mathsf{ctx}]\!], [\![\Gamma \vdash A]\!], [\![\Gamma, x : A \vdash B]\!]\right)\left([\![\Gamma \vdash \lambda x : A.\, b]\!], [\![\Gamma \vdash a]\!]\right) \\
&= E_\Pi\left([\![\Gamma\ \mathsf{ctx}]\!], [\![\Gamma \vdash A]\!], [\![\Gamma, x : A \vdash B]\!]\right) \\
&\qquad \left(I_\Pi\left([\![\Gamma\ \mathsf{ctx}]\!], [\![\Gamma \vdash A]\!], [\![\Gamma, x : A \vdash B]\!]\right)\left([\![\Gamma, x : A \vdash b]\!]\right), [\![\Gamma \vdash a]\!]\right) \\
&= S\left([\![\Gamma\ \mathsf{ctx}]\!], [\![\Gamma \vdash A]\!], [\![\Gamma, x : A \vdash B]\!]\right)\left([\![\Gamma, x : A \vdash b]\!], [\![\Gamma \vdash a]\!]\right) \quad .
\end{aligned}$$

If $x \notin \mathsf{FV}(b)$, then $[\![\Gamma, x : A \vdash b]\!] = \nu_A \circ [\![\Gamma \vdash b]\!]$, and thus, by (10) in Definition 5.1.19, $[\![\Gamma \vdash (\lambda x : A.\, b)\, a]\!] = [\![\Gamma \vdash b]\!] = [\![\Gamma \vdash b[a/x]]\!]$. Henceforth, assume $x \in \mathsf{FV}(b)$ and proceed by induction on the structure of the syntactical term b.

- If b is a variable, it has to be x, thus by (10) in Definition 5.1.19,

$$\begin{aligned}
&[\![\Gamma \vdash (\lambda x : A.\, x)\, a]\!] \\
&= S\left([\![\Gamma\ \mathsf{ctx}]\!], [\![\Gamma \vdash A]\!], [\![\Gamma, x : A \vdash A]\!]\right)\left([\![\Gamma, x : A \vdash x]\!], [\![\Gamma \vdash a]\!]\right) \\
&= S\left([\![\Gamma\ \mathsf{ctx}]\!], [\![\Gamma \vdash A]\!], [\![\Gamma, x : A \vdash A]\!]\right)\left(\chi^\Gamma_A, [\![\Gamma \vdash a]\!]\right) \\
&\cong [\![\Gamma \vdash a]\!] = [\![\Gamma \vdash x[a/x]]\!] \quad .
\end{aligned}$$

- If b is the application $p\, q$ then

$$\begin{aligned}
&[\![\Gamma \vdash (\lambda x : A.\, p\, q)\, a]\!] \\
&= S\left([\![\Gamma\ \mathsf{ctx}]\!], [\![\Gamma \vdash A]\!], [\![\Gamma, x : A \vdash B]\!]\right)\left([\![\Gamma, x : A \vdash p\, q]\!], [\![\Gamma \vdash a]\!]\right) \\
&= S\left([\![\Gamma\ \mathsf{ctx}]\!], [\![\Gamma \vdash A]\!], [\![\Gamma, x : A \vdash B]\!]\right) \\
&\qquad \left(E_\Pi\left([\![\Gamma, x : A\ \mathsf{ctx}]\!], [\![\Gamma, x : A \vdash Q]\!], [\![\Gamma, x : A, z : Q \vdash P]\!]\right)\right. \\
&\qquad\qquad \left.\left([\![\Gamma, x : A, z : Q \vdash p]\!], [\![\Gamma, x : A \vdash q]\!]\right), [\![\Gamma \vdash a]\!]\right) \quad ,
\end{aligned}$$

  where $\Gamma, x : A, z : Q \vdash p : P$, $\Gamma, x : A \vdash q : Q$, and $B = P[q/z]$. Posing $Q_a = S([\![\Gamma\ \mathsf{ctx}]\!], [\![\Gamma \vdash A]\!], [\![\Gamma \vdash \mathcal{U}_i]\!])([\![\Gamma, x : A \vdash Q]\!], [\![\Gamma \vdash a]\!])$, and applying the induction hypothesis to B,

$$\begin{aligned}
&\cong S([\![\Gamma\ \mathsf{ctx}]\!], [\![\Gamma \vdash A]\!], \\
&\qquad S([\![\Gamma, x : A\ \mathsf{ctx}]\!], [\![\Gamma, x : A \vdash Q]\!], [\![\Gamma, x : A, z : Q \vdash \mathcal{U}_i]\!]) \\
&\qquad\quad ([\![\Gamma, x : A, z : Q \vdash P]\!], [\![\Gamma, x : A \vdash q]\!])) \\
&\qquad \left(E_\Pi\left([\![\Gamma, x : A\ \mathsf{ctx}]\!], [\![\Gamma, x : A \vdash Q]\!], [\![\Gamma, x : A, z : Q \vdash P]\!]\right)\right. \\
&\qquad\qquad \left.\left([\![\Gamma, x : A, z : Q \vdash p]\!], [\![\Gamma, x : A \vdash q]\!]\right), [\![\Gamma \vdash a]\!]\right)
\end{aligned}$$

  which can be reduced by applying the properties of S (Point (10) of Definition 5.1.19), and the induction hypothesis to Q_a, yielding $Q_a \cong [\![\Gamma \vdash Q[a/x]]\!]$,

$$\cong E_\Pi([\![\Gamma\ \mathsf{ctx}]\!], [\![\Gamma \vdash Q[a/x]]\!],$$

$$
\begin{aligned}
&F_\Pi^{-1}(\llbracket \Gamma\ \mathsf{ctx} \rrbracket, \llbracket \Gamma \vdash Q[a/x] \rrbracket) \\
&\qquad (S(\llbracket \Gamma\ \mathsf{ctx} \rrbracket, \llbracket \Gamma \vdash A \rrbracket, \llbracket \Gamma, x : A \vdash \mathcal{U}_i \rrbracket) \\
&\qquad\quad (F_\Pi(\llbracket \Gamma, x : A\ \mathsf{ctx} \rrbracket, \llbracket \Gamma, x : A \vdash Q \rrbracket) \\
&\qquad\qquad (\llbracket \Gamma, x : A, z : Q \vdash P \rrbracket, \llbracket \Gamma \vdash a \rrbracket)))) \\
&(S(\llbracket \Gamma\ \mathsf{ctx} \rrbracket, \llbracket \Gamma \vdash A \rrbracket, \\
&\quad F_\Pi(\llbracket \Gamma, x : A\ \mathsf{ctx} \rrbracket, \llbracket \Gamma, x : A \vdash Q \rrbracket) \\
&\qquad (\llbracket \Gamma, x : A, z : Q \vdash P \rrbracket))(\llbracket \Gamma, x : A \vdash p \rrbracket, \llbracket \Gamma \vdash a \rrbracket), \\
&\ S(\llbracket \Gamma\ \mathsf{ctx} \rrbracket, \llbracket \Gamma \vdash A \rrbracket, \llbracket \Gamma, x : A \vdash Q \rrbracket)(\llbracket \Gamma, x : A \vdash q \rrbracket, \llbracket \Gamma \vdash a \rrbracket))
\end{aligned}
$$

then, applying the induction hypothesis again,

$$
\begin{aligned}
&= E_\Pi(\llbracket \Gamma\ \mathsf{ctx} \rrbracket, \llbracket \Gamma \vdash Q[a/x] \rrbracket, \\
&\qquad F_\Pi^{-1}(\llbracket \Gamma\ \mathsf{ctx} \rrbracket, \llbracket \Gamma \vdash Q[a/x] \rrbracket) \\
&\qquad\quad (\llbracket \Gamma \vdash \Pi z : Q[a/x].\, P[a/x] \rrbracket)) \\
&\qquad (\llbracket \Gamma \vdash p[a/x] \rrbracket, \llbracket \Gamma \vdash q[a/x] \rrbracket) \\
&= E_\Pi(\llbracket \Gamma\ \mathsf{ctx} \rrbracket, \llbracket \Gamma \vdash Q[a/x] \rrbracket, \llbracket \Gamma, z : Q[a/x] \vdash P[a/x] \rrbracket) \\
&\qquad (\llbracket \Gamma \vdash p[a/x] \rrbracket, \llbracket \Gamma \vdash q[a/x] \rrbracket) \\
&= \llbracket \Gamma \vdash p[a/x]\, q[a/x] \rrbracket = \llbracket \Gamma \vdash (p\, q)[a/x] \rrbracket \ .
\end{aligned}
$$

- If b is the abstraction $\lambda y : C.\, p$, in which we may safely assume that $y \notin \mathsf{FV}(a)$ since we can freely rename bound variables, and calling

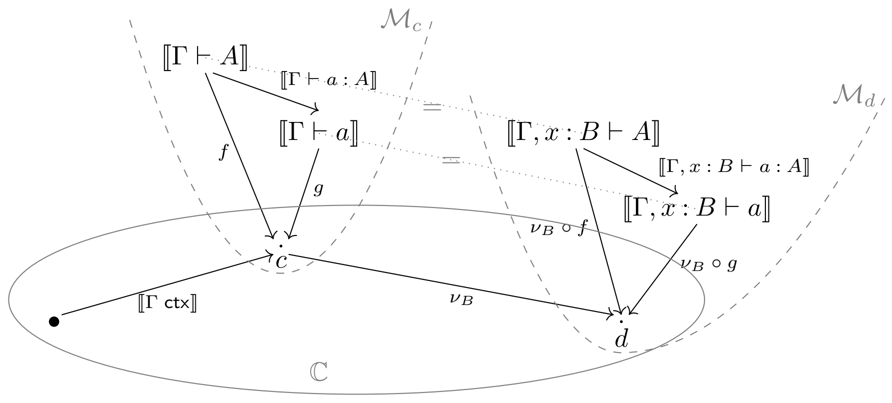
$$
\begin{aligned}
C_a &= S(\llbracket \Gamma\ \mathsf{ctx} \rrbracket, \llbracket \Gamma \vdash A \rrbracket, \llbracket \Gamma, x : A \vdash \mathcal{U}_i \rrbracket)(\llbracket \Gamma, x : A \vdash C \rrbracket, \llbracket \Gamma \vdash a \rrbracket) \ , \\
P_a &= F_\Pi^{-1}(\llbracket \Gamma\ \mathsf{ctx} \rrbracket, C_a)(S(\llbracket \Gamma\ \mathsf{ctx} \rrbracket, \llbracket \Gamma \vdash A \rrbracket, \llbracket \Gamma, x : A \vdash \mathcal{U}_i \rrbracket) \\
&\qquad (F_\Pi(\llbracket \Gamma, x : A\ \mathsf{ctx} \rrbracket, \llbracket \Gamma, x : A \vdash C \rrbracket) \\
&\qquad\quad (\llbracket \Gamma, x : A, y : X \vdash P \rrbracket), \llbracket \Gamma \vdash a \rrbracket)) \ , \\
p_a &= I_\Pi^{-1}(\llbracket \Gamma\ \mathsf{ctx} \rrbracket, C_a, P_a)(S(\llbracket \Gamma\ \mathsf{ctx} \rrbracket, \llbracket \Gamma \vdash A \rrbracket, \llbracket \Gamma, x : A \vdash \mathcal{U}_i \rrbracket) \\
&\qquad (I_\Pi(\llbracket \Gamma, x : A\ \mathsf{ctx} \rrbracket, \llbracket \Gamma, x : A \vdash C \rrbracket, \llbracket \Gamma, x : A, y : X \vdash P \rrbracket) \\
&\qquad\quad (\llbracket \Gamma, x : A, y : C \vdash p \rrbracket), \llbracket \Gamma \vdash a \rrbracket)) \ ,
\end{aligned}
$$

  by Point (10c) in Definition 5.1.19,

$$
\begin{aligned}
&\llbracket \Gamma \vdash (\lambda x : A.\, (\lambda y : C.\, p))\, a \rrbracket \\
&= S(\llbracket \Gamma\ \mathsf{ctx} \rrbracket, \llbracket \Gamma \vdash A \rrbracket, \llbracket \Gamma, x : A \vdash \Pi y : C.\, P \rrbracket) \\
&\qquad (I_\Pi(\llbracket \Gamma, x : A\ \mathsf{ctx} \rrbracket, \llbracket \Gamma, x : A \vdash C \rrbracket, \llbracket \Gamma, x : A, y : C \vdash P \rrbracket) \\
&\qquad\quad (\llbracket \Gamma, x : A, y : P \vdash p \rrbracket), \llbracket \Gamma \vdash a \rrbracket) \\
&\cong I_\Pi(\llbracket \Gamma\ \mathsf{ctx} \rrbracket, C_a, P_a)(p_a)
\end{aligned}
$$

  applying the induction hypothesis in C_a, P_a, and p_a,

$$
\begin{aligned}
&\cong I_\Pi(\llbracket \Gamma\ \mathsf{ctx} \rrbracket, \llbracket \Gamma \vdash C[a/x] \rrbracket, \llbracket \Gamma, y : C[a/x] \vdash P[a/x] \rrbracket) \\
&\qquad (\llbracket \Gamma, y : C[a/x] \vdash p[a/x] \rrbracket) \\
&= \llbracket \Gamma \vdash \lambda y : C[a/x].\, p[a/x] \rrbracket = \llbracket \Gamma \vdash (\lambda y : C.\, p)[a/x] \rrbracket \ .
\end{aligned}
$$



$\mathcal{M}_c$
$[\![\Gamma \vdash A]\!]$
$[\![\Gamma \vdash a : A]\!]$
$[\![\Gamma \vdash a]\!]$
f
g
$=$
$=$
$\mathcal{M}_d$
$[\![\Gamma, x : B \vdash A]\!]$
$[\![\Gamma, x : B \vdash a : A]\!]$
$[\![\Gamma, x : B \vdash a]\!]$
$\nu_B \circ f$
$\nu_B \circ g$
c
$[\![\Gamma \text{ ctx}]\!]$
ν_B
d
$\mathbb{C}$

Figure 5.1: How interpretation works.

- Finally, if b is the product $\Pi y : C.\, P$, in which we may safely assume that $y \notin \mathsf{FV}(a)$, and calling

$$
\begin{aligned}
C_a &= S([\![\Gamma\ \mathsf{ctx}]\!], [\![\Gamma \vdash A]\!], [\![\Gamma, x : A \vdash \mathcal{U}_i]\!])([\![\Gamma, x : A \vdash C]\!], [\![\Gamma \vdash a]\!]) \ , \\
P_a &= F_\Pi^{-1}([\![\Gamma\ \mathsf{ctx}]\!], C_a)(S([\![\Gamma\ \mathsf{ctx}]\!], [\![\Gamma \vdash A]\!], [\![\Gamma, x : A \vdash \mathcal{U}_i]\!]) \\
&\qquad (F_\Pi([\![\Gamma, x : A\ \mathsf{ctx}]\!], [\![\Gamma, x : A \vdash C]\!]) \\
&\qquad\quad ([\![\Gamma, x : A, y : C \vdash P]\!]), [\![\Gamma \vdash a]\!])) \ ,
\end{aligned}
$$

then

$$
\begin{aligned}
&[\![\Gamma \vdash (\lambda x : A.\, (\Pi y : C.\, P))\, a]\!] \\
&= S([\![\Gamma\ \mathsf{ctx}]\!], [\![\Gamma \vdash A]\!], [\![\Gamma, x : A \vdash \mathcal{U}_i]\!]) \\
&\qquad (F_\Pi([\![\Gamma, x : A\ \mathsf{ctx}]\!], [\![\Gamma, x : A \vdash C]\!]) \\
&\qquad\quad ([\![\Gamma, x : A, y : C \vdash P]\!]), [\![\Gamma \vdash a]\!]) \\
&\cong F_\Pi([\![\Gamma\ \mathsf{ctx}]\!], C_a)(P_a)
\end{aligned}
$$

applying the induction hypothesis in C_a and P_a,

$$
\begin{aligned}
&\cong F_\Pi([\![\Gamma\ \mathsf{ctx}]\!], [\![\Gamma \vdash C[a/x]]\!]) \\
&\qquad ([\![\Gamma, y : C[a/x] \vdash P[a/x]]\!]) \\
&= [\![\Gamma \vdash \Pi y : C[a/x].\, P[a/x]]\!] = [\![\Gamma \vdash (\Pi y : C.\, P)[a/x]]\!] \ . \qquad \square
\end{aligned}
$$

Therefore, since $[\![\Gamma \vdash f\, a]\!]$ is an object of

$$
\mathcal{M}_\Gamma \uparrow S([\![\Gamma\ \mathsf{ctx}]\!], [\![\Gamma \vdash A]\!], [\![\Gamma, x : A \vdash B]\!])([\![\Gamma, x : A \vdash f]\!], [\![\Gamma \vdash a]\!]) \ ,
$$

then $[\![\Gamma \vdash f\, a]\!] \in \mathsf{Obj}\ (\mathcal{M}_\Gamma \uparrow [\![\Gamma \vdash B[a/x]]\!])$ by Lemma 5.2.2, thus the arrow $[\![\Gamma \vdash f\, a : B[a/x]]\!]$ is defined, see (3) in Definition 5.2.1.

Proposition 5.2.3. *Given a ML-category and an interpretation over it with the notation as above, if* $\Gamma \vdash a : A$ *and* $\Delta \vdash a : A$ *are valid, and* $\Gamma \subseteq \Delta$*, then* $[\![\Delta \vdash a : A]\!] = [\![\Gamma \vdash a : A]\!]$.

Proof. If Ξ ctx is valid and Ξ' is a permutation of Ξ, then $[\![\Xi\ \mathsf{ctx}]\!] = [\![\Xi'\ \mathsf{ctx}]\!]$ by (5) in Definition 5.1.19. Thus, in particular, if Δ ctx is a valid permutation of Γ ctx, $[\![\Gamma \vdash a : A]\!] = [\![\Delta \vdash a : A]\!]$.

By hypothesis, Δ ctx can be permuted to the valid context Γ, Θ ctx: first, one moves in the front the typed variables in Γ, and in the rear the others, taking care of preserving the relative ordering of the two pieces; then, one rearranges the front part. Hence, $[\![\Delta \vdash a : A]\!] = [\![\Gamma, \Theta \vdash a : A]\!] = [\![\Gamma \vdash a : A]\!]$ by definition. □

Corollary 5.2.4. *Given a ML-category and an interpretation over it with the notation as before, if* $\Gamma \vdash a : A$ *is valid then* $[\![\Gamma \vdash a : A]\!] = [\![\mathsf{AV}(a) \vdash a : A]\!]$.

Proof. By Proposition 5.2.3. □

Fact 5.2.5. *If* $\Gamma \sim \Delta$ *are equivalent contexts, see Definition 3.3.18, then* $[\![\Gamma\ \mathsf{ctx}]\!] \cong [\![\Delta\ \mathsf{ctx}]\!]$ *by Point (4) of Definition 5.1.19. If, in particular,* $\Gamma \approx \Delta$, *then* $[\![\Gamma\ \mathsf{ctx}]\!] = [\![\Delta\ \mathsf{ctx}]\!]$ *by Point (5) of Definition 5.1.19.*

Fact 5.2.6. *In the basic system, if* $\Gamma \vdash a : A$ *and* $\Delta \vdash b : B$ *are two equivalent judgements,* $[\![\Gamma \vdash a]\!] \cong [\![\Delta \vdash b]\!]$. *Thus, if* $\mathfrak{M}$ *is an ML-category,* $\mathcal{M}_\Gamma \uparrow [\![\Gamma \vdash A]\!] \cong \mathcal{M}_\Delta \uparrow [\![\Delta \vdash B]\!]$.

This property will be extended to inductive theories in the following.

Definition 5.2.7. A model $\mathfrak{M} = \langle \mathcal{M}, [\![\cdot]\!] \rangle$ for a theory T is composed by a ML-category and an interpretation over it such that, for every $\gamma \in T$, γ is valid.

Theorem 5.2.8 (Soundness)**.** *Let* B *be the set of derivable judgements in the basic system. Then all the judgements in* B *are valid in every model.*

Proof. By induction on the structure of derivations:

- if the premises are valid, then by Definition 5.2.1 and the properties of categories the conclusions of $\mathsf{ctx}{-}\mathsf{EMP}$, $\mathsf{ctx}{-}\mathsf{EXT}$, Vble, $\equiv{-}\mathsf{refl}$, $\equiv{-}\mathsf{sym}$, $\equiv{-}\mathsf{trans}$, $\equiv{-}\mathsf{subst}$, $\equiv{-}\mathsf{subst}{-}\mathsf{eq}$, $\mathcal{U}{-}\mathsf{intro}$, $\mathcal{U}{-}\mathsf{cumul}$, $\Pi{-}\mathsf{form}$, $\Pi{-}\mathsf{intro}$, and $\Pi{-}\mathsf{elim}$ are valid;

- the observations after Definition 5.2.1 show that, when the premises are valid, also the conclusions of the rules $\Pi{-}\mathsf{form}{-}\mathsf{eq}$, $\Pi{-}\mathsf{intro}{-}\mathsf{eq}$, $\Pi{-}\mathsf{elim}{-}\mathsf{eq}$, and $\Pi{-}\mathsf{uniq}$ are valid;

- Lemma 5.2.2 shows that if the premises are valid, so is the conclusion of the $\Pi{-}\mathsf{comp}$ rule. □

At this point it becomes clear how [18] inspired this semantics: the definitions of ML-category and interpretation are a rephrasing of the proof theoretic properties of type theory, in particular, how substitution in proofs is performed gives the key to show Lemma 5.2.2. This lemma determines the constraints to put into the semantics, but these constraints are nothing but the description of substitution in proof, that is, Definition 3.3.6.

Inductive types

In the following, the interpretations of dependent products, abstractions and applications will always be obtained applying the functors F_Π, I_Π and E_Π to the interpretations of the proper terms-in-context. However, to simplify the notation, sometimes we will write $[\![\Gamma \vdash \Pi x : A.\, B]\!]$ instead of $F_\Pi([\![\Gamma\ \mathsf{ctx}]\!], [\![\Gamma \vdash A]\!])([\![\Gamma, x : A \vdash B]\!])$, and similarly for the other constructions.

Definition 5.2.9. Given a T-ML-category $\mathfrak{M}^i = \langle \mathfrak{M}; \{F_\tau, \{I_{\tau,\mathbb{K}}\}_\mathbb{K}, E_\tau\}_\tau\rangle$, for each inductive type τ the interpretation of Definition 5.2.1 is extended as follows:

1. if $\Gamma \vdash \Pi\,(x:F)_n\,.\mathcal{U}_i : \mathcal{U}_{i+1}$ is valid, then $\Gamma \vdash \tau : \Pi\,(x:F)_n\,.\mathcal{U}_i$ is valid and $[\![\Gamma \vdash \tau]\!] = F_\tau([\![\Gamma\ \mathsf{ctx}]\!], [\![\Gamma \vdash \Pi\,(x:F)_n\,.\mathcal{U}_i]\!])(\cdot)$; the interpretation of the type-in-context $\Gamma \vdash \Pi\,(x:F)_n\,.\mathcal{U}_i$ can be computed from the interpretations of the $\Gamma \vdash F_i$ through the functor F_Π.

2. if the judgement $\Gamma \vdash \Pi\,(x:F)_{n'}\,.\,\Pi\,(y:I)_m\,.\,\tau\,x'_1\,\cdots\,x'_n : \mathcal{U}_i$ is valid, then the judgement $\Gamma \vdash \mathbb{K} : \Pi\,(x:F)_{n'}\,.\,\Pi\,(y:I)_m\,.\,\tau\,x'_1\,\cdots\,x'_n$ is valid and $[\![\Gamma \vdash \mathbb{K}]\!] = I_{\tau,\mathbb{K}}([\![\Gamma\ \mathsf{ctx}]\!], [\![\Gamma \vdash \Pi\,(x:F)_{n'}\,.\,\Pi\,(y:I)_m\,.\,\tau\,x'_1\,\cdots\,x'_n]\!])(\cdot)$. As above, the latter interpretation can be easily obtained through F_Π and E_Π.

3. let

$$\begin{aligned} T_{\mathsf{ind}} := \Pi\,(x:F)_n\,.\,(\Pi C : (\Pi\,(x:F)_n\,.\,\tau\,x_1\,\cdots\,x_n \to \mathcal{U}_h).\\ (\Pi_{i=1}^k c_i : (\Pi\,(x:F)_{n'_i}\,.\,\Pi\,(y:I)_{m_i}\,.\\ C\,x'_1\,\cdots\,x'_{n_i}\,(\mathbb{K}\,x'_1\,\cdots\,x'_{n_i}\,y_1\,\cdots\,y_{m_i})).\\ (\Pi e : \tau\,x_1\,\cdots\,x_n.\,C\,x_1\,\cdots\,x_n\,e)))\ . \end{aligned}$$

   If $\Gamma \vdash T_{\mathsf{ind}} : \mathcal{U}_{h+1}$ is valid, then $\Gamma \vdash \mathsf{ind}_\tau : T_{\mathsf{ind}}$ is valid and $[\![\Gamma \vdash \mathsf{ind}_\tau]\!] = E_\tau([\![\Gamma\ \mathsf{ctx}]\!], [\![\Gamma \vdash T_{\mathsf{ind}}]\!])(\cdot)$.

Notice that:

- since $[\![\Gamma \vdash \tau]\!] \in \mathsf{Obj}\ (\mathcal{M}_\Gamma \uparrow [\![\Gamma \vdash \Pi\,(x:F)_n\,.\mathcal{U}_i]\!])$, then the arrow $[\![\Gamma \vdash \tau : \Pi\,(x:F)_n\,.\mathcal{U}_i]\!]$ is well defined, see (3) of Definition 5.2.1;
- since $[\![\Gamma \vdash \mathbb{K}]\!] \in \mathsf{Obj}\ (\mathcal{M}_\Gamma \uparrow [\![\Gamma \vdash \Pi\,(x:F)_{n'}\,.\,\Pi\,(y:I)_m\,.\,\tau\,x'_1\,\cdots\,x'_n]\!])$, the arrow $[\![\Gamma \vdash \mathbb{K} : \Pi\,(x:F)_{n'}\,.\,\Pi\,(y:I)_m\,.\,\tau\,x'_1\,\cdots\,x'_n]\!]$ is well defined;
- since $[\![\Gamma \vdash \mathsf{ind}_\tau]\!] \in \mathsf{Obj}\ (\mathcal{M}_\Gamma \uparrow [\![\Gamma \vdash T_{\mathsf{ind}}]\!])$, the arrow $[\![\Gamma \vdash \mathsf{ind}_\tau : T_{\mathsf{ind}}]\!]$ is well defined;
- if the premises of $\tau\mathsf{-comp}_i$ are valid, then

  $$\begin{aligned} &E_\tau([\![\Gamma\ \mathsf{ctx}]\!], [\![\Gamma \vdash \mathsf{ind}_\tau\,T_1\,\cdots\,T_n\,C\,c_1\,\cdots\,c_k\,(\mathbb{K}_i\,T_1\,\cdots\,T_n\,p_1\,\cdots\,p_m)]\!])(\cdot)\\ &\cong [\![\Gamma \vdash c_i\,T_1\,\cdots\,T_n\,p'_1\,\cdots\,p'_m]\!] \end{aligned}$$

  by Point (3c) of Definition 5.1.24, allowing to interpret the conclusion of the computation rule;

- if $\Gamma \vdash e : \tau T_1, \ldots, T_n$ is valid, then

$$
\begin{aligned}
&E_\tau(\llbracket \Gamma\ \mathsf{ctx} \rrbracket, \llbracket \Gamma \vdash \mathsf{ind}_\tau\, T_1, \cdots T_n \\
&\quad (\lambda\, (x : F)_n \,.\, (\lambda z : \tau\, x_1 \,\cdots\, x_n . \tau\, x_1 \,\cdots\, x_n)) \\
&\quad \Big(\lambda\, (x : F)_{n_1} \,.\, \Big(\lambda\, (y : I)_{m'_1} \,.\, \mathbb{K}_1\, x_1 \,\cdots\, x_{n_1}\, y_1 \,\cdots\, y_{m_1}\Big)\Big) \\
&\quad \vdots \\
&\quad \Big(\lambda\, (x : F)_{n_k} \,.\, \Big(\lambda\, (y : I)_{m'_k} \,.\, \mathbb{K}_k\, x_1 \,\cdots\, x_{n_k}\, y_1 \,\cdots\, y_{m_k}\Big)\Big) \\
&\quad e \rrbracket)(\cdot) \cong \llbracket \Gamma \vdash e \rrbracket
\end{aligned}
$$

by Point (3d) of Definition 5.1.24, allowing to interpret the conclusion of the uniqueness rule.

Lemma 5.2.10 (Substitution). *If $\Gamma, x : A \vdash b : B$ and $\Gamma \vdash a : A$ are valid, then $\llbracket \Gamma \vdash (\lambda x : A.\, b)a \rrbracket \cong \llbracket \Gamma \vdash b[a/x] \rrbracket$.*

Proof. If b is the conclusion of a rule in the basic system, the statement holds repeating the reasoning of Lemma 5.2.2. Suppose that b is the conclusion of a formation, introduction or elimination rule for an inductive type, and that $x \in \mathsf{FV}(b)$, and proceed by induction.

