A Proof and Formalization of the Initiality Conjecture of Dependent Type Theory

Menno de Boer

A Proof and Formalization of the Initiality Conjecture of Dependent Type Theory

Menno de Boer

Printed by E-Print AB 2020
Distributor: Department of Mathematics, Stockholm University

Abstract

In this licentiate thesis we present a proof of the initiality conjecture for Martin-Löf's type theory with $\mathbf{0}$, $\mathbf{1}$, $\mathbf{N}$, $A + B$, $\Pi_A B$, $\Sigma_A B$, $\mathbf{Id}_A(u, v)$, a countable hierarchy of universes $(\mathbf{U}_i)_{i \in \mathbb{N}}$ closed under these type constructors and with type of elements $(\mathbf{El}_i(a))_{i \in \mathbb{N}}$. We employ the categorical semantics of contextual categories. The proof is based on a formalization in the proof assistant Agda done by Guillaume Brunerie and the author. This work was part of a joint project with Peter LeFanu Lumsdaine and Anders Mörtberg, who are developing a separate formalization of this conjecture with respect to categories with attributes and using the proof assistant Coq over the UniMath library instead. Results from this project are planned to be published in the future.

We start by carefully setting up the syntax and rules for the dependent type theory in question followed by an introduction to contextual categories. We then define the partial interpretation of raw syntax into a contextual category and we prove that this interpretation is total on well-formed input. By doing so, we define a functor from the term model, which is built out of the syntax, into any contextual category and we show that any two such functors are equal. This establishes that the term model is initial among contextual categories. At the end we discuss details of the formalization and future directions for research. In particular, we discuss a memory issue that arose in type checking the formalization and how it was resolved.

Sammanfattning

In denna licentiatavhandling presenterar vi ett bevis av initialitetsförmodan för Martin-Löfs typteori med $\mathbf{0}$, $\mathbf{1}$, $\mathbf{N}$, $A + B$, $\Pi_A B$, $\Sigma_A B$, $\mathbf{Id}_A(u, v)$, en uppräknelig hierarki av universum $(\mathbf{U}_i)_{i \in \mathbb{N}}$ slutna under dessa typkonstruktorer och med typ av element $(\mathbf{El}_i(a))_{i \in \mathbb{N}}$. Vi använder den kategoriska semantiken för kontextuella kategorier. Beviset är baserat på en formalisering i bevisassistenten Agda utförd av Guillaume Brunerie och författaren. Detta var en del av ett gemensamt projekt med Peter LeFanu Lumsdaine och Anders Mörtberg, vilka arbetar med en separat formalisering av denna förmodan med avseende på kategorier med attribut och använder bevisassistenten Coq över UniMath biblioteket istället. Resultaten från detta projekt planeras publiceras i framtiden.

Vi börjar med att noggrant beskriva syntax och regler för den beroende typteorin i fråga följt av en introduktion om kontextuella kategorier. Sedan definierar vi en partiell tolkning av rå syntax i en kontextuella kategori och vi bevisar att denna tolkning är total på välformad indata. Genom att göra det definierar vi en funktor från termmodellen som konstruerats från syntaxen i en kontextuell kategori och vi visar att två sådana funktorer är lika. Detta fastställer att termmodellen är initial bland kontextuella kategorier. I slutet kommer vi att diskutera vår formalisering och eventuella framtida forskningsinriktningar. I synnerhet diskuterar vi ett minnesproblem som uppstod under typcheckning av formaliseringen och hur det kan lösas.

Acknowledgements

I would like to take this opportunity to thank a number of people that have helped me during my work on this licentiate thesis and my PhD studies in general. First and foremost to my main supervisor Dr. Peter LeFanu Lumsdaine for his excellent support, great insights and our many fruitful discussions and secondly to Dr. Guillaume Brunerie for our collaboration during this particular project and his many insights in working with Agda. Next, to Prof. Erik Palmgren[†] for his time being my second supervisor, who sadly passed away after the first year of my PhD studies. A special thanks to Dr. Alexander Berglund for stepping in as the replacement second supervisor, but also for his support as director of PhD studies at the department of mathematics at Stockholm University.

I would also like to thank the department of mathematics at Stockholm University in general for the nice atmosphere to work in and especially the Logic Group for the many excellent seminar talks and interesting discussions regarding various topics. Also thanks to the Dutch Delegation for making after work hours just as enjoyable.

Finally, I thank my girlfriend Carolien for her support and patience during the time living in different countries.

Contents

1. Introduction

Dependent type theories have been introduced to model mathematics starting with the AUTOMATH project by de Bruijn [dB73]. Objects like $\mathbb{R}^n$ can be understood as *depending* on $n \in \mathbb{N}$. Crucially, even mathematical propositions themselves, such as $\forall x.P(x)$ and $\exists x.P(x)$, can be identified with dependent types. This observation is called the *Curry-Howard isomorphism* or *proposition as types*.

Per Martin-Löf expanded on these ideas by proposing a *constructive foundational system* based on types in [ML75] and later [ML84]. Later, Vladimir Voevodsky built upon these [Voe06] and introduced the *univalence axiom* resulting in what is now know as *homotopy type theory*. The main source on this development is the HoTT book [UFP13].

One of the main features of type theoretic foundations is that they are suitable for computer implementation. Indeed, homotopy type theory has been formalized in projects like the HoTT library [BGL+16] and UniMat library [VAG+].

One difficulty in working with a dependent type theory as a formal system is the handling of its syntax. Among other things one needs to properly deal with variables, substitution and possibly multiple derivations of a given judgment. For this reason it can be preferable to work in a *semantic model* instead, such as, but not limited to: contextual categories, categories with families, categories with attributes or comprehension categories. In these settings the syntactic subtleties disappear. However, the structure they need in order to interpret more complicated syntactic constructions can become unreadable. These issues have been discussed in [KL20, Section 1.2].

Ideally, one could move back and forth between the syntactic and semantic representation of type theory and work in the one that is more appropriate for the given situation. This is similar to the *soundness* and *completeness* theorems for first order predicate logic. In the setting of categorical semantics, the counterpart to this process is called *initiality*.

In this licentiate thesis we will present a proof of the initiality conjecture for a dependent type theory with respect to the categorical semantics of contextual categories. It is based on a formalization done by Guillaume Brunerie and the author which is available at https://github.com/guillaumebrunerie/ initiality.

The specific version on which this thesis is based is commit 17c2477 (March 27, 2020) and consists roughly of 11000 lines of code. The formalization was part of a project together with Peter LeFanu Lumsdaine and Anders Mörtberg. This thesis focuses on the author's contributions. As ever, it is impossible to completely disentangle one collaborator's contributions from others', but for the material covered in this thesis, the author was either a primary or equal contributor, except at a few points where explicitly noted otherwise (included for context and completeness). In the Agda formalization, the contributions of the author and Brunerie can be viewed in the repository's history.

We have been uniform in the treatment of the type constructors we consider, making extensions of the results to larger systems, by adding additional constructors and axioms, transparent. Moreover, the formalization ensures all details have been properly checked. However, the author hopes that in the future a proof of initiality is presented for a general dependent type theory for which the presented proof can help to better understand the difficulties that may arise.

At the moment of writing the required memory to type check the entire formalization quickly exceeds that of most personal computers. This is currently being fixed by forcing Agda to erase unnecessary data, which can be justified metatheoretically. The hope is that in the near future this bump can be solved. We will discuss this particular issue at the point in the proof and when exploring the formalization itself in Chapter 5.

1.1 Historic overview of initiality

The main reference on the initiality conjecture is the book by Streicher [Str91], in which initiality was shown for the calculus of constructions. A common consensus among the community has been that these methods can be extrapolated to larger theories without complications, although the process would be long and tedious. Therefore, the conjecture is referred to as a 'folklore' result.

A strong advocate for solving this controversy was Vladimir Voevodsky†. He argued that even though the interpretation of various rules had been studied, the interpretation of dependent type theory itself has yet remained open and [Str91] was the only "substantial non-trivial analog of this conjecture known".[1] Since Voevodsky's original post, there have been discussions at various mathematical fora about the subject.[2]

[1] https://homotopytypetheory.org/2015/01/11/hott-is-not-an-interpretation-of-mltt-into-abstract-homotopy-theory.

[2] for instance: https://groups.google.com/forum/#!searchin/homotopytypetheory/initiality%7Csort:date/homotopytypetheory/1hic3vFc6n0/sNX47YIoAQAJ, https://nforum.ncatlab.org/discussion/8854/beijing-talk.

One response was the *initiality project* on nLab[1] set up by Michael Shulman around September 2019.[2] Its aim was to crowdsource the long and tedious details to a larger group of mathematicians in a similar vein to the write-up of the HoTT book [UFP13] and the Polymath project. At the moment of writing this project has come to a standstill and it is currently undecided whether it will start up again in the future.

A different response by Peter LeFanu Lumsdaine was to start a project around October that same year to formalize the conjecture in a proof assistant instead. He was joined in this project by Guillaume Brunerie, Anders Mörtberg and the author. The project was subsequently split into two teams: Brunerie and the author would work on a formalization in Agda using the categorical semantic of contextual categories, while Lumsdaine and Mörtberg would work on a formalization in Coq over the UniMath library and using categories with attributes instead.

Precise statement

One of the challenges of initiality is stating the problem at hand sufficiently precisely. Informally, one can state it as:

> The syntax of any dependent type theory forms a category, called the *syntactic category* or *term model*, whose structure depends on the rules of the theory. This category is *initial* among all categories possessing this structure, i.e. there exists a unique structure preserving functor from it to any other such category.

As mentioned before, the initiality conjecture can be read as the categorical analogue of *soundness* and *completeness* for first order logic. The above statement raises at least two questions:

- What do we consider to be 'a dependent type theory'?
- What do we mean by 'all categories sharing this structure'?

Regarding the first question, it is still open what we consider by a general dependent type theory. Work in this direction has been made by Taichi Uemura [Uem19] and independently by Andrej Bauer, Philipp Haselwarter and Peter LeFanu Lumsdaine, although the latter has not yet been published.[3] As such,

[1] https://ncatlab.org/nlab/show/Initiality+Project.

[2] announcement: https://golem.ph.utexas.edu/category/2018/09/a_communal_proof_of_an_initial.html.

[3] Slides for a talk by Peter LeFanu Lumsdaine given at EUTypes 2018 can be found at https://cs.au.dk/fileadmin/user_upload/PeterLumsdaine_general-dependent-type-theories.pdf.

we can only tackle the initiality conjecture for a specific type theory, but a future goal is still to have a proof of initiality for general type theories. Nevertheless, having access to a proof for particular cases can help give insight in the development of the general case.

For the second question, several categorical semantics have been proposed that capture the structure of the term model, e.g. contextual categories, categories with families and categories with attributes to name a few. Some of these notions have been shown to be equivalent [ALV18]. Ideally, initiality is independent of any such choice. In this thesis we will employ the categorical semantics given by contextual categories. This particular semantics was also used in [Str91]. It can be represented as an essentially algebraic theory which made it very suitable to implement in a proof assistant like Agda.

Treatments in the literature

In this section we want to discuss more in depth and give credit to various cases in the literature that have aimed to tackle the initiality conjecture by stating the type theory and the categorical semantics used.

As stated previously [Str91] has been the main source. The type theory under consideration was the *calculus of constructions*. This type theory is significantly smaller than the kinds used today, such as HoTT or UniMath.

In [Hof97, Remark 2.5.8], initiality is stated for a type theory with Π-types, a natural number type, and identity types. It is also sketched how to extend this by a unit and universe type. It is one of the few write-ups that properly treats the problem, although it leaves a significant part to the reader. One can therefore only accept the results after they have checked the omitted details themselves.

The previously mentioned initiality project on nLab aimed for a dependent type theory including only Π-types, although with extensions in mind. It used the categorical semantics of categories with families. One of its goals was to write out all of the details once and for all.

In [Yam17], results from [Hof97] are used and expanded upon. Additionally, a system called the *equational theory* $\lambda^{=}_{1,X}$ is considered which has unique derivations, allowing for the interpretation function to be defined differently.

In [Cas14], initiality is presented for a type theory with Π-types and one universe. A technique is presented for producing unique derivations by compressing a given one, which is very specific to the particular presentation of this system. However, it is unclear whether this technique can be extrapolated to more complicated systems.

There are other sources in the literature that state initiality either with or without proof. Regardless, there is still dissatisfaction in the community about the status of the conjecture. Because of this, it seems appropriate to also

mention to what extend this thesis claims to be sufficient.

The author aims in this write-up to be detailed enough leaving no room for subtleties to be hiding in the gaps. In particular, we have been careful to state all definitions we use and any proofs that are omitted in this write-up can be checked in the formalization. The formalization is self-contained and publicly available. It can be verified to contain precisely the content it claims by whoever wishes to do so. Finally, we have given a uniform treatment of the various type constructors, making it clear how to extend to a larger system in written form or by contributing to our formalization.

1.2 Metatheory

As this licentiate thesis aims to prove a statement about a foundational system, we will state the particular metatheory we will work in. However, the goal has been to write statements and proofs in a way so they can be read both in a (constructive) type theoretic and a classical set-theoretic foundation. This is in a similar vein as [AL19, Section 2.1].

The minimal foundation system in which our arguments will work is a variant of Martin-Löf's intensional type theory including: Σ-types, with η; Π-types, with η and function extensionality; inductive definitions such as 0, 1, 2, $\mathbb{N}$ and W-types; quotients; two universes closed under these notions; propositional truncation and propositional extensionality. For Martin-Löf's original presentation, see [ML75] and [ML84]. For an example in which propositional truncation and quotients are treated we refer to [UFP13].

That being said, the arguments will work in any foundational system that includes these principles such as classical ZFC, intuitionistic IFZ or an appropriate extension of the calculus of inductive constructions. This latter is closely related to the metatheory of our formalization, which was done in the proof assistant Agda. We will expand on this in the following sections and in Chapter 5.

We will not impose any additional restrictions on the equality of types such as univalence or UIP. As such, the body of this thesis should be compatible with univalence and the interpretation of types as classical sets. For readability we will use 'set' instead of 'type' on a metalevel and write in a conventional mathematical language being confident that a reader with a type theoretic background can make the translation without much effort.

Some type theoretic issues do not exist in a more classical interpretation and a reader coming from such a background is free to ignore them.

Inductive definitions

In this section we briefly recall the common notation for inductive definitions. This can either be read as inductively defined types, as the smallest set generated/closed under the given inference rules or as an algebra freely generated by these generators.

As a basic example, one can define the natural numbers $\mathbb{N}$ inductively by the inference rules

$$\frac{}{0 \in \mathbb{N}} \qquad \frac{n \in \mathbb{N}}{n+1 \in \mathbb{N}}$$

which can be read as stating that 0 is a natural number and if n is a natural number, so is $n+1$. It is also common to write $S(n)$ instead of $n+1$, which highlights that '$n+1$' is just a syntactic expression.

Defining a function from $\mathbb{N}$ to any other set is done by the process of induction/recursion. As an example, we define addition by

$$\begin{aligned} \_ + \_ &: \mathbb{N} \times \mathbb{N} \to \mathbb{N} \\ m + 0 &\coloneqq m \\ m + (n+1) &\coloneqq (m+n) + 1. \end{aligned}$$

We say in this case that addition is defined by *structural induction* on its second argument. Inductively defined sets can depend on additional parameters. An example of this are the finite sets $\mathrm{Fin}(n)$ for $n \in \mathbb{N}$ which can be inductively generated by the rules

$$\frac{}{0_n \in \mathrm{Fin}(n+1)} \qquad \frac{k \in \mathrm{Fin}(n)}{k +_n 1 \in \mathrm{Fin}(n+1)}.$$

Informally we can think of $\mathrm{Fin}(0) = \emptyset$, $\mathrm{Fin}(1) = \{0\}$, $\mathrm{Fin}(2) = \{0, 1\}$, etc.

For $k \in \mathrm{Fin}(n)$ and $m \in \mathbb{N}$ we define the expression $k +_n m \in \mathrm{Fin}(n+m)$ by structural induction on m

$$\begin{aligned} \_ +_n \_ &: \mathrm{Fin}(n) \times \mathbb{N} \to \mathrm{Fin}(n+m) \\ k +_n 0 &\coloneqq k \\ k +_n (m+1) &\coloneqq (k +_n m) +_{m+n} 1. \end{aligned}$$

As an example of the conventional mathematical notation we will employ in this thesis, we use $k < n$ in written text instead of $k \in \mathrm{Fin}(n)$.

Propositions

Mathematical logic requires a notion of logical propositions. In a classical setting this role is filled by the two-element set $\{0, 1\}$. In intuitionistic set theory IZF, or intuitionistic higher-order logic, this is filled more generally by the subobject classifier (in topos-theoretic language), i.e. $\mathcal{P}(1)$. In a type theoretic setting this role can be filled by considering h-propositions, as in univalent foundations [UFP13], or by a separate universe of propositions, as in the calculus of constructions [CH88].

The kind of propositions we use in our formalization is a hierarchy of universes of *strict* propositions sProp as described in [GCST19] to which we refer a reader interested in its metatheoretical properties. The strictness refers to any two elements of a proposition being *judgementally* equal. It is worth noting that strictness should not be required for the results in this thesis. The other main features of sProp are that any type can be squashed to a proposition, any proposition can be lifted to a type and its use is compatible with both UIP and univalence. However, in our formalization we consider equality to be squashed into sProp. The development only relies on the *mere* existence of certain equalities and should not need UIP.

Finally, we also assume *proposition extensionality*: any two logically equivalent propositions are equal.

Quotients

Quotients are an important tool when dealing with the initiality conjecture as they are needed to define the term model. A reader who intends to follow our arguments in a classical background should have no problem with their use.

In a type theoretic setting, this topic is less clear and various different approaches for introducing quotient types have been proposed. An overview and discussion can be found in [Li15, Chapter 3]. We present here the particular kind of quotients we have used in the formalization. The implementation is due to Brunerie and similar to the presentation in [Li15, Section 3.1.1].

Given a set X and a sProp-valued equivalence relation $\sim$ we can form the set $X/\!\sim$. It comes equipped with a function $[-] : X \to X/\!\sim$ and for each $x, y \in X$ such that $x \sim y$ an equality $[x] = [y]$. It satisfies the following dependent elimination rule: for a family of sets $P : X/\!\sim \to \mathsf{Set}$ together with a function $f : (x : X) \to P([x])$ such that if $x \sim y$ we have $f([x]) = f([y])$, we get a function $\overline{f} : X/\!\sim \to P(x)$ satisfying the computation rule $\overline{f}([x]) = f(x)$.

The quotients we are considering can be shown to be *effective*, i.e. if $[x] = [y]$ then $x \sim y$. This requires propositional extensionality, which is essentially a formalization of the proof in [Vel15, Proposition 1].

2. Dependent Type Theory

In this chapter we start by setting up the dependent type theory that will be addressed in this thesis. A brief informal description of a dependent type theory is as a many-sorted language, whose sorts are called *types*. Its deduction rules deal with statements we call *judgments*. Any particular judgment is made from a given *context* which is, roughly speaking, a finite list of types in which we allow an entry to 'depend' on all its predecessors. All of these notions will be treated in this chapter and no additional background knowledge is assumed.

In this thesis we show the initiality conjecture for Martin-Löf's intensional type theory with the following type constructors: $\mathbf{0}$, $\mathbf{1}$, $\mathbf{N}$, $A + B$, $\Pi_{A,B}$, $\Sigma_{A,B}$, $\mathbf{Id}_A(u, v)$ and a countable hierarchy of universes $(\mathbf{U}_i)_{i\in\mathbb{N}}$, closed under the type constructors and with type of elements $\mathbf{El}_i(a)$, for a given $i \in \mathbb{N}$. We will refer to this type theory as MLTT. All of these constructors have already been presented in [ML84].

Because the results in this thesis are about the interplay between the syntax and semantics of dependent type theory, we have chosen to thoroughly include the precise definitions of syntax we use. Additionally, at the moment there is no standard convention and approaches in the literature vary in many details. Although these different approaches usually do not matter, it will matter for us.

A reader that is familiar with the setup of syntax for dependent type theory should be able to skip most of this chapter. However, we do advise to skim through it and take note of certain conventions/notation.

2.1 Raw syntax

Just as one has to define the concept of 'terms', 'formulae' and 'derivations' inductively in traditional first order logic, we too must properly define the syntax of our system.

Types and terms

We will adopt the use of de Bruijn indices instead of named variables. This allows us to define type and term expressions indexed by a natural number, which indicates the length of a context in which they can be formed. This approach has proven to be very suitable for formalization.

Let us start by defining the families of sets that contain type and term expressions over a context of a given length.

Definition 2.1.1. The sets $\mathrm{TyExpr}(n)$ and $\mathrm{TmExpr}(n)$, where $n \in \mathbb{N}$, of *raw type and term expressions*, are inductively generated by the following clauses:

$$\frac{}{\mathbf{0} \in \mathrm{TyExpr}(n)} \qquad \frac{}{\mathbf{1} \in \mathrm{TyExpr}(n)} \qquad \frac{}{\mathbf{N} \in \mathrm{TyExpr}(n)}$$

$$\frac{A, B \in \mathrm{TyExpr}(n)}{A + B \in \mathrm{TyExpr}(n)} \qquad \frac{A \in \mathrm{TyExpr}(n) \qquad B \in \mathrm{TyExpr}(n+1)}{\Pi_A B \in \mathrm{TyExpr}(n)}$$

$$\frac{A \in \mathrm{TyExpr}(n) \qquad B \in \mathrm{TyExpr}(n+1)}{\Sigma_A B \in \mathrm{TyExpr}(n)}$$

$$\frac{A \in \mathrm{TyExpr}(n) \qquad u, v \in \mathrm{TmExpr}(n)}{\mathbf{Id}_A(u, v) \in \mathrm{TyExpr}(n)}$$

$$\frac{i \in \mathbb{N}}{\mathbf{U}_i \in \mathrm{TyExpr}(n)} \qquad \frac{i \in \mathbb{N} \qquad v \in \mathrm{TmExpr}(n)}{\mathbf{El}_i(v) \in \mathrm{TyExpr}(n)}$$

$$\frac{l < n}{\mathbf{x}_l \in \mathrm{TmExpr}(n)}$$

$$\frac{P \in \mathrm{TyExpr}(n+1) \qquad u \in \mathrm{TmExpr}(n)}{\mathbf{empty\_elim}(P, u) \in \mathrm{TmExpr}(n)} \qquad \frac{}{\star \in \mathrm{TmExpr}(n)}$$

$$\frac{P \in \mathrm{TyExpr}(n+1) \qquad d_\star \in \mathrm{TmExpr}(n) \qquad u \in \mathrm{TmExpr}(n)}{\mathbf{unit\_elim}(P, d_\star, u) \in \mathrm{TmExpr}(n)}$$

$$\frac{}{\mathbf{zero} \in \mathrm{TmExpr}(n)} \qquad \frac{u \in \mathrm{TmExpr}(n)}{\mathbf{suc}(u) \in \mathrm{TmExpr}(n)}$$

$$\frac{\begin{array}{c} P \in \mathrm{TyExpr}(n+1) \\ d_{\mathbf{zero}} \in \mathrm{TmExpr}(n) \qquad d_{\mathbf{suc}} \in \mathrm{TmExpr}(n+2) \qquad u \in \mathrm{TmExpr}(n) \end{array}}{\mathbf{ind}(P, d_{\mathbf{zero}}, d_{\mathbf{suc}}, u) \in \mathrm{TmExpr}(n)}$$

$$\frac{A, B \in \mathrm{TyExpr}(n) \qquad a \in \mathrm{TmExpr}(n)}{\mathbf{inl}(A, B, a) \in \mathrm{TmExpr}(n)}$$

$$\frac{A, B \in \mathrm{TyExpr}(n) \qquad b \in \mathrm{TmExpr}(n)}{\mathbf{inr}(A, B, b) \in \mathrm{TmExpr}(n)}$$

$$\frac{\begin{array}{c} A, B \in \mathrm{TyExpr}(n) \qquad P \in \mathrm{TyExpr}(n+1) \\ d_{\mathbf{inl}}, d_{\mathbf{inr}} \in \mathrm{TmExpr}(n+1) \qquad u \in \mathrm{TmExpr}(n) \end{array}}{\mathbf{match}(A, B, P, d_{\mathbf{inl}}, d_{\mathbf{inr}}, u) \in \mathrm{TmExpr}(n)}$$

$$\frac{A \in \mathrm{TyExpr}(n) \qquad B \in \mathrm{TyExpr}(n+1) \qquad u \in \mathrm{TmExpr}(n+1)}{\lambda(A, B, u) \in \mathrm{TmExpr}(n)}$$

$$\frac{\begin{array}{c} A \in \mathrm{TyExpr}(n) \qquad B \in \mathrm{TyExpr}(n+1) \\ f \in \mathrm{TmExpr}(n) \qquad a \in \mathrm{TmExpr}(n) \end{array}}{\mathbf{app}(A, B, f, a) \in \mathrm{TmExpr}(n)}$$

$$\frac{\begin{array}{c} A \in \mathrm{TyExpr}(n) \qquad B \in \mathrm{TyExpr}(n+1) \\ a \in \mathrm{TmExpr}(n) \qquad b \in \mathrm{TmExpr}(n+1) \end{array}}{\mathbf{pair}(A, B, a, b) \in \mathrm{TmExpr}(n)}$$

$$\frac{A \in \mathrm{TyExpr}(n) \qquad B \in \mathrm{TyExpr}(n+1) \qquad u \in \mathrm{TmExpr}(n)}{\mathbf{pr}_1(A, B, u) \in \mathrm{TmExpr}(n)}$$

$$\frac{A \in \mathrm{TyExpr}(n) \qquad B \in \mathrm{TyExpr}(n+1) \qquad u \in \mathrm{TmExpr}(n)}{\mathbf{pr}_2(A, B, u) \in \mathrm{TmExpr}(n)}$$

$$\frac{A \in \mathrm{TyExpr}(n) \qquad a \in \mathrm{TmExpr}(n)}{\mathbf{refl}(A, a) \in \mathrm{TmExpr}(n)}$$

$$\frac{\begin{array}{c} A \in \mathrm{TyExpr}(n) \\ P \in \mathrm{TyExpr}(n+3) \qquad d_{\mathbf{refl}} \in \mathrm{TmExpr}(n+1) \qquad u \in \mathrm{TmExpr}(n) \end{array}}{\mathbf{J}(A, P, d_{\mathbf{refl}}, u, v, p) \in \mathrm{TmExpr}(n)}$$

$$\frac{i \in \mathbb{N}}{0_i \in \mathrm{TmExpr}(n)} \qquad \frac{i \in \mathbb{N}}{1_i \in \mathrm{TmExpr}(n)} \qquad \frac{i \in \mathbb{N}}{\mathbf{n}_i \in \mathrm{TmExpr}(n)}$$

$$\frac{i \in \mathbb{N}}{\mathbf{u}_i \in \mathrm{TmExpr}(n)} \qquad \frac{i \in \mathbb{N} \qquad a, b \in \mathrm{TmExpr}(n)}{a +_i b \in \mathrm{TmExpr}(n)}$$

$$\frac{i \in \mathbb{N} \qquad a \in \mathrm{TmExpr}(n) \qquad b \in \mathrm{TmExpr}(n+1)}{\pi_i(a, b) \in \mathrm{TmExpr}(n)}$$

$$\frac{i \in \mathbb{N} \qquad a \in \mathrm{TmExpr}(n) \qquad b \in \mathrm{TmExpr}(n+1)}{\sigma_i(a, b) \in \mathrm{TmExpr}(n)}$$

$$\frac{i \in \mathbb{N} \qquad a, u, v \in \mathrm{TmExpr}(n)}{\mathbf{id}_i(a, u, v) \in \mathrm{TmExpr}(n)}$$