- If $b = \tau$,

$$
\begin{aligned}
&\llbracket \Gamma \vdash (\lambda x : A.\, \tau)\, a \rrbracket \\
&= S(\llbracket \Gamma\ \mathsf{ctx} \rrbracket, \llbracket \Gamma \vdash A \rrbracket, \llbracket \Gamma, x : A \vdash \Pi\, (x : F)_n \,.\, \mathcal{U}_i \rrbracket) \\
&\quad (F_\tau(\llbracket \Gamma, x : A\ \mathsf{ctx} \rrbracket, \llbracket \Gamma, x : A \vdash \Pi\, (x : F)_n \,.\, \mathcal{U}_i \rrbracket)(\cdot), \llbracket \Gamma \vdash a \rrbracket) \\
&\cong F_\tau(\llbracket \Gamma\ \mathsf{ctx} \rrbracket, \\
&\qquad S(\llbracket \Gamma\ \mathsf{ctx} \rrbracket, \llbracket \Gamma \vdash A \rrbracket, \\
&\qquad\quad S(\llbracket \Gamma\ \mathsf{ctx} \rrbracket, \llbracket \Gamma \vdash A \rrbracket, \llbracket \Gamma, x : A \vdash \mathcal{U}_{i+1} \rrbracket) \\
&\qquad\qquad (\llbracket \Gamma, x : A \vdash \Pi\, (x : F)_n \,.\, \mathcal{U}_i \rrbracket, \llbracket \Gamma \vdash a \rrbracket))(\cdot)
\end{aligned}
$$

and, by induction hypothesis,

$$
\begin{aligned}
&\cong F_\tau(\llbracket \Gamma\ \mathsf{ctx} \rrbracket, \llbracket \Gamma \vdash (\Pi\, (x : F)_n \,.\, \mathcal{U}_i)[a/x] \rrbracket)(\cdot) \\
&= \llbracket \Gamma \vdash \tau[a/x] \rrbracket \enspace .
\end{aligned}
$$

- If $b = \mathbb{K}$,

$$
\begin{aligned}
&\llbracket \Gamma \vdash (\lambda x : A.\, \mathbb{K})\, a \rrbracket \\
&= S(\llbracket \Gamma\ \mathsf{ctx} \rrbracket, \llbracket \Gamma \vdash A \rrbracket, \llbracket \Gamma, x : A \vdash \Pi\, (x : F)_{n'} \,.\, \Pi\, (y : I)_m \,.\, \tau\, x'_1 \,\cdots\, x'_n \rrbracket) \\
&\quad (I_{\tau,\mathbb{K}}(\llbracket \Gamma, x : A\ \mathsf{ctx} \rrbracket, \llbracket \Gamma, x : A \vdash \Pi\, (x : F)_{n'} \,. \\
&\qquad\qquad \Pi\, (y : I)_m \,.\, \tau\, x'_1 \,\cdots\, x'_n \rrbracket)(\cdot), \llbracket \Gamma \vdash a \rrbracket) \\
&\cong I_{\tau,\mathbb{K}}(\llbracket \Gamma\ \mathsf{ctx} \rrbracket, S(\llbracket \Gamma\ \mathsf{ctx} \rrbracket, \llbracket \Gamma \vdash A \rrbracket, \llbracket \Gamma, x : A \vdash \mathcal{U}_i \rrbracket) \\
&\qquad (\llbracket \Gamma, x : A \vdash \Pi\, (x : F)_{n'} \,.\, \Pi\, (y : I)_m \,.\, \tau\, x'_1 \,\cdots\, x'_n \rrbracket, \llbracket \Gamma \vdash a \rrbracket))(\cdot)
\end{aligned}
$$

and, by induction hypothesis,

$$
\begin{aligned}
&\cong I_{\tau,\mathbb{K}}(\llbracket \Gamma\ \mathsf{ctx} \rrbracket, \llbracket \Gamma \vdash (\Pi\, (x : F)_{n'} \,.\, \Pi\, (y : I)_m \,.\, \tau\, x'_1 \,\cdots\, x'_n)[a/x] \rrbracket)(\cdot) \\
&= \llbracket \Gamma \vdash \mathbb{K}[a/x] \rrbracket \enspace .
\end{aligned}
$$

- If $b = \mathsf{ind}_\tau$,

$$\begin{aligned}
&[\![\Gamma \vdash (\lambda x : A.\, \mathsf{ind}_\tau)\, a]\!] \\
&= S([\![\Gamma\ \mathsf{ctx}]\!], [\![\Gamma \vdash A]\!], [\![\Gamma, x : A \vdash T_{\mathsf{ind}}]\!]) \\
&\qquad (E_\tau([\![\Gamma, x : A\ \mathsf{ctx}]\!], [\![\Gamma, x : A \vdash T_{\mathsf{ind}}]\!])(\cdot), [\![\Gamma \vdash a]\!]) \\
&\cong E_\tau([\![\Gamma\ \mathsf{ctx}]\!], S([\![\Gamma\ \mathsf{ctx}]\!], [\![\Gamma \vdash A]\!], [\![\Gamma, x : A \vdash \mathcal{U}_{h+1}]\!]) \\
&\qquad\qquad ([\![\Gamma, x : A \vdash T_{\mathsf{ind}}]\!], [\![\Gamma \vdash a]\!]))(\cdot)
\end{aligned}$$

  and, by induction hypothesis,

$$\begin{aligned}
&\cong E_\tau([\![\Gamma\ \mathsf{ctx}]\!], [\![\Gamma \vdash (T_{\mathsf{ind}})[a/x]]\!])(\cdot) \\
&= [\![\Gamma \vdash \mathsf{ind}_\tau[a/x]]\!] \ .
\end{aligned}$$

□

Proposition 5.2.3, Corollary 5.2.4 and Fact 5.2.6 trivially extend to the inductive system and T-ML-categories.

Definition 5.2.11. A *model* for an inductive theory T is composed by a T-ML-category and an interpretation over it such that, for every $\gamma \in T$, γ is valid.

Theorem 5.2.12 (Soundness)**.** *Let T be a set of derivable judgements on an inductive system, i.e., the basic system extended with a finite number of inductive types. Then all the judgements in T are valid in every model.*

Proof. Since a model for an inductive system extends a model for the basic system, and Theorem 5.2.8 proves the soundness of the basic system, it is enough to prove the statement when γ is the conclusion of a rule for an inductive type:

- if the premises are valid, then the conclusions of $\tau{-}\mathsf{form}$, $\tau{-}\mathsf{intro}_i$ and $\tau{-}\mathsf{elim}$ are valid by Definition 5.2.9;
- the observations after Definition 5.2.9 show that, when the premises are valid, also the conclusions of the rules $\tau{-}\mathsf{comp}_i$ and $\tau{-}\mathsf{uniq}$ are valid. □

5.3 Classifying model and completeness

This section is devoted to construct the syntactical categories for the basic system and a generic inductive theory, and to prove that the respective semantics are complete.

In the following, when considering the basic system, the theory T is the system B, obtained applying the rules illustrated in Section 3.1. When dealing with inductive types, T is an inductive theory, i.e., B extended with a finite number of inductive types as in Section 3.2.

Basic system

Definition 5.3.1. The set $\mathcal{O}_T$ is defined as the minimal one satisfying

1. if $(\Gamma\ \mathsf{ctx}) \in T$ then $[\Gamma\ \mathsf{ctx}] = [\Gamma\ \mathsf{ctx}]_{\approx} \in \mathcal{O}_T$;
2. if $(\Gamma \vdash a : A) \in T$ then $[\Gamma \vdash a] = (a, [\mathsf{AV}(a)\ \mathsf{ctx}]) \in \mathcal{O}_T$, with $\mathsf{AV}(a)$ introduced in Definition 3.3.1.

Definition 5.3.2. Given a pair $(a, b) \in \mathcal{O}_T \times \mathcal{O}_T$, $\mathsf{dom}(a, b) = a$ and $\mathsf{cod}(a, b) = b$. Also, if $(a, b) \in \mathcal{O}_T \times \mathcal{O}_T$ and $(b, c) \in \mathcal{O}_T \times \mathcal{O}_T$, then $(b, c) \circ (a, b) = (a, c)$.

Definition 5.3.3. The set $\mathcal{A}_T \subseteq \mathcal{O}_T \times \mathcal{O}_T$ is defined as the minimal one such that

1. if $(\Gamma, x : A\ \mathsf{ctx}) \in T$ then $\nu_A = ([\Gamma\ \mathsf{ctx}], [\Gamma, x : A\ \mathsf{ctx}]) \in \mathcal{A}_T$;
2. if $(\Gamma\ \mathsf{ctx}), (\Delta\ \mathsf{ctx}) \in T$ and $\Gamma \sim \Delta$ then $([\Gamma\ \mathsf{ctx}], [\Delta\ \mathsf{ctx}]) \in \mathcal{A}_T$;
3. if $(\Gamma \vdash a : A) \in T$ then $[\Gamma \vdash a : A] = ([\Gamma \vdash A], [\Gamma \vdash a]) \in \mathcal{A}_T$ and $\epsilon_{\Gamma\vdash a:A} = ([\Gamma \vdash a], [\mathsf{AV}(a)\ \mathsf{ctx}]) \in \mathcal{A}_T$;
4. if $(\Gamma \vdash a \equiv b : A) \in T$ then $[\Gamma \vdash a \equiv b : A] = ([\Gamma \vdash a], [\Gamma \vdash b]) \in \mathcal{A}_T$;
5. if $a \in \mathcal{O}_T$ then $\mathsf{id}_a = (a, a) \in \mathcal{A}_T$;
6. if $(a, b) \in \mathcal{A}_T$ and $(b, c) \in \mathcal{A}_T$ then $(b, c) \circ (a, b) \in \mathcal{A}_T$.

It must be remarked that $(a, b) \in \mathcal{A}_T$ is uniquely identified by its endpoints.

Definition 5.3.4. The *syntactical category* $\mathbb{M}_T$ has $\mathcal{O}_T$ as objects, and $\mathcal{A}_T$ as arrows, and its composition is $\circ$ as in Definition 5.3.2.

Fact 5.3.5. *The syntactical category is a category.*

Proof. First, Definition 5.3.4 is well given: by Proposition 3.3.9, if $(\Gamma \vdash a : A) \in T$ then $(\Gamma \vdash A : \mathcal{U}_i) \in T$ thus $[\Gamma \vdash a : A]$ is properly defined; also, by Proposition 3.3.8, if $(\Gamma \vdash a \equiv b : A) \in T$ then $(\Gamma \vdash a : A) \in T$ and $(\Gamma \vdash b : A) \in T$, thus $[\Gamma \vdash a]$ and $[\Gamma \vdash b]$ are defined.

Finally, if $f = (a, b) \in \mathcal{A}_T$ then $f \circ \mathsf{id}_a = (a, b) \circ (a, a) = (a, b) = f$ and $\mathsf{id}_b \circ f = (b, b) \circ (a, b) = (a, b) = f$. If $f = (a, b)$, $g = (b, c)$, $h = (c, d)$ are in $\mathcal{A}_T$ then $h \circ (g \circ f) = h \circ ((b, c) \circ (a, b)) = (c, d) \circ (a, c) = (a, d) = (b, d) \circ (a, b) = ((c, d) \circ (b, c)) \circ f = (h \circ g) \circ f$. □

As Dr. Maschio pointed out, the syntactical category is a partial order.

Definition 5.3.6. The *syntactical context category* $\mathbb{C}_T$ is the full subcategory of $\mathbb{M}_T$ whose objects are $\{[\Gamma\ \mathsf{ctx}] : (\Gamma\ \mathsf{ctx}) \in T\}$.

Fact 5.3.7. *The category $\mathbb{C}_T$ has $[\bullet\ \mathsf{ctx}]$ as the initial object, and each arrow ν_A is indecomposable.*

Proof. By construction, each arrow ν_A is indecomposable.

Since $(\bullet\ \mathsf{ctx}) \in T$, $[\bullet\ \mathsf{ctx}]$ is defined. Let $(\Gamma\ \mathsf{ctx}) \in T$, then $\Gamma = x_1{:}A_1, \ldots, x_n{:} A_n$, so

$$[\bullet\ \mathsf{ctx}] \xrightarrow{\nu_{A_1}} [x_1 : A_1\ \mathsf{ctx}] \xrightarrow{\nu_{A_2}} \cdots \xrightarrow{\nu_{A_n}} [\Gamma\ \mathsf{ctx}]$$

thus the composition is the pair $([\bullet\ \mathsf{ctx}], [\Gamma\ \mathsf{ctx}])$. If $\Delta \approx \Gamma$, the same construction yields $([\bullet\ \mathsf{ctx}], [\Delta\ \mathsf{ctx}])$, but $[\Delta\ \mathsf{ctx}] = [\Delta\ \mathsf{ctx}]_\approx = [\Gamma\ \mathsf{ctx}]_\approx = [\Gamma\ \mathsf{ctx}]$ proving the uniqueness of the arrow $[\bullet\ \mathsf{ctx}] \to [\Gamma\ \mathsf{ctx}]$. □

Fact 5.3.8. *If $(\Gamma, x : A\ \mathsf{ctx}), (\Delta, y : B\ \mathsf{ctx}) \in T$ such that $[\Gamma\ \mathsf{ctx}] \cong [\Delta\ \mathsf{ctx}]$ and $(\Gamma \vdash A \equiv B \in \mathcal{U}_i) \in T$, then $[\Gamma, x : A\ \mathsf{ctx}] \cong [\Delta, y : B\ \mathsf{ctx}]$.*

Proof. By Definition 5.3.3 $[\Gamma\ \mathsf{ctx}] \cong [\Delta\ \mathsf{ctx}]$ implies $\Gamma \sim \Delta$, thus $\Gamma, x{:}A \sim \Delta, y{:}B$ and $\mathsf{cod}(\nu_A^{[\Gamma\ \mathsf{ctx}]}) = [\Gamma, x : A\ \mathsf{ctx}] \cong [\Delta, y : B\ \mathsf{ctx}] = \mathsf{cod}(\nu_B^{[\Delta\ \mathsf{ctx}]})$ by Point (2) of Definition 5.3.3. □

Fact 5.3.9. *If* $(\Gamma \vdash A : \mathcal{U}_i) \in T$ *and* $(\Gamma \vdash B : \mathcal{U}_j) \in T$ *then* $\nu_{\nu_B \circ A} \circ \nu_B = \nu_{\nu_A \circ B} \circ \nu_A$.

Proof. By Corollary 3.3.5, $(\Gamma\ \mathsf{ctx}) \in T$, so $[\Gamma\ \mathsf{ctx}] \in \mathsf{Obj}\,\mathbb{C}_T$. Moreover, it holds that $(\Gamma, x : A, y : B\ \mathsf{ctx}) \in T$ and $(\Gamma, y : B, x : A\ \mathsf{ctx}) \in T$, with x and y fresh variables. By Definition 3.3.18, $[\Gamma, x : A, y : B\ \mathsf{ctx}]_{\approx} = [\Gamma, y : B, x : A\ \mathsf{ctx}]_{\approx}$, hence

$$\begin{aligned}
&\nu_{\nu_B \circ A} \circ \nu_B \\
&= ([\Gamma, y : B\ \mathsf{ctx}], [\Gamma, y : B, x : A\ \mathsf{ctx}]) \circ ([\Gamma\ \mathsf{ctx}], [\Gamma, y : B\ \mathsf{ctx}]) \\
&= \big([\Gamma\ \mathsf{ctx}], [\Gamma, y : B, x : A]_{\approx}\big) \\
&= \big([\Gamma\ \mathsf{ctx}], [\Gamma, x : A, y : B]_{\approx}\big) \\
&= ([\Gamma, x : A\ \mathsf{ctx}], [\Gamma, x : A, y : B\ \mathsf{ctx}]) \circ ([\Gamma\ \mathsf{ctx}], [\Gamma, x : A\ \mathsf{ctx}]) \\
&= \nu_{\nu_A \circ B} \circ \nu_A \quad ,
\end{aligned}$$

proving the statement. □

Fact 5.3.10. *For every* $i \in \mathbb{N}$ *and* $(\Gamma\ \mathsf{ctx}) \in T$, $[\Gamma \vdash \mathcal{U}_i] = (\mathcal{U}_i, [\bullet\ \mathsf{ctx}])$.

Proof. Since $(\vdash \mathcal{U}_i : \mathcal{U}_{i+1}) \in T$, $[\vdash \mathcal{U}_i] = (\mathcal{U}_i, [\bullet\ \mathsf{ctx}])$. Also, $(\Gamma \vdash \mathcal{U}_i : \mathcal{U}_{i+1}) \in T$ and $\mathsf{AV}(\mathcal{U}_i)$ is empty, so $[\Gamma \vdash \mathcal{U}_i] = (\mathcal{U}_i, [\bullet\ \mathsf{ctx}])$. □

Fact 5.3.11. *For every* $[\Gamma\ \mathsf{ctx}] \in \mathsf{Obj}\,\mathbb{C}_T$*, the slice category*

$$\mathcal{M}_\Gamma^T = \langle \mathbb{M}_T / [\Gamma\ \mathsf{ctx}]\,; \big\{([\bullet\ \mathsf{ctx}], [\Gamma\ \mathsf{ctx}]) \circ \epsilon_{\vdash \mathcal{U}_i : \mathcal{U}_{i+1}}\big\}_{i \in \mathbb{N}} \rangle$$

is iso-stable.

Proof. Suppose $f \colon a \to a$ in $\mathcal{M}_\Gamma^T$. Since $a = (b, [\Gamma\ \mathsf{ctx}])$, $f \colon b \to b$ in $\mathbb{M}_T$, thus $f = (b, b)$ since arrows are uniquely identified by their endpoints, see Definition 5.3.3. Hence, $f = \mathsf{id}_b$ in $\mathbb{M}_T$ and thus $f = \mathsf{id}_a$ in $\mathcal{M}_\Gamma^T$. □

Fact 5.3.12. *If* $(\Gamma \vdash a \equiv b : A) \in T$*, then* $[\Gamma \vdash a \equiv b : A]$ *is an isomorphism both in* $\mathbb{M}_T$ *and in* $\mathcal{M}_\Gamma^T$.

Proof. If $(\Gamma \vdash a \equiv b : A) \in T$ then $(\Gamma \vdash b \equiv a : A) \in T$, too, so $[\Gamma \vdash a \equiv b : A] \in \mathcal{A}_T$. It is immediate to see that the composition of these two arrows yields the identity, in both ways. Hence $[\Gamma \vdash a \equiv b : A]$ is an isomorphism. □

Proposition 5.3.13. *For every* $[\Gamma\ \mathsf{ctx}] \in \mathsf{Obj}\,\mathbb{C}_T$*, the slice category* $\mathcal{M}_\Gamma^T$ *is typical.*

Proof. Let $c \in \mathsf{Obj}\,\mathcal{M}_\Gamma^T$ and consider its domain in $\mathbb{M}_T$:

- if $\mathsf{dom}(c) = [\bullet\ \mathsf{ctx}]$ then, by Corollary 3.3.23, there are a and A such that $(\vdash a : A) \in T$ and a is not inhabited. Thus, by Point (3) of Definition 5.3.3, $\epsilon_{\vdash a:A}$ exists, and it has to be indecomposable, proving the statement.
- if $\mathsf{dom}(c) = [\Delta, x : A\ \mathsf{ctx}]$ then ν_A is the sought indecomposable arrow.

- if $\mathsf{dom}(c) = [\Gamma \vdash a]$ then $(\Gamma \vdash a : A) \in T$ for some type A. By Proposition 3.3.21, if A is not unique up to isomorphisms then $\Gamma \vdash a : \Pi(x : G)_l.\mathcal{U}_j$ and $\Gamma \vdash a : \Pi(x : G)_l.\mathcal{U}_k$. Thus, if $l = 0$, consider A as the minimal universe $\mathcal{U}_i$ such that $\Gamma \vdash a : \mathcal{U}_i$. Otherwise, the choice of A among the different types of a is free. Thus $[\Gamma \vdash a : A]$ has to be indecomposable and $\mathsf{cod}\,([\Gamma \vdash a : A]) = [\Gamma \vdash a]$. ☐

Proposition 5.3.14. *For every* $[\Gamma\ \mathsf{ctx}] \in \mathsf{Obj}\,\mathbb{C}_T$*, the slice category*

$$\langle \mathcal{M}^T_\Gamma ; \{([\Gamma \vdash \mathcal{U}_i]\,, [\Gamma\ \mathsf{ctx}])\}_{i \in \mathbb{N}} \rangle$$

has universes.

Proof. First $([\Gamma \vdash \mathcal{U}_i]\,, [\Gamma\ \mathsf{ctx}]) = !_{[\Gamma\ \mathsf{ctx}]} \circ \epsilon_{\vdash \mathcal{U}_i : \mathcal{U}_{i+1}}$ with $!_{[\Gamma\ \mathsf{ctx}]}$ the unique arrow from the initial object $[\bullet\ \mathsf{ctx}]$, see Fact 5.3.7. Thus $([\Gamma \vdash \mathcal{U}_i]\,, [\Gamma\ \mathsf{ctx}]) \in \mathsf{Obj}\,\mathcal{M}^T_\Gamma$.

Consider $\langle \mathcal{M}^T_\Gamma ; ([\Gamma \vdash \mathcal{U}_0]\,, [\Gamma\ \mathsf{ctx}]) \rangle$: for every $i \in \mathbb{N}$, $(\Gamma \vdash \mathcal{U}_i : \mathcal{U}_{i+1}) \in T$, so $([\Gamma \vdash \mathcal{U}_i]\,, [\Gamma\ \mathsf{ctx}])$ is a weak universe by a straightforward induction.

Also, $\langle \mathcal{M}^T_\Gamma ; ([\Gamma \vdash \mathcal{U}_0]\,, [\Gamma\ \mathsf{ctx}]) \rangle$ has weak universes because, for every $c \in \mathsf{Obj}\,\mathcal{M}^T_\Gamma$,

- either $\mathsf{dom}(c) = [\Delta\ \mathsf{ctx}]$ and thus $\Delta \subseteq \Gamma$, then $([\Gamma \vdash \mathcal{U}_i]\,, [\Delta\ \mathsf{ctx}])$ is an arrow of $\mathcal{M}^T_\Gamma$ for each $i \in \mathbb{N}$, whose codomain is c, obtained composing $\epsilon_{\Delta \vdash \mathcal{U}_i : \mathcal{U}_{i+1}}$ with the ν maps of the difference $\Gamma \setminus \Delta$;
- or $\mathsf{dom}(c) = [\Gamma \vdash a]$ and thus there is $(\Gamma \vdash a : A) \in T$, so $(\Gamma \vdash A : \mathcal{U}_i) \in T$ for some $i \in \mathbb{N}$ by Proposition 3.3.9, hence $[\Gamma \vdash a : A] \circ [\Gamma \vdash A : \mathcal{U}_i]$ is an arrow of $\mathcal{M}^T_\Gamma$ whose codomain is c.

Since $(\Gamma \vdash \mathcal{U}_i : \mathcal{U}_{i+1}) \in T$, the arrow $[\Gamma \vdash \mathcal{U}_i : \mathcal{U}_{i+1}] : ([\Gamma \vdash \mathcal{U}_{i+1}]\,, [\Gamma\ \mathsf{ctx}]) \to ([\Gamma \vdash \mathcal{U}_i]\,, [\Gamma\ \mathsf{ctx}])$ is in $\mathcal{M}^T_\Gamma$ for every $i \in \mathbb{N}$. Moreover, by Proposition 3.3.21, this arrow is indecomposable, so $\langle \mathcal{M}^T_\Gamma ; \{([\Gamma \vdash \mathcal{U}_i]\,, [\Gamma\ \mathsf{ctx}])\}_{i \in \mathbb{N}} \rangle$ is a category with universes. ☐

Definition 5.3.15. If T is the basic system, its *syntactical category* is $\mathbb{M}_T$ of Definition 5.3.4 extended with the following families of functors. Since the groupoids of terms are groupoids, i.e., categories of isomorphisms, it suffices to define the functors for Π on the objects of their category of definition. Whenever $(\Gamma \vdash A : \mathcal{U}_i) \in T$,

$$\begin{aligned}&\mathcal{F}_T([\Gamma\ \mathsf{ctx}], [\Gamma \vdash A])\\ &\quad (([\Gamma, x : A \vdash B], [\Gamma, x : A\ \mathsf{ctx}])) = ([\Gamma \vdash \Pi x : A.B], [\Gamma\ \mathsf{ctx}])\ ,\end{aligned}$$

and for each instance of a Π-type

$$\begin{aligned}&\mathcal{I}_T([\Gamma\ \mathsf{ctx}], [\Gamma \vdash A], [\Gamma, x : A \vdash B])\\ &\quad (([\Gamma, x : A \vdash b], [\Gamma, x : A\ \mathsf{ctx}])) = ([\Gamma \vdash \lambda x : A.b], [\Gamma\ \mathsf{ctx}])\ ,\\ &\mathcal{E}_T([\Gamma\ \mathsf{ctx}], [\Gamma \vdash A], [\Gamma, x : A \vdash B])\\ &\quad (([\Gamma \vdash f], [\Gamma\ \mathsf{ctx}]), ([\Gamma \vdash a], [\Gamma\ \mathsf{ctx}])) = ([\Gamma \vdash f\,a], [\Gamma\ \mathsf{ctx}])\ .\end{aligned}$$

Define $\chi^{[\Gamma\ \mathsf{ctx}]}_{[\Gamma \vdash A]} = [\Gamma, x : A \vdash x]$.

Definition 5.3.16. The canonical interpretation $[\![\cdot]\!]_T$ over T is defined as follows:

- if $(\Gamma\ \mathsf{ctx}) \in T$ then $[\![\Gamma\ \mathsf{ctx}]\!]_T = !_{[\Gamma\ \mathsf{ctx}]}$;
- if there is A such that $(\Gamma \vdash a : A) \in T$, then $[\![\Gamma \vdash a]\!]_T = ([\Gamma \vdash a], [\Gamma\ \mathsf{ctx}])$;
- if $(\Gamma \vdash a : A) \in T$ then $[\![\Gamma \vdash a : A]\!]_T = ([\Gamma \vdash A], [\Gamma \vdash a])$;
- if $(\Gamma \vdash a \equiv b : A) \in T$ then $[\![\Gamma \vdash a \equiv b : A]\!]_T = ([\Gamma \vdash a], [\Gamma \vdash b])$.

Proposition 5.3.17. *The category*

$$\mathfrak{M}_T = \langle \mathbb{M}_T; \mathbb{C}_T, \{([\Gamma \vdash \mathcal{U}_i], [\Gamma\ \mathsf{ctx}])\}_{i\in\mathbb{N}}, \nu, [\bullet\ \mathsf{ctx}], \mathcal{F}_T, \mathcal{I}_T, \mathcal{E}_T \rangle$$

is a ML-category and $[\![\cdot]\!]_T$ is an interpretation over it.

Proof. To simplify the presentation, the proof follows the points of Definition 5.1.19.

1. Fact 5.3.5, Definition 5.3.6, and Fact 5.3.7 prove the point to hold.
2. Fact 5.3.10 and Proposition 5.3.14 prove the point.
3. Fact 5.3.7 proves that ν_A^c is indecomposable, and, evidently, for every $c = [x_1 : A_1, \ldots, x_n : A_n\ \mathsf{ctx}] \in \mathsf{Obj}\,\mathbb{C}_T$, $!_c = \nu_{A_n} \circ \cdots \circ \nu_{A_1}$.
4. Fact 5.3.8 proves the point.
5. Fact 5.3.9 proves the point.
6. Since $(\Gamma, x : A \vdash x : A) \in T$ implies $(\Gamma \vdash A : \mathcal{U}_i) \in T$ by Lemma 3.3.7 and Proposition 3.3.9, $\chi_{[\Gamma\vdash A]}^{[\Gamma\ \mathsf{ctx}]}$ has type $[\Gamma \vdash A]$ as required. Moreover, by Proposition 3.3.14, $\chi_{[\Gamma\vdash A]}^{[\Gamma\ \mathsf{ctx}]} \neq \nu_A \circ e$ for every term e in $\mathcal{M}_{[\Gamma\ \mathsf{ctx}]}$.
7. By Definition 5.3.15, $\mathcal{F}_T$ is appropriately indexed and
   a) each instance has the right domain and codomain;
   b) if $(\Gamma \vdash A : \mathcal{U}_i) \in T$, $(\Gamma \vdash \Pi x : A. B : \mathcal{U}_j) \in T$, and $k = \max(i, j)$, then $\mathcal{F}_T([\Gamma\ \mathsf{ctx}], [\Gamma \vdash A])(([\Gamma, x : A \vdash B], [\Gamma, x : A\ \mathsf{ctx}])) = ([\Gamma \vdash \Pi x : A. B], [\Gamma\ \mathsf{ctx}])$ is a term of type $[\Gamma \vdash \mathcal{U}_k]$ because of Point (3) of Definition 5.3.3;
   c) if $\Gamma \sim \Delta$, $[\Gamma \vdash A] \cong [\Delta \vdash C]$ and $([\Gamma, x : A \vdash B], [\Gamma, x : A\ \mathsf{ctx}]) \cong ([\Delta, y : C \vdash D], [\Delta, y : C\ \mathsf{ctx}])$,

$$\begin{aligned} &\mathcal{F}_T([\Gamma\ \mathsf{ctx}], [\Gamma \vdash A])(([\Gamma, x : A \vdash B], [\Gamma, x : A\ \mathsf{ctx}])) \\ &= ([\Gamma \vdash \Pi x : A. B], [\Gamma\ \mathsf{ctx}]) \\ &\cong ([\Delta \vdash \Pi y : C. D], [\Delta\ \mathsf{ctx}]) \\ &= \mathcal{F}_T([\Delta\ \mathsf{ctx}], [\Delta \vdash C])(([\Delta, y : C \vdash D], [\Delta, y : C\ \mathsf{ctx}])) \end{aligned}$$

   by closure under the $\Pi{-}\mathsf{form}{-}\mathsf{eq}$ rule and Fact 5.2.6; the condition on arrows is automatic since every functor preserves isomorphisms and the domain and codomain are groupoids;

d) if z does not occur in A and B,

$$\begin{aligned}&\mathcal{F}_T([\Gamma, z : C \text{ ctx}], [\Gamma, z : C \vdash A])\\&\quad(([\Gamma, z : C, x : A \vdash B], [\Gamma, z : C, x : A \text{ ctx}]))\\&= ((\Pi x : A.\, B, [AV(\Pi x : A.\, B) \text{ ctx}]), [\Gamma, z : C \text{ ctx}])\\&= ([\Gamma \text{ ctx}], [\Gamma, z : C \text{ ctx}]) \circ\\&\quad((\Pi x : A.\, B, [AV(\Pi x : A.\, B) \text{ ctx}]), [\Gamma \text{ ctx}])\\&= \nu_C \circ \mathcal{F}_T([\Gamma \text{ ctx}], [\Gamma \vdash A])(([\Gamma, x : A \vdash B], [\Gamma, x : A \text{ ctx}]))\ ;\end{aligned}$$

e) let $h\colon a \to b$ be an isomorphism such that $\mathcal{F}_T([\Gamma \text{ ctx}], [\Gamma \vdash A])(g) = h$ for some arrow g. Then,

$$\begin{aligned}h = (&\mathcal{F}_T([\Gamma \text{ ctx}], [\Gamma \vdash A])(([\Gamma, x : A \vdash B], [\Gamma, x : A \text{ ctx}])),\\&\mathcal{F}_T([\Gamma \text{ ctx}], [\Gamma \vdash A])(([\Gamma, x : A \vdash C], [\Gamma, x : A \text{ ctx}])))\end{aligned}$$

and $g = ([\Gamma, x : A \vdash B], [\Gamma, x : A \vdash C])$. By considering the inverse image h^{-1}, $g^{-1} = ([\Gamma, x : A \vdash C], [\Gamma, x : A \vdash B])$, which has to exist in the domain, showing that g is an isomorphism.

8. By Definition 5.3.15, $\mathcal{I}_T$ is appropriately indexed and

a) $\mathcal{I}_T([\Gamma \text{ ctx}], [\Gamma \vdash A], [\Gamma, x : A \vdash B])(([\Gamma, x : A \vdash b], [\Gamma, x : A \text{ ctx}])) = ([\Gamma \vdash \lambda x : A.\, b], [\Gamma \text{ ctx}])$ is a term of type $[\Gamma \vdash \Pi x : A.\, B]$ because of Point (3) of Definition 5.3.3;

b) calculating

$$\begin{aligned}&\mathcal{I}_T([\Gamma \text{ ctx}], [\Gamma \vdash A], [\Gamma, x : A \vdash \Pi x_1 : D_1. \cdots .\Pi x_l : D_l.\mathcal{U}_j])\\&\quad(([\Gamma, x : A \vdash b], [\Gamma, x : A \text{ ctx}]))\\&= ([\Gamma \vdash \lambda x : A.\, b], [\Gamma \text{ ctx}])\\&= \mathcal{I}_T([\Gamma \text{ ctx}], [\Gamma \vdash A], [\Gamma, x : A \vdash \Pi x_1 : D_1. \cdots .\Pi x_l : D_l.\mathcal{U}_k])\\&\quad(([\Gamma, x : A \vdash b], [\Gamma, x : A \text{ ctx}]))\end{aligned}$$

c) as already noticed in the case of $\mathcal{F}_T$, it suffices to prove that $\mathcal{I}_T$ respects isomorphisms on objects.
If $\Gamma \sim \Delta$, $[\Gamma \vdash A] \cong [\Delta \vdash C]$, $[\Gamma, x : A \vdash B] \cong [\Delta, x : C \vdash D]$ and $[\Gamma, x : A \vdash b] \cong [\Delta, x : C \vdash d]$,

$$\begin{aligned}&\mathcal{I}_T([\Gamma \text{ ctx}], [\Gamma \vdash A], [\Gamma, x : A \vdash B])\\&\quad(([\Gamma, x : A \vdash b], [\Gamma, x : A \text{ ctx}]))\\&= ([\Gamma \vdash \lambda x : A.\, b], [\Gamma \text{ ctx}])\\&\cong ([\Delta \vdash \lambda x : C.\, d], [\Delta \text{ ctx}])\\&= \mathcal{I}_T([\Delta \text{ ctx}], [\Delta \vdash C], [\Delta, x : C \vdash D])\\&\quad(([\Delta, x : C \vdash d], [\Delta, x : C \text{ ctx}]))\end{aligned}$$

by closure under the Π−intro−eq rule and Fact 5.2.6;

d) if z does not occur in A, B, and b,

$$\mathcal{I}_T([\Gamma, z : C \text{ ctx}], [\Gamma, z : C \vdash A], [\Gamma, z : C \vdash B])$$

$$\begin{aligned}
&(([\Gamma, z : C, x : A \vdash b], [\Gamma, z : C, x : A \ \mathsf{ctx}])) \\
&= ((\lambda x : A.\, b, [AV(\lambda x : A.\, b) \ \mathsf{ctx}]), [\Gamma, z : C \ \mathsf{ctx}]) \\
&= ([\Gamma \ \mathsf{ctx}], [\Gamma, z : C \ \mathsf{ctx}]) \circ \\
&\quad ((\lambda x : A.\, b, [AV(\lambda x : A.\, b) \ \mathsf{ctx}]), [\Gamma \ \mathsf{ctx}]) \\
&= \nu_C \circ \mathcal{I}_T([\Gamma \ \mathsf{ctx}], [\Gamma \vdash A], [\Gamma, x : A \vdash B]) \\
&\qquad\quad (([\Gamma, x : A \vdash B], [\Gamma, x : A \ \mathsf{ctx}])) \ \ ;
\end{aligned}$$

e) let $(\Gamma \vdash f : \Pi x : A.\, B) \in T$ and define

$$\mathcal{J}_T([\Gamma \ \mathsf{ctx}], [\Gamma \vdash A], [\Gamma, x : A \vdash B])$$

as a functor

$$\begin{aligned}
&(\mathbb{M}_T/[\Gamma \ \mathsf{ctx}] \uparrow ([\Gamma \vdash \Pi x : A.\, B], [\Gamma \ \mathsf{ctx}])) \\
&\to (\mathbb{M}_T/[\Gamma, x : A \ \mathsf{ctx}] \uparrow \mathcal{F}_T^{-1}([\Gamma \ \mathsf{ctx}], [\Gamma \vdash A]) \\
&\qquad\qquad\qquad\qquad (([\Gamma \vdash \Pi x : A.\, B], [\Gamma \ \mathsf{ctx}]))
\end{aligned}$$

such that

$$\begin{aligned}
&\mathcal{J}_T([\Gamma \ \mathsf{ctx}], [\Gamma \vdash A], [\Gamma, x : A \vdash B])(([\Gamma \vdash f], [\Gamma \ \mathsf{ctx}])) \\
&= ([\Gamma, x : A \vdash f\, x], [\Gamma, x : A \ \mathsf{ctx}]) \ \ ,
\end{aligned}$$

which suffices to define $\mathcal{J}_T$ as it acts on groupoids in an iso-stable category. Hence

$$\begin{aligned}
&\big(\mathcal{J}_T([\Gamma \ \mathsf{ctx}], [\Gamma \vdash A], [\Gamma, x : A \vdash B]) \circ \\
&\ \ \mathcal{I}_T([\Gamma \ \mathsf{ctx}], [\Gamma \vdash A], [\Gamma, x : A \vdash B])\big) \\
&(([\Gamma, x : A \vdash b], [\Gamma, x : A \ \mathsf{ctx}])) \\
&\cong ([\Gamma, x : A \vdash (\lambda x : A.\, b)\, x], [\Gamma, x : A \ \mathsf{ctx}]) \\
&\cong ([\Gamma, x : A \vdash b], [\Gamma, x : A \ \mathsf{ctx}])
\end{aligned}$$

by closure under the $\Pi-\mathsf{comp}$ rule, and

$$\begin{aligned}
&\big(\mathcal{I}_T([\Gamma \ \mathsf{ctx}], [\Gamma \vdash A], [\Gamma, x : A \vdash B]) \circ \\
&\ \ \mathcal{J}_T([\Gamma \ \mathsf{ctx}], [\Gamma \vdash A], [\Gamma, x : A \vdash B])\big)(([\Gamma \vdash f], [\Gamma \ \mathsf{ctx}])) \\
&\cong ([\Gamma \vdash \lambda x : A.\, f\, x], [\Gamma \ \mathsf{ctx}]) \cong ([\Gamma \vdash f], [\Gamma \ \mathsf{ctx}])
\end{aligned}$$

by closure under the $\Pi-\mathsf{uniq}$ rule. So $\mathcal{I}_T([\Gamma \ \mathsf{ctx}], [\Gamma \vdash A], [\Gamma, x : A \vdash B])$ is an equivalence of categories whose inverse is $\mathcal{J}_T([\Gamma \ \mathsf{ctx}], [\Gamma \vdash A], [\Gamma, x : A \vdash B])$.

9. By Definition 5.3.15, $\mathcal{E}_T$ is appropriately indexed and

   a) assuming $\Gamma \vdash a : A$, $\Gamma \vdash A : \mathcal{U}_i$, $\Gamma, x : A \vdash b : B$, and $\Gamma, x : A \vdash B : \mathcal{U}_i$, $\mathcal{E}_T([\Gamma \ \mathsf{ctx}], [\Gamma \vdash A], [\Gamma, x{:}A \vdash B])(([\Gamma, x{:}A \vdash b], [\Gamma, x{:}A \ \mathsf{ctx}])) = ([\Gamma \vdash \lambda x : A.\, b], [\Gamma \ \mathsf{ctx}])$ is a term of type $[\Gamma \vdash \Pi x : A.\, B]$ in the universe $\mathcal{U}_i$ because of Point (3) of Definition 5.3.3;

   b) $\mathcal{E}_T([\Gamma \ \mathsf{ctx}], [\Gamma \vdash A], [\Gamma, x : A \vdash B])(([\Gamma \vdash f], [\Gamma \ \mathsf{ctx}]), ([\Gamma \vdash a], [\Gamma \ \mathsf{ctx}]))$ is a term of type $B[a/x]$ because of conditions (3) and (4) in Definition 5.3.3, and closure under the $\Pi-\mathsf{elim}$ rule;

c) calculating,

$$\begin{aligned}
&\mathcal{E}_T([\Gamma\ \mathsf{ctx}], [\Gamma \vdash A], [\Gamma, x : A \vdash \Pi x_1 : D_1. \cdots . \Pi x_l : D_l.\mathcal{U}_j])\\
&\quad(([\Gamma \vdash f], [\Gamma\ \mathsf{ctx}]), ([\Gamma \vdash a], [\Gamma\ \mathsf{ctx}]))\\
&= ([\Gamma \vdash f\,a], [\Gamma\ \mathsf{ctx}])\\
&= \mathcal{E}_T([\Gamma\ \mathsf{ctx}], [\Gamma \vdash A], [\Gamma, x : A \vdash \Pi x_1 : D_1. \cdots . \Pi x_l : D_l.\mathcal{U}_k])\\
&\quad(([\Gamma \vdash f], [\Gamma\ \mathsf{ctx}]), ([\Gamma \vdash a], [\Gamma\ \mathsf{ctx}]))
\end{aligned}$$

and

$$\begin{aligned}
&\mathcal{E}_T([\Gamma\ \mathsf{ctx}], [\Gamma \vdash \Pi x_1 : D_1. \cdots . \Pi x_l : D_l.\mathcal{U}_j], [\Gamma, x : A \vdash B])\\
&\quad(([\Gamma \vdash f], [\Gamma\ \mathsf{ctx}]), ([\Gamma \vdash a], [\Gamma\ \mathsf{ctx}]))\\
&= ([\Gamma \vdash f\,a], [\Gamma\ \mathsf{ctx}])\\
&= \mathcal{E}_T([\Gamma\ \mathsf{ctx}], [\Gamma \vdash \Pi x_1 : D_1. \cdots . \Pi x_l : D_l.\mathcal{U}_k], [\Gamma, x : A \vdash B])\\
&\quad(([\Gamma \vdash f], [\Gamma\ \mathsf{ctx}]), ([\Gamma \vdash a], [\Gamma\ \mathsf{ctx}]))\ ;
\end{aligned}$$

d) as already noticed in the case of $\mathcal{F}_T$ and $\mathcal{I}_T$, it suffices to prove that $\mathcal{E}_T$ respects isomorphisms on objects. If $\Gamma \sim \Delta$, $[\Gamma \vdash A] \cong [\Delta \vdash C]$, $[\Gamma, x : A \vdash B] \cong [\Delta, x : C \vdash D]$, $[\Gamma \vdash f] \cong [\Delta \vdash g]$, and $[\Gamma \vdash a] \cong [\Delta \vdash c]$,

$$\begin{aligned}
&\mathcal{E}_T([\Gamma\ \mathsf{ctx}], [\Gamma \vdash A], [\Gamma, x : A \vdash B])\\
&\quad(([\Gamma \vdash f], [\Gamma\ \mathsf{ctx}]), ([\Gamma \vdash a], [\Gamma\ \mathsf{ctx}]))\\
&= ([\Gamma \vdash f\,a], [\Gamma\ \mathsf{ctx}])\\
&\cong ([\Delta \vdash g\,c], [\Delta\ \mathsf{ctx}])\\
&= \mathcal{E}_T([\Delta\ \mathsf{ctx}], [\Delta \vdash C], [\Delta, x : C \vdash D])\\
&\quad(([\Delta \vdash g], [\Delta\ \mathsf{ctx}]), ([\Delta \vdash c], [\Delta\ \mathsf{ctx}]))
\end{aligned}$$

by closure under the $\Pi-\mathsf{elim}-\mathsf{eq}$ rule and Fact 5.2.6;

e) if z does not occur in A, B, f and a,

$$\begin{aligned}
&\mathcal{E}_T([\Gamma, z : C\ \mathsf{ctx}], [\Gamma, z : C \vdash A], [\Gamma, z : C \vdash B])\\
&\quad(([\Gamma, z : C \vdash f], [\Gamma, z : C\ \mathsf{ctx}]), ([\Gamma, z : C \vdash a], [\Gamma, z : C\ \mathsf{ctx}]))\\
&= ((f\,a, [AV(f\,a)\ \mathsf{ctx}]), [\Gamma, z : C\ \mathsf{ctx}])\\
&= ([\Gamma\ \mathsf{ctx}], [\Gamma, z : C\ \mathsf{ctx}]) \circ ((f\,a, [AV(f\,a)\ \mathsf{ctx}]), [\Gamma\ \mathsf{ctx}])\\
&= \nu_C \circ \mathcal{E}_T([\Gamma\ \mathsf{ctx}], [\Gamma \vdash A], [\Gamma, x : A \vdash B])\\
&\qquad(([\Gamma \vdash f], [\Gamma\ \mathsf{ctx}]), ([\Gamma \vdash a], [\Gamma\ \mathsf{ctx}]))\ .
\end{aligned}$$

10. the auxiliary family of functors S is indexed as $S([\Gamma\ \mathsf{ctx}], [\Gamma \vdash A], [\Gamma, x{:}A \vdash B])$ with A a type in $\Gamma\ \mathsf{ctx}$ and B a type in $\Gamma, x : A\ \mathsf{ctx}$, and

a) $S([\Gamma\ \mathsf{ctx}], [\Gamma \vdash A], [\Gamma, x : A \vdash B])(([\Gamma, x : A \vdash b], [\Gamma, x : A\ \mathsf{ctx}]), ([\Gamma \vdash a], [\Gamma\ \mathsf{ctx}]))$ is defined to be

$$\begin{aligned}
&\mathcal{E}_T([\Gamma\ \mathsf{ctx}], [\Gamma \vdash A], [\Gamma, x : A \vdash B])\\
&\quad(\mathcal{I}_T([\Gamma\ \mathsf{ctx}], [\Gamma \vdash A], [\Gamma, x : A \vdash B])\\
&\qquad(([\Gamma, x : A \vdash b], [\Gamma, x : A\ \mathsf{ctx}])), ([\Gamma \vdash a], [\Gamma\ \mathsf{ctx}]))
\end{aligned}$$

$$\begin{aligned}&\cong \mathcal{E}_T([\Gamma\ \mathsf{ctx}],[\Gamma\vdash A],[\Gamma,x:A\vdash B])\\&\qquad(([\Gamma\vdash \lambda x:A.\,b],[\Gamma\ \mathsf{ctx}]),([\Gamma\vdash a],[\Gamma\ \mathsf{ctx}]))\\&\cong([\Gamma\vdash(\lambda x:A.\,b)\,a],[\Gamma\ \mathsf{ctx}])\cong([\Gamma\vdash b[a/x]],[\Gamma\ \mathsf{ctx}])\end{aligned}$$

by closure under the $\Pi{-}\mathsf{comp}$ rule;

b) calculating,

$$\begin{aligned}&S([\Gamma\ \mathsf{ctx}],[\Gamma\vdash A],[\Gamma,x:A\vdash A])\\&\quad(([\Gamma,x:A\vdash x],[\Gamma,x:A\ \mathsf{ctx}]),([\Gamma\vdash a],[\Gamma\ \mathsf{ctx}]))\\&\cong([\Gamma\vdash x[a/x]],[\Gamma\ \mathsf{ctx}])=([\Gamma\vdash a],[\Gamma\ \mathsf{ctx}])\ \ ;\end{aligned}$$

c) if $\Gamma\vdash p:B$, i.e., x does not occur in p,

$$\begin{aligned}&S([\Gamma\ \mathsf{ctx}],[\Gamma\vdash A],[\Gamma,x:A\vdash B])\\&\quad(([\Gamma,x:A\vdash p],[\Gamma,x:A\ \mathsf{ctx}]),([\Gamma\vdash a],[\Gamma\ \mathsf{ctx}]))\\&\cong([\Gamma\vdash p[a/x]],[\Gamma\ \mathsf{ctx}])=([\Gamma\vdash p],[\Gamma\ \mathsf{ctx}])\ \ ;\end{aligned}$$

d) calculating, it is long but easy to verify the condition on substitution applied to the Π-elimination functor:

$$\begin{aligned}&\mathcal{E}_T([\Gamma,x:A\ \mathsf{ctx}],[\Gamma,x:A\vdash Q],[\Gamma,x:A,y:Q\vdash P])\\&\qquad(([\Gamma,x:A\vdash p],[\Gamma,x:A\ \mathsf{ctx}]),([\Gamma,x:A\vdash q],[\Gamma,x:A\ \mathsf{ctx}]))\\&=([\Gamma,x:A\vdash p\,q],[\Gamma,x:A\ \mathsf{ctx}])\ \ ,\end{aligned}$$

$$\begin{aligned}Q_a&=S([\Gamma\ \mathsf{ctx}],[\Gamma\vdash A],[\Gamma,x:A\vdash \mathcal{U}_i])\\&\qquad(([\Gamma,x:A\vdash Q],[\Gamma,x:A\ \mathsf{ctx}]),([\Gamma\vdash a],[\Gamma\ \mathsf{ctx}]))\\&\cong([\Gamma\vdash Q[a/x]],[\Gamma\ \mathsf{ctx}])\end{aligned}$$

then

$$\begin{aligned}&S([\Gamma\ \mathsf{ctx}],[\Gamma\vdash A],\\&\qquad S([\Gamma,x:A\ \mathsf{ctx}],[\Gamma,x:A\vdash Q],[\Gamma,x:A,y:Q\vdash \mathcal{U}_i])\\&\qquad\quad(([\Gamma,x:A,y:Q\vdash P],[\Gamma,x:A,y:Q,\ \mathsf{ctx}]),\\&\qquad\qquad([\Gamma,x:A\vdash q],[\Gamma,x:A\ \mathsf{ctx}])))\\&\quad(([\Gamma,x:A\vdash p\,q],[\Gamma,x:\ \mathsf{ctx}]),([\Gamma\vdash a],[\Gamma\ \mathsf{ctx}]))\\&\cong S([\Gamma\ \mathsf{ctx}],[\Gamma\vdash A],[\Gamma,x:A\vdash P[q/y]])\\&\quad(([\Gamma,x:A\vdash p\,q],[\Gamma,x:\ \mathsf{ctx}]),([\Gamma\vdash a],[\Gamma\ \mathsf{ctx}]))\\&\cong([\Gamma\vdash(p\,q)[a/x]],[\Gamma\ \mathsf{ctx}])\\&\cong([\Gamma\vdash p[a/x]\,q[a/x]],[\Gamma\ \mathsf{ctx}])\\&\cong\mathcal{E}_T([\Gamma\ \mathsf{ctx}],[\Gamma\vdash Q[a/x]],[\Gamma,y:Q[a/x]\vdash P[a/x]])\\&\qquad(([\Gamma,y:Q[a/x]\vdash p[a/x]],[\Gamma,y:Q[a/x]\ \mathsf{ctx}]),\\&\qquad\ ([\Gamma\vdash q[a/x]],[\Gamma\ \mathsf{ctx}]))\\&\cong\mathcal{E}_T([\Gamma\ \mathsf{ctx}],Q_a,\\&\qquad\mathcal{F}_T^{-1}([\Gamma\ \mathsf{ctx}],Q_a)\\&\qquad\quad(([\Gamma\vdash\Pi y:Q[a/x].\,P[a/x]],[\Gamma\ \mathsf{ctx}])))\end{aligned}$$

$$
\begin{aligned}
&\qquad (S([\Gamma\ \mathsf{ctx}],[\Gamma\vdash A],[\Gamma,x:A\vdash \Pi y:Q[a/x].\,P[a/x]])\\
&\qquad\quad (([\Gamma,x:A\vdash p],[\Gamma,x:A\ \mathsf{ctx}]),([\Gamma\vdash a],[\Gamma\ \mathsf{ctx}])),\\
&\qquad S([\Gamma\ \mathsf{ctx}],[\Gamma\vdash A],[\Gamma,x:A\vdash Q])\\
&\qquad\quad (([\Gamma,x:A\vdash q],[\Gamma,x:A\ \mathsf{ctx}]),([\Gamma\vdash a],[\Gamma\ \mathsf{ctx}])))\\
&\cong \mathcal{E}_T([\Gamma\ \mathsf{ctx}],Q_a,\\
&\qquad \mathcal{F}_T^{-1}([\Gamma\ \mathsf{ctx}],Q_a)\\
&\qquad\qquad (S([\Gamma\ \mathsf{ctx}],[\Gamma\vdash A],[\Gamma,x:A\vdash \mathcal{U}_i])\\
&\qquad\qquad\quad (([\Gamma,x:A\vdash \Pi y:Q.\,P],[\Gamma,x:A\ \mathsf{ctx}]),\\
&\qquad\qquad\quad ([\Gamma\vdash a],[\Gamma\ \mathsf{ctx}]))))\\
&\qquad (S([\Gamma\ \mathsf{ctx}],[\Gamma\vdash A],[\Gamma,x:A\vdash \Pi y:Q[a/x].\,P[a/x]])\\
&\qquad\quad (([\Gamma,x:A\vdash p],[\Gamma,x:A\ \mathsf{ctx}]),([\Gamma\vdash a],[\Gamma\ \mathsf{ctx}])),\\
&\qquad S([\Gamma\ \mathsf{ctx}],[\Gamma\vdash A],[\Gamma,x:A\vdash Q])\\
&\qquad\quad (([\Gamma,x:A\vdash q],[\Gamma,x:A\ \mathsf{ctx}]),([\Gamma\vdash a],[\Gamma\ \mathsf{ctx}])))\\
&\cong \mathcal{E}_T([\Gamma\ \mathsf{ctx}],Q_a,\\
&\qquad \mathcal{F}_T^{-1}([\Gamma\ \mathsf{ctx}],Q_a)\\
&\qquad\qquad (S([\Gamma\ \mathsf{ctx}],[\Gamma\vdash A],[\Gamma,x:A\vdash \mathcal{U}_i])\\
&\qquad\qquad\quad (\mathcal{F}_T([\Gamma,x:A\ \mathsf{ctx}],[\Gamma,x:A\vdash Q])\\
&\qquad\qquad\qquad (([\Gamma,x:A,y:Q\vdash P],[\Gamma,x:A,y:Q\ \mathsf{ctx}])),\\
&\qquad\qquad\qquad ([\Gamma\vdash a],[\Gamma\ \mathsf{ctx}]))))\\
&\qquad (S([\Gamma\ \mathsf{ctx}],[\Gamma\vdash A],[\Gamma,x:A\vdash \Pi y:Q[a/x].\,P[a/x]])\\
&\qquad\quad (([\Gamma,x:A\vdash p],[\Gamma,x:A\ \mathsf{ctx}]),([\Gamma\vdash a],[\Gamma\ \mathsf{ctx}])),\\
&\qquad S([\Gamma\ \mathsf{ctx}],[\Gamma\vdash A],[\Gamma,x:A\vdash Q])\\
&\qquad\quad (([\Gamma,x:A\vdash q],[\Gamma,x:A\ \mathsf{ctx}]),([\Gamma\vdash a],[\Gamma\ \mathsf{ctx}])))\ .
\end{aligned}
$$

e) calculating,

$$
\begin{aligned}
&\mathcal{I}_T([\Gamma,x:A\ \mathsf{ctx}],[\Gamma,x:A\vdash D],[\Gamma,x:A,y:D\vdash B])\\
&\quad (([\Gamma,x:A,y:D\vdash b],[\Gamma,x:A,y:D\ \mathsf{ctx}]))\\
&= ([\Gamma,x:A\vdash \lambda y:D.\,b],[\Gamma,x:A\ \mathsf{ctx}])\ ,
\end{aligned}
$$

$$
\begin{aligned}
D_a &= S([\Gamma\ \mathsf{ctx}],[\Gamma\vdash A],[\Gamma,x:A\vdash \mathcal{U}_i])\\
&\quad (([\Gamma,x:A\vdash D],[\Gamma,x:A\ \mathsf{ctx}]),([\Gamma\vdash a],[\Gamma\ \mathsf{ctx}]))\\
&\cong ([\Gamma\vdash D[a/x]],[\Gamma\ \mathsf{ctx}])\ ,
\end{aligned}
$$

$$
\begin{aligned}
B_a &= \mathcal{F}_T^{-1}([\Gamma\ \mathsf{ctx}],D_a)\\
&\qquad (S([\Gamma\ \mathsf{ctx}],[\Gamma\vdash A],[\Gamma,x:A\vdash \mathcal{U}_i])\\
&\qquad\quad (\mathcal{F}_T([\Gamma,x:A\ \mathsf{ctx}],[\Gamma,x:A\vdash D])\\
&\qquad\qquad (([\Gamma,x:A,y:D\vdash B],[\Gamma,x:A,y:D\ \mathsf{ctx}])),\\
&\qquad\qquad ([\Gamma\vdash a],[\Gamma\ \mathsf{ctx}])))\\
&\cong ([\Gamma,y:D[a/x]\vdash B[a/x]],[\Gamma,y:D[a/x]\ \mathsf{ctx}])\ ,
\end{aligned}
$$

$$
b_a = \mathcal{I}_T^{-1}([\Gamma\ \mathsf{ctx}],B_a,D_a)
$$

$$\begin{aligned}
&(S([\Gamma\ \mathsf{ctx}], [\Gamma \vdash A], [\Gamma, x : A \vdash \mathcal{U}_i]) \\
&\quad(\mathcal{I}_T([\Gamma, x : A\ \mathsf{ctx}], [\Gamma, x : A \vdash D]) \\
&\qquad(([\Gamma, x : A, y : D \vdash b], [\Gamma, x : A, y : D\ \mathsf{ctx}])), \\
&\quad([\Gamma \vdash a], [\Gamma\ \mathsf{ctx}]))) \\
\cong\ &([\Gamma, y : D[a/x] \vdash B[a/x]], [\Gamma, y : D[a/x]\ \mathsf{ctx}]) \ .
\end{aligned}$$

Hence,

$$\begin{aligned}
&S([\Gamma\ \mathsf{ctx}], [\Gamma \vdash A], \mathcal{F}_T([\Gamma, x : A\ \mathsf{ctx}], [\Gamma, x : A \vdash D]) \\
&\qquad(([\Gamma, x : A, y : D \vdash B], [\Gamma, x : A, y : D\ \mathsf{ctx}]))) \\
&\quad(([\Gamma, x : A \vdash \lambda y : D.\, b], [\Gamma, x : A\ \mathsf{ctx}]), ([\Gamma \vdash a], [\Gamma\ \mathsf{ctx}])) \\
\cong\ &S([\Gamma\ \mathsf{ctx}], [\Gamma \vdash A], [\Gamma, x : A \vdash \Pi y : D.\, B]) \\
&\quad(([\Gamma, x : A \vdash \lambda y : D.\, b], [\Gamma, x : A\ \mathsf{ctx}]), ([\Gamma \vdash a], [\Gamma\ \mathsf{ctx}])) \\
\cong\ &([\Gamma \vdash (\lambda y : D.\, b)[a/x]], [\Gamma\ \mathsf{ctx}]) \\
\cong\ &([\Gamma \vdash \lambda y : D[a/x].\, b[a/x]], [\Gamma\ \mathsf{ctx}]) \\
\cong\ &\mathcal{I}_T([\Gamma\ \mathsf{ctx}], D_a, B_a)(b_a) \ .
\end{aligned}$$

f) calculating,

$$\begin{aligned}
&\mathcal{F}_T([\Gamma, x : A\ \mathsf{ctx}], [\Gamma, x : A \vdash D] \\
&\quad(([\Gamma, x : A, y : D \vdash B], [\Gamma, x : A, y : D\ \mathsf{ctx}])) \\
=\ &([\Gamma, x : A \vdash \Pi y : D.\, B], [\Gamma, x : A\ \mathsf{ctx}]) \ ,
\end{aligned}$$

$$\begin{aligned}
D_a &= S([\Gamma\ \mathsf{ctx}], [\Gamma \vdash A], [\Gamma, x : A \vdash \mathcal{U}_i]) \\
&\quad(([\Gamma, x : A \vdash D], [\Gamma, x : A\ \mathsf{ctx}]), ([\Gamma \vdash a], [\Gamma\ \mathsf{ctx}])) \\
&\cong ([\Gamma \vdash D[a/x]], [\Gamma\ \mathsf{ctx}]) \ ,
\end{aligned}$$

$$\begin{aligned}
B_a &= \mathcal{F}_T^{-1}([\Gamma\ \mathsf{ctx}], D_a) \\
&\qquad(S([\Gamma\ \mathsf{ctx}], [\Gamma \vdash A], [\Gamma, x : A \vdash \mathcal{U}_i]) \\
&\qquad\quad(\mathcal{F}_T([\Gamma, x : A\ \mathsf{ctx}], [\Gamma, x : A \vdash D]) \\
&\qquad\qquad(([\Gamma, x : A, y : D \vdash B], [\Gamma, x : A, y : D\ \mathsf{ctx}])), \\
&\qquad\qquad([\Gamma \vdash a], [\Gamma\ \mathsf{ctx}]))) \\
&\cong ([\Gamma, y : D[a/x] \vdash B[a/x]], [\Gamma, y : D[a/x]\ \mathsf{ctx}]) \ ,
\end{aligned}$$

then

$$\begin{aligned}
&S([\Gamma\ \mathsf{ctx}], [\Gamma \vdash A], [\Gamma, x : A \vdash \mathcal{U}_i]) \\
&\quad(([\Gamma, x : A \vdash \Pi y : D.\, B], [\Gamma, x : A\ \mathsf{ctx}]), ([\Gamma \vdash a], [\Gamma\ \mathsf{ctx}])) \\
\cong\ &([\Gamma \vdash (\Pi y : D.\, B)[a/x]], [\Gamma\ \mathsf{ctx}]) \\
\cong\ &([\Gamma \vdash \Pi y : D[a/x].\, B[a/x]], [\Gamma\ \mathsf{ctx}]) \\
\cong\ &\mathcal{F}_T([\Gamma\ \mathsf{ctx}], D_a)(B_a) \ .
\end{aligned}$$

g) by a simple calculation

$$S([\Gamma\ \mathsf{ctx}], [\Gamma \vdash A],$$

$$
\begin{aligned}
&S([\Gamma, x : A\ \mathsf{ctx}], [\Gamma, x : A \vdash B], [\Gamma, x : A, y : B \vdash \mathcal{U}_i])\\
&\quad(([\Gamma, x : A, y : B \vdash E], [\Gamma, x : A, y : B\ \mathsf{ctx}]),\\
&\quad\ ([\Gamma, x : A \vdash b], [\Gamma, x : A\ \mathsf{ctx}])))\\
&(S([\Gamma, x : A\ \mathsf{ctx}], [\Gamma, x : A \vdash B], [\Gamma, x : A, y : B \vdash E])\\
&\quad(([\Gamma, x : A, y : B \vdash e], [\Gamma, x : A, y : B\ \mathsf{ctx}]),\\
&\quad\ ([\Gamma, x : A \vdash b], [\Gamma, x : A\ \mathsf{ctx}])), ([\Gamma \vdash a], [\Gamma\ \mathsf{ctx}]))\\
\cong\ & S([\Gamma\ \mathsf{ctx}], [\Gamma \vdash A], [\Gamma, x : A \vdash E[b/y]])\\
&\quad(([\Gamma, x : A \vdash e[b/y]], [\Gamma, x : A\ \mathsf{ctx}]), ([\Gamma \vdash a], [\Gamma\ \mathsf{ctx}]))\\
\cong\ & ([\Gamma \vdash e[b/y][a/x]], [\Gamma\ \mathsf{ctx}])\\
\cong\ & ([\Gamma \vdash e[a/x][b/y]], [\Gamma\ \mathsf{ctx}])\\
\cong\ & S([\Gamma\ \mathsf{ctx}], [\Gamma \vdash B], [\Gamma, y : B \vdash E[a/x]])\\
&\quad(([\Gamma, y : B \vdash e[a/x]], [\Gamma, y : B\ \mathsf{ctx}]), ([\Gamma \vdash a], [\Gamma\ \mathsf{ctx}]))\\
\cong\ & S([\Gamma\ \mathsf{ctx}], [\Gamma \vdash B],\\
&\quad S([\Gamma, y : B\ \mathsf{ctx}], [\Gamma, y : B \vdash A], [\Gamma, y : B, x : A \vdash \mathcal{U}_i])\\
&\qquad(([\Gamma, y : B, x : A \vdash E], [\Gamma, y : B, x : A\ \mathsf{ctx}]),\\
&\qquad\ ([\Gamma, y : B \vdash a], [\Gamma, y : B\ \mathsf{ctx}])))\\
&\quad(S([\Gamma, y : B\ \mathsf{ctx}], [\Gamma, y : B \vdash A], [\Gamma, y : B, x : A \vdash E])\\
&\qquad(([\Gamma, y : B, x : A \vdash e], [\Gamma, y : B, x : A\ \mathsf{ctx}]),\\
&\qquad\ ([\Gamma, y : B \vdash a], [\Gamma, y : B\ \mathsf{ctx}])), ([\Gamma \vdash b], [\Gamma\ \mathsf{ctx}]))\ .
\end{aligned}
$$

Finally, it is immediate to check that $[\![\cdot]\!]_T$ satisfies all the conditions to be an interpretation, see Definition 5.2.1. ☐

Theorem 5.3.18 (Classifying model)**.** *If $\mathfrak{M} = \langle \mathbb{M}, [\![\cdot]\!] \rangle$ is a model for T, then there is functor $\mathfrak{I} \colon \mathfrak{M}_T \to \mathfrak{M}$ which preserves the interpretation.*

Proof. Define $\mathfrak{I} \colon \mathbb{M}_T \to \mathbb{M}$ following Definitions 5.3.1 and 5.3.3:

- if $(\Gamma\ \mathsf{ctx}) \in T$, $\mathfrak{I}([\Gamma\ \mathsf{ctx}]) = \mathsf{cod}([\![\Gamma\ \mathsf{ctx}]\!])$;
- if $(\Gamma \vdash a : A) \in T$, $\mathfrak{I}([\Gamma \vdash a]) = \mathsf{dom}([\![\Gamma \vdash a]\!])$;
- if $(\Gamma, x : A\ \mathsf{ctx}) \in T$, $\mathfrak{I}(\nu_A) = \nu_A$ where the former ν_A is an arrow of $\mathbb{M}_T$ while the latter is an arrow of $\mathbb{M}$;
- if $(\Gamma \vdash a : A) \in T$, $\mathfrak{I}([\Gamma \vdash a : A]) = [\![\Gamma \vdash a : A]\!]$, and $\mathfrak{I}(\epsilon_{\Gamma \vdash a:A}) = [\![\mathsf{AV}(a) \vdash a]\!]$;
- if $(\Gamma \vdash a \equiv b : A) \in T$, $\mathfrak{I}([\Gamma \vdash a \equiv b : A]) = [\![\Gamma \vdash a \equiv b : A]\!]$;
- $\mathfrak{I}(\mathsf{id}_a) = \mathsf{id}_{\mathfrak{a}}$ and $\mathfrak{I}(f \circ g) = \mathfrak{I}(f) \circ \mathfrak{I}(g)$.