Remark 2.1.2. We will refer to the various operations above, such as $+$ or $\mathbf{pr}_1$, as *type* and *term constructors*. We will see later that a constructor is well-formed over a given context of length n, if all of its input is well-formed over that same, but possible extended, context. As an example, $\Pi_A B$ will be well-formed over a context of length n, if A is well-formed over it, and B is well-formed over this context *extended* by A.

However, it is important to note that there are a priori no such restrictions on the input a given type/term constructor can take, i.e. $\Pi_A B \in \mathrm{TyExpr}(n)$ for any type expressions $A \in \mathrm{TyExpr}(n)$ and $B \in \mathrm{TyExpr}(n+1)$. This is why we call the sets $\mathrm{TyExpr}(n)$ and $\mathrm{TmExpr}(n)$ *raw* syntax.

In the process of setting up the rules for derivations, one will include precisely the specification that a raw expression is well-formed.

We give a bit of intuition about the syntax presented above. This intuition will be justified once we have stated the rules of the system.

- $\mathbf{0}$ is the *empty type*, which does not contain any terms. One can view $\mathbf{empty\_elim}(P, u)$ as the principle of explosion.
- $\mathbf{1}$ is the *unit type*, which is generated by $\star$. This will be highlighted with $\mathbf{unit\_elim}(P, d_\star, u)$.
- $\mathbf{N}$ is the *type of natural numbers*, which is generated by **zero** and the successor function $\mathbf{suc}(u)$, while $\mathbf{ind}(P, d_{\mathbf{zero}}, d_{\mathbf{suc}}, u)$ represents proof by induction.
- $A+B$ is the *coproduct/sum type* of A and B. Terms are generated by terms of type A via $\mathbf{inl}(A, B, a)$ and by terms of type B by $\mathbf{inr}(A, B, b)$. We can view $\mathbf{match}(A, B, P, d_{\mathbf{inl}}, d_{\mathbf{inr}}, u)$ as a case analysis on u, returning $d_{\mathbf{inl}}$ and $d_{\mathbf{inr}}$ on a terms coming from A and B, respectively.
- $\Pi_A B$ is the *type of dependent functions* from A to B. We can construct terms using lambda abstraction $\lambda(A, B, u)$, and eliminate them using application $\mathbf{app}(A, B, f, a)$.
- $\Sigma_A B$ is the *type of dependent pairs* of terms of A and B. We can construct terms by $\mathbf{pair}(A, B, a, b)$, and given a term of this type we can project to its components using $\mathbf{pr}_1(A, B, u)$ and $\mathbf{pr}_2(A, B, u)$.
- $\mathbf{Id}_A(u, v)$ is the *type of identifications* of u and v in A. A term of $\mathbf{Id}_A(u, v)$ can be seen as an internal equality between u and v. It is generated by the reflexivity identification $\mathbf{refl}(A, a)$ of type $\mathbf{Id}_A(a, a)$. The term $\mathbf{J}(A, P, d_{\mathbf{refl}}, u, v, p)$ captures what is know as path induction.
- $\mathbf{U}_i$ is the *type of the universe at level i*. It is generated by terms that code the 'basic' types 0_i, 1_i and $\mathbf{n}_i$ and closed under codes of the other type formers. For example if a, b are of type $\mathbf{U}_i$, the terms $a+b$, $\pi_i(a, b)$ and $\sigma_i(a, b)$ will also be of type $\mathbf{U}_i$. Moreover, $\mathbf{U}_{i+1}$ contains a code $\mathbf{u}_i$ of $\mathbf{U}_i$. If $v : \mathbf{U}_i$, the type $\mathbf{El}_i(v)$ it the *type of elements* of v. For example, $\mathbf{El}_i(\mathbf{n}_i)$ will be precisely the type of natural numbers $\mathbf{N}$. In light of this, for a of type $\mathbf{U}_i$ and u and v of type $\mathbf{El}_i(a)$, the term $\mathbf{id}_i(a, u, v)$ will also be of type $\mathbf{U}_i$.
- If we work in a context of length n, then $\mathbf{x}_l$ is the *variable of the type at position l*. As previously mentioned, we will use *de Bruijn indices* to index the variables, i.e. in a context of length n, $\mathbf{x}_0$ will be of the *last* type added to the context while, while $\mathbf{x}_{n-1}$ will be of the very *first*. For a slightly different approach using de Buijn indices, see [AAD07].

Remark 2.1.3. Observe that $\mathbf{x}_l$ is not just a variable, but *the* variable of the type at position l. Each position in a context will have precisely one variable associated with it. If one wants multiple variables of a given type, one will be forced to extend the context by multiple copies of the same type.

It is not uncommon in the literature to suppress some of the symbols in the syntax defined above, if certain parts can be deduced from the surroundings. For instance, one might see $\mathbf{app}(f, a)$, $\mathbf{pair}(a, b)$, or $\mathbf{refl}(a)$.

Contexts and context morphisms

Now that we have our first building blocks in the form of type and term expressions, we can expand our raw syntax to include *raw contexts* and *raw context morphisms*. We already informally introduced a context as a list of length n whose entries are types that 'depend' on previous ones. The following definition makes this precise.

Definition 2.1.4. The set $\mathrm{Ctx}(n)$, where $n \in \mathbb{N}$, of *raw contexts of length n* is inductively generated by the following two clauses:

$$\frac{}{\diamond \in \mathrm{Ctx}(0)} \qquad \frac{\Gamma \in \mathrm{Ctx}(n) \qquad A \in \mathrm{TyExpr}(n)}{\Gamma, A \in \mathrm{Ctx}(n+1)}.$$

Remark 2.1.5. Again, we emphasize that a priori there is no assumption on whether A in the second case is actually well-formed in the given context Γ, just that it is raw expression for the given length. This is what makes $\mathrm{Ctx}(n)$ part of the *raw* syntax.

We refer to the symbol $\diamond$ as the *empty context*. A context $(((\diamond, A), B), C)$ is written simply as (A, B, C).

If we have a context $\Gamma \in \mathrm{Ctx}(n)$, and a position $l < n$ one ought be able to retrieve some type $A \in \mathrm{Ctx}(n - l)$ at that given position in Γ. Intuitively, one expects $\mathbf{x}_l$ to be of type A over the context Γ, however that does not work as $\mathbf{x}_l \in \mathrm{TmExpr}(n)$ while $A \in \mathrm{TyExpr}(n - l)$. We will be able to address this after introducing the concept of *weakening*.

A raw context morphisms consists of a list of n term expressions, all over a context of length m. The intuitive picture is that a context morphism δ is a *translation* from a context Γ to a context Δ.

Definition 2.1.6. The set $\mathrm{CtxMor}(m, n)$ for $m, n \in \mathbb{N}$ of *raw context morphisms* is inductively generated by the following two clauses:

$$\frac{}{! \in \mathrm{CtxMor}(m, 0)} \qquad \frac{\delta \in \mathrm{CtxMor}(m, n) \qquad u \in \mathrm{TmExpr}(m)}{\delta, u \in \mathrm{CtxMor}(m, n+1)}.$$

The symbol ! is called the *terminal (context) morphism*. Again we write a morphism $(((!, u), v), w)$ simply as (u, v, w).

A prime example of a context morphism is the *identity context morphism* $\mathrm{id}_n \in \mathrm{CtxMor}(n, n)$ which consists of $(x_{n-1}, \ldots, x_1, x_0)$. However, we will only be able to define this after we have introduced the concept of *weakening*.

2.2 Operations on raw syntax

It is possible to define various operations on raw syntax. Two of these are extremely important: *weakening* and *substitution*. In this section we will introduce these two operations and show to what extent they commute.

Weakening

Suppose we are given an element $A \in \mathrm{TyExpr}(n)$, which intuitively meant that A is a type expression in a context of length n. Now, imagine we were to insert an additional type at a position k in this context (beginning and end point allowed). Then A should still be 'the same' type expression if we swap all occurrences of $\mathbf{x}_l$ with $l \geq k$ for $\mathbf{x}_{l+1}$. This concept is captured by the operation called *weakening*. For this we first need a helper function, that will capture the variable shifts.

Definition 2.2.1. For $n \in \mathbb{N}$ and $k < n + 1$, we define a function

$$\mathrm{wVar}_k(-) : \mathrm{Fin}(n) \to \mathrm{Fin}(n + 1)$$

by structural induction on k:

$$\begin{aligned} \mathrm{wVar}_0(l) &:= l + 1 \\ \mathrm{wVar}_{k+1}(0) &:= 0 \\ \mathrm{wVar}_{k+1}(l + 1) &:= \mathrm{wVar}_k(l) + 1 \end{aligned}$$

Remark 2.2.2. One needs to be a bit careful to check that the above is well defined, i.e. we have omitted all the subscripts that indicate the specific finite set we are dealing with.

To define weakening for a general constructor, we need to compensate for any input over an extension of the context by increasing the position of the weakening accordingly.

Definition 2.2.3. For $n \in \mathbb{N}$ and $k < n + 1$, we define two functions

$$\mathrm{wTy}_k(-) : \mathrm{TyExpr}(n) \to \mathrm{TyExpr}(n + 1)$$

$$\mathrm{wTm}_k(-) : \mathrm{TmExpr}(n) \to \mathrm{TmExpr}(n+1)$$

by a mutual structural induction on $\mathrm{TyExpr}(n)$ and $\mathrm{TmExpr}(n)$. The variable case is given by

$$\mathrm{wTm}_k(\mathbf{x}_l) \coloneqq \mathbf{x}_{\mathrm{wVar}_k(l)},$$

while for *any* other clause, it follows this heuristic pattern:

If $\mathbf{construction}(i, A, u, \dots)$ comes from the inductive clause

$$\frac{i \in \mathbb{N} \qquad A \in \mathrm{TyExpr}(n + m_A) \qquad u \in \mathrm{TmExpr}(n + m_u) \qquad \dots}{\mathbf{construction}(i, A, u, \dots) \in \text{T-Expr}(n)},$$

then it is mapped to

$$\mathbf{construction}(i, \mathrm{wTy}_{k+m_A}(A), \mathrm{wTm}_{k+m_u}(u), \dots).$$

This is a raw expression by apply the induction hypotheses and the inference rule

$$\frac{\begin{array}{c} i \in \mathbb{N} \qquad \mathrm{wTy}_{k+m_A}(A) \in \mathrm{TyExpr}((n+1) + m_A) \\ \mathrm{wTm}_{k+m_u}(u) \in \mathrm{TmExpr}((n+1) + m_u) \end{array}}{\mathbf{construction}(i, \mathrm{wTy}_{k+m_A}(A), \mathrm{wTm}_{k+m_u}(u), \dots) \in \text{T-Expr}(n+1)}$$

To explain the heuristic definition, so let us expand it by giving two examples. Suppose we wish to construct $\mathrm{wTy}_k(\Pi_A B)$. We know that the expression $\Pi_A B$ is formed using

$$\frac{A \in \mathrm{TyExpr}(n) \qquad B \in \mathrm{TyExpr}(n+1)}{\Pi_A B \in \mathrm{TyExpr}(n)}$$

so $m_A = 0$ and $m_B = 1$. By induction hypothesis, we know that $\mathrm{wTy}_k(A) \in \mathrm{TyExpr}(n+1)$ and $\mathrm{wTy}_{k+1}(B) \in \mathrm{TyExpr}((n+1)+1)$ are already constructed and we see that

$$\frac{\mathrm{wTy}_k(A) \in \mathrm{TyExpr}(n+1) \qquad \mathrm{wTy}_{k+1}(B) \in \mathrm{TyExpr}((n+1)+1)}{\Pi_{(\mathrm{wTy}_k(A))}(\mathrm{wTy}_{k+1}(B)) \in \mathrm{TyExpr}(n+1)}$$

ensures a raw expression which is assigned to $\mathrm{wTy}_k(\Pi_A B)$. For a more complicated example, consider the $\mathbf{J}$-constructor

$$\frac{\begin{array}{c} A \in \mathrm{TyExpr}(n) \\ P \in \mathrm{TyExpr}(n+3) \qquad d_{\mathbf{refl}} \in \mathrm{TmExpr}(n+1) \qquad u \in \mathrm{TmExpr}(n) \end{array}}{\mathbf{J}(A, P, d_{\mathbf{refl}}, u, v, p) \in \mathrm{TmExpr}(n)}$$

with $m_A = 0$, $m_P = 3$, $m_{d_{\mathbf{refl}}} = 1$ and $m_u = 0$. By induction hypothesis we have constructions

$$\begin{aligned}
\mathrm{wTy}_k(A) &\in \mathrm{TyExpr}(n+1),\\
\mathrm{wTy}_{k+3}(P) &\in \mathrm{TyExpr}((n+3)+1) = \mathrm{TyExpr}((n+1)+3),\\
\mathrm{wTm}_{k+1}(d_{\mathbf{refl}}) &\in \mathrm{TyExpr}((n+1)+1),\\
\mathrm{wTm}_k(u) &\in \mathrm{TmExpr}(n+1),
\end{aligned}$$

which allows us to determine that

$$\mathbf{J}(\mathrm{wTy}_k(A), \mathrm{wTy}_{k+3}(P), \mathrm{wTm}_{k+1}(d_{\mathbf{refl}}), \mathrm{wTm}_k(u))$$

is indeed an element of $\mathrm{TyExpr}(n+1)$.

Remark 2.2.4. We note that in a setting with named variables weakening is just the identity, which is an argument for that approach.

We can extend the concept of weakening to operations on contexts and context morphisms. The first captures the process we briefly mentioned at the beginning of this subsection: inserting a type into a given context at a certain position.

Definition 2.2.5. For $n \in \mathbb{N}$ and $k < n+1$, we define a function

$$\mathrm{wCtx}_k(-,-) : \mathrm{Ctx}(n) \times \mathrm{TyExpr}(n-k) \to \mathrm{Ctx}(n+1)$$

by structural induction on k and $\mathrm{Ctx}(n)$

$$\begin{aligned}
\mathrm{wCtx}_0(\Gamma, B) &:= (\Gamma, B)\\
\mathrm{wCtx}_{k+1}((\Gamma, A), B) &:= (\mathrm{wCtx}_k(\Gamma, B), \mathrm{wTy}_k(A)).
\end{aligned}$$

The observation in this definition is that it is not enough to just insert the type B at the indicated position k. We need to make sure all types that come *after* k are weakened appropriately for the end result to be accepted in the clauses of Definition 2.1.4.

The connection between the weakening of types and terms will be that if A and u are well-formed expression in a context Γ of length n, and B is any type expression containing at most $n-k$ variables, then $\mathrm{wTy}_k(A)$ and $\mathrm{wTm}_k(u)$ will be well-formed expressions over $\mathrm{wCtx}_k(\Gamma, B)$.

We can now also define the process of looking up a type expression from a given concept, as we alluded to earlier.

Definition 2.2.6. For $n \in \mathbb{N}$ we define a function

$$(-)_{(-)} : \mathrm{Ctx}(n) \times \mathrm{Fin}(n) \to \mathrm{TyExpr}(n)$$

by structural induction on $\mathrm{Fin}(n)$ and $\mathrm{Ctx}(n)$

$$\begin{aligned}(\Gamma, A)_0 &\coloneqq \mathrm{wTy}_0(A),\\ (\Gamma, A)_{l+1} &\coloneqq \mathrm{wTy}_0(\Gamma_l).\end{aligned}$$

The idea behind Γ_l, is retrieving $A \in \mathrm{TyExpr}(n-l)$ from the position l in Γ, and weakening it appropriately, so that it is becomes a raw expression over the entire context Γ. It will serve as the type of $\mathbf{x}_l$.

Weakening of a context morphism amounts to simply weaken all the terms in the list

Definition 2.2.7. For $m, n \in \mathbb{N}$ and $k < n+1$ we define a function

$$\mathrm{wMor}_k(-) : \mathrm{CtxMor}(m,n) \to \mathrm{CtxMor}(m+1,n)$$

by structural induction

$$\begin{aligned}\mathrm{wMor}_k(!) &\coloneqq !\\ \mathrm{wMor}_k(\delta, u) &\coloneqq (\mathrm{wMor}_k(\delta), \mathrm{wTm}_k(u))\end{aligned}$$

A definition that uses the above and which will prove to be very important is taking a morphism, weakening it at the last position and adding to it the last variable.

Definition 2.2.8. For $m, n \in \mathbb{N}$ we define a function

$$\begin{aligned}&\mathrm{wMor{+}}(-) : \mathrm{CtxMor}(m,n) \to \mathrm{CtxMor}(m+1,n+1)\\ &\mathrm{wMor{+}}(\delta) \coloneqq (\mathrm{wMor}_0(\delta), \mathbf{x}_0)\end{aligned}$$

We use this to define the identity morphism we have described earlier.

Definition 2.2.9. For $n \in \mathbb{N}$, we define $\mathrm{id}_n \in \mathrm{CtxMor}(n,n)$ by induction on n

$$\begin{aligned}\mathrm{id}_0 &\coloneqq !\\ \mathrm{id}_{n+1} &\coloneqq \mathrm{wMor{+}}(\mathrm{id}_n)\end{aligned}$$

Observe that $\mathrm{id}_1 = \mathbf{x}_0$, $\mathrm{id}_2 = (\mathbf{x}_1, \mathbf{x}_0)$, $\mathrm{id}_3 = (\mathbf{x}_2, \mathbf{x}_1, \mathbf{x}_0)$, etc. as intended. We will sometimes drop the subscript and write simply id if the length can be inferred from context.

It turns out $\mathrm{wMor{+}}(-)$ is related to a more general operation on $\mathrm{CtxMor}(m,n)$ which we denote by $\mathrm{insertCtxMor}_k(-,-)$. It is the process of inserting $u \in \mathrm{TmExpr}(m)$ at position k in the sequence of $\delta \in \mathrm{CtxMor}(m,n)$.

Definition 2.2.10. For $m, n \in \mathbb{N}$ and $k < n + 1$, we define the function

$$\text{insertCtxMor}_k(-,-) : \text{CtxMor}(m,n) \times \text{TmExpr}(m) \to \text{CtxMor}(m, n+1)$$

by structural induction on k

$$\begin{aligned} \text{insertCtxMor}_0(\delta, t) &\coloneqq (\delta, t) \\ \text{insertCtxMor}_{k+1}((\delta, u), t) &\coloneqq (\text{insertCtxMor}_k(\delta, t), u) \end{aligned}$$

Observe the similarities between this definition and $\text{wCtx}_k(-)$. However, in this case we don't have to weaken anything.

Substitution

Suppose that $A \in \text{TyExpr}(n)$ and we are also given $\delta \in \text{CtxMor}(m, n)$. Then we should be able construct the type expression $A[\delta] \in \text{TyExpr}(m)$, which one gets by replacing any mentioning of $\mathbf{x}_l$ in A by the term u at position l in δ. We call this process *total substitution*. An important special case of this will be *term substitution*, in which $\delta \in \text{CtxMor}(n, n+m)$ and starts with id_n.

As we did with weakening, we first define a helper function that deals with the variable case.

Definition 2.2.11. For $m, n \in \mathbb{N}$ we define a function

$$(-)[-] : \text{Fin}(n) \times \text{CtxMor}(m,n) \to \text{TmExpr}(m)$$

by structural induction on $\text{Fin}(n)$ and $\text{CtxMor}(m, n)$:

$$\begin{aligned} 0[\delta, t] &\coloneqq t \\ (k+1)[\delta, t] &\coloneqq k[\delta], \end{aligned}$$

Now we can define total substitution for arbitrary expressions. Because we use de Bruijn indices we do not have to worry about capture. Instead, to compensate for input defined over an extension of the context we need $\text{wMor+}(-)$ to compensate.