This definition is well given since it does not depend on the representatives in the equivalence classes. This fact follows from the definition of $\approx$, and Fact 5.2.6.

A simple check against Definition 5.3.16 shows that $\mathfrak{I}([\![\cdot]\!]_T) \cong [\![\cdot]\!]$. ☐

Theorem 5.3.19 (Completeness)**.** *Let γ be a judgement which is valid in every model for T. Then γ is derivable from T.*

Proof. Since γ is valid in every model for T, it is valid in $\mathfrak{M}_T$. Hence, it is an arrow of $\mathbb{M}_T$, thus, by Definition 5.3.3, $\gamma \in T$. Since T is closed under derivation in the basic system, γ is derivable from T. $\square$

Inductive types

Since the proof theory in Section 3.3 comprehends inductive types, all the constructions and results obtained so far for the basic system B trivially extend to any inductive theory T. Thus, we can extend the syntactical category for the basic system to T as follows

Definition 5.3.20. If T is an inductive theory, its *syntactical category* is the one of Definition 5.3.15 extended with the following families of functors; as for the basic system, it suffices to define the functors on the objects of their category of definition. For each instance of a formation, introduction and elimination rule for a type τ,

- $\mathcal{F}_\tau([\Gamma\ \mathsf{ctx}], [\Gamma \vdash \Pi\,(x:F)_n\,.\,\mathcal{U}_i])(\cdot) = ([\Gamma \vdash \tau], [\Gamma\ \mathsf{ctx}])$;
- $\mathcal{I}_{\tau,\mathbb{K}}([\Gamma\ \mathsf{ctx}], [\Gamma \vdash \Pi\,(x:F)_{n'}\,.\,\Pi\,(y:I)_m\,.\,\tau\, x'_1 \cdots x'_n])(\cdot)$
$= ([\Gamma \vdash \mathbb{K}], [\Gamma\ \mathsf{ctx}])$;
- $\mathcal{E}_\tau([\Gamma\ \mathsf{ctx}], [\Gamma \vdash T_{\mathsf{ind}}])(\cdot) = ([\Gamma \vdash \mathsf{ind}_\tau], [\Gamma\ \mathsf{ctx}])$.

The canonical interpretation $[\![\cdot]\!]_T$ over T extends the interpretation for the basic system of Definition 5.3.16 in the obvious way.

Proposition 5.3.21. *$\mathfrak{M}^i_T = \langle \mathfrak{M}_T, \{\mathcal{F}_\tau, \{\mathcal{I}_{\tau,\mathbb{K}}\}, \mathcal{E}_\tau\}\rangle$ is a T-ML-category, and $[\![\cdot]\!]_T$ is an interpretation over it.*

Proof. $\mathfrak{M}_T$ is a ML-category by Proposition 5.3.17, thus it is enough to show that the requirements of Definition 5.1.24 are satisfied. To simplify the presentation, the proof follows the points of the definition.

1. By Definition 5.3.20, $\mathcal{F}_\tau$ is appropriately indexed and
   a) each instance has the right domain and codomain;
   b) if $\Gamma \sim \Delta$, then by Fact 5.2.6
   $$\begin{aligned}&\mathcal{F}_\tau([\Gamma\ \mathsf{ctx}], [\Gamma \vdash \Pi\,(x:F)_n\,.\,\mathcal{U}_i])\\ &\cong \mathcal{F}_\tau([\Delta\ \mathsf{ctx}], [\Gamma \vdash \Pi\,(x:F)_n\,.\,\mathcal{U}_i])\ .\end{aligned}$$
2. By Definition 5.3.20, $\mathcal{I}_{\tau,\mathbb{K}}$ is appropriately indexed and
   a) each instance has the right domain and codomain;
   b) if $\Gamma \sim \Delta$, then by Fact 5.2.6
   $$\begin{aligned}&\mathcal{I}_{\tau,\mathbb{K}}([\Gamma\ \mathsf{ctx}], [\Gamma \vdash \Pi\,(x:F)_{n'}\,.\,\Pi\,(y:I)_m\,.\,\tau\, x'_1 \cdots x'_n])\\ &\cong \mathcal{I}_{\tau,\mathbb{K}}([\Delta\ \mathsf{ctx}], [\Gamma \vdash \Pi\,(x:F)_{n'}\,.\,\Pi\,(y:I)_m\,.\,\tau\, x'_1 \cdots x'_n])\ .\end{aligned}$$
3. By Definition 5.3.20, if τ is non recursive $\mathcal{E}_\tau$ is appropriately indexed and
   a) each instance has the right domain and codomain;

b) if $\Gamma \sim \Delta$, then by Fact 5.2.6

$$\begin{aligned}&\mathcal{E}_\tau([\Gamma\ \mathsf{ctx}], [\Gamma \vdash T_{\mathsf{ind}}])\\ &\cong \mathcal{E}_\tau([\Delta\ \mathsf{ctx}], [\Gamma \vdash T_{\mathsf{ind}}])\ ;\end{aligned}$$

c) by closure under $\tau-\mathsf{comp}_i$ rule,

$$\begin{aligned}&([\Gamma \vdash \mathsf{ind}_\tau\, T_1 \cdots T_n\, C\, c_1 \cdots c_k\\ &\qquad (\mathbb{K}_i\, T_1 \cdots T_n\, p_1 \cdots p_m)], [\Gamma\ \mathsf{ctx}])\\ &\cong ([\Gamma \vdash c_i\, T_1 \cdots T_n\, p'_1 \cdots p'_m], [\Gamma\ \mathsf{ctx}])\ ;\end{aligned}$$

d) by closure under $\tau-\mathsf{uniq}$ rule,

$$\begin{aligned}&([\Gamma \vdash \mathsf{ind}_\tau\, T_1, \cdots T_n\\ &\qquad (\lambda\,(x:F)_n\,.\,(\lambda z : \tau\, x_1 \cdots x_n . \tau\, x_1 \cdots x_n))\\ &\qquad \Big(\lambda\,(x:F)_{n_1}\,.\,\Big(\lambda\,(y:I)_{m'_1}\,.\,\mathbb{K}_1\, x_1 \cdots x_{n_1}\, y_1 \cdots y_{m_1}\Big)\Big)\\ &\qquad \vdots\\ &\qquad \Big(\lambda\,(x:F)_{n_k}\,.\,\Big(\lambda\,(y:I)_{m'_k}\,.\,\mathbb{K}_k\, x_1 \cdots x_{n_k}\, y_1 \cdots y_{m_k}\Big)\Big)\\ &\qquad e], [\Gamma\ \mathsf{ctx}])\\ &\cong ([\Gamma \vdash e], [\Gamma\ \mathsf{ctx}])\ .\end{aligned}$$

In the recursive case, the above points hold analogously.

4. Focusing on the auxiliary functor S,

a) by definition,

$$\begin{aligned}&S([\Gamma\ \mathsf{ctx}], [\Gamma \vdash A], [\Gamma, x : A \vdash \mathcal{U}_i])\\ &\qquad (\mathcal{F}_\tau([\Gamma\ \mathsf{ctx}], [\Gamma \vdash \Pi\,(x:F)_n\,.\mathcal{U}_i])(\cdot), ([\Gamma \vdash a], [\Gamma\ \mathsf{ctx}]))\\ &= S([\Gamma\ \mathsf{ctx}], [\Gamma \vdash A], [\Gamma, x : A \vdash \mathcal{U}_i])\\ &\qquad (([\Gamma \vdash \tau], [\Gamma\ \mathsf{ctx}]), ([\Gamma \vdash a], [\Gamma\ \mathsf{ctx}]))\\ &\cong ([\Gamma \vdash \tau[a/x]], [\Gamma\ \mathsf{ctx}])\\ &\cong \mathcal{F}_\tau([\Gamma\ \mathsf{ctx}], [\Gamma \vdash (\Pi\,(x:F)_n\,.\mathcal{U}_i)[a/x]])(\cdot)\\ &\cong \mathcal{F}_\tau([\Gamma\ \mathsf{ctx}], S([\Gamma\ \mathsf{ctx}], [\Gamma \vdash A], [\Gamma, x : A \vdash \mathcal{U}_i])\\ &\qquad\quad ([\Gamma \vdash \Pi\,(x:F)_n\,.\mathcal{U}_i], [\Gamma \vdash a]))\ ;\end{aligned}$$

b) by definition,

$$\begin{aligned}&S([\Gamma\ \mathsf{ctx}], [\Gamma \vdash A], [\Gamma, x : A \vdash \mathcal{U}_i])\\ &\qquad (\mathcal{I}_{\tau,\mathbb{K}}([\Gamma\ \mathsf{ctx}], [\Gamma \vdash \Pi\,(x:F)_{n'}\,.\,\Pi\,(y:I)_m\,.\,\tau\, x'_1 \cdots x'_n])(\cdot),\\ &\qquad\ ([\Gamma \vdash a], [\Gamma\ \mathsf{ctx}]))\\ &= S([\Gamma\ \mathsf{ctx}], [\Gamma \vdash A], [\Gamma, x : A \vdash \mathcal{U}_i])\\ &\qquad (([\Gamma \vdash \mathbb{K}], [\Gamma\ \mathsf{ctx}]), ([\Gamma \vdash a], [\Gamma\ \mathsf{ctx}]))\\ &\cong ([\Gamma \vdash \mathbb{K}[a/x]], [\Gamma\ \mathsf{ctx}])\\ &\cong \mathcal{I}_{\tau,\mathbb{K}}([\Gamma\ \mathsf{ctx}], [\Gamma \vdash (\Pi\,(x:F)_{n'}\,.\,\Pi\,(y:I)_m\,.\,\tau\, x'_1 \cdots x'_n)[a/x]])(\cdot)\\ &\cong \mathcal{I}_{\tau,\mathbb{K}}([\Gamma\ \mathsf{ctx}], S([\Gamma\ \mathsf{ctx}], [\Gamma \vdash A], [\Gamma, x : A \vdash \mathcal{U}_i])\\ &\qquad\quad ([\Gamma \vdash \Pi\,(x:F)_{n'}\,.\,\Pi\,(y:I)_m\,.\,\tau\, x'_1 \cdots x'_n], [\Gamma \vdash a]))\ ;\end{aligned}$$

c) by definition,

$$\begin{aligned} & S([\Gamma\ \mathsf{ctx}], [\Gamma \vdash A], [\Gamma, x : A \vdash \mathcal{U}_i]) \\ & \quad (\mathcal{E}_\tau([\Gamma\ \mathsf{ctx}], [\Gamma \vdash T_{\mathsf{ind}}])(\cdot), ([\Gamma \vdash a], [\Gamma\ \mathsf{ctx}])) \\ & = S([\Gamma\ \mathsf{ctx}], [\Gamma \vdash A], [\Gamma, x : A \vdash \mathcal{U}_i]) \\ & \quad (([\Gamma \vdash \mathsf{ind}_\tau], [\Gamma\ \mathsf{ctx}]), ([\Gamma \vdash a], [\Gamma\ \mathsf{ctx}])) \\ & \cong ([\Gamma \vdash \mathsf{ind}_\tau[a/x]], [\Gamma\ \mathsf{ctx}]) \\ & \cong \mathcal{F}_\tau([\Gamma\ \mathsf{ctx}], [\Gamma \vdash (T_{\mathsf{ind}})[a/x]])(\cdot) \\ & \cong \mathcal{F}_\tau([\Gamma\ \mathsf{ctx}], S([\Gamma\ \mathsf{ctx}], [\Gamma \vdash A], [\Gamma, x : A \vdash \mathcal{U}_i]) \\ & \qquad ([\Gamma \vdash T_{\mathsf{ind}}], [\Gamma \vdash a])) \ . \end{aligned}$$

It is immediate to check that $[\![\cdot]\!]_T$ satisfies all the conditions to be an interpretation, see Definition 5.2.9. □

Theorem 5.3.22 (Classifying model)**.** *If $\mathfrak{M}^i$ is a model for T, then there is a functor $\mathfrak{J} \colon \mathfrak{M}^i_T \to \mathfrak{M}^i$ which preserves the interpretation.*

Proof. Analogous to the proof of Theorem 5.3.18. □

Theorem 5.3.23 (Completeness)**.** *Let γ be a judgement which is valid in every model for an inductive theory T. Then γ is derivable from T.*

Proof. Since γ is valid in every model for T, it is valid in $\mathfrak{M}^i_T$. Hence it is an arrow of $\mathbb{M}_T$, thus, by Definition 5.3.3, $\gamma \in T$. Since T is closed under derivation in an inductive system, γ is derivable from T in the inductive system. □

The classifying model can be used to obtain some proof theoretical results about 1-HoTT theories.

Corollary 5.3.24. *If $\Gamma \vdash \Pi x : A.\, B \equiv \Pi x : C.\, D : \mathcal{U}_i$ then $\Gamma \vdash A \equiv C : \mathcal{U}_i$ and $\Gamma, x : A \vdash B \equiv D : \mathcal{U}_i$.*

Proof. In the classifying model, the hypothesis become

$$[\![\Gamma \vdash \Pi x : A.\, B]\!] \cong [\![\Gamma \vdash \Pi x : C.\, D]\!]$$

thus, by definition of interpretation

$$\begin{aligned} & F_\Pi([\![\Gamma\ \mathsf{ctx}]\!], [\![\Gamma \vdash A]\!])([\![\Gamma, x : A \vdash B]\!]) \\ \cong\ & F_\Pi([\![\Gamma\ \mathsf{ctx}]\!], [\![\Gamma \vdash C]\!])([\![\Gamma, x : C \vdash D]\!]) \end{aligned}$$

Since F_Π is conservative, see Definition 5.1.19,

$$\begin{aligned} & [\![\Gamma, x : A \vdash B]\!] \\ & = F_\Pi^{-1}([\![\Gamma\ \mathsf{ctx}]\!], [\![\Gamma \vdash A]\!])([\![\Gamma \vdash \Pi x : A.\, B]\!]) \\ & \cong F_\Pi^{-1}([\![\Gamma\ \mathsf{ctx}]\!], [\![\Gamma \vdash C]\!])([\![\Gamma \vdash \Pi x : C.\, D]\!]) \\ & = [\![\Gamma, x : C \vdash D]\!] \ . \end{aligned}$$

By definition of classifying model, it follows $[\![\Gamma, x : A\ \mathsf{ctx}]\!] \cong [\![\Gamma, x : C\ \mathsf{ctx}]\!]$, so, in particular $[\![\Gamma \vdash A]\!] \cong [\![\Gamma \vdash C]\!]$. Since F_Π respects isomorphisms, see Definition 5.1.19, it follows $[\![\Gamma, x : A \vdash B]\!] \cong [\![\Gamma, x : A \vdash D]\!]$. By completeness, $\Gamma \vdash A \equiv C : \mathcal{U}_i$ and $\Gamma, x : A \vdash B \equiv D : \mathcal{U}_i$. □

Corollary 5.3.25. $\Gamma \nvdash \Pi x : A.\, B \equiv \mathcal{U}_i : \mathcal{U}_h$.

Proof. Suppose $\Gamma \vdash \Pi x : A.\, B \equiv \mathcal{U}_i : \mathcal{U}_h$. Then $\Gamma \vdash \Pi x : A.\, B : \mathcal{U}_h$, so there is $\mathcal{U}_j$ minimal in which $\Gamma \vdash \Pi x : A.\, B : \mathcal{U}_j$. Since $\Gamma \vdash \mathcal{U}_i : \mathcal{U}_{i+1}$, $\Gamma \vdash \Pi x : A.\, B : \mathcal{U}_{i+1}$ by Corollary 3.3.10, so $j \leqslant i+1$. Suppose $j \leqslant i$: then $\Gamma \vdash \Pi x : A.\, B : \mathcal{U}_i$, thus $\Gamma \vdash \mathcal{U}_i : \mathcal{U}_i$ by Corollary 3.3.10, contradicting the fact that the hierarchy of universes does not collapse, see [58]. Hence $j = i+1$, so $\Gamma \vdash \Pi x : A.\, B \equiv \mathcal{U}_i : \mathcal{U}_{i+1}$.

In the classifying model, this means $[\![\Gamma \vdash \Pi x : A.\, B]\!] \cong [\![\Gamma \vdash \mathcal{U}_i]\!]$, so $F_\Pi^{-1}([\![\Gamma\ \mathsf{ctx}]\!], [\![\Gamma \vdash A]\!])([\![\Gamma \vdash \Pi x : A.\, B]\!]) = [\![\Gamma, x : A \vdash B]\!]$ and the functor is defined on $[\![\Gamma \vdash \mathcal{U}_i]\!]$: $F_\Pi^{-1}([\![\Gamma\ \mathsf{ctx}]\!], [\![\Gamma \vdash A]\!])([\![\Gamma \vdash \mathcal{U}_i]\!]) = [\![\Gamma, x : A \vdash C]\!]$ for some C.

However $F_\Pi([\![\Gamma\ \mathsf{ctx}]\!], [\![\Gamma \vdash A]\!])([\![\Gamma, x : A \vdash C]\!]) = [\![\Gamma \vdash \Pi x : A.\, C]\!] = [\![\Gamma \vdash \mathcal{U}_i]\!]$, so by Definitions 5.3.1 and 5.3.2 $\Pi x{:}A.\, C = \mathcal{U}_i$, literally, which is impossible. □

The semantics presented in this chapter, in the light of the obtained results, is mathematically perfect: it is sound and complete, with completeness arising from a classifying model. However, the semantics seems complex and ad hoc.

As already said, its complexity is technical, but the overall idea is simple: except for universes, which are structural elements, types are transformations of the framework into itself. The exact meaning of this description has been clarified in Definitions 5.1.19 and 5.1.24, where the frameworks, ML- and T-ML-categories, are presented along with the transformations, coded as families of functors. The nature of these transformations is easy to grasp: the canonical models, see Propositions 5.3.17 and 5.3.21, make evident how the transformations are conceived, and since these models are classifying, the transformations can be conceived in this way only. Wearing the glasses of this point of view, the complexity of the semantics reduces to a mere technical fact, which poses no more difficulties than other semantics for other systems, see, e.g., Heyting categories versus first-order intuitionistic logic [54].

The ad hoc aspect is apparent. It is clear that the presentation from Section 5.1 to Section 5.3 follows the scholar style, which emphasises the progressive construction of the semantics from the roots. And it is clear to every working mathematician that the results have not been obtained by conceiving them in this way. The ad hoc aspect is a by-product of the style of the presentation. In the following, the main structure of the definitions is decomposed to show where they come from and why. In other words, the part the scholar presentation hides.

The initial idea is simple: different objects may be equal; equal objects are interchangeable. The process of exchanging one object with another is a morphism, so equality is naturally modelled by isomorphisms. As rough as it may sound, the idea is deeply grounded in the proof theory: the $\equiv-$subst rule, Corollary 3.3.10, and Proposition 3.3.15 prove that judgemental equivalence has the interchangeability property; Corollary 4.1.12 shows that normal forms are unique up to conversion, i.e., judgemental equality. Hence the notion of equality gets established, and also, as a side effect, the notion of object we informally used above. In fact, equality acts on objects, so necessarily objects have to be contexts, terms, and types. However, terms and types do not exist in isolations: they live in a context, see Proposition 3.3.14, hence the building blocks of the semantics must be contexts and terms-in-context. Contexts are equivalent when they are permutations of the same set of declarations, as for

Proposition 3.3.11, and every context is generated from the empty one. These features are directly coded by Points (1) to (5) of Definition 5.1.19.

Typing has to relate $\Gamma \vdash a$ to $\Gamma \vdash A$ to get $\Gamma \vdash a : A$: the natural way to look at terms-in-context in a category is to think them as objects, as justified before, and also as objects over the context Γ, which is achieved by the technical instrument of slice categories. So $\Gamma \vdash a : A$ becomes a pair of objects in the slice category over Γ, each one representing $\Gamma \vdash a$ and $\Gamma \vdash A$ respectively: this is easy to achieve if we model $\vdash$ as an arrow to Γ; similarly, we model $:$ as an arrow from $\Gamma \vdash A$ to $\Gamma \vdash a$. Each extension in the context introduces a new variable via the $\mathsf{ctx{-}EXT}$ rule: Point (6) of Definition 5.1.19 is a direct translation of this.

The typing arrows, modelling the colon in $\Gamma \vdash a : A$, must be indecomposable to fix that each term has a minimal type. This leads to have a notion of factorisation which is aware of equalities, i.e., isomorphisms (Definitions 5.1.1 and 5.1.2), and objects which are equal to themselves in a unique way (Definition 5.1.4). Since type theory makes sense if and only if every term lies in some type, Definition 5.1.7 directly captures this sentence.

Universes are special: they are, at the same time, types and the structure allowing to define the concept of being a type. Definition 5.1.8 fixes what means to be a universe in the most basic sense: containing the fundamental universe, which is given. Putting all together, the chain $\{\mathcal{U}_i\}_{i \in \mathbb{N}}$ of universes arises, see Corollary 5.1.10, as a consequence of types and equalities, as summarised in Fact 5.1.15.

Therefore, the structural part of the semantics is by no means ad hoc: it is the necessary consequence of a natural line to describe the formal system, in which judgemental equality is the centre of interest.

The pillar of the presented semantics is to consider types as transformations. This idea comes from the definition of inductive types, and the analysis of Chapter 4 about normalisation. The syntax suggests that inductive types are generated, or constructed if you prefer, from already existing ones combining equality, function spaces, and new constants. These constants are the *creative* part of a type: they allow to synthesise the new terms and types. The ultimate purpose of transformations is to replicate inside the semantics this process of construction. Of course, also Π-types are modelled by transformations, since they are inductive types, too. In the syntax, encoding Π-types as inductive following Section 3.2, will lead to a circular definition, but in the semantics it is possible, although they still provide the foundational features to express the other inductive types. As a matter of fact, Π-types are not function spaces, at least, not in the sense of set theory, nor exactly in the sense of category theory, as described in Chapter 2 about Seely's work [90]. Our initial belief was that a transformational semantics is better suited to capture Π-types' nature, and the soundness and completeness results confirm that the belief is solidly grounded.

Working in a categorical framework, transformations are rendered as functors; the formation, introduction, and elimination rules become families of functors, indexed by their parameters. Although it is somewhat arbitrary to choose what is a parameter and what is an index, we believe our choice is good: the conditions about the functors are naturally expressible, once their rationale is explained. This choice is reflected in the domain and codomain of the functors: both groupoid of terms in the right type. Many aspects must be considered: each family of functors must preserve universes, that is, it has to respect the cumulative nature of universes; concretely, it has to implement the $\mathcal{U}-\mathsf{cumul}$ rule in the parameters, reflecting half of Lemma 3.3.7. Each family of functors must respect isomorphisms: while this is automatic on arguments, it has to be required on parameters. This requirement corresponds to preserve equality, as already discussed. Each family of functors has to be stable with respect to context extension, essentially implementing weakening. In addition, each family of functors has a constraint to characterise its role in the theory: formation, introduction, or elimination. The family F_Π must model the fact that a Π-space is fully identified by its domain and codomain; hence, the right condition to require is that it has to be injective on objects. However, this property is categorically bad, not being stable under categorical equivalence, so the more appealing notion of conservativity has been employed, which implies injectivity on objects. The family I_Π has to preserve η-equivalence. In our framework, as it becomes crystal clear in Proposition 5.3.17, for this reason the maps into each member of the family have to be categorical equivalences. The family E_Π is more complex: in fact, it is easier to constrain its behaviour having a semantic notion of substitution S, which links E_Π with I_Π. In fact, Point (10) of Definition 5.1.19 says that S acts as a substitution operator, Points (10a) to (10g), following the usual definition of substitution on terms, and it says that $E_\Pi \circ I_\Pi = S$, leaving out parameters, which is nothing but the $\Pi-\mathsf{comp}$ rule. The families F_τ and I_τ of Definition 5.1.24 are required to respect equivalent contexts as it was for F_Π and I_Π: they inherit the other good properties from F_Π and I_Π. The family E_τ, in addition, has to implement $\tau-\mathsf{comp}$, Point (3c) of Definition 5.1.24, and $\tau-\mathsf{uniq}$, Point (3d). And, finally, S must behave as a substitution on the τ type, Point (4) of Definition 5.1.24.

Therefore, the semantic conditions on the transformations are nothing but the natural implementation of the types, as defined in the intended meaning the syntax conveys, plus a few requirements due to the choice of parameters. Nothing is forced in the semantics to make the soundness and the completeness results to go through; thus, when properly explained and not covered by the categorical language, nothing is ad hoc.

The previous explanation leaves one point open: apart the canonical one, are there other models for a theory? The trivial answer is "yes": one can easily throw in elements in the canonical model to coherently extend it to a larger model, for example, considering an extension to the theory, in which the new elements are invisible, i.e., they do not lie in the image of the interpretation of the theory. Of course, this answer in unsatisfactory.

A much more interesting answer comes from Theorem 4.3.12. That result could be understood as saying that considering just irreducible contexts, terms, and types suffices to build a model, which is different from the canonical one. Because each judgement γ can be interpreted in an equivalent irreducible judgement, interpretation has the features to conclude along the lines of Section 5.3 that the irreducible model $\mathfrak{M}_T^{\mathsf{irr}}$ is classifying for the theory T.

This facts opens the door to a new, yet unexplored room. In fact, it is clear that $\mathfrak{M}_T^{\mathsf{irr}} \simeq \mathfrak{M}_T$, the two classifying models are categorically equivalent. Considering the corresponding adjoint situation, the functor $\mathfrak{I}\colon \mathfrak{M}_T \to \mathfrak{M}_T^{\mathsf{irr}}$ models computation, and its adjoint $\mathfrak{J}$ in the equivalence is the inclusion, thus showing that $\mathfrak{M}_T^{\mathsf{irr}}$ is a reflective subcategory of $\mathfrak{M}_T$.

Finally, it has been said that the semantics discussed so far is *point-free*. In fact, since every model can be derived from the classifying model $\mathfrak{M}_T$, it suffices to justify the claim on it. As a category, its objects, see Definition 5.3.1, are the (equivalence class of) context and pairs of terms and their minimal context of definition; in turn, arrows are pair of objects, see Definition 5.3.3. This structure provides the basis on which the semantics is constructed: Propositions and Facts 5.3.5, 5.3.7, 5.3.8, 5.3.9, 5.3.10, 5.3.11, 5.3.12, 5.3.13, and 5.3.14 recognise that the properties to describe the structural aspects of Martin-Löf type theory are present. Then, Definitions 5.3.15 and 5.3.20 add the appropriate endofunctors to interpret the dependent product and the other inductive types.