Definition 2.2.12. For $m, n \in \mathbb{N}$ we define two function

$$\begin{aligned} (-)[-] &: \text{TyExpr}(n) \times \text{CtxMor}(m,n) \to \text{TyExpr}(m) \\ (-)[-] &: \text{TmExpr}(n) \times \text{CtxMor}(m,n) \to \text{TmExpr}(m) \end{aligned}$$

by a mutual structural induction on $\text{TyExpr}(n)$ and $\text{TmExpr}(n)$. The variable case is given by

$$\mathbf{x}_l[\delta] \coloneqq l[\delta]$$

while for *any* other clause, it follows this heuristic pattern:

If **construction**$(i, A, u, \dots)$ comes from the clause

$$\frac{i \in \mathbb{N} \qquad A \in \mathrm{TyExpr}(n + m_A) \qquad u \in \mathrm{TmExpr}(n + m_u) \qquad \dots}{\textbf{construction}(i, A, u, \dots) \in \text{T-Expr}(n)},$$

then it is mapped to

$$\textbf{construction}(i, A[\mathrm{wMor}+^{m_A}(\delta)], u[\mathrm{wMor}+^{m_u}(\delta)], \dots),$$

where $\mathrm{wMor}+^{m}(-)$ is the m-times iteration of $\mathrm{wMor}+(-)$. This is a raw expression by

$$\frac{\begin{array}{c} i \in \mathbb{N} \qquad A[\mathrm{wMor}+^{m_A}(\delta)] \in \mathrm{TyExpr}(m + m_A) \\ u[\mathrm{wMor}+^{m_u}(\delta)] \in \mathrm{TmExpr}(m + m_u) \qquad \dots \end{array}}{\textbf{construction}(i, A[\mathrm{wMor}+^{m_A}(\delta)], u[\mathrm{wMor}+^{m_u}(\delta)], \dots) \in \text{T-Expr}(m)}$$

Again we present two examples to unpack the heuristic representation. Suppose we wish to determine $(\Pi_A B)[\delta]$ from the above. Recall that $\Pi_A B$ is formed using the clause

$$\frac{A \in \mathrm{TyExpr}(n) \qquad B \in \mathrm{TyExpr}(n + 1)}{\Pi_A B \in \mathrm{TyExpr}(n)}$$

so $m_A = 0$ and $m_B = 1$. By induction hypothesis, we have constructed

$$\begin{aligned} A[\mathrm{wMor}+^{0}(\delta)] &= A[\delta] \in \mathrm{TyExpr}(m) \\ B[\mathrm{wMor}+^{1}(\delta)] &= B[\mathrm{wMor}+(\delta)] \in \mathrm{TyExpr}(m + 1). \end{aligned}$$

We thus find a raw expression

$$\frac{A[\delta] \in \mathrm{TyExpr}(m) \qquad B[\mathrm{wMor}+(\delta)] \in \mathrm{TyExpr}(m + 1)}{\Pi_{(A[\delta])}(B[\mathrm{wMor}+(\delta)]) \in \mathrm{TyExpr}(m)}$$

which is assigned to $(\Pi_A B)[\delta]$. Taking the **J**-constructor again as a more complicated example, we have

$$\frac{\begin{array}{c} \qquad\qquad A \in \mathrm{TyExpr}(n) \\ P \in \mathrm{TyExpr}(n + 3) \qquad d_{\mathbf{refl}} \in \mathrm{TmExpr}(n + 1) \qquad u \in \mathrm{TmExpr}(n) \end{array}}{\mathbf{J}(A, P, d_{\mathbf{refl}}, u, v, p)A, P, d_{\mathbf{refl}}, u \in \mathrm{TmExpr}(n)}$$

with $m_A = 0$, $m_P = 3$, $m_{d_{\mathbf{refl}}} = 1$ and $m_u = 0$. By induction hypothesis we have constructions

$$A[\mathrm{wMor}+^{0}(\delta)] = A[\delta] \in \mathrm{TyExpr}(m),$$

$$P[\mathrm{wMor}+^3(\delta)] \in \mathrm{TyExpr}(m+3),$$
$$d[\mathrm{wMor}+^1(\delta)] = d[\mathrm{wMor}+(\delta)] \in \mathrm{TyExpr}(m+1),$$
$$u[\mathrm{wMor}+^0(\delta)] = u[\delta] \in \mathrm{TmExpr}(m),$$

from which we can conclude that

$$\mathbf{J}(A[\delta], P[\mathrm{wMor}+^3(\delta)], d_{\mathbf{refl}}[\mathrm{wMor}+(\delta)], u[\delta])$$

is indeed an element of $\mathrm{TyExpr}(m)$.

Total substitution can be extended to the realm of context morphisms, simply by substituting in all of their terms.

Definition 2.2.13. For $m, n, p \in \mathbb{N}$ we define a function

$$(-)[-] : \mathrm{CtxMor}(n,p) \times \mathrm{CtxMor}(m,n) \to \mathrm{CtxMor}(m,p)$$

by structural induction on $\mathrm{CtxMor}(n,p)$:

$$!\,[\delta] \coloneqq !$$
$$(\theta, u)[\delta] \coloneqq (\theta[\delta], u[\delta]).$$

As mentioned earlier, *term substitution* is a special case of total substitution.

Definition 2.2.14. For $p, n \in \mathbb{N}$, and $t_0, \ldots, t_{p-1} \in \mathrm{TmExpr}(n)$, we define a function

$$(-)[t_0, \ldots, t_{p-1}] : \mathrm{TyExpr}(n+p) \to \mathrm{TyExpr}(n)$$
$$A[t_0, \ldots, t_{p-1}] \coloneqq A[\mathrm{id}_n, t_0, \ldots, t_{p-1}]$$

and

$$(-)[t_0, \ldots, t_{p-1}] : \mathrm{TmExpr}(n+p) \to \mathrm{TmExpr}(n)$$
$$u[t_0, \ldots, t_{p-1}] \coloneqq u[\mathrm{id}_n, t_0, \ldots, t_{p-1}].$$

Remark 2.2.15. In the body of this thesis, we will only need the above for at most three terms at a time. However, we will describe the general form nonetheless.

Syntactic equalities

What follows is a list of lemmas about the interplay of the various operations we have defined. They should not come as a surprise given the intuition behind every operation. For proofs we refer to the formalization. The equalities represented here are sometimes less general than possible, mostly because we

only need term substitution up to a maximum of three terms. Regardless, generalization should not be difficult to formalize.

Weakening commutes with itself in the sense that if we first weaken at position k and then at position $k' \leq k$, the end result is the same as first weakening at position k' and then at position $k+1$.

Lemma 2.2.16. *For $m, n \in \mathbb{N}$ and $k' \leq k < n$ we have for any $A \in \mathrm{TyExpr}(n)$, $u \in \mathrm{TmExpr}(n)$ and $\delta \in \mathrm{CtxMor}(m, n)$*

$$\begin{aligned}
\mathrm{wTy}_{k'}(\mathrm{wTy}_k(A)) &= \mathrm{wTy}_{k+1}(\mathrm{wTy}_{k'}(A)),\\
\mathrm{wTm}_{k'}(\mathrm{wTm}_k(u)) &= \mathrm{wTm}_{k+1}(\mathrm{wTm}_{k'}(u)),\\
\mathrm{wMor}_{k'}(\mathrm{wMor}_k(\delta)) &= \mathrm{wMor}_{k+1}(\mathrm{wMor}_{k'}(\delta)).
\end{aligned}$$

□

As a consequence, there is an interplay between Γ_l and $\mathrm{wCtx}_k(\Gamma, B)$. Getting the type at position l, and then weakening it at position k is the same as first weakening the context at k and getting the type at the shifted position of l.

Lemma 2.2.17. *For $n \in \mathbb{N}$, $k < n+1$, $l < n$, $B \in \mathrm{TyExpr}(n-k)$ and $\Gamma \in \mathrm{Ctx}(n)$*

$$\mathrm{wTy}_k(\Gamma_l) = (\mathrm{wCtx}_k(\Gamma, B))_{\mathrm{wVar}_k(l)}.$$

□

Weakening commutes with total substitution in the sense that if one weakens an expression in which one has first substituted a morphism, the end result is the same as substituting the weakened morphism instead.

Lemma 2.2.18. *For $m, n, p \in \mathbb{N}$, $k < m$ and $\delta \in \mathrm{CtxMor}(m, n)$ we have for any $A \in \mathrm{TyExpr}(n)$, $u \in \mathrm{TmExpr}(n)$ and $\theta \in \mathrm{CtxMor}(n, p)$*

$$\begin{aligned}
\mathrm{wTy}_k(A[\delta]) &= A[\mathrm{wMor}_k(\delta)],\\
\mathrm{wTm}_k(u[\delta]) &= u[\mathrm{wMor}_k(\delta)],\\
\mathrm{wTm}_k(\theta[\delta]) &= \theta[\mathrm{wMor}_k(\delta)].
\end{aligned}$$

□

If we substitute a morphism, in which we inserting a term at position k, into an expression that was just weakened at that same position, the end result is the same as simply substituting the original morphism into the original expression.

Lemma 2.2.19. *For $m, n, p \in \mathbb{N}$, $k \leq 3$, $\delta \in \mathrm{CtxMor}(m, n)$ and $t \in \mathrm{TmExpr}(m)$ we have for any $A \in \mathrm{TyExpr}(n)$, $u \in \mathrm{TmExpr}(n)$ and $\theta \in \mathrm{CtxMor}(n, p)$*

$$\begin{aligned}
\mathrm{wTy}_k(A)[\mathrm{insertCtxMor}_k(\delta, t)] &= A[\delta],\\
\mathrm{wTm}_k(u)[\mathrm{insertCtxMor}_k(\delta, t)] &= u[\delta],\\
\mathrm{wMor}_k(\theta)[\mathrm{insertCtxMor}_k(\delta, t)] &= \theta[\delta].
\end{aligned}$$

which specializes in the case of $k = 0$ to

$$\begin{aligned}
\mathrm{wTy}_0(A)[\delta, t] &= A[\delta], \\
\mathrm{wTm}_0(u)[\delta, t] &= u[\delta], \\
\mathrm{wMor}_0(\theta)[\delta, t] &= \theta[\delta].
\end{aligned}$$ □

Substitution by the identity morphism has no effect.

Lemma 2.2.20. *For $n, p \in \mathbb{N}$ we have for any $A \in \mathrm{TyExpr}(n)$, $u \in \mathrm{TmExpr}(n)$ and $\theta \in \mathrm{CtxMor}(n, p)$*

$$\begin{aligned}
A[\mathrm{id}_n] &\equiv A, \\
u[\mathrm{id}_n] &\equiv u, \\
\theta[\mathrm{id}_n] &\equiv \theta. \\
\mathrm{id}_p[\theta] &\equiv \theta.
\end{aligned}$$ □

Total substitution is associative, i.e. substituting by two morphisms one by one is the same as substituting once by the morphisms substituted into each other.

Lemma 2.2.21. *For m, n, p, q, $\delta \in \mathrm{CtxMor}(q, n)$ and $\theta \in \mathrm{CtxMor}(m, q)$ we have for any $A \in \mathrm{TyExpr}(n)$, $u \in \mathrm{TmExpr}(n)$ and $\varphi \in \mathrm{CtxMor}(n, p)$*

$$\begin{aligned}
A[\delta[\theta]] &\equiv (A[\delta])[\theta], \\
u[\delta[\theta]] &\equiv (u[\delta])[\theta], \\
\varphi[\delta[\theta]] &\equiv (\varphi[\delta])[\theta].
\end{aligned}$$ □

Using what we have so far we can show that weakening also commutes with term substitution in the following sense

Lemma 2.2.22. *For $p \leq 3$, $n \in \mathbb{N}$, $k < n$ and $t_0, \ldots, t_{p-1} \in \mathrm{TmExpr}(n)$ we have for any $A \in \mathrm{TyExpr}(n + p)$ and $u \in \mathrm{TmExpr}(n + p)$*

$$\begin{aligned}
&\mathrm{wTy}_k(A[t_0, \ldots, t_{p-1}]) \equiv (\mathrm{wTy}_{k+p}(A))[\mathrm{wTm}_k(t_0), \ldots, \mathrm{wTm}_k(t_{p-1})], \\
&\mathrm{wTm}_k(u[t_0, \ldots, t_{p-1}]) \equiv (\mathrm{wTm}_{k+p}(u))[\mathrm{wTm}_k(t_0), \ldots, \mathrm{wTm}_k(t_{p-1})].
\end{aligned}$$ □

Similarly, term substitution commutes with total substitution.

Lemma 2.2.23. *For $p \leq 3$, $m, n \in \mathbb{N}$, $\delta \in \mathrm{CtxMor}(m, n)$ and $t_0, \ldots, t_{p-1} \in \mathrm{TmExpr}(n)$ we have for any $A \in \mathrm{TyExpr}(n + p)$ and $u \in \mathrm{TmExpr}(n + p)$*

$$\begin{aligned}
(A[t_0, \ldots, t_{p-1}])[\delta] &\equiv (A[\mathrm{wMor}{+}^p(\delta)])[t_0[\delta], \ldots, t_{p-1}[\delta]), \\
(u[t_0, \ldots, t_{p-1}])[\delta] &\equiv (u[\mathrm{wMor}{+}^p(\delta)])[t_0[\delta], \ldots, t_{p-1}[\delta]).
\end{aligned}$$

Which specializes in the case of $t_0, \dots, t_{p-1} \in \mathrm{TmExpr}(n+1)$ *and* $t \in \mathrm{TmExpr}(n)$ *to*

$$(\mathrm{wTy}_p(A))[t_0, \dots, t_{p-1}])[t] \equiv A[t_0[t], \dots, t_{p-1}[t]],$$
$$(\mathrm{wTm}_p(u))[t_0, \dots, t_{p-1}])[t] \equiv u[t_0[t], \dots, t_{p-1}[t]]. \qquad \square$$

Term substituting into an expression that has been repeatedly weakened at 0 has the expected effect.

Lemma 2.2.24. *For* $p, q \in \mathbb{N}$ *with* $q \leq p \leq 3$, $n \in \mathbb{N}$ *and* $t_0, \dots, t_{p-1} \in \mathrm{TmExpr}(n)$ *we have for any* $A \in \mathrm{TyExpr}(n + (p-q))$ *and* $u \in \mathrm{TmExpr}(n + (p-q))$

$$(\mathrm{wTy}_0^q(A))[t_0, \dots, t_{p-1}] \equiv A[t_0, \dots, t_{p-1-q}],$$
$$(\mathrm{wTm}_0^q(u))[t_0, \dots, t_{p-1}] \equiv u[t_0, \dots, t_{p-1-q}]. \qquad \square$$

Finally, we can switch from a weakening to a total substitution in the following way

Lemma 2.2.25. *For* $p \leq 3$ *and* $m, n \in \mathbb{N}$ *we have for any* $A \in \mathrm{TyExpr}(n+p)$, $u \in \mathrm{TmExpr}(n+p)$ *and* $\theta \in \mathrm{CtxMor}(m, n)$

$$\mathrm{wTy}_p(A) \equiv A[\mathrm{wMor}{+}^p(\mathrm{wMor}_0(\mathrm{id}_n))],$$
$$\mathrm{wTm}_p(u) \equiv u[\mathrm{wMor}{+}^p(\mathrm{wMor}_0(\mathrm{id}_n))],$$
$$\mathrm{wMor}_p(\theta) \equiv \theta[\mathrm{wMor}{+}^p(\mathrm{wMor}_0(\mathrm{id}_n))]. \qquad \square$$

Remark 2.2.26. The above might give the impression we could have defined weakening in terms of substitution in the first place. However, recall that in order to define $\mathrm{wMor}_k(-)$ we relied on the definition of weakening.

2.3 Derivations

We are now in a position to talk about the kind of statements the system will derive and the rules that will allow one to derive said statements.

Judgments

A general statement in dependent type theory is called a *judgment*. Judgments over a given context come in four flavors.

Definition 2.3.1. The set Judgments(n) of *raw judgments* is the disjoint union of elements from

- $\mathrm{Ctx}(n) \times \mathrm{TyExpr}(n)$ which we denote by $\Gamma \vdash A$ and to be read as: A is a well-formed type expression in context Γ,
- $\mathrm{Ctx}(n) \times \mathrm{TyExpr}(n) \times \mathrm{TyExpr}(n)$ which we denote by $\Gamma \vdash A \equiv B$ and to be read as: A and B are *judgmentally* equal type expression in context Γ,
- $\mathrm{Ctx}(n) \times \mathrm{TmExpr}(n) \times \mathrm{TmExpr}(n)$ which we denote by $\Gamma \vdash u : A$ and to be read as: u is a well-formed term expression of type A in context Γ,
- $\mathrm{Ctx}(n) \times \mathrm{TmExpr}(n) \times \mathrm{TmExpr}(n) \times \mathrm{TyExpr}(n)$ which we denote by $\Gamma \vdash u \equiv v : A$ and to be read as: u and v are *judgmentally* equal term expressions of type A in context Γ.

Remark 2.3.2. One might wonder whether a judgment like $\Gamma \vdash u : A$ should also be understood to contain the information that A is well-formed type expression in context Γ. This is not the case from the get-go, but will be a property of the system we set up.

Rules

A *deduction rule* has the form

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

where the $\mathcal{J}_i$ and $\mathcal{J}$ are all judgments. The $\mathcal{J}_i$ are called *hypotheses* or *premises* and $\mathcal{J}$ the *conclusion*. Given a set of deduction rules, one defines deduction trees in the standard way. A judgment $\mathcal{J}$ is said to be derivable if there exists a deduction tree with conclusion $\mathcal{J}$. We will often conflate a judgment with the question whether it is derivable.

The rules for a Martin-Löf's style type theory come in two flavors. The first being the *structural rules*, which deals with the core properties of the system and ensure that it is well-behaved and the second being the *logical rules* which deal with the additional types one has added to the system. The logical rules of a given type usually include:

- *formation rules*, which indicate when the given type constructor is well-formed.
- *introduction rules*, which indicate which canonical terms are well-formed of this type.

- *elimination rules*, which indicate how to deal with arbitrary well-formed terms of this type.
- *computation rules*, which indicate how the elimination rules acts on canonical terms.

We will indicate rules of this kind by -FORM, -INTRO, -ELIM and -COMP. One can also add some additional logical rules. Examples are *congruence rules* which state that types/terms constructors preserve judgmental equality and η *rules*. We abbreviate these by -CONG and -ETA. More involved examples from the literature which we have not included in our system, are for instance *function extensionality* and the *univalence axiom*.

We will deliberately present all the rules of the system we consider and write down all the premises of a given rule, as it is not uncommon in the literature to leave some of these implicit.

Definition 2.3.3. The deduction rules of MLTT consist of

structural rules:

$$\frac{\Gamma \vdash A}{\Gamma \vdash A \equiv A}\ \textsc{Ty}\text{-REFL} \qquad \frac{\Gamma \vdash A \equiv B}{\Gamma \vdash B \equiv A}\ \textsc{Ty}\text{-SYMM}$$

$$\frac{\Gamma \vdash A \equiv B \qquad \Gamma \vdash B \qquad \Gamma \vdash B \equiv C}{\Gamma \vdash A \equiv C}\ \textsc{Ty}\text{-TRANS}$$

$$\frac{\Gamma \vdash u : A}{\Gamma \vdash u \equiv u : A}\ \textsc{Tm}\text{-REFL} \qquad \frac{\Gamma \vdash u \equiv v : A}{\Gamma \vdash v \equiv u : A}\ \textsc{Tm}\text{-SYMM}$$

$$\frac{\Gamma \vdash u \equiv v : A \qquad \Gamma \vdash v : A \qquad \Gamma \vdash v \equiv w : A}{\Gamma \vdash u \equiv w : A}\ \textsc{Tm}\text{-TRANS}$$

$$\frac{\Gamma \vdash u : A \qquad \Gamma \vdash A \qquad \Gamma \vdash A \equiv B}{\Gamma \vdash u : B}\ \textsc{Conv}$$

$$\frac{\Gamma \vdash u \equiv v : A \qquad \Gamma \vdash A \qquad \Gamma \vdash A \equiv B}{\Gamma \vdash u \equiv v : B}\ \textsc{ConvEq}$$

$$\frac{\Gamma \vdash \Gamma_l}{\Gamma \vdash \mathbf{x}_l : \Gamma_l}\ \textsc{Var}$$

Logical rules:

$$\frac{}{\Gamma \vdash \mathbf{0}}\ \mathbf{0}\text{-FORM} \qquad \frac{\Gamma, \mathbf{0} \vdash P \qquad \Gamma \vdash u : \mathbf{0}}{\Gamma \vdash \mathbf{empty\_elim}(P, u) : P[u]}\ \mathbf{0}\text{-ELIM}$$

$$\frac{}{\Gamma \vdash \mathbf{1}}\ \mathbf{1}\text{-FORM} \qquad \frac{}{\Gamma \vdash \star : \mathbf{1}}\ \star\text{-INTRO}$$

$$\frac{\Gamma, \mathbf{1} \vdash P \qquad \Gamma \vdash d_\star : P[\star] \qquad \Gamma \vdash u : \mathbf{1}}{\Gamma \vdash \mathbf{unit\_elim}(P, d_\star, u) : P[u]}\ \mathbf{1}\text{-ELIM}$$

$$\frac{\Gamma, \mathbf{1} \vdash P \qquad \Gamma \vdash d_\star : P[\star]}{\Gamma \vdash \mathbf{unit\_elim}(P, d_\star, \star) \equiv d_\star : P[\star]}\ \star\text{-COMP}$$

$$\frac{}{\Gamma \vdash \mathbf{N}}\ \mathbf{N}\text{-FORM}$$

$$\frac{}{\Gamma \vdash \mathbf{zero} : \mathbf{N}}\ \mathbf{zero}\text{-INTRO} \qquad \frac{\Gamma \vdash u : \mathbf{N}}{\Gamma \vdash \mathbf{suc}(u) : \mathbf{N}}\ \mathbf{suc}\text{-INTRO}$$

$$\frac{\Gamma, \mathbf{N} \vdash P \qquad \Gamma \vdash d_{\mathbf{zero}} : P[\mathbf{zero}] \qquad \Gamma, \mathbf{N}, P \vdash d_{\mathbf{suc}} : (\mathrm{wTy}_1(\mathrm{wTy}_1(P)))[\mathbf{suc}(\mathbf{x}_1)] \qquad \Gamma \vdash u : \mathbf{N}}{\Gamma \vdash \mathbf{ind}(P, d_{\mathbf{zero}}, d_{\mathbf{suc}}, u) : P[u]}\ \mathbf{ind}\text{-ELIM}$$

$$\frac{\Gamma, \mathbf{N} \vdash P \qquad \Gamma \vdash d_{\mathbf{zero}} : P[\mathbf{zero}] \qquad \Gamma, \mathbf{N}, P \vdash d_{\mathbf{suc}} : (\mathrm{wTy}_1(\mathrm{wTy}_1(P)))[\mathbf{suc}(\mathbf{x}_1)]}{\Gamma \vdash \mathbf{ind}(P, d_{\mathbf{zero}}, d_{\mathbf{suc}}, \mathbf{zero}) \equiv d_{\mathbf{zero}} : P[\mathbf{zero}]}\ \mathbf{zero}\text{-COMP}$$

$$\frac{\Gamma, \mathbf{N} \vdash P \qquad \Gamma \vdash d_{\mathbf{zero}} : P[\mathbf{zero}] \qquad \Gamma, \mathbf{N}, P \vdash d_{\mathbf{suc}} : (\mathrm{wTy}_1(\mathrm{wTy}_1(P)))[\mathbf{suc}(\mathbf{x}_1)] \qquad \Gamma \vdash u : \mathbf{N}}{\Gamma \vdash \mathbf{ind}(P, d_{\mathbf{zero}}, d_{\mathbf{suc}}, \mathbf{suc}(u)) \equiv d_{\mathbf{suc}}[u, \mathbf{ind}(P, d_{\mathbf{zero}}, d_{\mathbf{suc}}, u)] : P[\mathbf{suc}(u)]}\ \mathbf{suc}\text{-COMP}$$

$$\frac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A + B}\ +\text{-FORM} \qquad \frac{\Gamma \vdash A \qquad \Gamma \vdash B \qquad \Gamma \vdash a : A}{\Gamma \vdash \mathbf{inl}(A, B, a) : A + B}\ \mathbf{inl}\text{-INTRO}$$

$$\frac{\Gamma \vdash A \qquad \Gamma \vdash B \qquad \Gamma \vdash b : B}{\Gamma \vdash \mathbf{inr}(A, B, b) : A + B}\ \mathbf{inr}\text{-INTRO}$$

$$\frac{\begin{array}{c}\Gamma \vdash A \qquad \Gamma \vdash B \qquad \Gamma, A + B \vdash P \\ \Gamma, A \vdash d_{\mathbf{inl}} : (\mathrm{wTy}_1(P))[\mathbf{inl}(\mathrm{wTy}_0(A), \mathrm{wTy}_0(B), \mathbf{x}_0)] \\ \Gamma, B \vdash d_{\mathbf{inr}} : (\mathrm{wTy}_1(P))[\mathbf{inr}(\mathrm{wTy}_0(A), \mathrm{wTy}_0(B), \mathbf{x}_0)] \\ \Gamma \vdash u : A + B\end{array}}{\Gamma \vdash \mathbf{match}(A, B, P, d_{\mathbf{inl}}, d_{\mathbf{inr}}, u) : P[u]}\ \mathbf{match}\text{-ELIM}$$

$$\frac{\begin{array}{c}\Gamma \vdash A \qquad \Gamma \vdash B \qquad \Gamma, A + B \vdash P \\ \Gamma, A \vdash d_{\mathbf{inl}} : (\mathrm{wTy}_1(P))[\mathbf{inl}(\mathrm{wTy}_0(A), \mathrm{wTy}_0(B), \mathbf{x}_0)] \\ \Gamma, B \vdash d_{\mathbf{inr}} : (\mathrm{wTy}_1(P))[\mathbf{inr}(\mathrm{wTy}_0(A), \mathrm{wTy}_0(B), \mathbf{x}_0)] \\ \Gamma \vdash a : A\end{array}}{\Gamma \vdash \mathbf{match}(A, B, P, d_{\mathbf{inl}}, d_{\mathbf{inr}}, \mathbf{inl}(A, B, a)) \equiv d_{\mathbf{inl}}[a] : P[\mathbf{inl}(A, B, a)]}\ \mathbf{inl}\text{-COMP}$$

$$\frac{\Gamma \vdash A \qquad \Gamma \vdash B \qquad \Gamma, A + B \vdash P \qquad \Gamma, A \vdash d_{\mathbf{inl}} : (\mathrm{wTy}_1(P))[\mathbf{inl}(\mathrm{wTy}_0(A), \mathrm{wTy}_0(B), \mathbf{x}_0)] \qquad \Gamma, B \vdash d_{\mathbf{inr}} : (\mathrm{wTy}_1(P))[\mathbf{inr}(\mathrm{wTy}_0(A), \mathrm{wTy}_0(B), \mathbf{x}_0)] \qquad \Gamma \vdash b : B}{\Gamma \vdash \mathbf{match}(A, B, P, d_{\mathbf{inl}}, d_{\mathbf{inr}}, \mathbf{inr}(A, B, b)) \equiv d_{\mathbf{inr}}[b] : P[\mathbf{inr}(A, B, b)]}\ \mathbf{inr}\text{-COMP}$$

$$\frac{\Gamma \vdash A \qquad \Gamma, A \vdash B}{\Gamma \vdash \Pi_A B}\ \Pi\text{-FORM}$$

$$\frac{\Gamma \vdash A \qquad \Gamma, A \vdash B \qquad \Gamma, A \vdash u : B}{\Gamma \vdash \lambda(A, B, u) : \Pi_A B}\ \lambda\text{-INTRO}$$

$$\frac{\Gamma \vdash A \qquad \Gamma, A \vdash B \qquad \Gamma \vdash f : \Pi_A B \qquad \Gamma \vdash a : A}{\Gamma \vdash \mathbf{app}(A, B, f, a) : B[a]}\ \mathbf{ap}\text{-ELIM}$$

$$\frac{\Gamma \vdash A \qquad \Gamma, A \vdash B \qquad \Gamma, A \vdash u : B \qquad \Gamma \vdash a : A}{\Gamma \vdash \mathbf{app}(A, B, \lambda(A, B, u), a) \equiv u[a] : B[a]}\ \lambda\text{-COMP}$$

$$\frac{\Gamma \vdash A \qquad \Gamma, A \vdash B}{\Gamma \vdash \Sigma_A B}\ \Sigma\text{-FORM}$$

$$\frac{\Gamma \vdash A \qquad \Gamma, A \vdash B \qquad \Gamma \vdash a : A \qquad \Gamma \vdash b : B[a]}{\Gamma \vdash \mathbf{pair}(A, B, a, b) : \Sigma_A B}\ \mathbf{pair}\text{-INTRO}$$

$$\frac{\Gamma \vdash A \qquad \Gamma, A \vdash B \qquad \Gamma \vdash u : \Sigma_A B}{\Gamma \vdash \mathbf{pr}_1(A, B, u) : A}\ \mathbf{pr}_1\text{-ELIM}$$

$$\frac{\Gamma \vdash A \qquad \Gamma, A \vdash B \qquad \Gamma \vdash u : \Sigma_A B}{\Gamma \vdash \mathbf{pr}_2(A, B, u) : B[\mathbf{pr}_1(A, B, u)]}\ \mathbf{pr}_2\text{-ELIM}$$

$$\frac{\Gamma \vdash A \qquad \Gamma, A \vdash B \qquad \Gamma \vdash a : A \qquad \Gamma \vdash b : B[a]}{\Gamma \vdash \mathbf{pr}_1(A, B, \mathbf{pair}(A, B, a, b)) \equiv a : A}\ \mathbf{pr}_1\text{-COMP}$$

$$\frac{\Gamma \vdash A \qquad \Gamma, A \vdash B \qquad \Gamma \vdash a : A \qquad \Gamma \vdash b : B[a]}{\Gamma \vdash \mathbf{pr}_2(A, B, \mathbf{pair}(A, B, a, b)) \equiv b : B[a]}\ \mathbf{pr}_2\text{-COMP}$$

$$\frac{\Gamma \vdash A \qquad \Gamma \vdash u : A \qquad \Gamma \vdash v : A}{\Gamma \vdash \mathbf{Id}_A(u, v)}\ \mathbf{Id}\text{-FORM}$$

$$\frac{\Gamma \vdash A \qquad \Gamma \vdash a : A}{\Gamma \vdash \mathbf{refl}(A, a) : \mathbf{Id}_A(a, a)}\ \mathbf{refl}\text{-INTRO}$$

$$\frac{\Gamma \vdash A \qquad \Gamma, A, \mathrm{wTy}_0(A), \mathbf{Id}_{\mathrm{wTy}_0(\mathrm{wTy}_0(A))}(\mathbf{x}_1, \mathbf{x}_0) \vdash P \qquad \Gamma, A \vdash d_{\mathbf{refl}} : (\mathrm{wTy}_3(P))[\mathbf{x}_0, \mathbf{x}_0, \mathbf{refl}(\mathrm{wTy}_0(A), \mathbf{x}_0)] \qquad \Gamma \vdash u : A \qquad \Gamma \vdash v : A \qquad \Gamma \vdash p : \mathbf{Id}_A(u, v)}{\Gamma \vdash \mathbf{J}(A, P, d_{\mathbf{refl}}, u, v, p) : P[u, v, p]}\ \mathbf{J}\text{-ELIM}$$

$$\frac{\Gamma \vdash A \qquad \Gamma, A, \mathrm{wTy}_0(A), \mathbf{Id}_{\mathrm{wTy}_0(\mathrm{wTy}_0(A))}(\mathbf{x}_1, \mathbf{x}_0) \vdash P \qquad \Gamma, A \vdash d_{\mathbf{refl}} : (\mathrm{wTy}_3(P))[\mathbf{x}_0, \mathbf{x}_0, \mathbf{refl}(\mathrm{wTy}_0(A), \mathbf{x}_0)] \qquad \Gamma \vdash a : A}{\Gamma \vdash \mathbf{J}(A, P, d_{\mathbf{refl}}, a, a, \mathbf{refl}(A, a)) \equiv d_{\mathbf{refl}}[a] : P[a, a, \mathbf{refl}(A, a)]}\ \mathbf{refl}\text{-COMP}$$

$$\frac{}{\Gamma \vdash \mathbf{U}_i}\ \mathbf{U}_i\text{-FORM} \qquad \frac{}{\Gamma \vdash 0_i : \mathbf{U}_i}\ 0_i\text{-INTRO} \qquad \frac{}{\Gamma \vdash 1_i : \mathbf{U}_i}\ 1_i\text{-INTRO}$$

$$\frac{}{\Gamma \vdash \mathbf{n}_i : \mathbf{U}_i}\ \mathbf{n}_i\text{-INTRO} \qquad \frac{}{\Gamma \vdash \mathbf{u}_i : \mathbf{U}_{i+1}}\ \mathbf{u}_i\text{-INTRO}$$

$$\frac{\Gamma \vdash a : \mathbf{U}_i \qquad \Gamma \vdash b : \mathbf{U}_i}{\Gamma \vdash a + b : \mathbf{U}_i}\ +\text{-INTRO}$$

$$\frac{\Gamma \vdash a : \mathbf{U}_i \qquad \Gamma, \mathbf{El}_i(a) \vdash b : \mathbf{U}_i}{\Gamma \vdash \pi_i(a, b) : \mathbf{U}_i}\ \pi_i\text{-INTRO}$$

$$\frac{\Gamma \vdash a : \mathbf{U}_i \qquad \Gamma, \mathbf{El}_i(a) \vdash b : \mathbf{U}_i}{\Gamma \vdash \sigma_i(a, b) : \mathbf{U}_i}\ \sigma_i\text{-INTRO}$$

$$\frac{\Gamma \vdash a : \mathbf{U}_i \qquad \Gamma \vdash u : \mathbf{El}_i(a) \qquad \Gamma \vdash v : \mathbf{El}_i(a)}{\Gamma \vdash \mathbf{id}_i(a, u, v) : \mathbf{U}_i}\ \mathbf{id}_i\text{-INTRO}$$

$$\frac{\Gamma \vdash v : \mathbf{U}_i}{\Gamma \vdash \mathbf{El}_i(v)}\ \mathbf{El}_i\text{-ELIM} \qquad \frac{}{\Gamma \vdash \mathbf{El}_i(0_i) \equiv \mathbf{0}}\ 0_i\text{-COMP}$$

$$\frac{}{\Gamma \vdash \mathbf{El}_i(1_i) \equiv \mathbf{1}}\ 1_i\text{-COMP} \qquad \frac{}{\Gamma \vdash \mathbf{El}_i(\mathbf{n}_i) \equiv \mathbf{N}}\ \mathbf{n}_i\text{-COMP}$$

$$\frac{}{\Gamma \vdash \mathbf{El}_i(\mathbf{u}_i) \equiv \mathbf{U}_i}\ \mathbf{u}_i\text{-COMP} \qquad \frac{\Gamma \vdash a : \mathbf{U}_i \qquad \Gamma \vdash b : \mathbf{U}_i}{\Gamma \vdash \mathbf{El}_i(a +_i b) \equiv \mathbf{El}_i(a) + \mathbf{El}_i(b)}\ +_i\text{-COMP}$$

$$\frac{\Gamma \vdash a : \mathbf{U}_i \qquad \Gamma, \mathbf{El}_i(a) \vdash b : \mathbf{U}_i}{\Gamma \vdash \mathbf{El}_i(\pi_i(a, b)) \equiv \Pi_{\mathbf{El}_i(a)} \mathbf{El}_i(b)}\ \pi_i\text{-COMP}$$

$$\frac{\Gamma \vdash a : \mathbf{U}_i \qquad \Gamma, \mathbf{El}_i(a) \vdash b : \mathbf{U}_i}{\Gamma \vdash \mathbf{El}_i(\sigma_i(a, b)) \equiv \Sigma_{\mathbf{El}_i(a)} \mathbf{El}_i(b)}\ \sigma_i\text{-COMP}$$

$$\frac{\Gamma \vdash a : \mathbf{U}_i \qquad \Gamma \vdash u : \mathbf{El}_i(a) \qquad \Gamma \vdash v : \mathbf{El}_i(a)}{\Gamma \vdash \mathbf{El}_i(\mathbf{id}_i(a, u, v)) \equiv \mathbf{Id}_{\mathbf{El}_i(a)}(u, v)}\ \mathbf{id}_i\text{-COMP}$$

η rules:

$$\frac{\Gamma \vdash A \qquad \Gamma \vdash B \qquad \Gamma \vdash u : A + B}{\Gamma \vdash u \equiv \mathbf{match}(A, B, \mathrm{wTy}_0(A + B), \mathbf{inl}(\mathrm{wTy}_0(A), \mathrm{wTy}_0(B), \mathbf{x}_0), \mathbf{inr}(\mathrm{wTy}_k(A), \mathrm{wTy}_k(B), \mathbf{x}_0), u) : A + B}\ +\text{-ETA}$$

$$\frac{\Gamma \vdash A \qquad \Gamma, A \vdash B \qquad \Gamma \vdash f : \Pi_A B}{\Gamma \vdash f \equiv \lambda(A, B, \mathbf{app}(\mathrm{wTy}_0(A), \mathrm{wTy}_1(B), \mathrm{wTm}_0(f), \mathbf{x}_0)) : \Pi_A B}\ \Pi\text{-ETA}$$

$$\frac{\Gamma \vdash A \qquad \Gamma, A \vdash B \qquad \Gamma \vdash u : \Sigma_A B}{\Gamma \vdash u \equiv \mathbf{pair}(A, B, \mathbf{pr}_1(A, B, u), \mathbf{pr}_2(A, B, u)) : \Sigma_A B}\ \Sigma\text{-ETA}$$

Congruence rules:

$$\frac{}{\Gamma \vdash 0_i \equiv 0_i}\ 0_i\text{-CONG} \qquad \frac{}{\Gamma \vdash 1_i \equiv 1_i}\ 1_i\text{-CONG} \qquad \frac{}{\Gamma \vdash \mathbf{N} \equiv \mathbf{N}}\ \mathbf{N}\text{-CONG}$$

$$\frac{\Gamma \vdash A \equiv A' \qquad \Gamma \vdash B \equiv B'}{\Gamma \vdash A + B \equiv A' + B'}\ +\text{-CONG}$$

$$\frac{\Gamma \vdash A \equiv A' \qquad \Gamma \vdash A \qquad \Gamma, A \vdash B \equiv B'}{\Gamma \vdash \Pi_A B \equiv \Pi_{A'} B'}\ \Pi\text{-CONG}$$

$$\frac{\Gamma \vdash A \equiv A' \qquad \Gamma \vdash A \qquad \Gamma, A \vdash B \equiv B'}{\Gamma \vdash \Sigma_A B \equiv \Sigma_{A'} B'}\ \Sigma\text{-CONG}$$

$$\frac{\Gamma \vdash A \equiv A' \qquad \Gamma \vdash u \equiv u' : A \qquad \Gamma \vdash v \equiv v' : A}{\Gamma \vdash \mathbf{Id}_A(u, v) \equiv \mathbf{Id}_{A'}(u', v')}\ \mathbf{Id}\text{-CONG}$$

$$\frac{\Gamma, \mathbf{0} \vdash P \equiv P' \qquad \Gamma \vdash u \equiv u' : \mathbf{0}}{\Gamma \vdash \mathbf{empty\_elim}(P, u) \equiv \mathbf{empty\_elim}(P', u') : P[u]}\ \mathbf{empty\_elim}\text{-CONG}$$

$$\frac{}{\Gamma \vdash \star \equiv \star : \mathbf{N}}\ \star\text{-CONG}$$

$$\frac{\Gamma, \mathbf{1} \vdash P \equiv P' \qquad \Gamma \vdash d_\star \equiv d'_\star : P[\star] \qquad \Gamma \vdash u \equiv u' : \mathbf{1}}{\Gamma \vdash \mathbf{unit\_elim}(P, d_\star, u) \equiv \mathbf{unit\_elim}(P', d'_\star, u') : P[u]}\ \mathbf{unit\_elim}\text{-CONG}$$

$$\frac{}{\Gamma \vdash \mathbf{zero} \equiv \mathbf{zero} : \mathbf{N}}\ \mathbf{zero}\text{-CONG} \qquad \frac{\Gamma \vdash u \equiv u' : \mathbf{N}}{\Gamma \vdash \mathbf{suc}(u) \equiv \mathbf{suc}(u') : \mathbf{N}}\ \mathbf{suc}\text{-CONG}$$

$$\frac{\Gamma, \mathbf{N} \vdash P \equiv P' \qquad \Gamma \vdash d_{\mathbf{zero}} \equiv d'_{\mathbf{zero}} : P[\mathbf{zero}] \qquad \Gamma, \mathbf{N}, P \vdash d_{\mathbf{suc}} \equiv d'_{\mathbf{suc}} : (\mathrm{wTy}_1(\mathrm{wTy}_1(P)))[\mathbf{suc}(\mathbf{x}_1)] \qquad \Gamma \vdash u \equiv u' : \mathbf{N}}{\Gamma \vdash \mathbf{ind}(P, d_{\mathbf{zero}}, d_{\mathbf{suc}}, u) \equiv \mathbf{ind}(P', d'_{\mathbf{zero}}, d'_{\mathbf{suc}}, u') : P[u]}\ \mathbf{ind}\text{-CONG}$$

$$\frac{\Gamma \vdash A \equiv A' \qquad \Gamma \vdash B \equiv B' \qquad \Gamma \vdash a \equiv a' : A}{\Gamma \vdash \mathbf{inl}(A, B, a) \equiv \mathbf{inl}(A', B', a') : A + B}\ \mathbf{inl}\text{-CONG}$$

$$\frac{\Gamma \vdash A \equiv A' \qquad \Gamma \vdash B \equiv B' \qquad \Gamma \vdash b \equiv b' : B}{\Gamma \vdash \mathbf{inr}(A, B, b) \equiv \mathbf{inr}(A', B', b') : A + B}\ \mathbf{inr}\text{-CONG}$$

$$\frac{\Gamma \vdash A \equiv A' \qquad \Gamma \vdash B \equiv B' \qquad \Gamma \vdash A \qquad \Gamma \vdash B \qquad \Gamma, A + B \vdash P \equiv P' \qquad \Gamma, A \vdash d_{\mathbf{inl}} \equiv d'_{\mathbf{inl}} : (\mathrm{wTy}_1(P))[\mathbf{inl}(A, B, a)A, B, \mathbf{x}_0] \qquad \Gamma, B \vdash d_{\mathbf{inr}} \equiv d'_{\mathbf{inr}} : (\mathrm{wTy}_1(P))[\mathbf{inr}(A, B, b)A, B, \mathbf{x}_0] \qquad \Gamma \vdash u \equiv u' : A + B}{\Gamma \vdash \mathbf{match}(A, B, P, d_{\mathbf{inl}}, d_{\mathbf{inr}}, u) \equiv \mathbf{match}(A', B', P', d'_{\mathbf{inl}}, d'_{\mathbf{inr}}, u') : P[u]}\ \mathbf{match}\text{-CONG}$$

$$\frac{\Gamma \vdash A \equiv A' \qquad \Gamma \vdash A \qquad \Gamma, A \vdash B \equiv B' \qquad \Gamma, A \vdash u \equiv u' : B}{\Gamma \vdash \lambda(A, B, u) \equiv \lambda(A', B', u') : \Pi_A B}\ \lambda\text{-CONG}$$

$$\frac{\Gamma \vdash A \equiv A' \qquad \Gamma \vdash A \qquad \Gamma, A \vdash B \equiv B' \qquad \Gamma \vdash f \equiv f' : \Pi_A B \qquad \Gamma \vdash a \equiv a' : A}{\Gamma \vdash \mathbf{app}(A, B, f, a) \equiv \mathbf{app}(A', B', f', a') : B[a]}\ \mathbf{ap}\text{-CONG}$$

$$\frac{\Gamma \vdash A \equiv A' \qquad \Gamma \vdash A \qquad \Gamma, A \vdash B \equiv B' \qquad \Gamma \vdash a \equiv a' : A \qquad \Gamma \vdash b \equiv b' : B[a]}{\Gamma \vdash \mathbf{pair}(A,B,a,b) \equiv \mathbf{pair}(A',B',a',b') : \Sigma_A B}\ \mathbf{pair}\text{-CONG}$$

$$\frac{\Gamma \vdash A \equiv A' \qquad \Gamma \vdash A \qquad \Gamma, A \vdash B \equiv B' \qquad \Gamma \vdash u \equiv u' : \Sigma_A B}{\Gamma \vdash \mathbf{pr}_1(A,B,u) \equiv \mathbf{pr}_1(A',B',u') : A}\ \mathbf{pr}_1\text{-CONG}$$

$$\frac{\Gamma \vdash A \equiv A' \qquad \Gamma \vdash A \qquad \Gamma, A \vdash B \equiv B' \qquad \Gamma \vdash u \equiv u' : \Sigma_A B}{\Gamma \vdash \mathbf{pr}_2(A,B,u) \equiv \mathbf{pr}_2(A',B',u') : B[\mathbf{pr}_1(A,B,u)]}\ \mathbf{pr}_2\text{-CONG}$$

$$\frac{\Gamma \vdash A \equiv A' \qquad \Gamma \vdash a \equiv a'}{\Gamma \vdash \mathbf{refl}(A,a) \equiv \mathbf{refl}(A',a') : \mathbf{Id}_A(a,a)}\ \mathbf{refl}\text{-CONG}$$

$$\frac{\begin{array}{c}\Gamma \vdash A \equiv A' \\ \Gamma \vdash A \qquad \Gamma, A, \mathrm{wTy}_0(A), \mathbf{Id}_{\mathrm{wTy}_0(\mathrm{wTy}_0(A))}(\mathbf{x}_1,\mathbf{x}_0) \vdash P \equiv P' \\ \Gamma, A \vdash d_{\mathbf{refl}} \equiv d'_{\mathbf{refl}} : (\mathrm{wTy}_3(P))[\mathbf{x}_0, \mathbf{refl}(\mathrm{wTy}_0(A), \mathbf{x}_0)] \\ \Gamma \vdash u \equiv u' : A \qquad \Gamma \vdash v \equiv v' : A \qquad \Gamma \vdash p \equiv p' : \mathbf{Id}_A(u,v)\end{array}}{\Gamma \vdash \mathbf{J}(A,P,d_{\mathbf{refl}},u,v,p) \equiv \mathbf{J}(A',P',d'_{\mathbf{refl}},u',v',p') : P[u,v,p]}\ \mathbf{J}\text{-CONG}$$

$$\frac{\Gamma \vdash a \equiv a' : \mathbf{U}_i \qquad \Gamma \vdash b \equiv b' : \mathbf{U}_i}{\Gamma \vdash a +_i b \equiv a' +_i b' : \mathbf{U}_i}\ +_i\text{-CONG}$$

$$\frac{\Gamma \vdash a \equiv a' : \mathbf{U}_i \qquad \Gamma \vdash a : \mathbf{U}_i \qquad \Gamma, \mathbf{El}_i(a) \vdash b \equiv b' : \mathbf{U}_i}{\Gamma \vdash \pi_i(a,b) \equiv \pi_i(a',b') : \mathbf{U}_i}\ \pi_i\text{-CONG}$$

$$\frac{\Gamma \vdash a \equiv a' : \mathbf{U}_i \qquad \Gamma \vdash a : \mathbf{U}_i \qquad \Gamma, \mathbf{El}_i(a) \vdash b \equiv b' : \mathbf{U}_i}{\Gamma \vdash \sigma_i(a,b) \equiv \sigma_i(a',b') : \mathbf{U}_i}\ \sigma_i\text{-CONG}$$

$$\frac{\Gamma \vdash a \equiv a' \mathbf{U}_i \qquad \Gamma \vdash u \equiv u' : \mathbf{El}_i(a) \qquad \Gamma \vdash v \equiv v' : \mathbf{El}_i(a)}{\Gamma \vdash \mathbf{id}_i(a,u,v) \equiv \mathbf{id}_i(a',u',v') : \mathbf{U}_i}\ \mathbf{id}_i\text{-CONG}$$

$$\frac{\Gamma \vdash v \equiv v' : \mathbf{U}_i}{\Gamma \vdash \mathbf{El}_i(v) \equiv \mathbf{El}_i(v')}\ \mathbf{El}_i\text{-CONG}$$

Remark 2.3.4.

- In the literature rules are often represented with named variables, to make more readable. As an example, consider the following presentation of **J** -ELIM in the style of the HoTT book [UFP13, Appendix A.2]:

$$\frac{\begin{array}{c}\Gamma, x : A, y : A, p : \mathbf{Id}_A(x,y) \vdash P \\ \Gamma, z : A \vdash c : P[z,z,\mathbf{refl}(z,A)/x,y,p] \\ \Gamma \vdash u : A \qquad \Gamma \vdash v : A \qquad \Gamma \vdash p' : \mathbf{Id}_A(u,v)\end{array}}{\Gamma \vdash \mathbf{J}(x.y.p.P, z.c, a, b, p') : P[u,v,p'/x,y,p]}\ \mathbf{J}\text{-ELIM}\ .$$

  Although aesthetically pleasing, the formal treatment of variable binding is usually unspecified and if specified, is usually done by de Bruijn indices.