Moving one step back, and comparing, e.g., to Tarski's semantics for first-order logic, there is a fundamental difference in the methodology: the primitive notion behind the semantics is not a universe of objects linked by functions and relations, expressible in the syntax, but rather the structure to model substitution[1], as already remarked after Theorem 5.2.8. In this respect, the key concept of type splits into two parts: the object which stands for the type in context, and the transformation which models the behaviour of the type. The distinction between the object and its behaviour, the latter being predominant to describe the semantics, is why the model is said to be point-free.

The topos-theoretical semantics of first-order logic and the simple theory of types [55, 54] uses a similar idea: although the syntactical elements are present in the models as suitable arrows, the core of the semantics, what gives meaning to objects, are the constructions one can perform in the categories, and, mostly important, how they are linked by adjunctions. However, there is an asymmetry in that semantics, which becomes completely clear when looking at the simple theory of types. In fact, the simple theory of types is, via the Curry-Howard isomorphism, just intuitionistic propositional logic. However, constructing the canonical models for these two formal systems in the topos-theoretic framework, they are completely different: hence, there is no direct correspondent of the proposition-as-type interpretation in the semantics. Oppositely, the present semantics, which incorporates the simple theory of types as a subsystem, provides a complete account of the isomorphism, since the logical interpretation is not distinguished from the type-theoretical one.

[1]See [18] to see how first-order logic admits a point-free, substitution based semantics.

Chapter 6

Homotopic features

In this chapter, the results of Chapters 4 and 5 will be extended to homotopy type theory. First, the prominent homotopic features will be introduced: function extensionality and univalence, the higher inductive types. The family of higher inductive types synthesised in Section 6.2 plus elementary truncation, see Section 6.3, satisfies the normalisation theorem 4.3.12 and the semantics, eventually getting soundness and completeness with a classifying model.

6.1 Function extensionality and univalence

The syntax of the most peculiar axioms characterising homotopy type theory, that is, function extensionality and univalence, is complex: a number of auxiliary definitions are used, which are introduced and discussed in Chapters 2 and 4 of [95]. They are summarised below with few further comments:

$$f \sim g :\equiv \Pi x : A.\, f\, x =_B g\, x$$

with $f, g : \Pi x : A.\, B$, which expresses that f and g are extensionally equal;

$$f \circ g :\equiv \lambda x : A.\, f\,(g\, x)$$

with $f : B \to C$ and $g : A \to B$, composition of non-dependent functions;

$$\mathsf{id}_A :\equiv \lambda x : A.\, x$$

with $A : \mathcal{U}_i$, the identity function of the type A;

$$\begin{aligned}\mathsf{happly}(f, g) :\equiv \mathsf{ind}_=&(\Pi x : A.\, B)\, f\, g\\ &(\lambda f, g : (\Pi x : A.\, B).\\ &\quad(\lambda p : (= (\Pi x : A.\, B)\, f\, g).\, f \sim g))\\ &(\lambda f : (\Pi x : A.\, B).\, \lambda x : A.\, \mathsf{refl}\,(f\, x))\end{aligned}$$

with $f, g : \Pi x : A.\, B$, so that $\mathsf{happly}(f, g) : f =_{\Pi x : A.\, B} g \to f \sim g$;

$$\begin{aligned}\mathsf{isequiv}(f) :\equiv\ &\Sigma g : (B \to A), \eta : (g \circ f \sim \mathsf{id}_A), \epsilon : (f \circ g \sim \mathsf{id}_B).\\ &\Pi x : A.\, f\,(\eta\, x) =_B \epsilon\,(f\, x)\end{aligned}$$

with $f : A \to B$, which says that f is an equivalence;

$$\begin{aligned}p_* :\equiv\ &\mathsf{ind}_=\, A\, x\, y\, (\lambda z, w : A.\, (P\, z \to P\, w))\\ &(\lambda z : A.\, \mathsf{id}_{P\, z})\, p\end{aligned}$$

with $p : x =_A y$ and $P : A \to \mathcal{U}_i$, so that $p_* : P\,x \to P\,y$, the *transport* of p in P;

$$\mathsf{idtoeqv}(A, B) :\equiv \lambda p : (A =_{\mathcal{U}_i} B).\, p_*$$

with $A, B : \mathcal{U}_i$.

To ease notation and help understanding, define the following abbreviations:

$$\begin{aligned}\mathsf{FunExt} :\equiv\ & \Pi A : \mathcal{U}_i, B : (A \to \mathcal{U}_i).\, \Pi f, g : (\Pi x : A.\, B\,x).\\ & \mathsf{isequiv}(\mathsf{happly}(f, g)) \enspace ,\end{aligned}$$

$$\mathsf{Univalence} :\equiv \Pi A, B : \mathcal{U}_i.\mathsf{isequiv}\left(\mathsf{idtoeqv}(A, B)\right) \enspace .$$

Then the axioms of function extensionality and univalence can be expressed as

$$\frac{\Gamma \vdash \mathsf{FunExt} : \mathcal{U}_{i+1}}{\Gamma \vdash \mathsf{funext} : \mathsf{FunExt}}\,\Pi{-}\mathsf{ext} \qquad \frac{\Gamma \vdash \mathsf{Univalence} : \mathcal{U}_{i+1}}{\Gamma \vdash \mathsf{univalence} : \mathsf{Univalence}}\,\mathcal{U}_i{-}\mathsf{univ}$$

with funext and $\mathsf{univalence}$ distinguished constants.

It is clear that these axioms are instances of the $k{-}\mathsf{intro}$ schema. Also, it is immediate to see how the present form is equivalent to the version in Section A.3 of [95], by $\Pi{-}\mathsf{elim}{-}\mathsf{eq}$ in one direction, and by Lemma 3.3.7 in the other.

6.2 Syntax of higher inductive types

This section is devoted to introduce a generic syntax for the higher inductive types with introduction rules of the form $t : \tau$ and $p : (=\ \tau\,x\,y)$. In the following, these types are referred to as 1-higher inductive types, to emphasise that they are limited to describe 0-paths (points) and 1-paths.

To simplify the notation, if p: $=\ A\,x\,y$, $u : P\,x$, $v : P\,y$, and $f : \Pi x : A.P$ define

$$(u =^P_p v) :\equiv (=\ A\,p_*\,u\,v) \enspace ,$$

$$\begin{aligned}\mathsf{adp}_f(p) :\equiv\ & \mathsf{ind}_=\ A\,x\,y\,(\lambda x : A.\, \lambda y : A.\, \lambda p : (=\ A\,x\,y).\\ & (=\ Px\,(p_*\,fx)\,fy))(\lambda x : A.\, \mathsf{refl}_{fx})\,p \enspace .\end{aligned}$$

These definitions come from Sections 2.3 and 6.2 of [95].

Formation and introduction rules

The formation rule $\tau{-}\mathsf{form}$ has the same pattern of the one for the inductive types, see Section 3.2. The same happens for the introduction rule $\tau{-}\mathsf{intro}$, which introduces canonical terms of type τ.

The novel structure of this subset of higher inductive types is introduced by the $\tau{-}\mathsf{intro}{-}{=}$ rules, which are related to paths of type $=\ \tau\,x\,y$. The scheme is

$$\frac{\begin{array}{l}\Gamma \vdash \big(\Pi\,(x : F)_{n'} .\, \Pi\,(z : P)_r .\ = (\tau\,x'_1 \cdots x'_n)\\ \qquad (\mathbb{K}_a x'_1 \cdots x'_n\,w^a_1 \cdots w^a_{s_a})(\mathbb{K}_b x'_1 \cdots x'_n\,w^b_1 \cdots w^b_{s_b})\big) : \mathcal{U}_i\end{array}}{\begin{array}{l}\Gamma \vdash \mathbb{P}_k : \big(\Pi\,(x : F)_{n'} .\, \Pi\,(z : P)_r .\ = (\tau\,x'_1 \cdots x'_n)\\ \qquad (\mathbb{K}_a x'_1 \cdots x'_n\,w^a_1 \cdots w^a_{s_a})(\mathbb{K}_b x'_1 \cdots x'_n\,w^b_1 \cdots w^b_{s_b})\big)\end{array}}\,\tau{-}\mathsf{intro}_k{-}{=}$$

where the sequences of the w^i_j contain the x_k and z_k.

Elimination rule

The simple elimination rule for the higher inductive types is the one for inductive types extended with the g_i

$$\frac{\begin{array}{l}\Gamma \vdash (\Pi\,(x:F)_n\,.\,(\Pi C : (\Pi\,(x:F)_n\,.\,\tau\,x_1\,\cdots\,x_n \to \mathcal{U}_h).\\ \qquad (\Pi_{i=1}^{h} c_i : (\Pi\,(x:F)_{n'_i}\,.\,\Pi\,(y:I)_{m_i}\,.\\ \qquad\qquad C\,x'_1\,\cdots\,x'_{n_i}\,(\mathbb{K}\,x'_1\,\cdots\,x'_{n_i}\,y_1\,\cdots\,y_{m_i})).\\ \qquad (\Pi_{i=1}^{h} g_i : (\Pi\,(x:F)_{n''_i}\,.\,\Pi\,(z:P)_{r_i}\,.\\ \qquad\qquad (c_a\underline{x}(\underline{w_a}) =^{C\underline{x}}_{\mathbb{P}_i\underline{xz}} c_b\underline{x}(\underline{w_b}))).\\ \qquad (\Pi e : \tau\,x_1\,\cdots\,x_n.\,C\,x_1\,\cdots\,x_n\,e)))) : \mathcal{U}_{h+1}\end{array}}{\begin{array}{l}\Gamma \vdash \mathsf{ind}_\tau : (\Pi\,(x:F)_n\,.\,(\Pi C : (\Pi\,(x:F)_n\,.\,\tau\,x_1\,\cdots\,x_n \to \mathcal{U}_h).\\ \qquad (\Pi_{i=1}^{k} c_i : (\Pi\,(x:F)_{n'_i}\,.\,\Pi\,(y:I)_{m_i}\,.\\ \qquad\qquad C\,x'_1\,\cdots\,x'_{n_i}\,(\mathbb{K}\,x'_1\,\cdots\,x'_{n_i}\,y_1\,\cdots\,y_{m_i})).\\ \qquad (\Pi_{i=1}^{k} g_i : (\Pi\,(x:F)_{n''_i}\,.\,\Pi\,(z:P)_{r_i}\,.\\ \qquad\qquad (c_a\underline{x}(\underline{w_a}) =^{C\underline{x}}_{\mathbb{P}_i\underline{xz}} c_b\underline{x}(\underline{w_b}))).\\ \qquad (\Pi e : \tau\,x_1\,\cdots\,x_n.\,C\,x_1\,\cdots\,x_n\,e))))\end{array}}\ \tau\mathsf{-elim}$$

where $\underline{x} = x_1 \cdots x_n$, $\underline{z} = z_1 \cdots z_r$ and $\underline{w_i} = w^i_1 \cdots w^i_{s_i}$.

The same happens in the recursive case.

Computation rules

The computation rules $\tau\mathsf{-comp}_i$ related to the introduction rules $\tau\mathsf{-intro}_i$ extend the ones for the inductive types requiring the existence of the g_i

$$\frac{\begin{array}{l}\Gamma \vdash C : (\Pi\,(x:F)_n\,.\,\tau\,x_1\,\cdots\,x_n \to \mathcal{U}_h)\\ \Gamma \vdash c_1 : \Big(\Pi\,(x:F)_{n'_1}\,.\,\Pi\,(y:I)_{m'_1}\,.\\ \qquad\quad C\,x'_1\,\cdots x'_{n_1}\,\left(\mathbb{K}_1\,x'_1\,\cdots x'_{n_1}\,y_1\,\cdots\,y_{m_1}\right)\Big)\\ \ldots\\ \Gamma \vdash c_k : \Big(\Pi\,(x:F)_{n'_k}\,.\,\Pi\,(y:I)_{m'_k}\,.\\ \qquad\quad C\,x'_1\,\cdots x'_{n_k}\,\left(\mathbb{K}_k\,x'_1\,\cdots x'_{n_k}\,y_1\,\cdots\,y_{m_k}\right)\Big)\\ \Gamma \vdash g_1 : \Big(\Pi\,(x:F)_{n''_1}\,.\,\Pi\,(z:P)_{r_1}\,.\,(c_{a_1}\underline{x}(\underline{w_{a_1}}) =^{C\underline{x}}_{\mathbb{P}_1\underline{xz}} c_{b_1}\underline{x}(\underline{w_{b_1}}))\Big)\\ \ldots\\ \Gamma \vdash g_l : \Big(\Pi\,(x:F)_{n''_l}\,.\,\Pi\,(z:P)_{r_l}\,.\,(c_{a_l}\underline{x}(\underline{w_{a_l}}) =^{C\underline{x}}_{\mathbb{P}_l\underline{xz}} c_{b_l}\underline{x}(\underline{w_{b_l}}))\Big)\\ \Gamma \vdash \mathbb{K}_i\,T_1\,\cdots\,T_n\,p_1\,\cdots\,p_m : \tau\,T_1\,\cdots\,T_n\end{array}}{\begin{array}{l}\Gamma \vdash \mathsf{ind}_\tau\,T_1\,\cdots\,T_n\,C\,c_1\,\cdots\,c_k\,g_1\,\cdots\,g_l\,(\mathbb{K}_i\,T_1\,\cdots\,T_n\,p_1\,\cdots\,p_m) \equiv\\ \qquad c_i\,T_1\,\cdots\,T_n\,p'_1\,\cdots\,p'_m : C\,T_1\,\cdots\,T_n\,(\mathbb{K}_i\,T_1\,\cdots\,T_n\,p_1\,\cdots\,p_m)\end{array}}\ \tau\mathsf{-comp}_i$$

The computation rules $\tau\mathsf{-comp}_i{-}{=}$ related to the $\tau\mathsf{-intro}_i{-}{=}$ do not have as a conclusion an equality judgement; indeed, they only state that a propositional equality type is inhabited.

$$\dfrac{\begin{array}{l}\Gamma \vdash C : (\Pi\,(x:F)_n \,.\, \tau\, x_1 \,\cdots\, x_n \to \mathcal{U}_h)\\ \Gamma \vdash c_1 : \Big(\Pi\,(x:F)_{n'_1} \,.\, \Pi\,(y:I)_{m'_1} \,.\\ \qquad C\, x'_1 \cdots x'_{n_1}\, \left(\mathbb{K}_1\, x'_1 \cdots x'_{n_1}\, y_1 \,\cdots\, y_{m_1}\right)\Big)\\ \ldots\\ \Gamma \vdash c_k : \Big(\Pi\,(x:F)_{n'_k} \,.\, \Pi\,(y:I)_{m'_k} \,.\\ \qquad C\, x'_1 \cdots x'_{n_k}\, \left(\mathbb{K}_k\, x'_1 \cdots x'_{n_k}\, y_1 \,\cdots\, y_{m_k}\right)\Big)\\ \Gamma \vdash g_1 : \Big(\Pi\,(x:F)_{n'_1} \,.\, \Pi\,(z:P)_{r_1} \,.\, (c_{a_1}\underline{x}(\underline{w_{a_1}}) =^{C\underline{x}}_{\mathbb{P}_1\underline{xz}} c_{b_1}\underline{x}(\underline{w_{b_1}}))\Big)\\ \ldots\\ \Gamma \vdash g_l : \Big(\Pi\,(x:F)_{n'_l} \,.\, \Pi\,(z:P)_{r_l} \,.\, (c_{a_l}\underline{x}(\underline{w_{a_l}}) =^{C\underline{x}}_{\mathbb{P}_l\underline{xz}} c_{b_l}\underline{x}(\underline{w_{b_l}}))\Big)\\ \Gamma \vdash \mathbb{P}_i\, T_1 \,\cdots\, T_n\, q_1 \,\cdots\, q_{r_i} :\\ \qquad = (\tau\, T_1 \,\cdots\, T_n)(\mathbb{K}_{a_i}\underline{T}(q_1^{a_i} \,\cdots\, q^{a_i}_{s_{a_i}}))(\mathbb{K}_{b_i}\underline{T}(q_1^{b_i} \,\cdots\, q^{b_i}_{s_{b_i}}))\end{array}}{\begin{array}{c}\Gamma \vdash \mathbb{C}_i : = (c_{a_i}\underline{T}(\underline{T}\underline{q_{b_i}}) =^{C\underline{T}}_{\mathbb{P}_l\underline{T}\underline{q}} c_{b_i}\underline{T}(\underline{T}\underline{q_{a_i}}))\\ (\mathsf{adp}_{\mathsf{ind}_\tau\, \underline{T}\, C\, \underline{c}\, \underline{d}}(\mathbb{P}_i\, \underline{T}\underline{q}))(g_i\, \underline{T}\underline{q})\end{array}}\ \tau{-}\mathsf{comp}^*_i{-}{=}$$

where the q^i_j correspond to the w^i_j.

Since the rule above is complex and difficult to analyse, although it closely resembles the shape of $\tau{-}\mathsf{comp}_i$, we will use instead the following less direct rule:

$$\dfrac{\begin{array}{c}\Gamma \vdash = (c_{a_i}\underline{T}(\underline{T}\underline{q_{a_i}}) =^{C\underline{T}}_{\mathbb{P}_l\underline{T}\underline{q}} c_{b_i}\underline{T}(\underline{T}\underline{q_{b_i}}))\\ (\mathsf{adp}_{\mathsf{ind}_\tau\, \underline{T}\, C\, \underline{c}\, \underline{d}}(\mathbb{P}_i\, \underline{T}\underline{q}))(g_i\, \underline{T}\underline{q}) : \mathcal{U}_{h+1}\end{array}}{\begin{array}{c}\Gamma \vdash \mathbb{C}_i : = (c_{a_i}\underline{T}(\underline{T}\underline{q_{a_i}}) =^{C\underline{T}}_{\mathbb{P}_l\underline{T}\underline{q}} c_{b_i}\underline{T}(\underline{T}\underline{q_{b_i}}))\\ (\mathsf{adp}_{\mathsf{ind}_\tau\, \underline{T}\, C\, \underline{c}\, \underline{d}}(\mathbb{P}_i\, \underline{T}\underline{q}))(g_i\, \underline{T}\underline{q})\end{array}}\ \tau{-}\mathsf{comp}_i{-}{=}$$

which is an instance of the $k{-}\mathsf{intro}$ schema. A tedious but easy derivation shows that, given the premises of $\tau{-}\mathsf{comp}^*_i{-}{=}$, the premise of $\tau{-}\mathsf{comp}_i{-}{=}$ can be derived, so $\tau{-}\mathsf{comp}_i{-}{=}$ is able to simulate $\tau{-}\mathsf{comp}^*_i{-}{=}$. Conversely, by Lemma 3.3.7, from the derivation of the premise of $\tau{-}\mathsf{comp}_i{-}{=}$ one could extract the premises of $\tau{-}\mathsf{comp}^*_i{-}{=}$, so the $\tau{-}\mathsf{comp}_i{-}{=}$ rule can be simulated by $\tau{-}\mathsf{comp}^*_i{-}{=}$. Hence, the two rules are equivalent: the $*$-version is more useful in applications since it resembles the first-order computation rule; oppositely, $\tau{-}\mathsf{comp}_i{-}{=}$ is easier to analyse, so this is the rule which will be adopted in the following of this chapter.

Uniqueness rule

As remarked in [95], general uniqueness rules pose a problem. In fact, only $\Pi{-}\mathsf{uniq}$ is included in [95], while the $\tau{-}\mathsf{uniq}$ rules are excluded, although a version which uses propositional equality can be usually derived, see Section 5.2 in [95]. The reason to exclude uniqueness rules lies in their meaning: what a uniqueness rule says is that the only inhabitants of an inductive type are the canonical terms up to equality. In the judgemental version of uniqueness, equality is conversion, see Corollary 4.1.12; in the propositional version, equality is the identity type, thus uniqueness states that every inhabitant of the type is an endpoint of some path whose other endpoint is a canonical term. Moving to higher order inductive types, even the simple sphere $\mathbb{S}^1$ violates this meaning of uniqueness: as explained in Section 6.1 of [95], $\mathsf{loop}, \mathsf{loop} \cdot \mathsf{loop}, \mathsf{loop}^{-1}, \mathsf{refl}$ are all different and this is essential to capture the homotopical nature of the sphere. In fact, stating that the only paths are the canonical ones, refl and loop, would make impossible to generate the fundamental Poincaré's group of the sphere, see [48, Chapters 1 and 4], [84, Chapters 3 and 11] and [74]. However, the results in Chapter 4 are immediately transported in a version of Martin-Löf type theory in which $\Pi-\mathsf{uniq}$ is the only uniqueness rule. Therefore, the analysis of the proof theory of [95] can safely start from those results, which have to be extended to univalence, function extensionality, and the higher inductive types.

6.3 Canonical higher inductive types

In the following the rules for the canonical 1-higher inductive types, whose introduction rules are of the form $t : \tau$ or $p : (= \tau\, x\, y)$, are written according to our syntax.

The canonical types with different introduction rules are the spheres $\mathbb{S}^i$ with $i > 1$, the generic cell complexes (and, in particular, the torus T^2), n-truncation with $n \geqslant 0$ and quotients. These types are not 1-higher inductive types. Although it is possible to generalise our definitions to k-higher inductive types, $k \in \mathbb{N}$, this is cumbersome and still ineffective to capture all these types.

Truncation (which is, indeed, -1-truncation) has introduction rules of the required form, but its elimination and computation rules do not behave accordingly to our generic syntax. We include the rules for truncation for completeness, and briefly discuss them. Also, the higher order computation rules are presented in the most convenient format for applications, see the comments at the end of the description of the computation rules in Section 6.2.

Circle

$$\frac{\Gamma \vdash \mathcal{U}_i : \mathcal{U}_{i+1}}{\Gamma \vdash \mathbb{S}^1 : \mathcal{U}_i}\ \mathbb{S}^1\mathsf{-form}$$

$$\frac{\Gamma \vdash \mathbb{S}^1 : \mathcal{U}_i}{\Gamma \vdash \mathsf{base} : \mathbb{S}^1}\ \mathbb{S}^1\mathsf{-intro}$$

$$\frac{\Gamma \vdash = \ \mathbb{S}^1\, \mathsf{base}\, \mathsf{base} : \mathcal{U}_i}{\Gamma \vdash \mathsf{loop} : = \ \mathbb{S}^1\, \mathsf{base}\, \mathsf{base}}\ \mathbb{S}^1\mathsf{-intro-=}$$

$$\frac{\Gamma \vdash \Pi C : (\mathbb{S}^1 \to \mathcal{U}_i).\, \Pi c_1 : C\, \mathsf{base}.\, \Pi\, g_1 : c_1 =^{C}_{\mathsf{loop}} c_1.\, \Pi x : \mathbb{S}^1.\, C x : \mathcal{U}_{i+1}}{\Gamma \vdash \mathsf{ind}_{\mathbb{S}^1} : \Pi C : (\mathbb{S}^1 \to \mathcal{U}_i).\, \Pi c_1 : C\, \mathsf{base}.\, \Pi\, g_1 : c_1 =^{C}_{\mathsf{loop}} c_1.\, \Pi x : \mathbb{S}^1.\, C x}\ \mathbb{S}^1\mathsf{-elim}$$

$$\frac{\begin{array}{l}\Gamma \vdash C : \mathbb{S}^1 \to \mathcal{U}_i \quad \Gamma \vdash c_1 : C\, \mathsf{base} \quad \Gamma \vdash g_1 : c_1 =^{C}_{\mathsf{loop}} c_1 \\ \Gamma \vdash \mathsf{base} : \mathbb{S}^1\end{array}}{\Gamma \vdash \mathsf{ind}_{\mathbb{S}^1}\, C\, c_1\, g_1\, \mathsf{base} \equiv c_1 : C\, \mathsf{base}}\ \mathbb{S}^1\mathsf{-comp}$$

$$\frac{\begin{array}{l}\Gamma \vdash C : \mathbb{S}^1 \to \mathcal{U}_i \quad \Gamma \vdash c_1 : C\,\mathsf{base} \quad \Gamma \vdash g_1 : c_1 =^C_{\mathsf{loop}} c_1 \\ \Gamma \vdash \mathsf{loop}\colon = \ \mathbb{S}^1\,\mathsf{base}\,\mathsf{base}\end{array}}{\Gamma \vdash \mathbb{C}\colon = (c_1 =^C_{\mathsf{loop}} c_1)\,(\mathsf{adp}_{\mathsf{ind}_{\mathbb{S}^1}\,C\,c_1\,g_1}\mathsf{loop})\,g_1}\ \mathbb{S}^1\mathsf{-comp-=}$$

Interval type

$$\frac{\Gamma \vdash \mathcal{U}_i : \mathcal{U}_{i+1}}{\Gamma \vdash I : \mathcal{U}_i}\ I\mathsf{-form}$$

$$\frac{\Gamma \vdash I : \mathcal{U}_i}{\Gamma \vdash 0_I : I}\ I\mathsf{-intro}_1 \qquad \frac{\Gamma \vdash I : \mathcal{U}_i}{\Gamma \vdash 1_I : I}\ I\mathsf{-intro}_2$$

$$\frac{\Gamma \vdash = \ I\,0_I\,1_I : \mathcal{U}_i}{\Gamma \vdash \mathsf{seg}\colon = \ I\,0_I\,1_I}\ I\mathsf{-intro-=}$$

$$\frac{\begin{array}{c}\Gamma \vdash \Pi C : (I \to \mathcal{U}_i).\, \Pi c_1 : C\,0_I.\, \Pi c_2 : C\,1_I. \\ \Pi\, g_1 : c_1 =^C_{\mathsf{seg}} c_2.\, \Pi x : I.\, Cx : \mathcal{U}_{i+1}\end{array}}{\begin{array}{c}\Gamma \vdash \mathsf{ind}_I : \Pi C : (I \to \mathcal{U}_i).\, \Pi c_1 : C\,0_I.\, \Pi c_2 : C\,1_I. \\ \Pi\, g_1 : c_1 =^C_{\mathsf{seg}} c_2.\, \Pi x : I.\, Cx\end{array}}\ I\mathsf{-elim}$$

$$\frac{\begin{array}{l}\Gamma \vdash C : I \to \mathcal{U}_i \quad \Gamma \vdash c_1 : C\,0_I \quad \Gamma \vdash c_2 : C\,1_I \\ \Gamma \vdash g_1 : c_1 =^C_{\mathsf{seg}} c_2 \quad \Gamma \vdash 0_I : I\end{array}}{\Gamma \vdash \mathsf{ind}_I\,C\,c_1\,c_2\,g_1\,\mathsf{base} \equiv c_1 : C\,0_I}\ I\mathsf{-comp}_1$$

$$\frac{\begin{array}{l}\Gamma \vdash C : I \to \mathcal{U}_i \quad \Gamma \vdash c_1 : C\,0_I \quad \Gamma \vdash c_2 : C\,1_I \\ \Gamma \vdash g_1 : c_1 =^C_{\mathsf{seg}} c_2 \quad \Gamma \vdash 1_I : I\end{array}}{\Gamma \vdash \mathsf{ind}_I\,C\,c_1\,c_2\,g_1\,\mathsf{base} \equiv c_2 : C\,1_I}\ I\mathsf{-comp}_2$$

$$\frac{\begin{array}{l}\Gamma \vdash C : I \to \mathcal{U}_i \quad \Gamma \vdash c_1 : C\,0_I \quad \Gamma \vdash c_2 : C\,1_I \\ \Gamma \vdash g_1 : c_1 =^C_{\mathsf{seg}} c_2 \quad \Gamma \vdash \mathsf{seg}\colon = \ I\,0_I\,1_I\end{array}}{\Gamma \vdash \mathbb{C}\colon = (c_1 =^C_{\mathsf{seg}} c_2)\,(\mathsf{adp}_{\mathsf{ind}_I\,C\,c_1\,c_2\,g_1}\mathsf{seg})\,g_1}\ I\mathsf{-comp-=}$$

Suspensions

As done in [95], we use Σ as type-symbol for suspensions, even though it is the same symbol used for the inductive type of dependent sum.