- Another big difference between our presentation of the rules compared to standard presentations, is that we have excluded *context judgments* so far (we will introduce this only later). For example consider the following presentation of the **N** -FORM

$$\frac{\vdash \Gamma}{\Gamma \vdash \mathbf{N}}\ \mathbf{N}\text{-FORM}.$$

  The decision to not include these additional premises is somewhat novel. We claim that this presentation results in the same system for well-formed contexts as the more standard presentation. Properly showing this will not be treated in this thesis and is left for future investigation.

- We have included all premises that introduce metavariables. For instance the premise $\Gamma \vdash A$ in Π -FORM

$$\frac{\Gamma \vdash A \qquad \Gamma, A \vdash B}{\Gamma \vdash \Pi_A B}\ \Pi\text{-FORM}.$$

  These kinds of premises are often left out in the literature.

- We have decided to include only some of the congruence rules. Other variants like

$$\frac{\Gamma \vdash A \equiv A' \qquad \Gamma \vdash A' \qquad \Gamma, A' \vdash B \equiv B' \qquad \Gamma, A \vdash u \equiv u' : B}{\Gamma \vdash \lambda(A, B, u) \equiv \lambda(A', B', u') : \Pi_{A'} B'}\ \lambda\text{-CONG}'$$

  are admissible, i.e. if the premises are derivable, so is the conclusion.

We extend the notion of derivability to contexts and context morphisms. The judgments for these consist of:

- $\mathrm{Ctx}(n)$ which we denote by $\vdash \Gamma$ and to be read as: Γ is a well-formed context,
- $\mathrm{Ctx}(n) \times \mathrm{Ctx}(n)$ which we denote by $\vdash \Gamma \equiv \Gamma'$ and to be read as: Γ and Γ' are judgmentally equal contexts,
- $\mathrm{Ctx}(m) \times \mathrm{CtxMor}(m, n) \times \mathrm{Ctx}(n)$ which we denote by $\Delta \vdash \delta : \Gamma$ and to be read as: δ is a well-formed context morphism from Δ to Γ,
- $\mathrm{Ctx}(m) \times \mathrm{CtxMor}(m, n) \times \mathrm{CtxMor}(m, n) \times \mathrm{Ctx}(n)$ which we denote by $\Delta \vdash \delta \equiv \delta' : \Gamma$ and to be read as: δ and δ' are judgmentally equal context morphisms from Δ to Γ.

As mentioned in Remark 2.3.4 it is not uncommon in the literature to include these kinds of judgments already from the beginning and use them as premises in some of the structural/logical rules of the system. In our setup we avoid these premises and as such we have decided to include them only now.

Definition 2.3.5. The deduction rules for contexts and context morphisms are given by

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

$$\frac{}{\vdash \diamond \equiv \diamond} \qquad \frac{\vdash \Gamma \equiv \Gamma' \qquad \Gamma \vdash A \equiv A'}{\vdash (\Gamma, A) \equiv (\Gamma', A')}$$

$$\frac{}{\Delta \vdash ! : \diamond} \qquad \frac{\Delta \vdash \delta : \Gamma \qquad \Delta \vdash u : A[\delta]}{\Delta \vdash (\delta, u) : \Gamma, A}$$

$$\frac{}{\Gamma \vdash ! \equiv ! : \diamond} \qquad \frac{\Delta \vdash \delta \equiv \delta' : \Gamma \qquad \Delta \vdash u \equiv u' : A[\delta]}{\Delta \vdash (\delta, u) \equiv (\delta', u') : \Gamma, A}$$

Remark 2.3.6. One might have expected some additional premises, but they turn out to be redundant. For the sake of simplicity we have not included them.

Admissible rules

As mentioned previously, there are a lot of additional rules one might want to include in the system described above. However, this turns out to be unnecessary, as they are *admissible*. We will list a number of them here. The benefit of not explicitly including them in the system, is that they would complicate situations in which we want to use induction on a derivation.

Admissible rules are often presented in the same way as ordinary deduction rules. However, they should be read differently: if the judgments in its premises are derivable, then so is its conclusion. The proofs of these are mostly known and standard, and thus we refer to our formalization for the details. Alternatively, see [Luo94, Chapter 3] for similar results for a smaller system.

We start with two very well known admissible rules that deal with substitution and weakening.

Lemma 2.3.7. *If* $\Gamma \vdash \mathcal{J}$*, then* $\mathrm{wCtx}_k(\Gamma, B) \vdash \mathrm{w}_k(\mathcal{J})$*, i.e.*

$$\frac{\Gamma \vdash A}{\mathrm{wCtx}_k(\Gamma, B) \vdash \mathrm{wTy}_k(A)} \qquad \frac{\Gamma \vdash u : A}{\mathrm{wCtx}_k(\Gamma, B) \vdash \mathrm{wTm}_k(u) : \mathrm{wTy}_k(A)}$$

$$\frac{\Gamma \vdash A \equiv A'}{\mathrm{wCtx}_k(\Gamma, B) \vdash \mathrm{wTy}_k(A) \equiv \mathrm{wTy}_k(A')}$$

$$\frac{\Gamma \vdash u \equiv u' : A}{\mathrm{wCtx}_k(\Gamma, B) \vdash \mathrm{wTm}_k(u) \equiv \mathrm{wTm}_k(u') : \mathrm{wTy}_k(A)}.$$

Lemma 2.3.8. *If* $\Gamma \vdash \mathcal{J}$ *and* $\Delta \vdash \delta : \Gamma$*, then* $\Gamma \vdash \mathcal{J}[\delta]$*, i.e.*

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

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

Additionally, we have

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

$$\frac{\Gamma \vdash u : A \qquad \Delta \vdash \delta : \Gamma \qquad \Delta \vdash \delta \equiv \delta' : \Gamma}{\Delta \vdash u[\delta] \equiv u[\delta'] : A[\delta]}.$$

Proof. The proof of these two lemmas is by mutual induction on the appropriate judgment in the premise of each rule and includes the following helper rules within the induction. The first couple dealing with the weakening of morphisms:

$$\frac{\Delta \vdash \delta : \Gamma}{\mathrm{wCtx}_k(\Delta, B) \vdash \mathrm{wMor}_k(\delta) : \Gamma}$$

$$\frac{\Delta \vdash \delta \equiv \delta' : \Gamma}{\mathrm{wCtx}_k(\Delta, B) \vdash \mathrm{wMor}_k(\delta) \equiv \mathrm{wMor}_k(\delta') : \Gamma} \qquad \frac{\Delta \vdash \delta : \Gamma \qquad \Delta \vdash A[\delta]}{\Delta, A[\delta] \vdash \mathrm{wMor{+}}(\delta) : \Gamma, A}$$

$$\frac{\Delta \vdash \delta : \Gamma \qquad \Delta \vdash \delta \equiv \delta' : \Gamma \qquad \Gamma \vdash A}{\Delta, A[\delta] \vdash \mathrm{wMor{+}}(\delta) \equiv \mathrm{wMor{+}}(\delta') : \Gamma, A}$$

which is shown by structural induction on δ. And finally, two dealing with cases for Γ_l:

$$\frac{\Delta \vdash \delta : \Gamma}{\Delta \vdash \mathbf{x}_l[\delta] : \Gamma_l[\delta]} \qquad \frac{\Delta \vdash \delta \equiv \delta' : \Gamma}{\Delta \vdash \mathbf{x}_l[\delta] \equiv \mathbf{x}_l[\delta'] : \Gamma_l[\delta]}$$

proven by structural induction on l. All of these helper rules as well as the main results are straightforward. □

We can combine the previous rules dealing with judgmental equality of substitution:

Corollary 2.3.9. *The following rules are admissible:*

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

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

Proof. Using the previous lemma and Tm-TRANS and Ty-TRANS. □

Next we have the fact that the identity morphism id_n is a well-formed endomorphism between any well-formed context of length n.

Lemma 2.3.10. *If* $\vdash \Gamma$, *then* $\Gamma \vdash \mathrm{id} : \Gamma$

Proof. By structural induction on Γ. □

This allows us to conclude that term substitution is also admissible.

Corollary 2.3.11. *For $p \leq 3$, the following rule is admissible:*

$$\frac{\Gamma \vdash t_0 : A_0 \qquad \dots \qquad \Gamma \vdash t_{p-1} : A_{p-1}[t_0, \dots, t_{p-2}] \qquad \Gamma, A_0, \dots, A_{p-1} \vdash \mathcal{J}}{\Gamma \vdash \mathcal{J}[t_0, \dots, t_{p-1}]}$$

to be read similarly as in Lemma 2.3.8.

Proof. By combining Lemma 2.3.8 and Lemma 2.3.10. □

From a well-formed context, we can derive its l-th type.

Lemma 2.3.12. *If $\vdash \Gamma$, then $\Gamma \vdash \Gamma_l$.*

Proof. By structural induction on l and Γ using the rules for weakening. The premise $\vdash \Gamma$ is needed in the base case, as one can conclude $\Gamma \vdash A$ from $\vdash \Gamma, A$ and then apply weakening. □

Next, we want to have conversion rules which state that derivable judgments are stable under judgmentally equal contexts. The asymmetry is due to the asymmetric choice in the clause for $\vdash \Gamma \equiv \Delta$.

Lemma 2.3.13. *The following rule is admissible:*

$$\frac{\Delta \vdash \mathcal{J} \qquad \vdash \Gamma \equiv \Delta \qquad \vdash \Gamma}{\Gamma \vdash \mathcal{J}}.$$

Proof. One needs an additional helper rule for the variable case

$$\frac{\vdash \Gamma \equiv \Delta}{\Gamma \vdash \Gamma_l \equiv \Delta_l},$$

which is proven by a straightforward induction on l. The lemma is then proven by mutual induction on the judgment. The only case where the asymmetry pops up is in the variable case. □

We are now in a position to prove a number of rules we call *prepositions*. These are of the form discussed in Remark 2.3.2.

We prove this simultaneously with the fact that judgmental equality between contexts and context morphisms is an equivalence relation.

Lemma 2.3.14. *Judgmental equality between contexts and context morphisms is an equivalence relation, i.e.*

$$\frac{\vdash \Gamma}{\vdash \Gamma \equiv \Gamma} \qquad \frac{\vdash \Gamma \equiv \Delta}{\vdash \Delta \equiv \Gamma} \qquad \frac{\vdash \Gamma \equiv \Delta \qquad \vdash \Delta \equiv \Theta}{\vdash \Gamma \equiv \Theta}$$

$$\frac{\Delta \vdash \delta : \Gamma}{\Delta \vdash \delta \equiv \delta : \Gamma} \qquad \frac{\vdash \Delta \qquad \vdash \Gamma \qquad \Delta \vdash \delta \equiv \theta : \Gamma}{\Delta \vdash \theta \equiv \delta : \Gamma}$$

$$\frac{\vdash \Delta \qquad \vdash \Gamma \qquad \Delta \vdash \delta \equiv \theta : \Gamma \qquad \Delta \vdash \theta \equiv \varphi : \Gamma}{\Delta \vdash \delta \equiv \varphi : \Gamma}$$

Lemma 2.3.15. *The following presuppositions are admissible:*

$$\frac{\vdash \Gamma \qquad \Gamma \vdash A \equiv B}{\Gamma \vdash A} \qquad \frac{\vdash \Gamma \qquad \Gamma \vdash A \equiv B}{\Gamma \vdash B}$$

$$\frac{\vdash \Gamma \qquad \Gamma \vdash u \equiv v : A}{\Gamma \vdash u : A} \qquad \frac{\vdash \Gamma \qquad \Gamma \vdash u \equiv v : A}{\Gamma \vdash v : A} \qquad \frac{\vdash \Gamma \qquad \Gamma \vdash u : A}{\Gamma \vdash A}$$

$$\frac{\vdash \Gamma \equiv \Delta}{\vdash \Gamma} \qquad \frac{\vdash \Gamma \equiv \Delta}{\vdash \Delta}$$

$$\frac{\vdash \Gamma \qquad \vdash \Delta \qquad \Delta \vdash \delta \equiv \theta : \Gamma}{\Delta \vdash \delta : \Gamma} \qquad \frac{\vdash \Gamma \qquad \vdash \Delta \qquad \Delta \vdash \delta \equiv \theta : \Gamma}{\Delta \vdash \theta : \Gamma}$$

Proof. The proof of these two lemmas is by a mutual induction. The reflexivity case for definitional equality of contexts and context morphism are dealt with using Ty-REFL and Tm-REFL. The remaining cases about contexts and context morphisms follow readily.

The presuppositions are all straightforward. Be aware that in rules like **ap** -ELIM, the left and right hand side of the judgmental equality are significantly different. □

We now give the admissible rules for substitutions and conversion extended to context morphisms. We could have already derived some of these previously, but they have not been necessary so far. We therefore collect them here.

Lemma 2.3.16. *The follow rules are admissible:*

$$\frac{\Theta \vdash \theta : \Gamma \qquad \Delta \vdash \delta : \Theta}{\Delta \vdash \theta[\delta] : \Gamma} \qquad \frac{\Theta \vdash \theta \equiv \theta' : \Gamma \qquad \Delta \vdash \delta : \Theta}{\Delta \vdash \theta[\delta] \equiv \theta'[\delta] : \Gamma}$$

$$\frac{\Theta \vdash \theta : \Gamma \qquad \Delta \vdash \delta : \Theta \qquad \Delta \vdash \delta \equiv \delta' : \Theta}{\Delta \vdash \theta[\delta] \equiv \theta[\delta'] : \Gamma}$$

$$\frac{\Theta \vdash \theta : \Gamma \qquad \Theta \vdash \theta \equiv \theta' : \Gamma \qquad \vdash \Theta \qquad \Delta \vdash \delta : \Theta \qquad \Delta \vdash \delta \equiv \delta' : \Theta}{\Delta \vdash \theta[\delta] \equiv \theta'[\delta'] : \Gamma}$$

$$\frac{\Delta \vdash \delta : \Gamma \qquad \vdash \Delta \equiv \Delta' \qquad \vdash \Gamma \equiv \Gamma'}{\Delta' \vdash \delta : \Gamma'}$$

$$\frac{\Delta \vdash \delta \equiv \delta' : \Gamma \qquad \vdash \Delta \equiv \Delta' \qquad \vdash \Gamma \equiv \Gamma'}{\Delta' \vdash \delta \equiv \delta' : \Gamma'}$$

Proof. By structural induction on the appropriate judgment. The need for the premise $\vdash \Theta$ can be seen as a similar requirement to a premise in Ty-TRANS and Tm-TRANS. □

Finally, given the extended list of admissible rules at our disposal, we can alter the premises from some of the rules we have so far. This includes core rules to the systems and admissible ones. This presentation turns out to be more convenient for the development later on.

Lemma 2.3.17. *The following rules are admissible:*

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

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

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

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

$$\frac{\vdash \Theta \qquad \vdash \Delta \qquad \Theta \vdash \theta : \Gamma \qquad \Delta \vdash \delta \equiv \delta' : \Theta}{\Delta \vdash \theta[\delta] \equiv \theta[\delta'] : \Gamma}$$

$$\frac{\vdash \Theta \qquad \vdash \Delta \qquad \Theta \vdash \theta \equiv \theta' : \Gamma \qquad \Delta \vdash \delta \equiv \delta' : \Theta}{\Delta \vdash \theta[\delta] \equiv \theta'[\delta'] : \Gamma}$$

$$\frac{\vdash \Delta \qquad \vdash \Gamma \qquad \Gamma \vdash A \qquad \Delta \vdash \delta \equiv \delta' : \Gamma}{\Delta, A[\delta] \vdash \mathrm{wMor+}(\delta) \equiv \mathrm{wMor+}(\delta') : \Gamma, A} \qquad \frac{\Delta \vdash A \qquad \vdash \Gamma \equiv \Delta}{\Gamma \vdash A}$$

$$\frac{\Delta \vdash A \equiv B \qquad \vdash \Gamma \equiv \Delta}{\Gamma \vdash A \equiv B} \qquad \frac{\Delta \vdash u : A \qquad \vdash \Gamma \equiv \Delta}{\Gamma \vdash u : A}$$

$$\frac{\Delta \vdash u \equiv v : A \qquad \vdash \Gamma \equiv \Delta}{\Gamma \vdash u \equiv v : A} \qquad \frac{\Gamma \vdash A \equiv B \qquad \Gamma \vdash B \equiv C}{\Gamma \vdash A \equiv C}$$

$$\frac{\Gamma \vdash u \equiv v : A \qquad \Gamma \vdash v \equiv w : A}{\Gamma \vdash u \equiv w : A}$$

□

We end by noting there might be more admissible rules in our system, but we have presented all required for our results.

3. Contextual Categories

In the previous chapter we have seen the syntactic world of dependent type theory. In this chapter we will discuss a semantic approach, which in our case will be the one of *contextual categories*. One might not be surprised that the semantics will lie in a categorical setting; we have already encountered a notion of 'context morphism' in the previous chapter which hinted in that direction. Indeed, we will start by establishing that we can form a category out of syntax.

We would like to emphasize that this is not the only (categorical) semantics for dependent type theory that have been proposed so far. However, it will be the one with respect to which we will prove the initiality conjecture.

3.1 Definition of contextual categories

We start by the observation that the syntax forms a category, which is given the name of *syntactic category* or *term model* in the literature. We need to introduce the following definitions.

Definition 3.1.1. A *derivable context* is a context $\Gamma \in \mathrm{Ctx}(n)$ such that $\vdash \Gamma$ holds. We usually denote a derivable context by its underlying context Γ. We define a relation on derivable contexts by

$$\Gamma \sim_{\mathrm{Ctx}} \Gamma' \text{ iff } \vdash \Gamma \equiv \Gamma.$$

This is an equivalence relations by Lemma 2.3.14.

We define a *derivable morphism* to be a triple (δ, Δ, Γ) such that $\Delta \in \mathrm{Ctx}(m)$ and $\Gamma \in \mathrm{Ctx}(n)$ are derivable contexts and $\delta \in \mathrm{CtxMor}(m, n)$ such that $\Delta \vdash \delta : \Gamma$ holds. We usually denote a derivable morphism (δ, Δ, Γ) by its underlying context morphism δ and write $\mathrm{dom}(\delta) \coloneqq \Delta$ and $\mathrm{cod}(\delta) \coloneqq \Gamma$ for the projections. We define a relation on derivable morphisms by

$$\begin{aligned} \delta \sim_{\mathrm{CtxMor}} \delta' \text{ iff } & \mathrm{dom}(\delta) \sim_{\mathrm{Ctx}} \mathrm{dom}(\delta'), \\ & \mathrm{cod}(\delta) \sim_{\mathrm{Ctx}} \mathrm{cod}(\delta') \text{ and} \\ & \mathrm{dom}(\delta) \vdash \delta \equiv \delta' : \mathrm{cod}(\delta). \end{aligned}$$

Note that inclusions like $\mathrm{dom}(\delta') \vdash \delta \equiv \delta' : \mathrm{cod}(\delta')$ into this relation would be redundant due to Lemma 2.3.16. This relation is an equivalence relation again by Lemma 2.3.14.

Remark 3.1.2. Observe that our convention for writing δ instead of the full data (δ, Δ, Γ) means it does not in general hold that $\delta = \delta$, as δ can represent two *different* derivable morphisms.

We will omit the subscripts in both relations if no confusion can arise. We are now in a position to properly introduce the underlying category of the term model.

Definition 3.1.3. The term model $\mathcal{C}(\mathrm{MLTT})$ is the category defined by

- objects: Derivable contexts modulo $\sim_{\mathrm{Ctx}}$. Note that all members of a given equivalence class will have the same length.
- morphisms: Derivable morphisms modulo $\sim_{\mathrm{CtxMor}}$. For an morphism $[\delta]$, its source is $[\mathrm{dom}(\delta)]$ and target is $[\mathrm{cod}(\delta)]$, which we write

  $$[\delta] : [\mathrm{dom}(\delta)] \to [\mathrm{cod}(\delta)].$$

  Note that the source and target of a morphism are well-defined by the definition of $\sim_{\mathrm{CtxMor}}$.
- composition: Given two morphisms $[\theta]$ and $[\delta]$ such that $[\mathrm{cod}(\delta)] = [\mathrm{dom}(\theta)]$, we define their composition

  $$[\theta] \circ [\delta] \coloneqq [\theta[\delta]] : [\mathrm{dom}(\delta)] \to [\mathrm{cod}(\theta)]$$

  by Lemma 2.3.16. This is well-defined by the same lemma. Moreover, this operation is associative by Lemma 2.2.21.
- identity morphism: Given an object $[\Gamma]$ with $\Gamma \in \mathrm{Ctx}(n)$, we define the identity on $[\Gamma]$ by

  $$\mathrm{id}_{[\Gamma]} \coloneqq [\mathrm{id}_n] : [\Gamma] \to [\Gamma]$$

  using Lemma 2.3.10. This is well-defined by noting that the expression id_n only depends on the length of the context Γ.

  Moreover, for any morphism $[\delta]$, we have

  $$\mathrm{id}_{[\mathrm{cod}(\delta)]} \circ [\delta] = [\mathrm{id}[\delta]] = [\delta]$$
  $$[\delta] \circ \mathrm{id}_{[\mathrm{dom}(\delta)]} = [\delta[\mathrm{id}]] = [\delta]$$

  by Lemma 2.2.20.

Remark 3.1.4. Be aware that writing $[\delta]$ instead of $[(\delta, \mathrm{dom}(\delta), \mathrm{cod}(\delta))]$ makes it so in general we cannot conclude $[\delta] = [\delta]$. Good examples of this phenomenon are the morphisms $[\mathrm{id}_n]$.

In particular, in the last part of the definition above we should have also shown, among other things, $\mathrm{dom}(\mathrm{id}[\delta]) \sim \mathrm{dom}(\delta)$. In this case $\mathrm{dom}(\mathrm{id}[\delta]) = \mathrm{dom}(\delta)$ so it follows immediately.

In general, we will not show the equivalence of the underlying contexts when showing two morphisms are equal, as it is often straightforward. For readers that are nonetheless interested, we refer to the formalization.

To investigate the properties that characterize the term model, we introduce the class of categories called *contextual categories*. This class of categories has been designed precisely to abstract away the fundamental properties of $\mathcal{C}(\mathrm{MLTT})$, and was introduced by Cartmell [Car86, Chapter 14] and later studied by Streicher [Str91]. They have also been studied by Voevodsky as C-systems in a series of papers, e.g. [Voe14] [Voe15] and [Voe16].

Definition 3.1.5. A *contextual category* is a (strict) category $\mathcal{C}$ together with the following additional structure:

- a grading of the objects of $\mathcal{C}$ as $\mathrm{Ob}_{\mathcal{C}} = \coprod_{n\in\mathbb{N}} \mathrm{Ob}_{\mathcal{C}}(n)$ such that there is a unique (on the nose) object $\mathrm{pt} \in \mathrm{Ob}_{\mathcal{C}}(0)$, which is a terminal object. We write ptmor_X for the unique morphism $X \to \mathrm{pt}$. For an object $X \in \mathrm{Ob}_{\mathcal{C}}(n)$, we refer to n as the *length* of X.

- for every object $X \in \mathrm{Ob}_{\mathcal{C}}(n+1)$, an object $\mathrm{ft}\, X \in \mathrm{Ob}_{\mathcal{C}}(n)$ (the *father* of X) and a morphism $\mathrm{p}_X : X \to \mathrm{ft}\, X$ which we denote by $\twoheadrightarrow$ in diagrams,

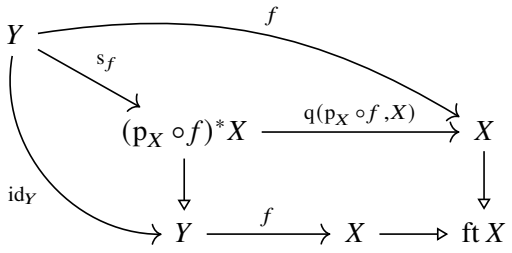
- for every object $X \in \mathrm{Ob}_{\mathcal{C}}(n+1)$ and every morphism $f : Y \to \mathrm{ft}\, X$ with $Y \in \mathrm{Ob}_{\mathcal{C}}(m)$, an object $f^*X \in \mathrm{Ob}_{\mathcal{C}}(m+1)$ such that $\mathrm{ft}(f^*X) = Y$ and a morphism $\mathrm{q}(f, X) : f^*X \to X$, such that the following diagram commutes

$$\begin{array}{ccc} f^*X & \xrightarrow{\mathrm{q}(f,X)} & X \\ \downarrow & & \downarrow \\ Y & \xrightarrow{f} & \mathrm{ft}\, X \end{array}$$

and such that

$$\begin{array}{ll} (\mathrm{id}_{\mathrm{ft}\, X})^* X = X & \mathrm{q}(\mathrm{id}_{\mathrm{ft}\, X}, X) = \mathrm{id}_X \\ (g \circ f)^* X = f^*(g^* X) & \mathrm{q}(g \circ f, X) = \mathrm{q}(g, X) \circ \mathrm{q}(f, g^* X). \end{array}$$

- for every morphism $f : Y \to X$ where $X \in \mathrm{Ob}_{\mathcal{C}}(n+1)$, a morphism $\mathrm{s}_f : Y \to (\mathrm{p}_X \circ f)^* X$ such that the following diagram commutes:



  and such that if $X = g^* U$ (for some object U and some morphism $g : \mathrm{ft}\, X \to \mathrm{ft}\, U$), then

$$\mathrm{s}_f = \mathrm{s}_{\mathrm{q}(g,U)\circ f} \ .$$

  Note that $\mathrm{q}(g, U) \circ f : Y \to U$ and that

$$\begin{aligned}(\mathrm{p}_U \circ \mathrm{q}(g, U) \circ f)^* U &= (g \circ \mathrm{p}_{g^* U} \circ f)^* U \\ &= (\mathrm{p}_{g^* U} \circ f)^* (g^* U) \\ &= (\mathrm{p}_X \circ f)^* X,\end{aligned}$$

  therefore s_f and $\mathrm{s}_{\mathrm{q}(g,U)\circ f}$ have the same domain and codomain.

Remark 3.1.6.

- We emphasize that we are dealing with a *strict* category and we thus consider the equations between the various operations up to *equality*, not merely isomorphism.

- The objects of a contextual category, together with the $\mathrm{p}_{(-)}$-morphisms form a *directed rooted tree* with root pt. This explains the origin of the name 'father'.

- An important example of a contextual category is the category Fam of families of sets. This was already considered in [Car86], although some issues have been raised in [Voe15]. We refer to the latter for a more detailed description.

- Contextual categories can be seen as models of an *essentially algebraic theory*. This already implies the existence of an initial contextual category. Because of this the initiality conjecture for contextual categories can be restated as the claim that this initial category is equivalent to the one defined by the syntax of a dependent type theory. We have decided not to go down this route.

To justify the claim that contextual categories are an abstraction of the properties of $\mathcal{C}(\mathrm{MLTT})$, let us start by establishing that $\mathcal{C}(\mathrm{MLTT})$ comes equipped with the structure of a contextual category.

Lemma 3.1.7. *The term model* $\mathcal{C}(\mathrm{MLTT})$ *has the structure of a contextual category.*

Proof. We already know that $\mathcal{C}(\mathrm{MLTT})$ is a (strict) category. In what follows, we define only the additional structure. The equations that they need to satisfy follow readily by syntactic equalities. For details we refer to the formalization.

- We set $\mathrm{Ob}_{\mathcal{C}(\mathrm{MLTT})}(n) \coloneqq \mathrm{Ctx}(n)/\sim$. Recall that $\diamond$ is the unique context in $\mathrm{Ctx}(0)$ and that it is derivable. For any other (derivable) context Γ we have the unique context moprhism ! which is derivable. This establishes that $[\diamond]$ satisfies the properties of pt.

- For an object $[\Gamma, A] \in \mathrm{Ob}_{\mathcal{C}(\mathrm{MLTT})}(n+1)$ we define

$$\mathrm{ft}[\Gamma, A] \coloneqq [\Gamma]$$

  by definition of a context being derivable. This is well-defined by definition of judgmental equality between contexts Definition 2.3.5.

  We also define

$$\mathrm{p}_{[\Gamma,A]} \coloneqq [\mathrm{wMor}_0(\mathrm{id}_n)] : [\Gamma, A] \to [\Gamma]$$

  by Lemma 2.3.7 and Lemma 2.3.10. This is well-defined since the expression $\mathrm{wMor}_0(\mathrm{id}_n)$ only depends on the length n.

- For an object $[\Gamma, A] \in \mathrm{Ob}_{\mathcal{C}(\mathrm{MLTT})}(n+1)$ and morphism $[\delta]$ with $[\mathrm{cod}(\delta)] = \mathrm{ft}[\Gamma, A]$ we define

$$[\delta]^*[\Gamma, A] \coloneqq [\mathrm{dom}(\delta), A[\delta]]$$

  by Lemma 2.3.8. This is well-defined by Corollary 2.3.9 and Lemma 2.3.13.

  We define a morphism

$$\mathrm{q}([\delta], [\Gamma, A]) \coloneqq [\mathrm{wMor}+(\delta)] : [\mathrm{dom}(\delta), A[\delta]] \to [\Gamma, A]$$

  by Lemma 2.3.7. This is well-defined by Lemma 2.3.16.

- Whenever we have a morphism $[\delta, t]$, it breaks down into a morphism $[\delta]$ with $\mathrm{dom}(\delta, t) = \mathrm{dom}(\delta)$, $\mathrm{cod}(\delta, t) = (\mathrm{cod}(\delta), A)$ and $\mathrm{dom}(\delta) \vdash t :$

$A[\delta]$ for some type expression A, by derivability of context morphisms Definition 2.3.5. Observe that

$$\mathrm{p}_{[\mathrm{cod}(\delta),A]} \circ [\delta, t] = [\mathrm{wMor}_0(\mathrm{id})] \circ [\delta, t] = [\mathrm{wMor}_0(\mathrm{id})[\delta, t]] = [\delta]$$

by Lemma 2.2.19 and Lemma 2.2.20. We define

$$\mathrm{s}_{[\delta,t]} = [\mathrm{id}, t] : [\mathrm{dom}(\delta)] \longrightarrow [\mathrm{dom}(\delta), A[\delta]] = [\delta]^*[\mathrm{cod}(\delta), A]$$

by Lemma 2.3.10. This is well-defined by Corollary 2.3.9.

Moreover, if $[\mathrm{cod}(\delta), A] = [\theta]^*[\Gamma, B]$ one can verify that

$$\mathrm{q}([\theta], [\Gamma, A]) \circ [\delta, t] = [\mathrm{wMor}_0(\theta)[\delta, t], t] = [\theta[\delta], t]$$

which implies that $\mathrm{s}_{[\delta,t]} = \mathrm{s}(\mathrm{q}([\theta], [\Gamma, A]) \circ [\delta, t])$, as its definition only depends on the final term of the morphism in question. □

The contextual category $\mathcal{C}(\mathrm{MLTT})$ can help sketch intuition for the various operations when working in an arbitrary contextual category.

Remark 3.1.8. It is important to note that up until this point the only rules from MLTT that have been used to construct the term model were structural. Indeed, the definition of a contextual category only captures this part of the system. The logical rules will be covered once we impose additional structure on a contextual category.

In addition, the definitions so far all go through even without quotienting by judgmental equality. This is due to the structural rules not introducing any non-trivial identifications. To the knowledge of the author, this fact has not been noted in the literature.

Moreover, for a system without any (judgemental) computation or η rules one should not need to take quotients when defining the logical structure of the term model. This could give rise to a different approach in setting up the term model. Because it is customary to take the quotients from the beginning we have not explored this yet. Since quotient types turn out to cause some complications in our formalization due to memory, this can be an argument for exploring this different approach in the future.

Finally, many authors have raised philosophical, proof-theoretical and computational issues with quotients. Our memory issue can be seen as yet another argument to have a foundational system without them. This could be achieved by a system without judgmental equality and all rules introducing non-trivial judgmental equalities replaced by appropriate terms of an identity type instead, see [Ber18] for an example in this direction.

3.2 Core structure

In this section we start by investigating the various structure and operations that can be defined from the core operation of a contextual category. These will be useful later on, once we try to interpret syntax. To help with intuition we will sketch these in the special case of $\mathcal{C}(\mathrm{MLTT})$. Some results are given here in a more general form than in the formalization, where they are shown only in the cases required for the main result.

Let us start by noting that we can iterate the basic operations in the following way:

Definition 3.2.1. Let $\mathcal{C}$ be a contextual category and $k, n \in \mathbb{N}$. If $X \in \mathrm{Ob}_{\mathcal{C}}(n+k)$, We denote by $\mathrm{ft}^k(X) \in \mathrm{Ob}_{\mathcal{C}}(n)$ and $\mathrm{p}^k_X : X \to \mathrm{ft}^k(X)$ the object and morphism defined inductively by the rules

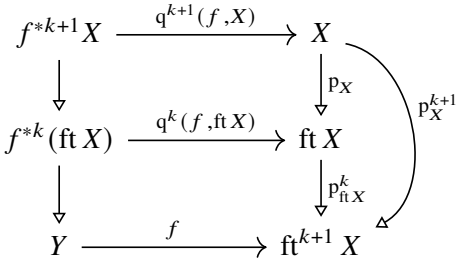
$$\mathrm{ft}^0 X \coloneqq X \qquad \mathrm{p}^0_X \coloneqq \mathrm{id}_X$$
$$\mathrm{ft}^{k+1} X \coloneqq \mathrm{ft}^k(\mathrm{ft}\, X) \qquad \mathrm{p}^{k+1}_X \coloneqq \mathrm{p}^k_{\mathrm{ft}\, X} \circ \mathrm{p}_X\,.$$

We will extend the use of $\twoheadrightarrow$ in diagrams to represent morphisms of the form p^k_X.

If moreover $Y \in \mathrm{Ob}_{\mathcal{C}}(m)$ and $f : Y \to \mathrm{ft}^k X$, we denote by $f^{*k}X \in \mathrm{Ob}_{\mathcal{C}}(m+k)$ and $\mathrm{q}^k(f, X) : f^{*k}X \to X$ the object and morphism defined inductively by the rules

$$f^{*0}X \coloneqq Y \qquad \mathrm{q}^0(f, X) \coloneqq f$$
$$f^{*k+1}X \coloneqq \mathrm{q}^k(f, \mathrm{ft}\, X)^*(X) \qquad \mathrm{q}^{k+1}(f, X) \coloneqq \mathrm{q}(\mathrm{q}^k(f, \mathrm{ft}\, X), X).$$

One can prove by induction that $\mathrm{ft}^k(f^{*k}) = Y$ as well as the initial claim that $\mathrm{q}^k(f, X) : f^{*k}(X) \to X$. In particular $\mathrm{ft}(f^{k+1}(X)) = f^{*k}(\mathrm{ft}\, X)$. All of the above fits in the following commutative diagram



which reduces to the 'one-step' diagram in Definition 3.1.5 for $k = 0$.

Example 3.2.2. In $\mathcal{C}(\mathrm{MLTT})$, this corresponds with repeating the operations we had defined on them:

$$\mathrm{ft}^k[\Gamma, A_0, \ldots, A_{k-1}] = [\Gamma]$$

$$\mathrm{p}^k([\Gamma, A_0, \dots, A_{k-1}] = [\mathrm{wMor}_0^k(\mathrm{id}_n)]$$
$$[\delta]^{*k}[\Gamma, A_0, \dots, A_{k-1}] = [\mathrm{dom}(\delta), A_0[\delta], \dots, A_{k-1}[\mathrm{wCmor}+^{k-1}(\delta)]]$$
$$\mathrm{q}^k([\delta], [\Gamma, A_0, \dots, A_{k-1}]) = [\mathrm{wCmor}+^k(\delta)]$$

Next we define a special class of morphism.

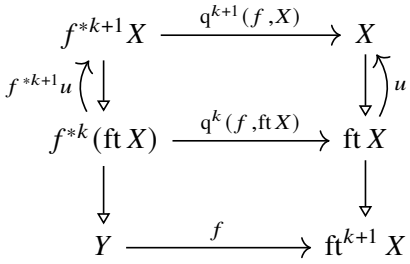
Definition 3.2.3. For $X \in \mathrm{Ob}_{\mathcal{C}}(n+1)$, a morphism $u : \mathrm{ft}(X) \to X$ is called a *term morphism* if $\mathrm{p}_X \circ u = \mathrm{id}_{\mathrm{ft}\, X}$.

Example 3.2.4. In $\mathcal{C}(\mathrm{MLTT})$ there is a one-to-one correspondence between term morphism and terms (up to judgmental equality). Given an object $[\Gamma]$ and a term expression t such that $\Gamma \vdash t : A$ for some type expression A, we define a morphism $[((\mathrm{id}, t), \Gamma, (\Gamma, A))]$ by Lemma 2.3.10 and Lemma 2.2.20. Observe that $\mathrm{p}_{[\Gamma, A]} \circ [\mathrm{id}, t] = \mathrm{id}_{[\Gamma]}$. On the other hand, given a term morphism $[\delta, t]$ in $\mathcal{C}(\mathrm{MLTT})$ it breaks down into a morphism $[\delta]$ satisfying $\mathrm{dom}(\delta, t) = \mathrm{dom}(\delta)$ and $\mathrm{cod}(\delta, t) = (\mathrm{cod}(\delta), A)$ for some type expression A. From it being a term morphism we can deduce $\mathrm{dom}(\delta) \vdash \mathrm{id} \equiv \delta : \mathrm{cod}(\delta)$ and thus $\mathrm{dom}(\delta) \vdash t : A$ by Lemma 2.3.8 and Lemma 2.2.20.

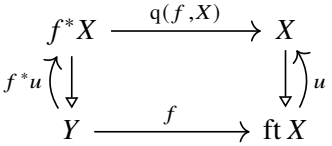
So, without loss of generality we can assume any term morphism in $\mathcal{C}(\mathrm{MLTT})$ to be of the form $[\mathrm{id}, t]$.

Note that we have already encountered an example of a term morphism, namely s_f for $f : Y \to X$ with $X \in \mathrm{Ob}_{\mathcal{C}}(n+1)$. In [Voe16, Section 3], the notation $\widetilde{\mathrm{Ob}}$ is used for the collection of all term morphisms. It is also not uncommon to refer to these morphisms simply as *sections*. We will define a $*$-operation on term morphisms which will model the substitution in terms.

Definition 3.2.5. For $k, n \in \mathbb{N}$, $X \in \mathrm{Ob}_{\mathcal{C}}(n+k+1)$, $Y \in \mathrm{Ob}_{\mathcal{C}}(m)$, $f : Y \to \mathrm{ft}^{k+1} X$ and a term morphism $u : \mathrm{ft}\, X \to X$ we define the term morphism $f^{*k+1}u : f^{*k}(\mathrm{ft}\, X) \to f^{*k+1}X$ by $\mathrm{s}(u \circ \mathrm{q}^k(f, \mathrm{ft}\, X))$. This fits in the previous diagram as



which reduces to



in the case $k = 0$. Observe that indeed

$$\mathrm{cod}(f^{*k+1}u) = (\mathrm{p}_X \circ u \circ \mathrm{q}^k(f, \mathrm{ft}\,X))^*X = (\mathrm{q}^k(f, \mathrm{ft}\,X))^*X = f^{*k+1}X,$$

as u is a term morphism.

Example 3.2.6. In $\mathcal{C}(\mathrm{MLTT})$ this operation becomes

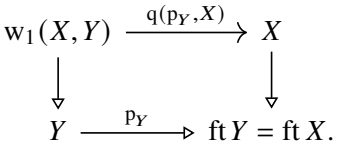
$$[\delta]^*[\mathrm{id}_n, t] = [\mathrm{id}_m, t[\delta]]$$
$$[\delta]^{*k+1}[\mathrm{id}_{n+k}, t] = [\mathrm{id}_{m+k}, t[\mathrm{wMor}+^k(\delta)]],$$

justifying why this represents 'substitution in terms'.

We will identify a few special cases of these iterated $*$-operations.

Definition 3.2.7. We introduce notation for the following special cases of the $*$-operation.

- **Weakening of objects:** Consider the case where $k \in \mathbb{N}$, $X \in \mathrm{Ob}_{\mathcal{C}}(n+k)$ and $Y \in \mathrm{Ob}_{\mathcal{C}}(n+1)$. If moreover $\mathrm{ft}\,Y = \mathrm{ft}^k X$, we write $\mathrm{w}_k(X, Y)$ for $\mathrm{p}_Y^{*k} X \in \mathrm{Ob}_{\mathcal{C}}(n+k+1)$. For $k = 0$ this can be though of as Y representing a context extension (X, B) of X by an arbitrary type B. For $k = 1$, we have the diagram



  In $\mathcal{C}(\mathrm{MLTT})$, this operations computes as

  $$\mathrm{w}_k([\Gamma, A_0, \ldots, A_{k-1}], [\Gamma, B]) = [\Gamma, B, \mathrm{wTy}_0(A_0), \ldots, \mathrm{wTy}_{k-1}(A_{k-1})],$$

  using Lemma 2.2.25. Notice the close resemblance with the definition of $\mathrm{wCtx}_k(\Gamma, B)$.

  This case is labeled by $T(Y, X)$ in [Voe16, Section 3]. In [Hof97, Section 2.4.1.2] the notation $X+$ is used for the variant of $\mathrm{w}_0(X, Y)$ in the realm of categories with attributes.

- **Weakening of term morphisms:** We extend the above to weakening of term morphisms. Let $k \in \mathbb{N}$, $X \in \mathrm{Ob}_{\mathcal{C}}(n+k+1)$ and $Y \in \mathrm{Ob}_{\mathcal{C}}(n+1)$ such that $\mathrm{ft}^{k+1} X = \mathrm{ft}\, Y$. If $t : \mathrm{ft}\, X \to X$ is a term morphism we write $\mathrm{w}_{k+1}(t, Y)$ for the morphism $\mathrm{p}_Y^{*k+1} t : \mathrm{w}_k(\mathrm{ft}\, X, Y) \to \mathrm{w}_{k+1}(X, Y)$. In $\mathcal{C}(\mathrm{MLTT})$ this becomes

$$\mathrm{w}_{k+1}([\mathrm{id}_{n+k}, t]) = [id_{n+k+1}, \mathrm{wTm}_k(t)].$$

  In [Voe16, Section 3] the notation $\widetilde{T}(Y, r)$ is used for this. In [Hof97, Section 2.4.1.2] again the notation $t+$ is used for $\mathrm{w}_0(t, Y)$.

- **Term substitution in objects:** Consider now the case where $k \in \mathbb{N}$, $X \in \mathrm{Ob}_{\mathcal{C}}(n+1+k)$ and we have a term morphism $u : \mathrm{ft}(\mathrm{ft}^k X) \to \mathrm{ft}^k X$. We will write $X[u]$ for the object $u^{*k} X \in \mathrm{Ob}_{\mathcal{C}}(n+k)$. For $k = 0$, we simply have $X[u] = \mathrm{ft}\, X$. For $k = 1$, we get the diagram

$$\begin{array}{ccc} X[u] & \xrightarrow{\mathrm{q}(u,X)} & X \\ \downarrow & & \downarrow \\ \mathrm{ft}(\mathrm{ft}\, X) & \xrightarrow{u} & \mathrm{ft}\, X. \end{array}$$

  In $\mathcal{C}(\mathrm{MLTT})$ this operation computes to

$$[\Gamma, A, B_0, \ldots, B_{k-1}][u] = [\Gamma, B_0[u], \ldots, B_{k-1}[\mathrm{wMor}+^{k-1}(u)]]$$

  which justifies its notation.

  This case is labeled by $S(s, X)$ in [Voe16, Section 3].

- **Term substitution in term morphisms:** Again, we extend term substitution to term morphisms. Let $k \in \mathbb{N}$, $X \in \mathrm{Ob}_{\mathcal{C}}(n+1+k+1)$ and $u : \mathrm{ft}(\mathrm{ft}^{k+1} X) \to \mathrm{ft}^{k+1} X$ a term morphism. If $t : \mathrm{ft}\, X \to X$ is another term morphism we write $t[u]$ for the morphism $u^{*k+1} t : (\mathrm{ft}\, X)[u] \to X[u]$. In $\mathcal{C}(\mathrm{MLTT})$ we find

$$[\mathrm{id}_{n+1+k}, t][u] = [\mathrm{id}_{n+k}, t[\mathrm{wMor}+^{k}(u)]].$$

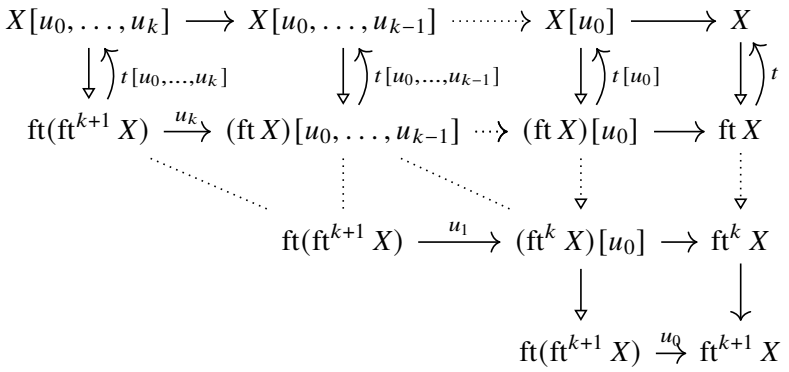
  In [Voe16, Section 3], this case is denoted by $\widetilde{S}(s, r)$.

- **Multiple term substitutions in a row:** The above only describes the substitution of one single term morphism, which can be iterated in the following way. If substitution up to k terms has been defined already and we have a sequence of term morphism $u_0, \ldots u_k$ such that $u_i : \mathrm{ft}(\mathrm{ft}^{k+1} X) \to (\mathrm{ft}^{k-i+1} X)[u_0, \ldots, u_{i-1}]$, we define

$$X[u_0, \ldots, u_k] \coloneqq (\ldots (X[u_0])[u_1] \ldots)[u_k],$$

$$t[u_0, \ldots, u_k] \coloneqq (\ldots(t[u_0])[u_1]\ldots)[u_k].$$

The following diagram illustrates this repeating process:



- **Variables:** We identity a special class of term morphisms that will represent variables. For $X \in \mathrm{Ob}_{\mathcal{C}}(n)$ and $l < n$, we first write $\mathrm{Ty}_l(X) \in \mathrm{Ob}_{\mathcal{C}}(n+1)$ for the object defined inductively by

$$\begin{aligned} \mathrm{Ty}_0(X) &\coloneqq \mathrm{w}_1(X, X) \\ \mathrm{Ty}_{l+1}(X) &\coloneqq \mathrm{w}_1(\mathrm{Ty}_l(\mathrm{ft}\, X), X). \end{aligned}$$

Observe that $\mathrm{ft}(\mathrm{Ty}_l(X)) = X$. We then write $\mathbf{x}_l(X)$ for the term morphism

$$\mathrm{s}_{\mathrm{p}^l_X} : X \to (\mathrm{p}_{\mathrm{ft}^l X} \circ \mathrm{p}^l_X)^*(\mathrm{ft}^l X).$$

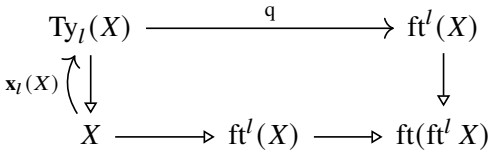
We show

$$\mathrm{Ty}_l(X) = (\mathrm{p}_{\mathrm{ft}^l X} \circ \mathrm{p}^l_X)^*(\mathrm{ft}^l X),$$

by structural induction on l. The case $l = 0$ follows by definition $\mathrm{w}_1(X, X) = \mathrm{p}^*_X(X)$. Suppose it holds for l. We find by expanding the definitions

$$\begin{aligned} \mathrm{Ty}_{l+1}(X) &= \mathrm{w}_1(\mathrm{Ty}_l(\mathrm{ft}\, X), X) \\ &= \mathrm{w}_1((\mathrm{p}_{\mathrm{ft}^l(\mathrm{ft}\, X)} \circ \mathrm{p}^l_{\mathrm{ft}\, X})^*(\mathrm{ft}^l(\mathrm{ft}\, X))), X) \\ &= \mathrm{p}^*_X((\mathrm{p}_{\mathrm{ft}^{l+1} X} \circ \mathrm{p}^l_{\mathrm{ft}\, X})^*(\mathrm{ft}^{l+1} X)) \\ &= (\mathrm{p}_{\mathrm{ft}^{l+1} X} \circ \mathrm{p}^l_{\mathrm{ft}\, X} \circ \mathrm{p}_X)^*(\mathrm{ft}^{l+1} X) \\ &= (\mathrm{p}_{\mathrm{ft}^{l+1} X} \circ \mathrm{p}^{l+1}_X)^*(\mathrm{ft}^{l+1} X) \end{aligned}$$

as required. The following diagram gives an illustration of these definitions



In $\mathcal{C}$(MLTT) one can show that

$$\mathrm{Ty}_l([\Gamma]) = [\Gamma, \Gamma_l],$$
$$\mathbf{x}_l([\Gamma]) = [\mathrm{id}_n, \mathbf{x}_l]$$

as expected.