$$\frac{\Gamma \vdash \Pi A : \mathcal{U}_i.\mathcal{U}_i : \mathcal{U}_{i+1}}{\Gamma \vdash \Sigma : \Pi A : \mathcal{U}_i.\mathcal{U}_i}\ \Sigma\mathsf{-form}$$

$$\frac{\Gamma \vdash \Pi A : \mathcal{U}_i.\, \Sigma\,A : \mathcal{U}_{i+1}}{\Gamma \vdash N : \Pi A : \mathcal{U}_i.\, \Sigma\,A}\ \Sigma\mathsf{-intro}_1 \qquad \frac{\Gamma \vdash \Pi A : \mathcal{U}_i.\, \Sigma\,A : \mathcal{U}_{i+1}}{\Gamma \vdash S : \Pi A : \mathcal{U}_i.\, \Sigma\,A}\ \Sigma\mathsf{-intro}_2$$

$$\frac{\Gamma \vdash \Pi A : \mathcal{U}_i.\, \Pi a : A.\ = \ (\Sigma\,A)\,(N\,A)\,(S\,A) : \mathcal{U}_{i+1}}{\Gamma \vdash \mathsf{merid} : \Pi A : \mathcal{U}_i.\, \Pi a : A.\ = \ (\Sigma\,A)\,(N\,A)\,(S\,A)}\ \Sigma\mathsf{-intro-=}$$

$$\frac{\begin{array}{l}\Gamma \vdash \Pi A:\mathcal{U}_i.\,\Pi C:(\Pi A:\mathcal{U}_i.\,\Sigma\, A \to \mathcal{U}_i).\\ \quad \Pi c_1:(\Pi A:\mathcal{U}_i.\,C\,A\,(N\,A)).\,\Pi c_2:(\Pi A:\mathcal{U}_i.\,C\,A\,(S\,A)).\\ \quad \Pi\, g_1:(\Pi A:\mathcal{U}_i, a:A.\,c_1\,A =^{C\,A}_{\mathsf{merid}\,A\,a} c_2\,A).\,\Pi x:\Sigma\,A.\,C\,A\,x:\mathcal{U}_{i+1}\end{array}}{\begin{array}{l}\Gamma \vdash \mathsf{ind}_\Sigma:\Pi A:\mathcal{U}_i.\,\Pi C:(\Pi A:\mathcal{U}_i.\,\Sigma\, A \to \mathcal{U}_i).\\ \qquad \Pi c_1:(\Pi A:\mathcal{U}_i.\,C\,A\,(N\,A)).\,\Pi c_2:(\Pi A:\mathcal{U}_i.\,C\,A\,(S\,A)).\\ \qquad \Pi\, g_1:(\Pi A:\mathcal{U}_i, a:A.\,c_1\,A =^{C\,A}_{\mathsf{merid}\,A\,a} c_2\,A).\,\Pi x:\Sigma\,A.\,C\,A\,x\end{array}}\ \Sigma\text{−elim}$$

$$\frac{\begin{array}{l}\Gamma \vdash C:(\Pi A:\mathcal{U}_i.\,\Sigma\,A \to \mathcal{U}_i)\\ \Gamma \vdash c_1:(\Pi A:\mathcal{U}_i.\,C\,A\,(N\,A)) \quad \Gamma \vdash c_2:(\Pi A:\mathcal{U}_i.\,C\,A\,(S\,A))\\ \Gamma \vdash g_1:(\Pi A:\mathcal{U}_i.\,\Pi a:A.\,c_1\,A =^{C\,A}_{\mathsf{merid}\,A\,a} c_2\,A)\\ \Gamma \vdash N\,A:\Sigma\,A\end{array}}{\Gamma \vdash \mathsf{ind}_\Sigma\,A\,C\,c_1\,c_2\,g_1\,(N\,A) \equiv c_1\,A : C\,A\,(N\,A)}\ \Sigma\text{−comp}_1$$

$$\frac{\begin{array}{l}\Gamma \vdash C:(\Pi A:\mathcal{U}_i.\,\Sigma\,A \to \mathcal{U}_i)\\ \Gamma \vdash c_1:(\Pi A:\mathcal{U}_i.\,C\,A\,(N\,A)) \quad \Gamma \vdash c_2:(\Pi A:\mathcal{U}_i.\,C\,A\,(S\,A))\\ \Gamma \vdash g_1:(\Pi A:\mathcal{U}_i.\,\Pi a:A.\,c_1\,A =^{C\,A}_{\mathsf{merid}\,A\,a} c_2\,A)\\ \Gamma \vdash S\,A:\Sigma\,A\end{array}}{\Gamma \vdash \mathsf{ind}_\Sigma\,A\,C\,c_1\,c_2\,g_1\,(S\,A) \equiv c_2\,A : C\,A\,(S\,A)}\ \Sigma\text{−comp}_2$$

$$\frac{\begin{array}{l}\Gamma \vdash C:(\Pi A:\mathcal{U}_i.\,\Sigma\,A \to \mathcal{U}_i)\\ \Gamma \vdash c_1:(\Pi A:\mathcal{U}_i.\,C\,A\,(N\,A)) \quad \Gamma \vdash c_2:(\Pi A:\mathcal{U}_i.\,C\,A\,(S\,A))\\ \Gamma \vdash g_1:(\Pi A:\mathcal{U}_i.\,\Pi a:A.\,c_1\,A =^{C\,A}_{\mathsf{merid}\,A\,a} c_2\,A)\\ \Gamma \vdash \mathsf{merid}\,A\,a{:}=\ (\Sigma\,A)\,(N\,A)\,(S\,A)\end{array}}{\begin{array}{l}\Gamma \vdash \mathbb{C}:=(c_1\,A =^{C\,A}_{\mathsf{merid}\,A\,a} c_2\,A)\\ \qquad (\mathsf{adp}_{\mathsf{ind}_\Sigma\,A\,C\,c_1\,c_2\,g_1}(\mathsf{merid}\,A\,a))\,(g_1\,A\,a)\end{array}}\ \Sigma\text{−comp}-=$$

Pushout

As for suspensions, the inl and inr symbols are used even if they collide with coproducts: we prefer to follow the traditional writing of [95].

$$\frac{\Gamma \vdash \Pi C:\mathcal{U}_i.\,\Pi A:\mathcal{U}_i.\,\Pi B:\mathcal{U}_i.\,\Pi f:C \to A.\,\Pi g:C \to B.\mathcal{U}_i:\mathcal{U}_{i+1}}{\Gamma \vdash \sqcup:\Pi C:\mathcal{U}_i.\,\Pi A:\mathcal{U}_i.\,\Pi B:\mathcal{U}_i.\,\Pi f:C \to A.\,\Pi g:C \to B.\mathcal{U}_i}\ \sqcup\text{−form}$$

$$\frac{\begin{array}{l}\Gamma \vdash \Pi C:\mathcal{U}_i.\,\Pi A:\mathcal{U}_i.\,\Pi B:\mathcal{U}_i.\\ \quad \Pi f:C \to A.\,\Pi g:C \to B.\,\Pi a:A.\sqcup C\,A\,B\,f\,g:\mathcal{U}_{i+1}\end{array}}{\begin{array}{l}\Gamma \vdash \mathsf{inl}:\Pi C:\mathcal{U}_i.\,\Pi A:\mathcal{U}_i.\,\Pi B:\mathcal{U}_i.\\ \qquad \Pi f:C \to A.\,\Pi g:C \to B.\,\Pi a:A.\sqcup C\,A\,B\,f\,g\end{array}}\ \sqcup\text{−intro}_1$$

$$\frac{\begin{array}{l}\Gamma \vdash \Pi C:\mathcal{U}_i.\,\Pi A:\mathcal{U}_i.\,\Pi B:\mathcal{U}_i.\\ \quad \Pi f:C \to A.\,\Pi g:C \to B.\,\Pi b:B.\sqcup C\,A\,B\,f\,g:\mathcal{U}_{i+1}\end{array}}{\begin{array}{l}\Gamma \vdash \mathsf{inr}:\Pi C:\mathcal{U}_i.\,\Pi A:\mathcal{U}_i.\,\Pi B:\mathcal{U}_i.\\ \qquad \Pi f:C \to A.\,\Pi g:C \to B.\,\Pi b:B.\sqcup C\,A\,B\,f\,g\end{array}}\ \sqcup\text{−intro}_2$$

$$\dfrac{\begin{array}{l}\Gamma \vdash \Pi C : \mathcal{U}_i .\, \Pi A : \mathcal{U}_i .\, \Pi B : \mathcal{U}_i .\, \Pi f : C \to A .\, \Pi g : C \to B .\, \Pi c : C . \\ \quad = (\sqcup\, C\, A\, B\, f\, g)(\mathsf{inl}\, C\, A\, B\, f\, g\, (f\, c))(\mathsf{inr}\, C\, A\, B\, f\, g\, (g\, c)) : \mathcal{U}_{i+1}\end{array}}{\begin{array}{l}\Gamma \vdash \mathsf{glue} : \Pi C : \mathcal{U}_i .\, \Pi A : \mathcal{U}_i .\, \Pi B : \mathcal{U}_i .\, \Pi f : C \to A .\, \Pi g : C \to B .\, \Pi c : C . \\ \qquad = (\sqcup\, C\, A\, B\, f\, g)(\mathsf{inl}\, C\, A\, B\, f\, g\, (f\, c))(\mathsf{inr}\, C\, A\, B\, f\, g\, (g\, c))\end{array}}\ \textsf{⊔−intro−=}$$

$$\dfrac{\begin{array}{l}\Gamma \vdash \Pi C : \mathcal{U}_i .\, \Pi A : \mathcal{U}_i .\, \Pi B : \mathcal{U}_i .\, \Pi f : C \to A .\, \Pi g : C \to B . \\ \quad \Pi P : (\Pi C : \mathcal{U}_i .\, \Pi A : \mathcal{U}_i .\, \Pi B : \mathcal{U}_i .\, \Pi f : C \to A .\, \Pi g : C \to B . \\ \qquad \sqcup\ C\, A\, B\, f\, g \to \mathcal{U}_i). \\ \quad \Pi c_1 : (\Pi C : \mathcal{U}_i .\, \Pi A : \mathcal{U}_i .\, \Pi B : \mathcal{U}_i .\, \Pi f : C \to A .\, \Pi g : C \to B . \\ \qquad \Pi a : A . P\, C\, A\, B\, f\, g\, (\mathsf{inl}\, C\, A\, B\, f\, g\, a)). \\ \quad \Pi c_2 : (\Pi C : \mathcal{U}_i .\, \Pi A : \mathcal{U}_i .\, \Pi B : \mathcal{U}_i .\, \Pi f : C \to A .\, \Pi g : C \to B . \\ \qquad \Pi b : B . P\, C\, A\, B\, f\, g\, (\mathsf{inr}\, C\, A\, B\, f\, g\, b)). \\ \quad \Pi g_1 : (\Pi C : \mathcal{U}_i .\, \Pi A : \mathcal{U}_i .\, \Pi B : \mathcal{U}_i .\, \Pi f : C \to A .\, \Pi g : C \to B . \\ \qquad \Pi c : C . (c_1\, C\, A\, B\, f\, g\, (f\, c)) \\ \qquad\qquad =^{P\, C\, A\, B\, f\, g}_{\mathsf{glue}\, C\, A\, B\, f\, g\, c} (c_2\ C\, A\, B\, f\, g\, (g\, c)) : \mathcal{U}_{i+1}\end{array}}{\begin{array}{l}\Gamma \vdash \mathsf{ind}_{\sqcup} : \Pi C : \mathcal{U}_i .\, \Pi A : \mathcal{U}_i .\, \Pi B : \mathcal{U}_i .\, \Pi f : C \to A .\, \Pi g : C \to B . \\ \quad \Pi P : (\Pi C : \mathcal{U}_i .\, \Pi A : \mathcal{U}_i .\, \Pi B : \mathcal{U}_i .\, \Pi f : C \to A .\, \Pi g : C \to B . \\ \qquad \sqcup\ C\, A\, B\, f\, g \to \mathcal{U}_i). \\ \quad \Pi c_1 : (\Pi C : \mathcal{U}_i .\, \Pi A : \mathcal{U}_i .\, \Pi B : \mathcal{U}_i .\, \Pi f : C \to A .\, \Pi g : C \to B . \\ \qquad \Pi a : A . P\, C\, A\, B\, f\, g\, (\mathsf{inl}\, C\, A\, B\, f\, g\, a)). \\ \quad \Pi c_2 : (\Pi C : \mathcal{U}_i .\, \Pi A : \mathcal{U}_i .\, \Pi B : \mathcal{U}_i .\, \Pi f : C \to A .\, \Pi g : C \to B . \\ \qquad \Pi b : B . P\, C\, A\, B\, f\, g\, (\mathsf{inr}\, C\, A\, B\, f\, g\, b)). \\ \quad \Pi g_1 : (\Pi C : \mathcal{U}_i .\, \Pi A : \mathcal{U}_i .\, \Pi B : \mathcal{U}_i .\, \Pi f : C \to A .\, \Pi g : C \to B . \\ \qquad \Pi c : C . (c_1\, C\, A\, B\, f\, g\, (f\, c)) \\ \qquad\qquad =^{P\, C\, A\, B\, f\, g}_{\mathsf{glue}\, C\, A\, B\, f\, g\, c} (c_2\ C\, A\, B\, f\, g\, (g\, c))\end{array}}\ \textsf{⊔−elim}$$

$$\dfrac{\begin{array}{l}\Gamma \vdash P : (\Pi C : \mathcal{U}_i .\, \Pi A : \mathcal{U}_i .\, \Pi B : \mathcal{U}_i .\, \Pi f : C \to A .\, \Pi g : C \to B . \\ \qquad \sqcup\ C\, A\, B\, f\, g \to \mathcal{U}_i). \\ \Gamma \vdash c_1 : (\Pi C : \mathcal{U}_i .\, \Pi A : \mathcal{U}_i .\, \Pi B : \mathcal{U}_i .\, \Pi f : C \to A .\, \Pi g : C \to B .\, \Pi a : A . \\ \qquad P\, C\, A\, B\, f\, g\, (\mathsf{inl}\, C\, A\, B\, f\, g\, a)). \\ \Gamma \vdash c_2 : (\Pi C : \mathcal{U}_i .\, \Pi A : \mathcal{U}_i .\, \Pi B : \mathcal{U}_i .\, \Pi f : C \to A .\, \Pi g : C \to B .\, \Pi b : B . \\ \qquad P\, C\, A\, B\, f\, g\, (\mathsf{inr}\, C\, A\, B\, f\, g\, b)). \\ \Gamma \vdash g_1 : (\Pi C : \mathcal{U}_i .\, \Pi A : \mathcal{U}_i .\, \Pi B : \mathcal{U}_i .\, \Pi f : C \to A .\, \Pi g : C \to B .\, \Pi c : C . \\ \qquad (c_1\, C\, A\, B\, f\, g\, (f\, c)) =^{P\, C\, A\, B\, f\, g}_{\mathsf{glue}\, C\, A\, B\, f\, g\, c} (c_2\ C\, A\, B\, f\, g\, (g\, c)) : \mathcal{U}_{i+1} \\ \Gamma \vdash \mathsf{inl}\, C\, A\, B\, f\, g\, a : \sqcup\, C\, A\, B\, f\, g\end{array}}{\begin{array}{l}\Gamma \vdash \mathsf{ind}_{\sqcup}\, C\, A\, B\, f\, g\, c_1\, c_2\, g_1\, (\mathsf{inl}\, C\, A\, B\, f\, g\, a) \equiv \\ \qquad c_1\, C\, A\, B\, f\, g\, a : P\, C\, A\, B\, f\, g\, (\mathsf{inl}\, C\, A\, B\, f\, g\, a)\end{array}}\ \textsf{⊔−comp}_1$$

$$\frac{\begin{array}{l}\Gamma \vdash P : (\Pi C : \mathcal{U}_i.\, \Pi A : \mathcal{U}_i.\, \Pi B : \mathcal{U}_i.\, \Pi f : C \to A.\, \Pi g : C \to B. \\ \qquad \sqcup\, C\, A\, B\, f\, g \to \mathcal{U}_i). \\ \Gamma \vdash c_1 : (\Pi C : \mathcal{U}_i.\, \Pi A : \mathcal{U}_i.\, \Pi B : \mathcal{U}_i.\, \Pi f : C \to A.\, \Pi g : C \to B.\, \Pi a : A. \\ \qquad P\, C\, A\, B\, f\, g\, (\mathsf{inl}\, C\, A\, B\, f\, g\, a)). \\ \Gamma \vdash c_2 : (\Pi C : \mathcal{U}_i.\, \Pi A : \mathcal{U}_i.\, \Pi B : \mathcal{U}_i.\, \Pi f : C \to A.\, \Pi g : C \to B.\, \Pi b : B. \\ \qquad P\, C\, A\, B\, f\, g\, (\mathsf{inr}\, C\, A\, B\, f\, g\, b)). \\ \Gamma \vdash g_1 : (\Pi C : \mathcal{U}_i.\, \Pi A : \mathcal{U}_i.\, \Pi B : \mathcal{U}_i.\, \Pi f : C \to A.\, \Pi g : C \to B.\, \Pi c : C. \\ \qquad (c_1\, C\, A\, B\, f\, g\, (f\, c)) =^{P\, C\, A\, B\, f\, g}_{\mathsf{glue}\, C\, A\, B\, f\, g\, c} (c_2\, C\, A\, B\, f\, g\, (g\, c)) : \mathcal{U}_{i+1} \\ \Gamma \vdash \mathsf{inr}\, C\, A\, B\, f\, g\, b : \sqcup\, C\, A\, B\, f\, g \end{array}}{\begin{array}{l}\Gamma \vdash \mathsf{ind}_{\sqcup}\, C\, A\, B\, f\, g\, c_1\, c_2\, g_1\, (\mathsf{inr}\, C\, A\, B\, f\, g\, b) \equiv \\ \qquad c_2\, C\, A\, B\, f\, g\, b : P\, C\, A\, B\, f\, g\, (\mathsf{inr}\, C\, A\, B\, f\, g\, b)\end{array}}\ \sqcup\mathsf{-comp}_2$$

$$\frac{\begin{array}{l}\Gamma \vdash P : (\Pi C : \mathcal{U}_i.\, \Pi A : \mathcal{U}_i.\, \Pi B : \mathcal{U}_i.\, \Pi f : C \to A.\, \Pi g : C \to B. \\ \qquad \sqcup\, C\, A\, B\, f\, g \to \mathcal{U}_i). \\ \Gamma \vdash c_1 : (\Pi C : \mathcal{U}_i.\, \Pi A : \mathcal{U}_i.\, \Pi B : \mathcal{U}_i.\, \Pi f : C \to A.\, \Pi g : C \to B.\, \Pi a : A. \\ \qquad P\, \mathsf{inl}\, C\, A\, B\, f\, g\, a). \\ \Gamma \vdash c_2 : (\Pi C : \mathcal{U}_i.\, \Pi A : \mathcal{U}_i.\, \Pi B : \mathcal{U}_i.\, \Pi f : C \to A.\, \Pi g : C \to B.\, \Pi b : B. \\ \qquad P\, \mathsf{inr}\, C\, A\, B\, f\, g\, b). \\ \Gamma \vdash g_1 : (\Pi C : \mathcal{U}_i.\, \Pi A : \mathcal{U}_i.\, \Pi B : \mathcal{U}_i.\, \Pi f : C \to A.\, \Pi g : C \to B.\, \Pi c : C. \\ \qquad (c_1\, C\, A\, B\, f\, g\, (f\, c)) =^{P\, C\, A\, B\, f\, g}_{\mathsf{glue}\, C\, A\, B\, f\, g\, c} (c_2\, C\, A\, B\, f\, g\, (g\, c)) : \mathcal{U}_{i+1} \\ \Gamma \vdash \mathsf{glue}\, C\, A\, B\, f\, g\, c \colon= (\sqcup\, C\, A\, B\, f\, g) \\ \qquad\qquad\qquad (\mathsf{inl}\, C\, A\, B\, f\, g\, (f\, c))(\mathsf{inr}\, C\, A\, B\, f\, g\, (g\, c)) \end{array}}{\begin{array}{l}\Gamma \vdash \mathbb{C} \colon= ((c_1\, C\, A\, B\, f\, g\, (f\, c)) =^{P\, C\, A\, B\, f\, g}_{\mathsf{glue}\, C\, A\, B\, f\, g\, c} (c_2\, C\, A\, B\, f\, g\, (g\, c))) \\ \qquad (\mathsf{adp}_{\mathsf{ind}_{\sqcup}\, C\, A\, B\, f\, g\, c_1\, c_2\, g_1} \mathsf{glue}\, C\, A\, B\, f\, g\, c)\, (g_1\, C\, A\, B\, f\, g\, c)\end{array}}\ \sqcup\mathsf{-comp-=}$$

Truncation

As can be seen in the following, the higher order introduction, elimination and computation rules for truncation do not behave accordingly to our generic syntax.

$$\frac{\Gamma \vdash \Pi A : \mathcal{U}_i. \mathcal{U}_i : \mathcal{U}_{i+1}}{\Gamma \vdash ||\cdot|| : \Pi A : \mathcal{U}_i. \mathcal{U}_i}\ ||\cdot||\mathsf{-form}$$

$$\frac{\Gamma \vdash \Pi A : \mathcal{U}_i.\, \Pi a : A.\, (||\cdot||\, A) : \mathcal{U}_{i+1}}{\Gamma \vdash |\cdot| : \Pi A : \mathcal{U}_i.\, \Pi a : A.\, (||\cdot||\, A)}\ ||\cdot||\mathsf{-intro}$$

$$\frac{\Gamma \vdash \Pi A : \mathcal{U}_i.\, \Pi x : (||\cdot||\, A).\, \Pi y : (||\cdot|| A).\ = (||\cdot||\, A)\, x\, y : \mathcal{U}_{i+1}}{\Gamma \vdash p : \Pi A : \mathcal{U}_i.\, \Pi x : ||\cdot||\, A.\, \Pi y : ||\cdot||\, A.\ = ||\cdot||\, A\, x\, y}\ ||\cdot||\mathsf{-intro-=}$$

$$\frac{\begin{array}{l}\Gamma \vdash \Pi A : \mathcal{U}_i.\, \Pi C : (\Pi A : \mathcal{U}_i.\, ||\cdot||\, A \to \mathcal{U}_i).\\ \quad \Pi c_1 : (\Pi A : \mathcal{U}_i.\, \Pi a : A.\, C\, A\, (|\cdot|\, A\, a)).\\ \quad \Pi\, g_1 : (\Pi A : \mathcal{U}_i. \Pi x : (||\cdot|| A).\, \Pi y : (||\cdot||\, A).\\ \qquad \Pi u : C\, A\, x.\, \Pi v : C\, A\, y.\, u =^{C\,A}_{p\,A\,x\,y} v).\\ \quad \Pi z : (||\cdot||\, A).\, C\, A\, z : \mathcal{U}_{i+1}\end{array}}{\begin{array}{l}\Gamma \vdash \mathsf{ind}_{||\cdot||} : \Pi A : \mathcal{U}_i.\, \Pi C : (\Pi A : \mathcal{U}_i.\, ||\cdot||\, A \to \mathcal{U}_i).\\ \qquad \Pi c_1 : (\Pi A : \mathcal{U}_i.\, \Pi a : A.\, C\, A\, (|\cdot|\, A\, a)).\\ \qquad \Pi\, g_1 : (\Pi A : \mathcal{U}_i. \Pi x : (||\cdot|| A).\, \Pi y : (||\cdot||\, A).\\ \qquad\quad \Pi u : C\, A\, x.\, \Pi v : C\, A\, y.\, u =^{C\,A}_{p\,A\,x\,y} v).\\ \qquad \Pi z : (||\cdot||\, A).\, C\, A\, z\end{array}}\ ||\cdot||-\mathsf{elim}$$

$$\frac{\begin{array}{l}\Gamma \vdash C : (\Pi A : \mathcal{U}_i.\, (||\cdot||\, A) \to \mathcal{U}_i)\\ \Gamma \vdash c_1 : (\Pi A : \mathcal{U}_i.\, \Pi a : A.\, C\, A\, (|\cdot|\, A\, a))\\ \Gamma \vdash g_1 : (\Pi A : \mathcal{U}_i. \Pi x : (||\cdot|| A).\, \Pi y : (||\cdot||\, A).\\ \qquad \Pi u : C\, A\, x.\, \Pi v : C\, A\, y.\, u =^{C\,A}_{p\,A\,x\,y} v)\\ \Gamma \vdash |\cdot|\, A\, a : (||\cdot||\, A)\end{array}}{\Gamma \vdash \mathsf{ind}_{||\cdot||}\, A\, c_1\, g_1 (|\cdot|\, A\, a) \equiv c_1\, A\, a : C\, A\, (|\cdot|\, A\, a)}\ ||\cdot||-\mathsf{comp}$$

$$\frac{\begin{array}{l}\Gamma \vdash C : (\Pi A : \mathcal{U}_i.\, (||\cdot||\, A) \to \mathcal{U}_i)\\ \Gamma \vdash c_1 : (\Pi A : \mathcal{U}_i.\, \Pi a : A.\, C\, A\, (|\cdot|\, A\, a))\\ \Gamma \vdash g_1 : (\Pi A : \mathcal{U}_i. \Pi x : (||\cdot|| A).\, \Pi y : (||\cdot||\, A).\\ \qquad \Pi u : C\, A\, x.\, \Pi v : C\, A\, y.\, u =^{C\,A}_{p\,A\,x\,y} v)\\ \Gamma \vdash p\, A\, x\, y : ||\cdot||\, A\, x\, y\end{array}}{\begin{array}{l}\Gamma \vdash \mathbb{C} \!:=\ ((\mathsf{ind}_{||\cdot||}\, A\, c_1\, g_1\, x) =^{C\,A}_{p\,A\,x\,y} (\mathsf{ind}_{||\cdot||}\, A\, c_1\, g_1\, y))\\ \qquad\qquad (\mathsf{adp}_{\mathsf{ind}_{||\cdot||}\, A\, c_1\, g_1}\, p\, A\, x\, y)\\ \qquad\qquad (g_1\, A\, x\, y\, (\mathsf{ind}_{||\cdot||}\, A\, c_1\, g_1\, x)\, (\mathsf{ind}_{||\cdot||}\, A\, c_1\, g_1\, y))\end{array}}\ ||\cdot||-\mathsf{comp}-=$$

6.4 Proof theory

Let θ be a 1-higher inductive type; let τ denote a generic, non-higher inductive type. Since $\Pi-\mathsf{ext}$, $\mathcal{U}_i-\mathsf{univ}$, $\theta-\mathsf{form}$, $\theta-\mathsf{intro}$, $\theta-\mathsf{intro}-=$, $\theta-\mathsf{elim}$, and $\theta-\mathsf{comp}-=$ are instance of the $k-\mathsf{intro}$ schema, all the results in Section 3.3 apply to them.

The $\theta-\mathsf{comp}$ rule affects Proposition 3.3.8: the induction in the proof has to be extended by an additional case, which proves the left- and the right-hand sides of the equivalence are derivable. The left-hand side is evidently derivable: the ind_θ term comes from $\theta-\mathsf{elim}$ and repeatedly applying $\Pi-\mathsf{elim}$ with the premises eventually yields the result. The right-hand side is derived analogously since the arguments of $\mathbb{K}_i$ in the last premise of $\theta-\mathsf{comp}$ are all derivable by Lemma 3.3.7. Therefore, all the results in Chapter 3 hold for all the newly introduced results.

Adapting Definition 4.1.1 to homotopy type theory, that is, dropping the $\tau-\mathsf{uniq}$ reduction for every inductive type τ, and adding the $\theta-\mathsf{comp}$ reductions for each 1-higher inductive type θ, which are obtained orienting the conclusions of the $\theta-\mathsf{comp}$ rules, all the results in Chapter 4 smoothly extend.

In fact, the change of Definition 4.1.1 affects:

- Proposition 4.1.2, which is immediate to adapt;

- Proposition 4.1.6, in which the $\theta-$comp case in the induction is completely analogous to the $\tau-$comp case, as the only difference between the two is the number of arguments in the left-hand side of the equivalence;
- Proposition 4.1.8, in which the $\theta-$comp reduction in the type behaves as the $\tau-$comp reduction, ultimately relying on the validity of Proposition 4.1.6;

To obtain Theorem 4.3.12, Proposition 4.3.10 has to be extended to 1-higher inductive types. Since the $\theta-$form and $\theta-$intro rules have the same shape of $\tau-$form and $\tau-$intro, they are already covered in the proof. As before, it is convenient to extract the pattern of these cases in the inductive proof of Proposition 4.3.10. The rules are instances of $k-$intro and the type $B = \Pi y_1 : C_1, \dots, y_n : C_n.\, D$ with D atomic. Since $B\,\sigma \in R(\mathcal{U}_i)$ by induction hypothesis, $(\Gamma \vdash B : \mathcal{U}_i)\sigma$ is strongly normalisable. Saturating the constant b by applying to it enough reducibility candidates $c_j \in R^{\sigma\circ[c_1/y_1,\dots,c_{j-1}/y_{j-1}]}_{\Gamma,\underline{y}:\underline{C}}(C_j)$, one derives by $\Pi-$elim and weakening $(\Gamma, \underline{y} : \underline{C} \vdash b\,c_1 \dots c_n : D)(\sigma \circ [\underline{c}/\underline{y}])$. This judgement is strongly normalisable since every reduction from it maps into reductions from $\Gamma\sigma$, c_j, and $D(\sigma \circ [\underline{c}/\underline{y}])$, which are easily seen to be strongly normalisable from the hypothesis, unless $(b\,c_1 \dots c_n)(\sigma \circ [\underline{c}/\underline{y}])$ reduces as a whole. However, this may happen only by a $\tau-$comp or $\theta-$comp reduction, whose shape ensures the reduced term to be strongly normalisable by the hypothesis. Hence $b\,c_1 \dots c_n \in R^{\sigma\circ[\underline{c}/\underline{y}]}_{\Gamma,\underline{y}:\underline{C}}(D)$. Thus, by iterating Definition 4.3.1 n times, $b\,\sigma \in R(B)$, as required. Applying this pattern to $\theta-$intro$-=$, $\theta-$elim, and $\theta-$comp$-=$, Proposition 4.3.10 is proved with no difficulties.

The very same pattern applies to $\Pi-$ext and $\mathcal{U}_i-$univ, so Proposition 4.3.10 holds even in presence of function extensionality and univalence.

Therefore, calling 1-*HoTT theory* any theory based on the basic system plus a finite number of inductive and 1-higher inductive types, and plus function extensionality and univalence, it holds that

Theorem 6.4.1 (Normalisation)**.** *Let T be a 1-HoTT theory. Then every derivable judgement in T is strongly normalisable. Moreover, irreducible judgements are unique up to conversion.*

We have seen that not every higher inductive type in [95] is 1-higher inductive. In principle, nothing prevents to extend the definitions in Section 6.2 to define k-higher inductive types for every $k \in \mathbb{N}$. Our unproved claim is that all the framework developed so far, and in particular Theorem 6.4.1, holds also for k-higher inductive types. There are two reasons why this path has not been pursued: while 1-higher inductive types are complex but manageable, k-higher inductive types are cumbersome, and quickly become unmanageable; most importantly, the collection of all k-higher inductive types is not exhaustive since, e.g., truncation does not fit into this pattern.

However, truncation validates all the results in Chapters 3 and 4. In fact, the $||\cdot||-$form, $||\cdot||-$intro, $||\cdot||-$intro$-=$, $||\cdot||-$elim, and $||\cdot||-$comp$-=$ rules are instances of the $k-$intro schema, so the results in Section 3.3 apply to them with no further proof. As for $||\cdot||-$comp, the corresponding case in the induction in the proof of Proposition 3.3.8 is completely analogous to the $\theta-$comp case of 1-higher inductive types. About normalisation, after adding the obvious $||\cdot||-\mathsf{comp}$ reduction to Definition 4.1.1 and observing that all the Propositions 4.1.2, 4.1.6 and 4.1.8 are immediately extended as in the 1-higher inductive case, it remains to prove Proposition 4.3.10 to derive Theorem 6.4.1 extended with truncation.

Since $||\cdot||-\mathsf{form}$, $||\cdot||-\mathsf{intro}$, $||\cdot||-\mathsf{intro}-=$, $||\cdot||-\mathsf{elim}$, and $||\cdot||-\mathsf{comp}-=$ are instances of $k-\mathsf{intro}$, proving Proposition 4.3.10 reduces to apply the previously discussed pattern to them, checking its conditions to hold. This step smoothly allows to derive Proposition 4.3.10.

In conclusion, our conjecture is that Theorem 6.4.1 holds for every HoTT-theory, and it can be proved along the lines of Theorem 4.3.12. What prevents to prove this conjecture is that there is no exhaustive syntactical characterisation of higher inductive types, see [95].

6.5 Semantics

This section is devoted to extend the semantics presented in Chapter 5 to the 1-higher inductive types described above.