In [Voe16, Section 3], $\mathbf{x}_0(X)$ is considered and denoted by $\delta(X)$. This notations stems from the observation that $\mathbf{x}_0(X)$ defines a *diagonal* map $X \to \mathrm{w}_1(X, X)$.

Semantic equalities

From the definitions of the previous section, we observe that we are able to define all the semantic counterparts of the operations we encountered in syntax in terms of semantic substitution. This is one of the benefits of the semantic side, however we do need to show that these semantic operations behave the same way as the syntactic ones. For instance, we need $\mathrm{w}_1(\mathbf{x}_0(X), B) = \mathbf{x}_1(\mathrm{w}_0(B, X))$ as morphisms in a arbitrary contextual category. Syntactically, this follows readily by definition of weakening: $\mathrm{wTm}_0(\mathbf{x}_0) = \mathbf{x}_1$.

In this section we will discuss various of these semantic equalities. We will follow a similar approach as for the syntactic equalities and refer to the formalization for the details of the proofs.

We start by showing that semantically, weakening and total substitution commute.

Lemma 3.2.8. *For $k \in \mathbb{N}$, $X \in \mathrm{Ob}_{\mathcal{C}}(n + k)$, $Y \in \mathrm{Ob}_{\mathcal{C}}(n + 1)$ such that* $\mathrm{ft}^k X = \mathrm{ft}\, Y$ *and $g : Z \to \mathrm{ft}^k X$ we have*

$$g^{*k+1}(\mathrm{w}_k(X, A)) = \mathrm{w}_k(g^{*k} X, g^* A).$$

Proof. We do not prove the lemma directly. Instead, by a straightforward induction on k one shows that the composition

$$\mathrm{q}^k(\mathrm{p}_Y, X) \circ \mathrm{q}^{k+1}(g, \mathrm{w}_k(X, Y)) : g^{*k+1}(\mathrm{w}_k(X, Y)) \to \mathrm{w}_k(X, A) \to X$$

is equal to the composition

$$\mathrm{q}^k(g, X) \circ \mathrm{q}^k(\mathrm{p}_{g^* A}, g^{*k} X) : \mathrm{w}_k(g^{*k} X, g^* Y) \to g^{*k} X \to X,$$

which implies their sources must be equal as well. □

Similarly, total substitution and term substitution commute.

Lemma 3.2.9. *For $X \in \mathrm{Ob}_{\mathcal{C}}(n+1+k)$, $u : \mathrm{ft}(\mathrm{ft}^k X) \to \mathrm{ft}^k X$ and $f : Y \to \mathrm{ft}(\mathrm{ft}^k X)$ we have*

$$f^{*k}(X[u]) = (\mathrm{q}(f, \mathrm{ft}^k X)^{*k} X)[f^*(u)].$$

Proof. Again, instead one shows by a straightforward induction on k one shows that the composition

$$\mathrm{q}^k(u, X) \circ \mathrm{q}^k(f, X[u]) : f^{*k}(X[u]) \to X[u] \to X$$

is equal to the composition

$$\begin{aligned}\mathrm{q}^k(\mathrm{q}(f, \mathrm{ft}^k X), X) \circ \mathrm{q}^k(f^* u, \mathrm{q}(f, \mathrm{ft}^k X)^{*k} X) :& \\ (\mathrm{q}(f, \mathrm{ft}^k X)^{*k} X)[f^* u] \to (\mathrm{q}(f, \mathrm{ft}^k X))^{*k} X \to X,&\end{aligned}$$

which gives the desired result. As an illustration, the base case is showing that the diagram

$$\begin{array}{ccc} f^* X & \xrightarrow{\mathrm{q}(f,X)} & X \\ {\scriptstyle f^*u}\uparrow & & \uparrow{\scriptstyle u} \\ Y & \xrightarrow{f} & \mathrm{ft}\, X \end{array}$$

commutes, which follows from

$$\begin{aligned}\mathrm{q}(f, X) \circ f^* u &= \mathrm{q}(p_X \circ u \circ f, X) \circ \mathrm{s}_{u \circ f} \\ &= u \circ f\end{aligned}$$

since u is a term morphism and the commuting diagram for s in Definition 3.1.5. □

For semantic equalities involving term morphisms, one often uses the following properties.

Lemma 3.2.10. *For appropriate morphism f and g we have*

$$\mathrm{s}(\mathrm{s}_g \circ f) = \mathrm{s}(g \circ f).$$

Proof. This follows immediately by

$$\mathrm{s}(\mathrm{s}_g \circ f) = \mathrm{s}(\mathrm{q}(\mathrm{p}_X \circ g, X) \circ \mathrm{s}_g \circ f) = \mathrm{s}(g \circ f).$$

using the basic properties of s. □

Lemma 3.2.11. *For a term morphism $u : \mathrm{ft}\, X \to X$ we have*

$$\mathrm{s}_u = u$$

Proof. The calculation

$$u = \mathrm{q}(\mathrm{p}_X \circ u, X) \circ \mathrm{s}_u = \mathrm{q}(\mathrm{id}_{\mathrm{ft}\, X}) \circ \mathrm{s}_u = \mathrm{id}_X \circ \mathrm{s}_u = \mathrm{s}_u$$

gives the desired result. □

Corollary 3.2.12. *Given a term morphism $u : \mathrm{ft}(X) \to X$ and appropriate morphisms f and g we have*

$$(g \circ f)^* u = f^*(g^* u),$$
$$(\mathrm{id})^* u = u.$$

□

The following shows that substitution in semantic variables behaves the same as the syntactic definition.

Lemma 3.2.13. *For $X \in \mathrm{Ob}_{\mathcal{C}}(n)$, $l \in \mathrm{Fin}(n)$, $Y \in \mathrm{Ob}_{\mathcal{C}}(m)$ and $g : Y \to X$ we have*

$$g^*(\mathbf{x}_0(X)) = \mathrm{s}_g$$
$$g^*(\mathbf{x}_{l+1}(X)) = (\mathrm{p}_X \circ g)^*(\mathbf{x}_l(\mathrm{ft}\, X))$$

Proof. Writing out the definitions and using Lemma 3.2.10. □

Remark 3.2.14. Observe the special case for a term morphism u is a term morphism

$$(\mathbf{x}_0(X))[u] = \mathrm{s}_u = u$$
$$(\mathbf{x}_{l+1}(X))[u] = (\mathrm{p}_X \circ u)^*(\mathbf{x}_l(\mathrm{ft}\, X)) = (\mathrm{id})^*(\mathbf{x}_l(\mathrm{ft}\, X)) = \mathbf{x}_l(\mathrm{ft}\, X),$$

which can be read as saying the intuition of Example 3.2.4 holds in any contextual category.

In the same vein, weakening of semantic variables behaves just like the definition in syntax.

Lemma 3.2.15. *For $k \in \mathbb{N}$, $l \in \mathrm{Fin}(n+k)$, $X \in \mathrm{Ob}_{\mathcal{C}}(n+k)$, $Y \in \mathrm{Ob}_{\mathcal{C}}(n+1)$ with $\mathrm{ft}(Y) = \mathrm{ft}^k(X)$ we have*

$$\mathrm{w}_{k+1}(\mathbf{x}_l(X), Y) = \mathbf{x}_{\mathrm{wVar}_k(l)}(\mathrm{w}_k(X, Y))$$

Proof. One start with an induction on k. The base case is straightforward. For $k+1$ one starts an additional induction on l. The base case is again straightforward, and the case $l+1$ uses both the general induction hypotheses as well as the case $k = 0$. Lemma 3.2.10 is used throughout. □

Inspecting the target of each of the above two morphisms we find:

Corollary 3.2.16. *For $k \in \mathbb{N}$, $l \in \mathrm{Fin}(n+k)$, $X \in \mathrm{Ob}_{\mathcal{C}}(n+k)$, $Y \in \mathrm{Ob}_{\mathcal{C}}(n+1)$ with $\mathrm{ft}\, Y = \mathrm{ft}^k X$ we have*

$$\mathrm{w}_{k+1}(\mathrm{Ty}_l(X), Y) = \mathrm{Ty}_{\mathrm{wVar}_k(l)}(\mathrm{w}_k(X, Y))$$

Next we observe that the $*$-operation can be understood as appending the morphism by variables as is the case in $\mathcal{C}(\mathrm{MLTT})$.

Lemma 3.2.17. *For $k \in \mathbb{N}$, $l \in \mathrm{Fin}(k)$, $X \in \mathrm{Ob}_{\mathcal{C}}(n+k)$, $Y \in \mathrm{Ob}_{\mathcal{C}}(m)$ and $g : Y \to \mathrm{ft}^k X$ we have*

$$g^{*k+1}(\mathbf{x}_l(X)) = \mathbf{x}_l(g^* X).$$

Proof. By an induction on k and essentially the same techniques as the previous lemmas. □

Again, inspecting the target of the above two morphism gives us:

Corollary 3.2.18. *For $k \in \mathbb{N}$, $l \in \mathrm{Fin}(k)$, $X \in \mathrm{Ob}_{\mathcal{C}}(n+k)$, $Y \in \mathrm{Ob}_{\mathcal{C}}(m)$ and $g : Y \to \mathrm{ft}^k(X)$ we have*

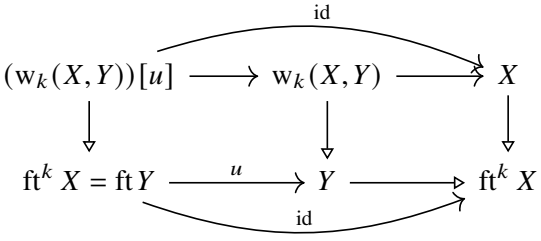
$$g^{*k+1}(\mathrm{Ty}_l(X)) = \mathrm{Ty}_l(g^{*k}(X)).$$

What follows is the interaction between semantic term substitution and weakening. We have seen this before in syntax by Lemma 2.2.19.

Lemma 3.2.19. *For $k \in \mathbb{N}$, $X \in \mathrm{Ob}_{\mathcal{C}}(n+k)$, $Y \in \mathrm{Ob}_{\mathcal{C}}(n+1)$ with $\mathrm{ft}(Y) = \mathrm{ft}^k(X)$ and term morphism $u : \mathrm{ft}(Y) \to Y$ we have*

$$(\mathrm{w}_k(X, Y))[u] = X$$

Proof. By induction on k. The main idea is sketched in the following commutative diagram



using that u is a term morphism. □

Lemma 3.2.20. *For $k \in \mathbb{N}$, $X \in \mathrm{Ob}_{\mathcal{C}}(n+k+1)$, $Y \in \mathrm{Ob}_{\mathcal{C}}(n+1)$ with $\mathrm{ft}\, Y = \mathrm{ft}^{k+1} X$ and term morphisms $t : \mathrm{ft}\, X \to X$ and $u : \mathrm{ft}\, Y \to Y$ we have*

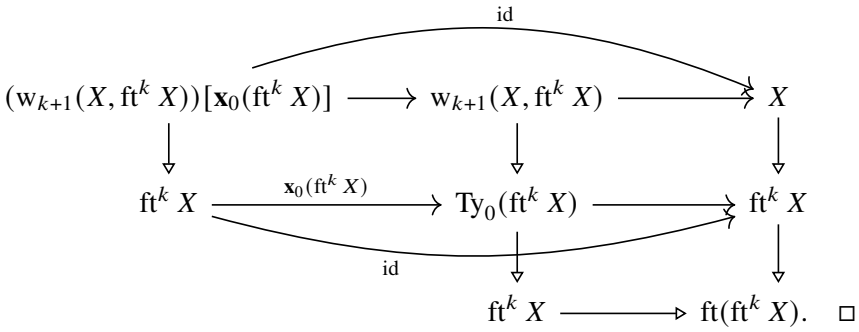
$$(\mathrm{w}_{k+1}(t, Y))[u] = t. \qquad \square$$

Finally, there is a special case for substitution by the last variable. This is due to the observation that $\mathbf{x}_0(X)$ not only computes to the identity after post composition with p but also with q by the commuting diagram in Definition 3.1.5.

Lemma 3.2.21. *For $k \in \mathbb{N}$ and $X \in \mathrm{Ob}_{\mathcal{C}}(n+1+k)$ we have*

$$(\mathrm{w}_{k+1}(X, \mathrm{ft}^k X))[\mathbf{x}_0(\mathrm{ft}^k X)] = X.$$

Proof. Straightforward induction on k. We sketch the main idea in a commutative diagram



□

3.3 Additional structure from logical rules

We will now proceed to translate all the type and term constructors as well as their logical rules into the realm of contextual categories. This is a very mechanical process and similar translations can for instance be found in [KL20] and [Hof97]. For every constructor we introduce an analogous structure on a contextual category. The properties this structure has are then decided by the logical rules that involve the particular constructor. Additionally, we need to include that the structure is stable under the $*$-operation. This ties in with substitution being a more primitive operation in semantics and not inductively defined as was the case in syntax. On the other hand, congruence rules do not have to be added, as we get them for free from the strictness of our category.

For this purpose we will introduce some additional notation that will improve readability. This is similar to the convention in [KL20]. Given an object

$X \in \mathrm{Ob}_{\mathcal{C}}(n)$, we mean by 'an object $(X.A)$' an object $A \in \mathrm{Ob}_{\mathcal{C}}(n+1)$ such that $\mathrm{ft}\, A = X$. We will write $(X.A_0.\cdots.A_k)$ instead of $((\cdots(X.A_0).A_1\cdots).A_k)$. We may write $(X.A) \in \mathrm{Ob}_{\mathcal{C}}(n+1)$ on its own without explicitly introducing $X \in \mathrm{Ob}_{\mathcal{C}}(n)$ first.

The translation procedure consist roughly of the following procedure. Given a rule, we equip a contextual with an operation taking, as input, objects and term morphisms corresponding with the premises of the rule and producing as output an object, term morphism, or equality corresponding to the conclusion. If a premise or the conclusion involves earlier defined structures, then the contextual category must be assumed to come equipped with it. This last part will appear less prevalent in our write-up, as we have bundled together various related structures. It is more present in our formalization.

Our translation will differ in one aspect compared to say [KL20] or [Hof97]. We have chosen to stick completely to the presentation of the rules of the system when defining the structure. As an example, consider the **suc** -INTRO rule

$$\frac{\Gamma \vdash u : \mathbf{N}}{\Gamma \vdash \mathbf{suc}(u) : \mathbf{N}}.$$

The corresponding structure in [Hof97, Definition 2.4.20] is a morphism

$$\mathrm{sucStr}(X) : \mathrm{NatStr}(X) \to \mathrm{NatStr}(X)$$

while in our setup the structure corresponds to a *term* morphism

$$\mathrm{sucStr}(X, u) : X \to \mathrm{NatStr}(X).$$

The form we use is slightly more verbose, but gives an entirely uniform translation for all rules. Due to this difference we have decided to include all structure explicitly instead of writing a heuristic definition. This should also help a reader planning to extend our system.

Throughout this section, let $\mathcal{C}$ be a contextual category.

Definition 3.3.1. A *Empty-type structure* on $\mathcal{C}$ consists of

1. For each object $X \in \mathrm{Ob}_{\mathcal{C}}(n)$ an object $(X.\mathrm{EmptyStr}(X))$, i.e. an object $\mathrm{EmptyStr}(X) \in \mathrm{Ob}_{\mathcal{C}}(n+1)$ such that $\mathrm{ft}(\mathrm{EmptyStr}(X)) = X$.

2. For each object $(X.\mathrm{EmptyStr}(X).P) \in \mathrm{Ob}_{\mathcal{C}}(n+2)$ and term morphism $u : X \to \mathrm{EmptyStr}(X)$ a term morphism

$$\mathrm{empty\_elimStr}(X, P, u) : X \to P[u].$$

3. Such that for each $g : Y \to X$ and the appropriate arguments

$$g^*(\mathrm{EmptyStr}(X)) = \mathrm{EmptyStr}(Y),$$
$$g^*(\mathrm{empty\_elimStr}(X, P, u)) = \mathrm{empty\_elimStr}(Y, g^{*2}(P), g^*(u))$$

Remark 3.3.2.

- Instead of Y we could also write $g^{*0}(X)$.
- Observe in the last equality that indeed

$$\mathrm{ft}(g^{*2}(P)) = g^*(\mathrm{ft}\, P) = g^*(\mathrm{EmptyStr}(X)) = \mathrm{EmptyStr}(Y)$$
$$g^*u : Y \to g^*(\mathrm{EmptyStr}(X)) = \mathrm{EmptyStr}(Y).$$

  which justifies the notation. We leave it up to the reader to check these kinds of justifications for themselves if necessary.

Definition 3.3.3. A *Unit-type structure* on $\mathcal{C}$ consists of

1. For each object $X \in \mathrm{Ob}_{\mathcal{C}}(n)$ an object $(X.\,\mathrm{UnitStr}(X))$.
2. For each object $X \in \mathrm{Ob}_{\mathcal{C}}(n)$ a term morphism

$$\star\text{-Str}(X) : X \to \mathrm{UnitStr}(X).$$

3. For each object $(X.\,\mathrm{UnitStr}(X).P) \in \mathrm{Ob}_{\mathcal{C}}(n+2)$, term morphism $d_\star : X \to P[\star\text{-Str}(X)]$ and term morphism $u : X \to \mathrm{UnitStr}(X)$ a term morphism

$$\mathrm{unit\_elimStr}(X, P, d_\star, u) : X \to P[u]$$

   satisfying

$$\mathrm{unit\_elimStr}(X, P, d_\star, \star\text{-Str}(X)) = d_\star.$$

4. Such that for each $g : Y \to X$ and the appropriate arguments

$$g^*(\mathrm{UnitStr}(X)) = \mathrm{UnitStr}(Y),$$
$$g^*(\star\text{-Str}(X)) = \star\text{-Str}(Y),$$
$$g^*(\mathrm{unit\_elimStr}(X, P, d_\star, u)) = \mathrm{unit\_elimStr}(Y, g^{*2}(P), g^*(d_\star), g^*(u)).$$

Definition 3.3.4. A *Nat-type structure* on $\mathcal{C}$ consists of

1. For each object $X \in \mathrm{Ob}_{\mathcal{C}}(n)$ an object $(X.\,\mathrm{NatStr}(X))$.
2. For each object $X \in \mathrm{Ob}_{\mathcal{C}}(n)$ a term morphism

$$\mathrm{zeroStr}(X) : X \to \mathrm{NatStr}(X).$$

3. For each object $X \in \mathrm{Ob}_{\mathcal{C}}(n)$ and term morphism $u : X \to \mathrm{NatStr}(X)$ a term morphism

$$\mathrm{sucStr}(X, u) : X \to \mathrm{NatStr}(X).$$

4. For each

   - object $(X.\,\mathrm{NatStr}(X).P) \in \mathrm{Ob}_{\mathcal{C}}(n+2)$,
   - term morphism $d_{\mathbf{zero}} : X \to P[\mathrm{zeroStr}(X)]$,
   - term morphism $d_{\mathbf{suc}} : P \to (\mathrm{w}_2(\mathrm{w}_2(P, \mathrm{NatStr}(X)), P))[\mathrm{sucStr}(P, \mathbf{x}_1(P))]$,
   - term morphism $u : X \to \mathrm{NatStr}(X)$,

   a term morphism

$$\mathrm{indStr}(X, P, d_{\mathbf{zero}}, d_{\mathbf{suc}}, u) : X \to P[u]$$

   satisfying

$$\begin{aligned}\mathrm{indStr}(X, P, d_{\mathbf{zero}}, d_{\mathbf{suc}}, \mathrm{zeroStr}(X)) &= d_{\mathbf{zero}},\\ \mathrm{indStr}(X, P, d_{\mathbf{zero}}, d_{\mathbf{suc}}, \mathrm{sucStr}(X, u)) &= d_{\mathbf{suc}}[u, \mathrm{indStr}(X, P, d_{\mathbf{zero}}, d_{\mathbf{suc}}, u)].\end{aligned}$$

5. Such that for each $g : Y \to X$ and the appropriate arguments

$$\begin{aligned}g^*(\mathrm{NatStr}(X)) &= \mathrm{NatStr}(Y),\\ g^*(\mathrm{zeroStr}(X)) &= \mathrm{zeroStr}(Y),\\ g^*(\mathrm{sucStr}(X, u)) &= \mathrm{sucStr}(Y, g^*(u)),\\ g^*(\mathrm{indStr}(X, P, d_{\mathbf{zero}}, d_{\mathbf{suc}}, u)) &= \mathrm{indStr}(Y, g^{*3}(P), g^*(d_{\mathbf{zero}}), g^{*2}(d_{\mathbf{suc}}), g^*(u)).\end{aligned}$$

Definition 3.3.5. A *Sum-type structure* on $\mathcal{C}$ consists of

1. For each pair of objects $(X.A), (X.B) \in \mathrm{Ob}_{\mathcal{C}}(n+1)$ an object $(X.\,\mathrm{SumStr}(X, A, B))$.