Categorical preliminaries

As done for the syntax, we need some definitions to simplify the notation. If $x, y \in \mathsf{Obj}\, \mathcal{M}_c \uparrow a$, define

$$\begin{aligned}\mathsf{eq}[a;x;y] :\equiv \bar{E}_\Pi(c, U_i, a, a, \bar{F}_\Pi(c, U_i, \chi^c_{U_i}, \chi_{U_i}^{\mathsf{cod}(\nu^c_{U_i})})(U_i))\\ (F_=(c, \bar{F}_\Pi(c, U_i, \chi^c_{U_i}, \chi_{U_i})(U_i))(\cdot), a, x, y) \enspace .\end{aligned}$$

Then, as for Definition 5.2.9, $\mathsf{eq}[a;x;y]$ is the semantic representation of $x =_a y$, which is, of course, an abuse of notation.

For each

$$p \in \mathsf{Obj}\,(\mathcal{M}_c \uparrow (\mathsf{eq}[a;x;y])) \enspace ,$$

define

$$\begin{aligned}p_* :\equiv\; & \bar{E}_\Pi(c, U_i, a, a, \bar{F}_\Pi(c, a, a)(U_i),\\ & \quad \bar{F}_\Pi(c, a, (E_\Pi(c, a, U_i)(P, x))), \mathsf{eq}[a;x;y], r)\\ & (E_=(c, r)(\cdot), a, x, y, \bar{I}_\Pi(c, a, a, U_i)\\ & \quad (F_\Pi(c, E_\Pi(c, a, U_i)(P, x))(E_\Pi(c, a, U_i)(P, y)),\\ & \quad \bar{I}_\Pi(c, a, E_\Pi(c, a, U_i)(P, x))(\chi_{E_\Pi(c,a,U_i)(P,x)}), p))\end{aligned}$$

with r calculated as in Definition 5.1.24 Point (3), instantiated with the objects $a_1, \ldots, a_n = U_i, \chi^c_{U_i}, \chi_{U_i}^{\mathsf{cod}(\nu^c_{U_i})}$, $a'_1, \ldots, a'_{n'} = U_i, \chi^c_{U_i}$, $d_1, \ldots, d_m = \varnothing$, and $a'_1, \ldots, a'_n = U_i, \chi^c_{U_i}, \chi_{U_i}$. Let q^* such that $p^* \in \mathcal{M}_c \uparrow q^*$. For each $u \in \mathsf{Obj}\,(\mathcal{M}_c \uparrow E_\Pi(c, a, U_i)(P, x))$ and $v \in \mathsf{Obj}\,(\mathcal{M}_c \uparrow E_\Pi(c, a, U_i)(P, y))$ define

$$\begin{aligned}(u =^P_p v) :\equiv\; & \bar{E}_\Pi(c, U_i, q^*, E_\Pi(c, a, U_i)(P, x),\\ & \quad E_\Pi(c, a, U_i)(P, y), \bar{F}_\Pi(c, U_i, \chi^c_{U_i}, \chi_{U_i})(U_i))\\ & (F_=(c, \bar{F}_\Pi(c, U_i, \chi^c_{U_i}, \chi_{U_i})(U_i))(\cdot), a, p^*, u, v) \enspace ;\end{aligned}$$

finally, for each $f \in \mathsf{Obj}\,(\mathcal{M}_c \uparrow F_\Pi(c,a)(P))$,

$$\begin{aligned}\mathsf{adp}_f(p) :\equiv\ & \bar{E}_\Pi(c, U_i, a, a, \bar{F}_\Pi(c, a, a, \mathsf{eq}[a; x; y])(T_{=(Px)(p^* fx)(fx)}),\\ & \quad F_\Pi(c,a)(\mathsf{eq}[E_\Pi(c,a,U_i)(P,x); E_\Pi(c,a,F_\Pi(c,a)(P))(f,x);\\ & \qquad E_\Pi(c,a,F_\Pi(c,a)(P))(f,y)]), \mathsf{eq}[a;x;y], r)\\ & (E_=(c,r)(\cdot), a, x, y, \bar{I}_\Pi(c,a,a,\mathsf{eq}[a;x;y], T_{=(Px)(p^* fx)(fx)})\\ & \quad (\mathsf{eq}[E_\Pi(c,a,U_i)(P,x); \bar{E}_\Pi(c, F_\Pi(c,a)(P), a, q^*)(p^*, f, x);\\ & \qquad E_\Pi(c,a,F_\Pi(c,a)(P))(f,x)]),\\ & \quad I_\Pi(c,a)(I_=(c, \bar{F}_\Pi(c, U_i, \chi^c_{U_i})(\mathsf{eq}[U_i; \chi^c_{U_i}, \chi_{U_i}^{\mathsf{cod}(\nu^c_{U_i})}]))(\cdot), p)) \quad ,\end{aligned}$$

with $T_{=(Px)(p^* fx)(fx)}$ such that

$$\begin{aligned}&\mathsf{eq}[E_\Pi(c,a,U_i)(P,x); \bar{E}_\Pi(c, F_\Pi(c,a)(P), a, q^*)(p^*, f, x);\\ &\quad E_\Pi(c,a,F_\Pi(c,a)(P))(f,x)] \in \mathsf{Obj}\,\mathcal{M}_c \uparrow T_{=(Px)(p^* fx)(fx)} \quad .\end{aligned}$$

Abusing notation, we defined the syntactical and semantic objects p^*, $(u =^b_p v)$ and $\mathsf{adp}_f(p)$ in the same way. It should be clear from the context whether we refer to syntax or semantics.

Definition 6.5.1. Let T be a 1-HoTT theory. A 1-HoTT-category is $\mathfrak{M}^H$, which extends the corresponding category $\mathfrak{M}^i$, see Definition 5.1.24, as follows

1. for each context c and each universe U_i, define an object fe, which interprets in our language the type FunExt; we leave to the reader the exact definition of fe. It is just the rewriting of FunExt as defined in Section 6.1 into the semantics, i.e., if c is the interpretation of Γ ctx, one should calculate $[\![\Gamma \vdash \mathsf{FunExt}]\!]$ in a T-ML-category with the fe object. Thus, FE is a family of functors indexed by c and fe such that
   - $FE(c, fe)\colon \{\cdot\} \to \mathcal{M}_c \uparrow fe$;
   - *FE respects equivalent contexts*: if $c \cong c'$, $FE(c, fe) \cong FE(c', fe)$ in the corresponding functor category.
2. for each context c and universe U_i, define an object u which interprets in our language the type $\mathsf{Univalence}$ interpreting $[\![\Gamma \vdash \mathsf{Univalence}]\!]$ as above. Thus, U is a family of functors indexed by c and u such that
   - $U(c, u)\colon \{\cdot\} \to \mathcal{M}_c \uparrow u$;
   - *U respects equivalent contexts*: if $c \cong c'$, $U(c, u) \cong U(c', u)$ in the corresponding functor category.
3. for each higher inductive type τ, the functor F_τ is defined as done for inductive types, and identified by an object p;
4. for each type τ and constructor $\mathbb{K}$, the functor $I_{\tau,\mathbb{K}}$ is defined as for inductive types, and identified by an object q;
5. each equality-constructor $\mathbb{P}$ of an inductive type τ is uniquely identified by an object q' representing the term in $\tau{-}\mathsf{intro}{-}{=}$, and an object s, which is constructed as q' but refers to the term in $\tau{-}\mathsf{comp}{-}{=}$. Then $I_{\tau,\mathbb{P}}$ is a family of functors indexed by c and q' such that

a) $I_{\tau,\mathbb{P}}(c,q') : \{\cdot\} \to \mathcal{M}_c \uparrow q'$;

b) $I_{\tau,\mathbb{P}}$ *respects equivalent contexts*: if $c \cong c'$ then $I_{\tau,\mathbb{P}}(c,q') \cong I_{\tau,\mathbb{P}}(c',q')$ in the corresponding functor category.

Moreover $C_{\tau,\mathbb{P}}$, interpreting the propositional computation rule for the constructor $\mathbb{P}$, is a family of functors indexed by c and s such that

a) $C_{\tau,\mathbb{P}}(c,s) \colon \{\cdot\} \to \mathcal{M}_c \uparrow s$;

b) $C_{\tau,\mathbb{P}}$ *respects equivalent contexts*: if $c \cong c'$, then in the corresponding functor category $C_{\tau,\mathbb{P}}(c,s) \cong C_{\tau,\mathbb{P}}(c',s)$;

6. in the non recursive case, each higher inductive type τ is uniquely associated to an object

$$\begin{aligned} r = \bar{F}_\Pi(&c, a_1, \dots, a_n, F_C, F_{c_1}, \dots, F_{c_k}, F_{g_1}, \dots, F_{g_l}, E_e, \\ &\bar{E}_\Pi(\mathsf{cod}(\nu_{E_e} \circ \nu_a), a_1, \dots, a_n, E_e, F_C)(\chi_{E_e}, \chi_{a_1}, \dots, \chi_{a_n})) \enspace , \end{aligned}$$

where F_{c_i}, E_e, ν_a, $\nu_{a,d}$, χ_{a_i} are as in Point (3) of Definition 5.1.24, and F_{g_i} represents $\Big(\Pi\,(x : F)_{n'_i} \,.\, \Pi\,(z : P)_{r_i} \,.\, (c_{a_i}\underline{x}(\underline{w_{a_i}}) =^{C\underline{x}}_{\mathbb{P}_l \underline{xz}} c_{b_l}\underline{x}(\underline{w_{b_i}}))\Big)$ in our model. Then E_τ is a family of functors indexed by c and r such that

a) $E_\tau \colon \{\cdot\} \to \mathcal{M}_c \uparrow r$;

b) E_τ *respects equivalent contexts*: if $c \cong c'$, then $E_\tau(c,r) \cong E_\tau(c',r)$;

c) given $C \in \mathsf{Obj}\,\mathcal{M}_c \uparrow F_C$, $c_i \in \mathcal{M}_c \uparrow F_{c_i}$ for $1 \leqslant i \leqslant k$, $g_i \in \mathcal{M}_c \uparrow F_{g_i}$ for $1 \leqslant i \leqslant l$, $T_i \in \mathcal{M}_c \uparrow a_i$ for $1 \leqslant i \leqslant n$, $p_i \in \mathcal{M}_c \uparrow d_i$ for $1 \leqslant i \leqslant m$,

$$\begin{aligned} &\bar{E}_\Pi(c, a_1, \dots, a_n, F_C, F_{c_1}, \dots, F_{c_k}, F_{g_1}, \dots, F_{g_l}, \\ &\quad \bar{E}_\Pi(c, a_1, \dots, a_n, p)(T_1, \dots, T_n, F_\tau(c,p)(\cdot))) \\ &\quad (E_\tau(c,r)(\cdot), T_1, \dots, T_n, C, c_1, \dots, c_k, g_1, \dots, g_l \\ &\quad (\bar{E}_\Pi(c, a_1, \dots, a_n, d_1, \dots, d_m, q) \\ &\qquad (I_{\tau,\mathbb{K}}(c,q)(\cdot), T_1, \dots, T_k, p_1, \dots, p_m))) \\ \cong\ &\bar{E}_\Pi(c, a_1, \dots, a_n, d_1, \dots, d_m, F_{c_i}) \\ &\quad (c_i, T_1, \dots, T_n, p_1, \dots, p_n) \enspace ; \end{aligned}$$

the extension to the recursive case is done as for the inductive types.

7. for each inductive type τ and constructors $\mathbb{K}$ and $\mathbb{P}$, the auxiliary functor S must satisfy the following conditions:

a) if $FE(\mathsf{cod}(\nu^c_a), fe)(\cdot)$ is a type in $\mathcal{M}_{\mathsf{cod}(\nu^c_a)}$ but not in $\mathcal{M}_{\mathsf{dom}(\nu^c_a)}$, then $S(c,a,U_i)(FE(\mathsf{cod}(\nu^c_a), fe)(\cdot), e) \cong FE(c, S(c,a,U_i)(fe,e))(\cdot)$;

b) if $U(\mathsf{cod}(\nu^c_a), u)(\cdot)$ is a type in $\mathcal{M}_{\mathsf{cod}(\nu^c_a)}$ but not in $\mathcal{M}_{\mathsf{dom}(\nu^c_a)}$, then $S(c,a,U_i)(U(\mathsf{cod}(\nu^c_a), u)(\cdot), e) \cong U(c, S(c,a,U_i)(u,e))(\cdot)$;

c) if $F_\tau(\mathsf{cod}(\nu^c_a), p)(\cdot)$ is a type in $\mathcal{M}_{\mathsf{cod}(\nu^c_a)}$ but not in $\mathcal{M}_{\mathsf{dom}(\nu^c_a)}$, then $S(c,a,U_i)(F_\tau(\mathsf{cod}(\nu^c_a), p)(\cdot), e) \cong F_\tau(c, S(c,a,U_i)(p,e))(\cdot)$;

d) if $I_{\tau,\mathbb{K}}(\mathsf{cod}(\nu_a^c), r)(\cdot)$ is a type in $\mathcal{M}_{\mathsf{cod}(\nu_a^c)}$ but not in $\mathcal{M}_{\mathsf{dom}(\nu_a^c)}$, then $S(c, a, U_i)(I_{\tau,\mathbb{K}}(\mathsf{cod}(\nu_a^c), q)(\cdot), e) \cong I_{\tau,\mathbb{K}}(c, S(c, a, U_i)(q, e))(\cdot)$;

e) if $I_{\tau,\mathbb{P}}(\mathsf{cod}(\nu_a^c), q')(\cdot)$ is a type in $\mathcal{M}_{\mathsf{cod}(\nu_a^c)}$ but not in $\mathcal{M}_{\mathsf{dom}(\nu_a^c)}$, then $S(c, a, U_i)(I_{\tau,\mathbb{P}}(\mathsf{cod}(\nu_a^c), q')(\cdot), e) \cong I_{\tau,\mathbb{P}}(c, S(c, a, U_i)(q', e))(\cdot)$;

f) if $C_{\tau,\mathbb{P}}(\mathsf{cod}(\nu_a^c), s)(\cdot)$ is a type in $\mathcal{M}_{\mathsf{cod}(\nu_a^c)}$ but not in $\mathcal{M}_{\mathsf{dom}(\nu_a^c)}$, then $S(c, a, U_i)(C_{\tau,\mathbb{P}}(\mathsf{cod}(\nu_a^c), s)(\cdot), e) \cong C_{\tau,\mathbb{P}}(c, S(c, a, U_i)(s, e))(\cdot)$;

g) if $E_\tau(\mathsf{cod}(\nu_a^c), e)(\cdot)$ is a type in $\mathcal{M}_{\mathsf{cod}(\nu_a^c)}$ but not in $\mathcal{M}_{\mathsf{dom}(\nu_a^c)}$, then $S(c, a, U_i)(E_\tau(\mathsf{cod}(\nu_a^c), r)(\cdot), e) \cong E_\tau(c, S(c, a, U_i)(r, e))(\cdot)$.

Following the discussion about uniqueness rules at the end of Section 6.2, we do not require a 1-HoTT-category to satisfy the requirements of Definition 5.1.24 about uniqueness, i.e., Point (3d).

As done in Section 6.4 for the proof theory, Definition 6.5.1 can be extended to interpret truncation. It is enough to add five functors, as follows.

1. For each context c, is defined the object $\tilde{p}$ interpreting in our language the term in $||\cdot||-\mathsf{form}$. Then, is defined $F_{||\cdot||}$ such that

   a) $F_{||\cdot||}(c, \tilde{p}) \colon \{\cdot\} \to \mathcal{M}_c \uparrow \tilde{p}$;

   b) $F_{||\cdot||}$ *respects equivalent contexts*: if $c \cong c'$ then $F_{||\cdot||}(c, \tilde{p}) \cong F_{||\cdot||}(c', \tilde{p})$ in the corresponding functor category;

   c) if $F_{||\cdot||}(\mathsf{cod}(\nu_a^c), \tilde{p})(\cdot)$ is a type in $\mathcal{M}_{\mathsf{cod}(\nu_a^c)}$ but not in $\mathcal{M}_{\mathsf{dom}(\nu_a^c)}$, then $S(c, a, U_i)(F_{||\cdot||}(\mathsf{cod}(\nu_a^c), \tilde{p})(\cdot), e) \cong F_{||\cdot||}(c, S(c, a, U_i)(\tilde{p}, e))(\cdot)$.

2. For each context c, is defined the object $\tilde{q}$ interpreting in our language the term in $||\cdot||-\mathsf{intro}$. Then, is defined $I_{||\cdot||}$ such that

   a) $I_{||\cdot||}(c, \tilde{q}) \colon \{\cdot\} \to \mathcal{M}_c \uparrow \tilde{q}$;

   b) $I_{||\cdot||}$ *respects equivalent contexts*: if $c \cong c'$ then $I_{||\cdot||}(c, \tilde{q}) \cong I_{||\cdot||}(c', \tilde{q})$ in the corresponding functor category;

   c) if $I_{||\cdot||}(\mathsf{cod}(\nu_a^c), \tilde{q})(\cdot)$ is a type in $\mathcal{M}_{\mathsf{cod}(\nu_a^c)}$ but not in $\mathcal{M}_{\mathsf{dom}(\nu_a^c)}$, then $S(c, a, U_i)(I_{||\cdot||}(\mathsf{cod}(\nu_a^c), \tilde{q})(\cdot), e) \cong I_{||\cdot||}(c, S(c, a, U_i)(\tilde{q}, e))(\cdot)$.

3. For each context c, is defined the object $\tilde{q}'$ interpreting in our language the term in $||\cdot||-\mathsf{intro}-=$. Then, is defined $I_{||\cdot||,=}$ such that

   a) $I_{||\cdot||,=}(c, \tilde{q}') \colon \{\cdot\} \to \mathcal{M}_c \uparrow \tilde{q}'$;

   b) $I_{||\cdot||,=}$ *respects equivalent contexts*: if $c \cong c'$ then $I_{||\cdot||,=}(c, \tilde{q}') \cong I_{||\cdot||,=}(c', \tilde{q}')$ in the corresponding functor category;

   c) if $I_{||\cdot||}(\mathsf{cod}(\nu_a^c), \tilde{q}')(\cdot)$ is a type in $\mathcal{M}_{\mathsf{cod}(\nu_a^c)}$ but not in $\mathcal{M}_{\mathsf{dom}(\nu_a^c)}$, then $S(c, a, U_i)(I_{||\cdot||,=}(\mathsf{cod}(\nu_a^c), \tilde{q}')(\cdot), e) \cong I_{||\cdot||,=}(c, S(c, a, U_i)(\tilde{q}', e))(\cdot)$.

4. For each context c, is defined the object $\tilde{s}$ interpreting in our language the term in $||\cdot||-\mathsf{comp}-=$. Then, is defined $C_{||\cdot||}$ such that

   a) $C_{||\cdot||}(c, \tilde{s}) \colon \{\cdot\} \to \mathcal{M}_c \uparrow \tilde{s}$;

   b) $C_{||\cdot||}$ *respects equivalent contexts*: if $c \cong c'$, $C_{||\cdot||}(c, \tilde{s}) \cong C_{||\cdot||}(c', \tilde{s})$ in the corresponding functor category;

c) if $C_{||\cdot||}(\mathsf{cod}(\nu_a^c), \tilde{s})(\cdot)$ is a type in $\mathcal{M}_{\mathsf{cod}(\nu_a^c)}$ but not in $\mathcal{M}_{\mathsf{dom}(\nu_a^c)}$, then $S(c, a, U_i)(C_{||\cdot||}(\mathsf{cod}(\nu_a^c), \tilde{s})(\cdot), e) \cong C_{||\cdot||}(c, S(c, a, U_i)(\tilde{s}, e))(\cdot)$.

5. For each context c, is defined the object $\tilde{r}$ interpreting in our language the term in $||\cdot||{-}\mathsf{elim}$. Then, is defined $E_{||\cdot||}$ such that

   a) $E_{||\cdot||}(c, \tilde{r})\colon \{\cdot\} \to \mathcal{M}_c \uparrow \tilde{r}$;

   b) $E_{||\cdot||}$ *respects equivalent contexts*: if $c \cong c'$, $E_{||\cdot||}(c, \tilde{r}) \cong E_{||\cdot||}(c', \tilde{r})$ in the corresponding functor category;

   c) a condition analogous to (6c) of Definition 6.5.1, requiring the existence of an isomorphism between the object interpreting in our language $\Gamma \vdash \mathsf{ind}_{||\cdot||}\, A\, c_1\, g_1(|\cdot|\, A\, a)$ and the one interpreting $\Gamma \vdash c_1\, A\, a$;

   d) if $E_{||\cdot||}(\mathsf{cod}(\nu_a^c), \tilde{r})(\cdot)$ is a type in $\mathcal{M}_{\mathsf{cod}(\nu_a^c)}$ but not in $\mathcal{M}_{\mathsf{dom}(\nu_a^c)}$, then $S(c, a, U_i)(E_{||\cdot||}(\mathsf{cod}(\nu_a^c), \tilde{r})(\cdot), e) \cong E_{||\cdot||}(c, S(c, a, U_i)(\tilde{r}, e))(\cdot)$.

The interpretation $[\![\cdot]\!]$ and all the results in the following can be extended to truncation in the obvious way.

Interpretation

The interpretation extends the one for inductive types:

Definition 6.5.2. Given a T-ML-category $\mathfrak{M}^H$, for each higher inductive type τ extend the interpretation of Definitions 5.2.1 and 5.2.9 as follows:

1. if $\Gamma \vdash \mathsf{FunExt} : \mathcal{U}_{i+1}$ is valid, then $\Gamma \vdash \mathsf{funext} : \mathsf{FunExt}$ is valid, and $[\![\Gamma \vdash \mathsf{funext}]\!] = FE([\![\Gamma\ \mathsf{ctx}]\!], [\![\Gamma \vdash \mathsf{FunExt}]\!])(\cdot)$;

2. if $\Gamma \vdash \mathsf{univalence} : \mathcal{U}_{i+1}$ is valid, then $\Gamma \vdash \mathsf{univalence} : \mathsf{Univalence}$ is valid, and $[\![\Gamma \vdash \mathsf{univalence}]\!] = U([\![\Gamma\ \mathsf{ctx}]\!], [\![\Gamma \vdash \mathsf{Univalence}]\!])(\cdot)$;

3. if $\Gamma \vdash \Pi\,(x:F)_n\,.\mathcal{U}_i : \mathcal{U}_{i+1}$ is valid, then $\Gamma \vdash \tau : \Pi\,(x:F)_n\,.\mathcal{U}_i$ is valid and $[\![\Gamma \vdash \tau]\!] = F_\tau([\![\Gamma\ \mathsf{ctx}]\!], [\![\Gamma \vdash \Pi\,(x:F)_n\,.\mathcal{U}_i]\!])(\cdot)$; the interpretation of the type-in-context $\Gamma \vdash \Pi\,(x:F)_n\,.\mathcal{U}_i$ can be computed from the interpretations of the $[\![\Gamma \vdash F_i]\!]$ through the functor F_Π;

4. if the judgement $\Gamma \vdash \Pi\,(x:F)_{n'}\,.\,\Pi\,(y:I)_m\,.\,\tau\, x'_1 \cdots x'_n : \mathcal{U}_i$ is valid, then the judgement $\Gamma \vdash \mathbb{K} : \Pi\,(x:F)_{n'}\,.\,\Pi\,(y:I)_m\,.\,\tau\, x'_1 \cdots x'_n$ is valid and $[\![\Gamma \vdash \mathbb{K}]\!] = I_{\tau,\mathbb{K}}([\![\Gamma\ \mathsf{ctx}]\!], [\![\Gamma \vdash \Pi\,(x:F)_{n'}\,.\,\Pi\,(y:I)_m\,.\,\tau\, x'_1 \cdots x'_n]\!])(\cdot)$. As above, the latter interpretation can be easily obtained through F_Π and E_Π;

5. if the judgement

$$\begin{aligned}\Gamma \vdash (\Pi\,(x:F)_{n'}\,.\,\Pi\,(z:P)_r\,.\, &= (\tau\, x'_1 \cdots x'_n)\\ &(\mathbb{K}_a x'_1 \cdots x'_n\, w^a_1 \cdots w^a_{s_a})(\mathbb{K}_b x'_1 \cdots x'_n\, w^b_1 \cdots w^b_{s_b})) : \mathcal{U}_i\end{aligned}$$

   is valid, then the judgement

$$\begin{aligned}\Gamma \vdash \mathbb{P} : (\Pi\,(x:F)_{n'}\,.\,\Pi\,(z:P)_r\,.\, &= (\tau\, x'_1 \cdots x'_n)\\ &(\mathbb{K}_a x'_1 \cdots x'_n\, w^a_1 \cdots w^a_{s_a})(\mathbb{K}_b x'_1 \cdots x'_n\, w^b_1 \cdots w^b_{s_b}))\end{aligned}$$

is valid and

$$\begin{aligned}\llbracket\Gamma \vdash \mathbb{P}\rrbracket = I_{\tau,\mathbb{P}}(\llbracket\Gamma\ \mathsf{ctx}\rrbracket, \llbracket\Gamma \vdash (\Pi\,(x:F)_{n'}\,.\,\Pi\,(z:P)_r\,.\, = (\tau\, x'_1\, \cdots\, x'_n)\\ (\mathbb{K}_a x'_1 \cdots x'_n\, w^a_1 \cdots w^a_{s_a})(\mathbb{K}_b x'_1 \cdots x'_n\, w^b_1 \cdots w^b_{s_b}))\rrbracket)(\cdot)\ \ ;\end{aligned}$$

6. if

$$\begin{aligned}&\Gamma \vdash C : (\Pi\,(x:F)_n\,.\,\tau\, x_1\, \cdots\, x_n \to \mathcal{U}_h)\\ &\Big\{\Gamma \vdash c_j : \Big(\Pi\,(x:F)_{n'_1}\,.\,\Pi\,(y:I)_{m'_j}\,.\\ &\qquad C\, x'_1 \cdots x'_{n_j}\ \Big(\mathbb{K}_1\, x'_1 \cdots x'_{n_j}\, y_1\, \cdots\, y_{m_j}\Big)\Big)\Big\}_{j=1}^{k}\\ &\Big\{\Gamma \vdash g_j : \Big(\Pi\,(x:F)_{n'_j}\,.\,\Pi\,(z:P)_{r_j}\,.\,(c_{a_j}\underline{x}(\underline{w_{a_j}}) =^{C\underline{x}}_{\mathbb{P}_i\underline{x}\underline{z}} c_{b_j}\underline{x}(\underline{w_{b_j}}))\Big)\Big\}_{j=1}^{l}\\ &\Gamma \vdash \mathbb{P}_i\, T_1\, \cdots\, T_n\, q_1\, \cdots\, q_{r_i} := (\tau\, T_1\, \cdots\, T_n)\\ &\qquad\qquad (\mathbb{K}_{a_i}\underline{T}(q^{a_i}_1\, \cdots\, q^{a_i}_{s_{a_i}}))(\mathbb{K}_{b_i}\underline{T}(q^{b_i}_1\, \cdots\, q^{b_i}_{s_{b_i}}))\end{aligned}$$

are valid, then the judgement

$$\Gamma \vdash \mathbb{C}_i := (c_{a_i}\underline{T}(\underline{T}\underline{q_{a_i}}) =^{C\underline{T}}_{\mathbb{P}_l\underline{T}\underline{q}} c_{b_i}\underline{T}(\underline{T}\underline{q_{b_i}}))(\mathsf{adp}_{\mathsf{ind}_\tau}\,\underline{T}\,C\,\underline{c}\,\underline{d}(\mathbb{P}_i\,\underline{T}\underline{q}))(g_i\,\underline{T}\underline{q})$$

is valid, and

$$\begin{aligned}\llbracket\Gamma \vdash \mathbb{C}_i\rrbracket = C_{\tau,\mathbb{P}_i}(\llbracket\Gamma\ \mathsf{ctx}\rrbracket, \llbracket\Gamma \vdash = (c_{a_i}\underline{T}(\underline{T}\underline{q_{a_i}}) =^{C\underline{T}}_{\mathbb{P}_l\underline{T}\underline{q}} c_{b_i}\underline{T}(\underline{T}\underline{q_{b_i}}))\\ (\mathsf{adp}_{\mathsf{ind}_\tau}\,\underline{T}\,C\,\underline{c}\,\underline{d}(\mathbb{P}_i\,\underline{T}\underline{q}))(g_i\,\underline{T}\underline{q})\rrbracket)\ \ ;\end{aligned}$$

7. let

$$\begin{aligned}T_{\mathsf{ind}} := \Pi\,(x:F)_n\,.\,(\Pi C : (\Pi\,(x:F)_n\,.\,\tau\, x_1\, \cdots\, x_n \to \mathcal{U}_h).\\ (\Pi^k_{i=1} c_i : (\Pi\,(x:F)_{n'_i}\,.\,\Pi\,(y:I)_{m_i}\,.\\ C\, x'_1\, \cdots\, x'_{n_i}\, (\mathbb{K}\, x'_1\, \cdots\, x'_{n_i}\, y_1\, \cdots\, y_{m_i})).\\ (\Pi^k_{i=1} g_i : (\Pi\,(x:F)_{n'_i}\,.\,\Pi\,(z:P)_{r_i}\,.\\ (c_a\underline{x}(\underline{w_a}) =^{C\underline{x}}_{\mathbb{P}_i\underline{x}\underline{z}} c_b\underline{x}(\underline{w_b}))).\\ (\Pi e : \tau\, x_1\, \cdots\, x_n.\, C\, x_1\, \cdots\, x_n\, e))))\ \ .\end{aligned}$$

If $\Gamma \vdash T_{\mathsf{ind}} : \mathcal{U}_{h+1}$ is valid, then $\Gamma \vdash \mathsf{ind}_\tau : T_{\mathsf{ind}}$ is valid and $\llbracket\Gamma \vdash \mathsf{ind}_\tau\rrbracket = E_\tau(\llbracket\Gamma\ \mathsf{ctx}\rrbracket, \llbracket\Gamma \vdash T_{\mathsf{ind}}\rrbracket)$.

Lemma 6.5.3 (Substitution)**.** *If* $\Gamma, x : A \vdash b : B$ *and* $\Gamma \vdash a : A$ *are valid, then* $\llbracket\Gamma \vdash (\lambda x : A.\, b)a\rrbracket \cong \llbracket\Gamma \vdash b[a/x]\rrbracket$.

Proof. It has the same structure of the poofs of Lemmas 5.2.2 and 5.2.10. The validity of the result is ensured by Point (7) of Definition 6.5.1. As an example, consider the case of function extensionality, i.e., $b = \mathsf{funext}$.

$$\begin{aligned}&\llbracket\Gamma \vdash (\lambda x : A.\,\mathsf{funext})\, a\rrbracket\\ &= S(\llbracket\Gamma\ \mathsf{ctx}\rrbracket, \llbracket\Gamma \vdash A\rrbracket, \llbracket\Gamma, x : A \vdash \mathsf{FunExt}\rrbracket)\\ &\quad (FE(\llbracket\Gamma, x : A\ \mathsf{ctx}\rrbracket, \llbracket\Gamma, x : A \vdash \mathsf{FunExt}\rrbracket)(\cdot), \llbracket\Gamma \vdash a\rrbracket)\\ &\cong FE(\llbracket\Gamma\ \mathsf{ctx}\rrbracket, S(\llbracket\Gamma\ \mathsf{ctx}\rrbracket, \llbracket\Gamma \vdash A\rrbracket, \llbracket\Gamma, x : A \vdash \mathcal{U}_{i+1}\rrbracket)\end{aligned}$$

$$([\![\Gamma, x : A \vdash \mathsf{FunExt}]\!], [\![\Gamma \vdash a]\!]))(\cdot)$$

and, by induction hypothesis,

$$\begin{aligned} &\cong FE([\![\Gamma\ \mathsf{ctx}]\!], [\![\Gamma \vdash (\mathsf{FunExt})[a/x]]\!])(\cdot) \\ &= [\![\Gamma \vdash \mathsf{funext}[a/x]]\!]\ . \end{aligned}$$ □

As already seen in Section 5.2 for inductive theories, Proposition 5.2.3, Corollary 5.2.4 and Fact 5.2.6 trivially extend to 1-HoTT theories.

Definition 6.5.4. A *model* for a 1-HoTT theory T is composed by a 1-HoTT-category and an interpretation over it such that, for every $\gamma \in T$, γ is valid.

Theorem 6.5.5 (Soundness)**.** *Let T be a set of derivable judgements on a 1-HoTT theory. Then all the judgements in T are valid in every model.*

Proof. Since a model for a 1-HoTT theory extends a model for an inductive system, and Theorem 5.2.12 proves the soundness of the inductive systems, it is enough to prove the statement when γ is the conclusion of a rule for an 1-higher inductive type or an axiom:

- if the premises are valid, then by Definition 6.5.2 the conclusions of $\tau-\mathsf{form}$, $\tau-\mathsf{intro}_i$, $\tau-\mathsf{elim}$, $\tau-\mathsf{intro}-=$, $\tau-\mathsf{comp}-=$, $\Pi-\mathsf{ext}$ and $\mathcal{U}_i-\mathsf{univ}$ are valid;
- the observations after Definition 6.5.2 show that, when the premises are valid, also the conclusion of the $\tau-\mathsf{comp}_i$ rule is valid. □

Classifying model and completeness

Section 6.4 allows to extend the results obtained in Section 5.3 to a 1-HoTT theory T. Thus, the syntactical category for the inductive types can be extended as follows.

Definition 6.5.6. If T is a 1-HoTT theory, its *syntactical category* is the one of Definitions 5.3.4 and 5.3.20 extended with the following families of functors:

- $\mathcal{FE}([\Gamma\ \mathsf{ctx}], [\Gamma \vdash \mathsf{FunExt}])(\cdot) = ([\Gamma \vdash \mathsf{funext}], [\Gamma\ \mathsf{ctx}])$,
- $\mathcal{U}([\Gamma\ \mathsf{ctx}], [\Gamma \vdash \mathsf{Univalence}])(\cdot) = ([\Gamma \vdash \mathsf{univalence}], [\Gamma\ \mathsf{ctx}])$,

and, for each instance of the rules for a 1-higher inductive type τ, the functors $\mathcal{F}_\tau$, $\mathcal{I}_{\tau,\mathbb{K}}$, $\mathcal{E}_\tau$, $\mathcal{I}_{\tau,\mathbb{P}}$, $\mathcal{C}_{\tau,\mathbb{P}}$ are defined, respectively, by the objects $([\Gamma \vdash \tau], [\Gamma\ \mathsf{ctx}])$, $([\Gamma \vdash \mathbb{K}_i], [\Gamma\ \mathsf{ctx}])$, $([\Gamma \vdash \mathsf{ind}_\tau], [\Gamma\ \mathsf{ctx}])$, $([\Gamma \vdash \mathbb{P}_i], [\Gamma\ \mathsf{ctx}])$ and $([\Gamma \vdash \mathbb{C}_i], [\Gamma\ \mathsf{ctx}])$, as done in Definition 5.3.20.

The canonical interpretation $[\![\cdot]\!]_T$ over T extends the interpretation for the basic system of Definition 5.3.16 in the obvious way.

Proposition 6.5.7. *$\mathfrak{M}^H_T$, defined extending $\mathfrak{M}^i_T$ with the functors of Definition 6.5.6, is a 1-HoTT-category, and $[\![\cdot]\!]_T$ is an interpretation over it.*

Proof. $\mathfrak{M}^i_T$ is a T-ML-category by Proposition 5.3.21, thus it is enough to show that the requirements of Definition 5.1.24 are satisfied. This is a long but trivial calculation, done as in the proof of Proposition 5.3.21. As an example, we check the requirements for function extensionality.

1. By Definition 6.5.6, $\mathcal{FE}$ is appropriately indexed and
   a) each instance has the right domain and codomain;
   b) if $\Gamma \sim \Delta$, then by Fact 5.2.6
   $$\begin{aligned}&\mathcal{FE}([\Gamma\ \mathsf{ctx}], [\Gamma \vdash \mathsf{FunExt}])\\ \cong\ &\mathcal{FE}([\Delta\ \mathsf{ctx}], [\Gamma \vdash \mathsf{FunExt}])\ .\end{aligned}$$

7. Focusing on the auxiliary functor S,
   a) by definition,
   $$\begin{aligned}&S([\Gamma\ \mathsf{ctx}], [\Gamma \vdash A], [\Gamma, x : A \vdash \mathcal{U}_i])\\ &\quad(\mathcal{FE}([\Gamma\ \mathsf{ctx}], [\Gamma \vdash \mathsf{FunExt}])(\cdot), ([\Gamma \vdash a], [\Gamma\ \mathsf{ctx}]))\\ =\ &S([\Gamma\ \mathsf{ctx}], [\Gamma \vdash A], [\Gamma, x : A \vdash \mathcal{U}_i])\\ &\quad(([\Gamma \vdash \mathsf{funext}], [\Gamma\ \mathsf{ctx}]), ([\Gamma \vdash a], [\Gamma\ \mathsf{ctx}]))\\ \cong\ &([\Gamma \vdash \mathsf{funext}[a/x]], [\Gamma\ \mathsf{ctx}])\\ \cong\ &\mathcal{FE}([\Gamma\ \mathsf{ctx}], [\Gamma \vdash \mathsf{FunExt}[a/x]])(\cdot)\\ \cong\ &\mathcal{FE}([\Gamma\ \mathsf{ctx}], S([\Gamma\ \mathsf{ctx}], [\Gamma \vdash A], [\Gamma, x : A \vdash \mathcal{U}_i])\\ &\qquad([\Gamma \vdash \mathsf{FunExt}], [\Gamma \vdash a]))\ .\end{aligned}$$

Hence, $\mathfrak{M}^H_T$ together with $[\![\cdot]\!]_T$ is a model. □

Theorem 6.5.8 (Classifying model)**.** *If $\mathfrak{M}^H$ is a model for T, then there is a functor $\mathfrak{J} \colon \mathfrak{M}^H_T \to \mathfrak{M}^H$ which preserves the interpretation.*

Proof. Analogous to the proof of Theorems 5.3.18 and 5.3.22. □

Theorem 6.5.9 (Completeness)**.** *Let γ be a judgement which is valid in every model for a 1-HoTT theory T. Then γ is derivable from T.*

Proof. Since γ is valid in every model for T, it is valid in $\mathfrak{M}^H_T$. Hence it is an arrow of $\mathbb{M}_T$, thus, by Definition 5.3.3, $\gamma \in T$. Since T is closed under derivation in a 1-HoTT system, γ is derivable from T in the 1-HoTT system. □

The conjecture at the end of Section 6.4, that every HoTT-theory makes Theorem 6.4.1, strong normalisation, true and its proof can be obtained along the lines of Theorem 4.3.12, extends to the semantics. We conjecture that every HoTT-theory can be interpreted in a suitable extension of ML-categories, similar to Definition 6.5.1, and this semantics is sound and complete, with a classifying model, the proof of these results being obtained extending Theorems 6.5.5, 6.5.8 and 6.5.9. As before, what prevents to verify this bold conjecture is the lack of an exhaustive syntactical characterisation of higher inductive types in their full generality. In fact, for each example of higher inductive types in [95], the double conjecture on proof theory and semantics holds.

As an example, consider the torus T^2. As described in [95, Section 6.6], its introduction constructors state that

- $b : T^2$;
- $p : b =_{T^2} b$;

- $q : b =_{T^2} b$;
- $t : p \cdot q = q \cdot p$, where $\cdot$ is the *concatenation* of [95, Lemma 2.1.2].

As emphasised in the Book, the induction principle, from which can be derived the elimination and computation rules via the inversion principle [76], is tricky; its exact formulation is too complex to be presented in this dissertation. Anyway, this leads to nine rules of the form $k-\mathsf{intro}$ ($T^2-\mathsf{form}$, $T^2-\mathsf{intro}$, $T^2-\mathsf{elim}$, $T^2-\mathsf{intro_p}-=$, $T^2-\mathsf{intro_q}-=$, $T^2-\mathsf{intro_t}-=$, $T^2-\mathsf{comp_p}-=$, $T^2-\mathsf{comp_q}-=$, $T^2-\mathsf{comp_t}-=$) and an equality rule, $T^2-\mathsf{comp}$.

Thus, as discussed for truncation at the end of Section 6.4, the torus validates all the results in Chapters 3 and 4. Also the semantics can be extended, as done for truncation in the discussion after Definition 6.5.1; in the case of T^2, nine new functors need to be defined, in order to interpret the formation, introduction and elimination rules, the three higher introduction rules and the three higher computation rules. The purpose of this example is twofold: the torus, and similar higher inductive types, satisfy our conjecture, even if is not a 1-HoTT type; moving outside 1-HoTT theories, the syntax, and thus the proofs of Theorems 6.4.1, 6.5.5, 6.5.8, and 6.5.9, become heavier and heavier, despite the core of the results and the methods to gain them are unchanged. This observation suggests that an agile and exhaustive syntactical characterisation of higher inductive types would lead to prove our bold conjectures. Unfortunately, such a characterisation is still lacking in the current literature.

Chapter 7

Conclusions

At the beginning of this dissertation, it has been said that the purpose of the present work is to answer one question: whether higher-order category theory is necessary to describe homotopy type theory. Since homotopy type theory is an open theory, in which parts are informally formal, that is, they are formal but not part of a formal framework, like higher order inductive types in their full generality, we considered a significant subset, 1-HoTT theory, and we proved in Chapters 5 and 6 that 1-category theory is enough to give a sound and complete semantics for it. Technically, the completeness results follow from the existence of a classifying model, which is a non evident fact: for example, the Tarski's semantics for first order logic fails in this respect, as no classifying model can be constructed[1].

The semantics abstracts, in the point-free sense described in the Introduction and discussed in Chapter 5, over the proof theory. As remarked in the Introduction, the proof theory of homotopy type theory has never been systematically developed by its own: most results are inherited from Martin-Löf type theory, see Chapter 2, and not always straightforwardly adapted to the new framework.

Then, in Chapters 3 and 4, a systematic account of the fundamental aspects of the proof theory in the framework of Martin-Löf type theory has been developed, prominently the strong normalisation theorem 4.3.12. This result closes the 40 years' quest for a bullet-proof normalisation of the intensional, multi-universe version of dependent type theory [1, 2, 10, 20, 37]. However, this result is still not perfect because, as e.g. in [2], it uses semantics to establish a couple of fundamental properties, in particular that Π is one-to-one and that no product is convertible to a universe.

The semantics for Martin-Löf type theory is developed in Chapter 5, where the soundness and completeness results are derived along with the existence of a classifying model, as already said. These results are naturally extended to 1-HoTT theories in Chapter 6, thus providing the full picture to build the semantics. Therefore, the dissertation concludes by providing a solid and conclusive answer to the initial question: **no**, 1-category theory suffices to fully describe a very large fragment of homotopy type theory. Also, the results of Chapter 6 suggest that 1-HoTT theories could be extended to n-HoTT theories with no major effort apart the complexity of the involved syntax, see the torus example at the end of Chapter 6.

Hence, we conjectured that the methods and the major results in this work (normalisation, soundness and completeness of the semantics, and the existence of a classifying model) can be extended to the whole HoTT. What

[1]This fact ultimately follows by the existence of the G sentence from Gödel incompleteness theorem [44]. In fact, a classifying model must make true either G or $\neg G$. Then, every model validates the same sentence, which contradicts the existence of non-standard models.

prevents to attempt this bold conjecture is the lack of an exhaustive syntactical characterisation of higher inductive types: this is still an open problem as remarked, e.g., in Section 6.13 of [95].

Bibliography

[1] Andreas Abel, Klaus Aehlig, and Peter Dybjer, *Normalization by evaluation for Martin-Löf type theory with one universe*, Electronic Notes in Theoretical Computer Science **173** (2007), 17–39, Proceedings of the 23rd Conference on the Mathematical Foundations of Programming Semantics.

[2] Andreas Abel, Thierry Coquand, and Peter Dybjer, *Normalization by evaluation for Martin-Löf type theory with typed equality judgements*, Proceedings of the 22nd Annual IEEE Symposium on Logic in Computer Science, IEEE Computer Society, 2007, pp. 3–12.

[3] Peter Aczel and Michael Rathjen, *Notes on constructive set theory*, Tech. Report 40, Institut Mittag–Leffler, 2000.

[4] ———, *Constructive set theory*, Book draft[1]: `https://www1.maths.leeds.ac.uk/~rathjen/book.pdf`, 2010.

[5] Agda, *Webpage*, url: `https://wiki.portal.chalmers.se/agda/pmwiki.php`, consulted on May 21st, 2019.

[6] Benedikt Ahrens, Paolo Capriotti, and Régis Spadotti, *Non-wellfounded trees in homotopy type theory*, 13th International Conference on Typed Lambda Calculi and Applications (Dagstuhl, Germany) (Thorsten Altenkirch, ed.), Leibniz International Proceedings in Informatics (LIPIcs), vol. 38, Schloß Dagstuhl–Leibniz-Zentrum für Informatik, 2015, pp. 17–30.

[7] Benedikt Ahrens, Krzysztof Kapulkin, and Michael Shulman, *Univalent categories and the Rezk completion*, Mathematical Structures in Computer Science **25** (2015), no. 5, 1010–1039.

[8] Benedikt Ahrens, Peter LeFanu Lumsdaine, and Vladimir Voevodsky, *Categorical structures for type theory in univalent foundations*, Logical Methods in Computer Science **14** (2018), no. 3, 1–18.

[9] Thorsten Altenkirch, Paolo Capriotti, and Nicolai Kraus, *Extending homotopy type theory with strict equality*, 25th EACSL Annual Conference on Computer Science Logic (Dagstuhl, Germany) (Jean-Marc Talbot and Laurent Regnier, eds.), Leibniz International Proceedings in Informatics (LIPIcs), vol. 62, Schloß Dagstuhl–Leibniz-Zentrum für Informatik, 2016, pp. 21:1–21:17.

[10] Thorsten Altenkirch, Martin Hofmann, and Thomas Streicher, *Reduction-free normalisation for a polymorphic system*, Proceedings 11th Annual IEEE Symposium on Logic in Computer Science, 1996, pp. 98–106.

[1]Courtesy of professor Rathjen.

[11] Jeremy Avigad, Krzysztof Kapulkin, and Peter LeFanu Lumsdaine, *Homotopy limits in type theory*, Mathematical Structures in Computer Science **25** (2015), no. 5, 1040—1070.

[12] Steve Awodey, *Natural models of homotopy type theory*, Mathematical Structures in Computer Science **28** (2018), no. 2, 241—286.

[13] Steve Awodey, Nicola Gambino, and Kristina Sojakova, *Inductive types in homotopy type theory*, 27^{th} Annual IEEE Symposium on Logic in Computer Science, 2012, pp. 95–104.

[14] ______, *Homotopy-initial algebras in type theory*, Journal of the ACM **63** (2017), no. 6, 51:1–51:45.

[15] Hendrik P. Barendregt, *The lambda calculus: Its syntax and semantics*, Studies in Logic and the Foundations of Mathematics, vol. 103, Elsevier, 1984.

[16] Andrej Bauer, Jason Gross, Peter LeFanu Lumsdaine, Michael Shulman, Matthieu Sozeau, and Bas Spitters, *The `HoTT` library: a formalization of homotopy type theory in `Coq`*, Proceedings of the 6^{th} ACM SIGPLAN Conference on Certified Programs and Proofs, Paris, France, January 16-17, 2017 (Yves Bertot and Viktor Vafeiadis, eds.), ACM, 2017, pp. 164–172.

[17] Michael J. Beeson, *Foundations of constructive mathematics*, Springer, 1985.

[18] Marco Benini, *Proof-oriented categorical semantics*, Concepts of Proof in Mathematics, Philosophy, and Computer Science (Dieter Probst and Peter Schuster, eds.), Ontos Mathematical Logic, vol. 6, De Gruyter, 2016, pp. 41–68.

[19] Marco Benini and Roberta Bonacina, *Well-quasi orders in a categorical setting*, Archive for Mathematical Logic **58** (2019), no. 3, 501–526.

[20] Ulrich Berger, Matthias Eberl, and Helmut Schwichtenberg, *Normalization by evaluation*, Prospects for Hardware Foundations: ESPRIT Working Group 8533 NADA — New Hardware Design Methods Survey Chapters (Bernhard Möller and John V. Tucker, eds.), Springer, 1998, pp. 117–137.

[21] Errett Bishop, *Foundations of constructive analysis*, Mcgraw-Hill, 1967.

[22] Errett Bishop and Douglas S. Bridges, *Constructive analysis*, Springer, 1985.

[23] Corrado Böhm and Alessandro Berarducci, *Automatic synthesis of typed lambda-programs on term algebras*, Theoretical Computer Science **39** (1985), 135–154.

[24] Roberta Bonacina, *Point-free categorical semantics for Martin-Löf type theory*, Master's thesis, Università degli Studi dell'Insubria, 2016.

[25] George Boole, *An investigation of the laws of thought: On which are founded the mathematical theories of logic and probabilities*, Cambridge Library Collection — Mathematics, Cambridge University Press, 2009.

[26] Francis Borceux, *Handbook of categorical algebra*, Encyclopedia of Mathematics and its Applications, vol. 1, Cambridge University Press, 1994.

[27] Nicolas Bourbaki, *Theory of sets*, Actualités scientifiques et industrielles, Springer, 2004.

[28] Carl B. Boyer, *A history of mathematics*, Wiley, 1991.

[29] Douglas S. Bridges and Fred Richman, *Varieties of constructive mathematics*, London Mathematical Society Lecture Note Series, vol. 97, Cambridge University Press, 1987.

[30] Luitzen E. J. Brouwer and Arend Heyting, *Collected works/L.E.J. Brouwer; edited by A. Heyting*, North-Holland, 1975.

[31] Felice Cardone and J. Roger Hindley, *History of lambda-calculus and combinatory logic*, Handbook of the History of Logic, vol. 5, Elsevier, 2006, pp. 723–817.

[32] Pierre Clairambault and Peter Dybjer, *The biequivalence of locally cartesian closed categories and Martin-Löf type theories*, Typed Lambda Calculi and Applications (Luke Ong, ed.), Springer, 2011, pp. 91–106.

[33] Cyril Cohen, Thierry Coquand, Simon Huber, and Anders Mörtberg, *Cubical type theory: A constructive interpretation of the univalence axiom*, 21^{st} International Conference on Types for Proofs and Programs (Tarmo Uustalu, ed.), Leibniz International Proceedings in Informatics (LIPIcs), vol. 69, Schloß Dagstuhl–Leibniz-Zentrum für Informatik, 2018, pp. 5:1–5:34.

[34] Loïc Colson, *About primitive recursive algorithms*, Automata, Languages and Programming, 16th International Colloquium, ICALP89, Stresa, Italy, July 11-15, 1989, Proceedings, 1989, pp. 194–206.

[35] Barry S. Cooper, *Computability theory*, Chapman and Hall/CRC, 2003.

[36] Coq, *Webpage*, url: `https://coq.inria.fr/`, consulted on May 21^{st}, 2019.

[37] Thierry Coquand and Arnaud Spiwack, *A proof of strong normalisation using domain theory*, 21^{st} Annual IEEE Symposium on Logic in Computer Science, 2006, pp. 307–316.

[38] Laura Crosilla, *The entanglement of logic and set theory, constructively*, Preprint[2] at `https://www.researchgate.net/publication/326741496_The_entanglement_of_logic_and_set_theory_constructively` Consulted on May 21^{st}, 2019, 2018.

[39] Pierre-Louis Curien, *Substitution up to isomorphism*, Fundamenta Informaticæ **19** (1993), no. 1–2, 51–85.

[40] Haskell B. Curry and Robert Feys, *Combinatory logic*, vol. 1, North-Holland, 1958.

[2]Courtesy of the author.

[41] Nicolaas G. de Bruijn, *Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the Church Rosser theorem*, Indagationes Mathematicae **75** (1972), no. 5, 381–392.

[42] Jean-Yves Girard, *Interprétation fonctionnelle et élimination des coupures de l'arithmétique d'ordre supérieur*, Phd thesis, Université Paris Diderot - Paris 7, 1972.

[43] Jean-Yves Girard, Yves Lafont, and Paul Taylor, *Proofs and types*, Cambridge Tracts in Theoretical Computer Science, vol. 7, Cambridge University Press, 1989.

[44] Kurt Gödel, *Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme, I*, Monatshefte für Mathematik und Physik **38** (1931), no. 1, 173–198.

[45] Robert Goldblatt, *Topoi: The categorical analysis of logic*, Dover Publications, 2006.

[46] George Grätzer, *General lattice theory*, 2$^{\text{nd}}$ ed., Birkhäuser, 2002.

[47] Alexandre Grothendieck and Jean-Louis Verdier, *Expose IV: Topos*, Theories des Topos et Cohomologie Etale des Schemas (SGA 4.1), Seminaire de Geometrie Algebrique Du Bois-Marie, Available at `http://www.grothendieckcircle.org/`, 1963–64, pp. 299–519.

[48] Allen Hatcher, *Algebraic topology*, Cambridge University Press, 2001.

[49] Martin Hofmann and Thomas Streicher, *The groupoid interpretation of type theory*, Twenty-Five Years of Constructive Type Theory (Giovanni Sambin and Jan Smith, eds.), Oxford Logic Guides, vol. 36, Oxford University Press, 1998, pp. 83–111.

[50] Kuen-Bang Hou Favonia, Eris Finster, Daniel R. Licata, and Peter LeFanu Lumsdaine, *A mechanization of the Blakers–Massey connectivity theorem in homotopy type theory*, 31$^{\text{st}}$ Annual ACM/IEEE Symposium on Logic in Computer Science, July 2016, pp. 1–10.

[51] William A. Howard, *The formulæ-as-types notion of construction*, To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism (J. Roger. Hindley and Jonathan P. Seldin, eds.), Academic Press, 1980, pp. 479–490.

[52] Bart Jacobs, *Categorical logic and type theory*, Studies in Logic and the Foundations of Mathematics, vol. 141, Elsevier, 1999.

[53] Peter T. Johnstone, *Stone spaces*, Cambridge Studies in Advanced Mathematics, Cambridge University Press, 1982.

[54] ———, *Sketches of an elephant: A topos theory compendium*, vol. 2, Oxford University Press, 2002.

[55] ———, *Sketches of an elephant: A topos theory compendium*, vol. 1, Oxford University Press, 2002.

[56] Morris Kline, *Mathematical thought from ancient to modern times:*, Mathematical Thought from Ancient to Modern Times, OUP USA, 1990.

[57] Dénes König, *Über eine Schlussweise aus dem Endlichen ins Unendliche*, Acta Litterarum ac Scientiarum Regiæ Universitatis Hungaricæ Francisco-Josephinæ Szeged **3** (1927), 121–130.

[58] James Ladyman and Stuart Presnell, *Universes and univalence in homotopy type theory*, Review of Symbolic Logic **12** (2019), no. 3, 426–455.

[59] Daniel R. Licata and Eric Finster, *Eilenberg-MacLane spaces in homotopy type theory*, Proceedings of the Joint Meeting of the 23rd EACSL Annual Conference on Computer Science Logic and the 29th Annual ACM/IEEE Symposium on Logic in Computer Science, ACM, 2014, pp. 66:1–66:9.

[60] Daniel R. Licata and Michael Shulman, *Calculating the fundamental group of the circle in homotopy type theory*, 28th Annual ACM/IEEE Symposium on Logic in Computer Science, 2013, pp. 223–232.

[61] Henri Lombardi and Claude Quitté, *Commutative algebra: Constructive methods — finite projective modules*, Springer, 2015.

[62] Jacob Lurie, *Higher topos theory (am-170)*, Princeton University Press, 2009.

[63] Sanders Mac Lane, *Categories for the working mathematician*, 2nd ed., Springer, 1998.

[64] Saunders Mac Lane and Ieke Moerdijk, *Sheaves in geometry and logic: A first introduction to topos theory*, Springer, 1992.

[65] Michael Makkai, *An algebraic look at propositional logic*, vol. 18, Associazione Italiana di Logica e sue Applicazioni, 1994.

[66] Andrey A. Markov, *Theory of algorithms*, TT 60-51085, Academy of Sciences of the USSR, 1954.

[67] Jean-Pierre Marquis, *From a geometrical point of view: A study of the history and philosophy of category theory*, Logic, Epistemology, and the Unity of Science, Springer, 2008.

[68] Per Martin-Löf, *An intuitionistic theory of types*, Tech. report, University of Stockholm, 1972.

[69] ———, *An intuitionistic theory of types: Predicative part*, Logic Colloquium '73 (H.E. Rose and J.C. Shepherdson, eds.), Studies in Logic and the Foundations of Mathematics, vol. 80, Elsevier, 1975, pp. 73 – 118.

[70] ———, *Constructive mathematics and computer programming*, Philosophical Transactions of the Royal Society of London. Series A. Mathematical and Physical Sciences **312** (1984), no. 1522, 501–518.

[71] ———, *Intuitionistic type theory. Notes by Giovanni Sambin of a series of lectures given in Padua, June 1980*, Bibliopolis, Napoli, 1984.

[72] ———, *An intuitionistic theory of types*, Twenty-Five Years of Constructive Type Theory (Giovanni Sambin and Jan Smith, eds.), Oxford Logic Guides, vol. 36, Oxford University Press, 1998, pp. 127–172.

[73] ———, *The Hilbert-Brouwer controversy resolved?*, One Hundred Years of Intuitionism (1907–2007): The Cerisy Conference (Mark van Atten, Pascal Boldini, Michel Bourdeau, and Gerhard Heinzmann, eds.), Birkhäuser, 2008, pp. 243–256.

[74] William S. Massey, *Algebraic topology: An introduction*, vol. 52, Springer, 1967.

[75] Paul F. Mendler, *Inductive definition in type theory*, Ph.D. thesis, Cornell University, USA, 1987.

[76] Sara Negri and Jan von Plato, *Structural proof theory*, Cambridge University Press, 2001.

[77] Bengt Nordström, Kent Petersson, and Jan M. Smith, *Programming in Martin-Löf's type theory*, Oxford University Press, 1990.

[78] Michel Parigot, *On the representation of data in lambda-calculus*, Proceedings of the 3rd Workshop on Computer Science Logic (London, UK, UK), CSL '89, Springer-Verlag, 1990, pp. 309–321.

[79] Benjamin C. Pierce, *Types and programming languages*, MIT Press, 2002.

[80] Henri M. Poincaré, *Analysis situs*, Journal de l'École Polytechnique **2** (1895), no. 1, 1–123.

[81] ———, *Cinquième complément à l'analysis situs*, Rendiconti del Circolo Matematico di Palermo (1884-1940) **18** (1904), no. 1, 45–110.

[82] Henri M. Poincaré, *La logique de l'infini*, Revue de Métaphysique et de Morale **17** (1909), 461–482.

[83] ———, *La logique de l'infini*, Scientia **12** (1912), 1–11.

[84] Joseph J. Rotman, *An introduction to algebraic topology*, vol. 119, Springer, 1988.

[85] Giovanni Sambin, *Formal topology*, url: `https://www.math.unipd.it/~sambin/maths-formaltopology.html`, consulted on June 4th, 2019.

[86] ———, *Intuitionistic formal spaces — a first communication*, Mathematical Logic and Its Applications (Dimiter G. Skordev, ed.), Springer US, Boston, MA, 1987, pp. 187–204.

[87] ———, *A course in formal topology*, 1998, European Summer School of Logic, Language, and Information[3].

[88] ———, *Some points in formal topology*, Theoretical Computer Science **305** (2003), no. 1, 347–408.

[3] Courtesy of M. Benini who attended the course.

[89] Peter Schuster, *Formal Zariski topology: Positivity and points*, Annals of Pure and Applied Logic **137** (2006), no. 1, 317–359.

[90] Robert A. G. Seely, *Locally Cartesian closed categories and type theory*, Mathematical Proceedings of the Cambridge Philosophical Society **95** (1984), 33–48.

[91] Christopher Strachey and Christopher P. Wadsworth, *Continuations: A mathematical semantics for handling full jumps*, Higher-Order and Symbolic Computation **13** (2000), no. 1, 135–152.

[92] Anne S. Troelstra, *Aspects of constructive mathematics*, Handbook of mathematical logic (Jon Barwise, ed.), Studies in Logic and the Foundations of Mathematics, vol. 90, Elsevier, 1977, pp. 973 – 1052.

[93] Anne S. Troelstra and Helmut Schwichtenberg, *Basic proof theory*, 2$^{\text{nd}}$ ed., Cambridge Tracts in Theoretical Computer Science, vol. 43, Cambrdige University Press, 2000.

[94] Anne S. Troelstra and Dirk van Dalen, *Constructivism in mathematics: An introduction. volume I*, Studies in Logic and the Foundations of Mathematics, vol. 121, Elsevier, 1998.

[95] The Univalent Foundations Program, *Homotopy type theory: Univalent foundations of mathematics*, Institute for Advanced Study: `https://homotopytypetheory.org/book`, 2013.

[96] Vladimir Voevodsky, *A very short note on homotopy λ-calculus*, `http://www.math.ias.edu/vladimir/files/2006_09_Hlambda.pdf`, 2006.

[97] ———, *The equivalence axiom and univalent models of type theory*, Tech. Report arXiv:1402.5556, arXiv, 2010.

[98] Michael A. Warren, *Homotopy theoretic aspects of constructive type theory*, Ph.D. thesis, Carnegie Mellon University, 2008.

[99] Benjamin Werner, *Une théorie des constructions inductives*, Ph.D. thesis, Université Paris VII, France, 1994.