2. For each pair of objects $(X.A), (X.B) \in \mathrm{Ob}_{\mathcal{C}}(n+1)$ and term morphism $a : X \to A$ a term morphism

$$\mathrm{inlStr}(X, A, B, a) : X \to \mathrm{SumStr}(X, A, B).$$

3. For each pair of objects $(X.A), (X.B) \in \mathrm{Ob}_{\mathcal{C}}(n+1)$ and term morphism $b : X \to B$ a term morphism

$$\mathrm{inrStr}(X, A, B, b) : X \to \mathrm{SumStr}(X, A, B).$$

4. For each

   - pair of objects $(X.A), (X.B) \in \mathrm{Ob}_{\mathcal{C}}(n+1)$,
   - object $(X.\,\mathrm{SumStr}(X,A,B).P) \in \mathrm{Ob}_{\mathcal{C}}(n+2)$,
   - term morphism $d_{\mathbf{inl}} : A \to (\mathrm{w}_2(P,A))[\mathrm{inlStr}(\mathrm{w}_1(A,A), \mathrm{w}_1(B,A), \mathbf{x}_0(A))]$,
   - term morphism $d_{\mathbf{inr}} : A \to (\mathrm{w}_2(P,B))[\mathrm{inrStr}(\mathrm{w}_1(A,B), \mathrm{w}_1(B,B), \mathbf{x}_0(B))]$,
   - term morphism $u : X \to \mathrm{SumStr}(X,A,B)$,

   a term morphism

   $$\mathrm{matchStr}(X,A,B,P,d_{\mathbf{inl}},d_{\mathbf{inr}},u) : X \to P[u]$$

   satisfying

   $$\mathrm{matchStr}(X,A,B,P,d_{\mathbf{inl}},d_{\mathbf{inr}},\mathrm{inlStr}(X,A,B,a)) = d_{\mathbf{inl}}[a],$$
   $$\mathrm{matchStr}(X,A,B,P,d_{\mathbf{inl}},d_{\mathbf{inr}},\mathrm{inrStr}(X,A,B,b)) = d_{\mathbf{inr}}[b].$$

5. Such that for each $g : Y \to X$ and the appropriate arguments

   $$g^*(\mathrm{SumStr}(X,A,B)) = \mathrm{SumStr}(Y, g^*(A), g^*(B)),$$
   $$g^*(\mathrm{inlStr}(X,A,B,a)) = \mathrm{inlStr}(Y, g^*(A), g^*(B), g^*(a)),$$
   $$g^*(\mathrm{inrStr}(X,A,B,b)) = \mathrm{inrStr}(Y, g^*(A), g^*(B), g^*(a)),$$
   $$g^*(\mathrm{matchStr}(X,A,B,P,d_{\mathbf{inl}},d_{\mathbf{inr}},u)) =$$
   $$\mathrm{matchStr}(Y, g^*(A), g^*(B), g^{*2}(P), g^{*2}(d_{\mathbf{inl}}), g^{*2}(d_{\mathbf{inr}}), g^*(u)).$$

6. And satisfies η-expansion if for the appropriate arguments

   $$\mathrm{matchStr}(X,A,B,\mathrm{w}_1(\mathrm{SumStr}(X,A,B), \mathrm{SumStr}(X,A,B)),$$
   $$\mathrm{inlStr}(A, \mathrm{w}_1(A,A), \mathrm{w}_1(B,A), \mathbf{x}_1(A)),$$
   $$\mathrm{inrStr}(B, \mathrm{w}_1(A,B), \mathrm{w}_1(B,B), \mathbf{x}_1(B)),$$
   $$u) = u.$$

Definition 3.3.6. A Π-*type structure* on $\mathcal{C}$ consists of

1. For each object $(X.A.B) \in \mathrm{Ob}_{\mathcal{C}}(n+2)$ an object $(X.\,\Pi\text{-}\mathrm{Str}(X,A,B))$.
2. For each object $(X.A.B) \in \mathrm{Ob}_{\mathcal{C}}(n+2)$ and term morphism $u : A \to B$ a term morphism

   $$\lambda\text{-}\mathrm{Str}(X,A,B,u) : X \to \Pi\text{-}\mathrm{Str}(X,A,B).$$

3. For each object $(X.A.B) \in \mathrm{Ob}_{\mathcal{C}}(n+2)$, term morphism $f : X \to \Pi\text{-Str}(X, A, B)$ and term morphism $a : X \to A$ a term morphism

$$\mathrm{app}(X, A, B, f, a) : X \to B[a]$$

satisfying

$$\mathrm{app}(X, A, B, \lambda\text{-Str}(X, A, B, u), a) = u[a].$$

4. Such that for each $g : Y \to X$ and the appropriate arguments

$$\begin{aligned} g^*(\Pi\text{-Str}(X, A, B)) &= \Pi\text{-Str}(Y, g^*(A), g^{*2}(B)), \\ g^*(\lambda\text{-Str}(X, A, B, u)) &= \lambda\text{-Str}(Y, g^*(A), g^{*2}(B), g^{*2}(u)), \\ g^*(\mathrm{app}(X, A, B, f, a)) &= \mathrm{app}(Y, g^*(A), g^{*2}(B), g^*(f), g^*(a)). \end{aligned}$$

5. And satisfies η-expansion if for the appropriate arguments

$$\lambda\text{-Str}(X, A, B, \mathrm{app}(A, \mathrm{w}_1(A, A), \mathrm{w}_2(B, A), \mathrm{w}_1(f, A), \mathbf{x}_0(A))) = f.$$

Definition 3.3.7. A Σ-*type structure* on $\mathcal{C}$ consists of

1. For each object $(X.A.B) \in \mathrm{Ob}_{\mathcal{C}}(n+2)$ an object $(X.\,\Sigma\text{-Str}(X, A, B))$.

2. For each object $(X.A.B) \in \mathrm{Ob}_{\mathcal{C}}(n+2)$, term morphism $a : X \to A$ and term morphism $b : X \to B[a]$ a term morphism

$$\mathrm{pairStr}(X, A, B, a, b) : X \to \Sigma\text{-Str}(X, A, B).$$

3. For each object $(X.A.B) \in \mathrm{Ob}_{\mathcal{C}}(n+2)$ and term morphism $u : X \to \Sigma\text{-Str}(X, A, B)$ a term morphism

$$\mathrm{pr}_1\text{-Str}(X, A, B, u) : X \to A$$

satisfying

$$\mathrm{pr}_1\text{-Str}(X, A, B, \mathrm{pairStr}(X, A, B, a, b)) = a.$$

4. For each object $(X.A.B) \in \mathrm{Ob}_{\mathcal{C}}(n+2)$ and term morphism $u : X \to \Sigma\text{-Str}(X, A, B)$ a term morphism

$$\mathrm{pr}_2\text{-Str}(X, A, B, u) : X \to B[\mathrm{pr}_1\text{-Str}(X, A, B, u)]$$

satisfying

$$\mathrm{pr}_2\text{-Str}(X, A, B, \mathrm{pairStr}(X, A, B, a, b)) = b.$$

5. Such that for each $g : Y \to X$ and the appropriate arguments

$$\begin{aligned} g^*(\Sigma\text{-Str}(X, A, B)) &= \Sigma\text{-Str}(Y, g^*(A), g^{*2}(B)), \\ g^*(\text{pairStr}(X, A, B, a, b)) &= \text{pairStr}(Y, g^*(A), g^{*2}(B), g^*(a), g^*(b)), \\ g^*(\text{pr}_1\text{-Str}(X, A, B, u)) &= \text{pr}_1\text{-Str}(Y, g^*(A), g^{*2}(B), g^*(u)), \\ g^*(\text{pr}_2\text{-Str}(X, A, B, u)) &= \text{pr}_2\text{-Str}(Y, g^*(A), g^{*2}(B), g^*(u)). \end{aligned}$$

6. And satisfying η-expansion if for the appropriate arguments

$$\text{pairStr}(X, A, B, \text{pr}_1\text{-Str}(X, A, B, u), \text{pr}_2\text{-Str}(X, A, B, u)) = u.$$

Definition 3.3.8. A *Id-type structure* on $\mathcal{C}$ consists of

1. For each object $(X.A) \in \text{Ob}_{\mathcal{C}}(n+1)$ and pair of term morphism $a, b : X \to A$ an object $(X.\,\text{IdStr}(X, A, a, b))$.

2. For each object $(X.A) \in \text{Ob}_{\mathcal{C}}(n+1)$ and term morphism $a : X \to A$ a term morphism

$$\text{reflStr}(X, A, a) : X \to \text{IdStr}(X, A, a, b).$$

3. For readability, we introduce the notation

$$\begin{aligned} \mathsf{Id}(A) \coloneqq \text{IdStr}(&\text{w}_1(A, A), \text{w}_1(\text{w}_1(A, A), \text{w}_1(A, A)), \\ &\mathbf{x}_1(\text{w}_1(A, A)), \mathbf{x}_0(\text{w}_1(A, A))). \end{aligned}$$

   Observe that $(X.A.\,\text{w}_1(A, A).\mathsf{Id}(A)) \in \text{Ob}_{\mathcal{C}}(n+3)$.

4. For each

   - object $(X.A) \in \text{Ob}_{\mathcal{C}}(n+1)$,
   - object $(\mathsf{Id}(A).P) \in \text{Ob}_{\mathcal{C}}(n+4)$,
   - term morphism $d_{\mathbf{refl}} : A \to (\text{w}_4(P, A))[\mathbf{x}_0(A), \mathbf{x}_0(A), \text{reflStr}(A, \text{w}_1(A, A), \mathbf{x}_0(A))]$,
   - term morphism $a : X \to A$,
   - term morphism $b : X \to A$,
   - term morphism $p : X \to \text{IdStr}(X, A, a, b)$,

   a term morphism

$$\text{J-Str}(X, A, P, d_{\mathbf{refl}}, a, b, p) : X \to P[a, b, p]$$

   satisfying

$$\text{J-Str}(X, A, P, d_{\mathbf{refl}}, a, a, \text{reflStr}(X, A, a)) = d_{\mathbf{refl}}[a].$$

5. Such that for each $g : Y \to X$ and the appropriate arguments

$$\begin{aligned} g^*(\mathrm{IdStr}(X, A, a, b)) &= \mathrm{IdStr}(Y, g^*(A), g^*(a), g^*(b)), \\ g^*(\mathrm{reflStr}(X, A, a)) &= \mathrm{reflStr}(Y, g^*(A), g^*(a)), \\ g^*(\text{J-Str}(X, A, P, d_{\mathbf{refl}}, a, b, p)) &= \\ \text{J-Str}(Y, g^*(A), g^{*4}(P), &g^{*2}(d_{\mathbf{refl}}), g^*(a), g^*(b), g^*(p)). \end{aligned}$$

Definition 3.3.9. A *hierarchy of universes* on $\mathcal{C}$ consists of

1. For each $i \in \mathbb{N}$ and object $X \in \mathrm{Ob}_{\mathcal{C}}(n)$ an object $(X.\,\text{U-Str}(i, X))$.
2. For each $i \in \mathbb{N}$ and object $X \in \mathrm{Ob}_{\mathcal{C}}(n)$ a term morphism
$$\text{u-Str}(i, X) : X \to \text{U-Str}(i + 1, X).$$
3. For each $i \in \mathbb{N}$, object $X \in \mathrm{Ob}_{\mathcal{C}}(n)$ and term morphism $v : X \to \text{U-Str}(i, X)$ an object $(X.\,\mathrm{ElStr}(i, X, v))$ satisfying
$$\mathrm{ElStr}(i + 1, X, \text{u-Str}(i, X)) = \text{U-Str}(i, X).$$
4. Such that for each $g : Y \to X$ and the appropriate arguments
$$\begin{aligned} g^*(\text{U-Str}(i, X)) &= \text{U-Str}(i, Y), \\ g^*(\text{u-Str}(i, X)) &= \text{u-Str}(i, Y), \\ g^*(\mathrm{ElStr}(i, X, v)) &= \mathrm{ElStr}(i, Y, g^*(v)). \end{aligned}$$
5. Additional, if $\mathcal{C}$ has
   - Empty-type structure: a term morphism
   $$\mathrm{emptyStr}(i, X) : X \to \text{U-Str}(i, X)$$
   such that
   $$\begin{aligned} g^*(\mathrm{emptyStr}(i, X)) &= \mathrm{emptyStr}(i, Y) \\ \mathrm{ElStr}(i, X, \mathrm{emptyStr}(i, X)) &= \mathrm{EmptyStr}(X). \end{aligned}$$
   - Unit-type structure: a term morphism
   $$\mathrm{unitStr}(i, X) : X \to \text{U-Str}(i, X)$$
   such that
   $$\begin{aligned} g^*(\mathrm{unitStr}(i, X)) &= \mathrm{unitStr}(i, Y) \\ \mathrm{ElStr}(i, X, \mathrm{unitStr}(i, X)) &= \mathrm{UnitStr}(X). \end{aligned}$$

- Nat-type structure: a term morphism

$$\mathrm{natStr}(i, X) : X \to \text{U-Str}(i, X)$$

such that

$$g^*(\mathrm{natStr}(i, X)) = \mathrm{natStr}(i, Y)$$
$$\mathrm{ElStr}(i, X, \mathrm{natStr}(i, X)) = \mathrm{NatStr}(X).$$

- Sum-type structure: for each term morphism $a : X \to \text{U-Str}(i, X)$ and term morphism $b : X \to \text{U-Str}(i, X)$ a term morphism

$$\mathrm{sumStr}(i, X, a, b) : X \to \text{U-Str}(i, X)$$

such that

$$g^*(\mathrm{sumStr}(i, X, a, b)) = \mathrm{sumStr}(i, Y, g^*(a), g^*(b))$$
$$\mathrm{ElStr}(i, X, \mathrm{sumStr}(i, X, a, b)) = \mathrm{SumStr}(X, \mathrm{ElStr}(i, X, a), \mathrm{ElStr}(i, X, b)).$$

- Π-type structure: for each term morphism $a : X \to \text{U-Str}(i, X)$ and term morphism $b : \mathrm{ElStr}(i, X, a) \to \text{U-Str}(i, \mathrm{ElStr}(i, X, a))$ a term morphism

$$\pi\text{-Str}(i, X, a, b) : X \to \text{U-Str}(i, X)$$

such that

$$g^*(\pi\text{-Str}(i, X, a, b)) = \pi\text{-Str}(i, Y, g^*(a), g^{*2}(b))$$
$$\mathrm{ElStr}(i, X, \pi\text{-Str}(i, X, a, b)) = \Pi\text{-Str}(X, \mathrm{ElStr}(i, X, a), \mathrm{ElStr}(i, \mathrm{ElStr}(i, X, a), b)).$$

- Σ-type structure: for each term morphism $a : X \to \text{U-Str}(i, X)$ and term morphism $b : \mathrm{ElStr}(i, X, a) \to \text{U-Str}(i, \mathrm{ElStr}(i, X, a))$ a term morphism

$$\sigma\text{-Str}(i, X, a, b) : X \to \text{U-Str}(i, X)$$

such that

$$g^*(\sigma\text{-Str}(i, X, a, b)) = \sigma\text{-Str}(i, Y, g^*(a), g^{*2}(b))$$
$$\mathrm{ElStr}(i, X, \sigma\text{-Str}(i, X, a, b)) = \Sigma\text{-Str}(X, \mathrm{ElStr}(i, X, a), \mathrm{ElStr}(i, \mathrm{ElStr}(i, X, a), b)).$$

- Id-type structure: for each term morphism $a : X \to \text{U-Str}(i, X)$ and pair of term morphisms $u, v : X \to \text{ElStr}(i, X, a)$ a term morphism $\text{idStr}(i, X, a, u, v) : X \to \text{U-Str}(i, X)$ such that

$$g^*(\text{idStr}(i, X, a, u, v)) = \text{idStr}(i, Y, g^*(a), g^*(u), g^*(v))$$
$$\text{ElStr}(i, X, \text{idStr}(i, X, a, u, v)) = \text{IdStr}(X, \text{ElStr}(i, X, a), u, v).$$

Definition 3.3.10. A contextual category $\mathcal{C}$ is considered *fully structured* if it is equipped with a chosen structure of all the previous mentioned ones including a hierarchy of universes and satisfies the various η-expansions.

The following justifies that claim that the above definition correctly capture all the constructors of MLTT.

Theorem 3.3.11. *The contextual category* $\mathcal{C}(\text{MLTT})$ *is fully structured.*

Proof. The proof essentially writes itself. Each structure is formed by the appropriate formation, introduction and elimination rules in addition with observations from Example 3.2.4 about terms morphisms in $\mathcal{C}(\text{MLTT})$. In particular, the structure corresponding to a type constructor on an object has the form

$$[\Gamma, \textbf{constructor}],$$

and the structure corresponding to term constructor has the form

$$[\text{id}, \textbf{constructor}] : [\Gamma] \to [\Gamma, A]$$

for an appropriate type expression A. As an example we set

$$\text{NatStr}([\Gamma]) \coloneqq [\Gamma, \mathbf{N}]$$

which is a derivable context by $\mathbf{N}$-FORM and

$$\text{sucStr}([\Gamma], [\text{id}, u]) \coloneqq [\text{id}, \textbf{suc}(u)] : [\Gamma] \to [\Gamma, \mathbf{N}]$$

which is derivable by **suc**-INTRO and relies on observations from Example 3.2.4 about term morphisms in $\mathcal{C}(\text{MLTT})$. The structures are all well-defined by the congruence rules. The various other equations that are needed to be satisfied follow by syntactic equalities, the computation rules or the η rules. We refer to the formalization for the details.

At a few places, one must use an additional syntactic equality in order to match the premises of a rule with the premises of the structure. For instance, in **ind**-ELIM, the type of $d_{\textbf{suc}}$ is

$$(\text{wTy}_1(\text{wTy}_1(P)))[\textbf{suc}(\mathbf{x}_1)]$$

which matches the object[1]

$$(\mathrm{w}_2(\mathrm{w}_2(P, \mathrm{NatStr}(X)), P))[\mathrm{sucStr}(P, \mathbf{x}_1(P))]$$

in $\mathcal{C}(\mathrm{MLTT})$ only up to syntactic equality, since weakening on objects in a contextual category is defined in terms of substitution. □

Remark 3.3.12. Even though the above proofs are straightforward, our current formalization fails to type check with a reasonable amount of RAM. To be precise, we have not been able to type check the formalization of the **J**-constructor, due to the complexity of $d_{\mathbf{refl}}$. This can be fixed by forcing unnecessary data to be erased. We will discuss this further once we get to the formalization in Chapter 5.

Morphisms between (structured) contextual categories

So far we have not mentioned what we consider a morphism between two contextual categories to be. This is crucial to even state the initiality conjecture. A morphism of contextual categories will be a functor that preserves all the structure *on the nose*. This agrees with the concept of homomorphisms when considering a contextual category as an essentially algebraic theory as in Remark 3.1.6 and as discussed in [KL20, Definition 1.2.7]. As we are interested in the system MLTT we will consider morphisms between fully structured contextual categories only, but it should be clear how they can be altered for more general settings.

Definition 3.3.13. Given two fully structured contextual categories $\mathcal{C}$ and $\mathcal{D}$, a *morphism of fully structured contextual categories* or *contextual morphism* for short is a functor $F : \mathcal{C} \to \mathcal{D}$ between the underlying categories, respecting the grading and preserving all the structure on the nose.

Additionally, it will be important to know when two contextual morphisms are equal. It will be enough to verify they are equal as functors.

Lemma 3.3.14. *If $F, G : \mathcal{C} \to \mathcal{D}$ are contextual morphisms between fully structured contextual categories such that*

- *for any object $X \in \mathrm{Ob}_{\mathcal{C}}$, $F(X) = G(X)$,*
- *for any morphism $f : X \to Y$ in $\mathcal{C}$, $F(f) = G(f)$,*

then $F = G$.

[1]Observe we need to increase the position by one, as semantically it is defined on *contexts* not *types*, i.e. it follows the same 'level' as the syntactic operation $\mathrm{wCtx}_k(-)$.

Proof. It should not be surprising that F and G will also agree on the additional data. However, we do need function extensionality in our metatheory to turn both premises in equalities on which we can pattern match. A reader with a more classical background should not have to worry about this. □

We now have all the ingredients ready to tackle the initiality conjecture, which will be the topic of the next chapter.

4. Initiality

Given everything we have introduced so far, we are finally in a position to properly state the initiality theorem:

Theorem 4.0.1. *The term model* $\mathcal{C}(\mathrm{MLTT})$ *is the initial fully structured contextual category, i.e. for any fully structured contextual category* $\mathcal{C}$*, there exists a unique contextual morphism* $\mathcal{C}(\mathrm{MLTT}) \to \mathcal{C}$.

We will prove this theorem in two steps. First, we show that for any contextual category $\mathcal{C}$, there exists a contextual morphism $\mathcal{C}(\mathrm{MLTT}) \to \mathcal{C}$. This map is commonly called the *interpretation function*. To achieve this, we first define a partial function on raw syntax and then show that this map is total for well-formed expressions. This technique is originally due to Streicher [Str91] and slightly modified by Hofmann [Hof97]. Next, we show that any two contextual morphisms $F, G : \mathcal{C}(\mathrm{MLTT}) \to \mathcal{C}$ are equal. We choose to take two arbitrary contextual morphism, in order to make the two steps independent of one another. Combining them will result in the proof of Theorem 4.0.1.

Throughout this chapter, let $\mathcal{C}$ be a fully structured contextual category.

4.1 Partial interpretation

As mentioned before we start by establishing the existence of a contextual morphism $\mathcal{C}(\mathrm{MLTT}) \to \mathcal{C}$. One might be tempted to define this functor directly. However, derivation are not unique due to rules like Conv. Because of this, we cannot define the components of such a functor just by induction on derivation trees without changing the syntax admitting either unique derivations or some coherence result. This is a known issue and discussed for instance in [Cur93], [CGH14] and [Yam17].

In this thesis we follow the approach contributed to Streicher [Str91]. This method was also suitable for formalization. We start by defining the partial interpretation operation on raw syntax. For this we will first be precise about what we mean by partial elements.

Definition 4.1.1. For a set X we write Partial X for the set of *partial elements* of X, i.e. subsets of X with at most one element or, equivalently, a map from

some proposition to X. A *partial operation* from Y to X is an operation $Y \to \mathrm{Partial}\, X$.

For $x \in \mathrm{Partial}\, X$ we say that x *is defined* and write $x\!\downarrow$ for short if x is non-empty. In the case that $x\!\downarrow$ we also write x for its unique element. This abuse of notation should not raise any confusion.

We introduce the following relation on partial elements. For $x, y \in \mathrm{Partial}\, X$ we write $x \leq y$ to mean that if $x\!\downarrow$, then $y\!\downarrow$ and $x = y$. Observe that if both $x \leq y$ and $y \leq x$ we find the usual *Kleene equality* which we write $x \simeq y$. We note that $\leq$ is a *partial order*.

Remark 4.1.2. It is discussed in [Kna18, Part III] that the above definition is a generalization of the classical approach of defining partial operations as total functions $Y \to X + 1$. It is also shown, among other things, that Partial forms a *monad*. We use this monadic structure to implement partial function in our formalization. However, in the body of this thesis we stick to a prose write-up.

We will view the operations on a fully structured contextual category also as partial operations. For example $\mathrm{SumStr}(X, A, B) \in \mathrm{Partial}\, \mathrm{Ob}_{\mathcal{C}}$ and is defined precisely when $\mathrm{ft}\, A = X$ and $\mathrm{ft}\, B = X$ and $\mathrm{sucStr}(X, u) \in \mathrm{Partial}\, \mathrm{Mor}_{\mathcal{C}}$ which is defined precisely when u is a term morphism $X \to \mathrm{NatStr}(X)$. We will treat the basic operations such as q and p similarly.

We follow the convention that if an expression contains an undefined subexpression it is itself undefined.

Definition 4.1.3. We define two operations

$$[\![-]\!]^{-}_{\mathsf{Ty}} : \mathrm{TyExpr}(n) \times \mathrm{Ob}_{\mathcal{C}}(n) \to \mathrm{Partial}\, \mathrm{Ob}_{\mathcal{C}}$$
$$[\![-]\!]^{-}_{\mathsf{Tm}} : \mathrm{TmExpr}(n) \times \mathrm{Ob}_{\mathcal{C}}(n) \to \mathrm{Partial}\, \mathrm{Mor}_{\mathcal{C}}$$

by mutual structural induction on raw syntax. We will often omit the subscripts.

$$\begin{aligned}
[\![\mathbf{0}]\!]^X &:= \mathrm{EmptyStr}(X) \\
[\![\mathbf{1}]\!]^X &:= \mathrm{UnitStr}(X) \\
[\![\mathbf{N}]\!]^X &:= \mathrm{NatStr}(X) \\
[\![A + B]\!]^X &:= \mathrm{SumStr}(X, [\![A]\!]^X, [\![B]\!]^X) \\
[\![\Pi_A B]\!]^X &:= \Pi\text{-}\mathrm{Str}(X, [\![A]\!]^X, [\![B]\!]^{([\![A]\!]^X)}) \\
[\![\Sigma_A B]\!]^X &:= \Sigma\text{-}\mathrm{Str}(X, [\![A]\!]^X, [\![B]\!]^{([\![A]\!]^X)}) \\
[\![\mathbf{Id}_A(u, v)]\!]^X &:= \mathrm{IdStr}(X, [\![A]\!]^X, [\![u]\!]^X, [\![v]\!]^X) \\
[\![\mathbf{U}_i]\!]^X &:= \mathrm{U}\text{-}\mathrm{Str}(i, X) \\
[\![\mathbf{El}_i(v)]\!]^X &:= \mathrm{ElStr}(X, [\![v]\!]^X)
\end{aligned}$$

$$
\begin{aligned}
[\![\mathbf{x}_l]\!]^X &:= \mathbf{x}_l(X)\\
[\![\mathbf{empty\_elim}(P,u)]\!]^X &:= \mathrm{empty\_elimStr}(X, [\![P]\!]^{\mathrm{EmptyStr}(X)}, [\![u]\!]^X)\\
[\![\star]\!]^X &:= \star\text{-Str}(X)\\
[\![\mathbf{unit\_elim}(P,d_\star,u)]\!]^X &:= \mathrm{unit\_elimStr}(X, [\![P]\!]^{\mathrm{UnitStr}(X)}, [\![d_\star]\!]^X, [\![u]\!]^X)\\
[\![\mathbf{zero}]\!]^X &:= \mathrm{zeroStr}(X)\\
[\![\mathbf{suc}(u)]\!]^X &:= \mathrm{sucStr}(X, [\![u]\!]^X)\\
[\![\mathbf{ind}(P,d_{\mathbf{zero}},d_{\mathbf{suc}},u)]\!]^X &:= \mathrm{indStr}(X, [\![P]\!]^{\mathrm{NatStr}(X)}, [\![d_{\mathbf{zero}}]\!]^X,\\
&\qquad [\![d_{\mathbf{suc}}]\!]^{([\![P]\!]^{\mathrm{NatStr}(X)})}, [\![u]\!]^X)\\
[\![\mathbf{inl}(A,B,a)]\!]^X &:= \mathrm{inlStr}(X, [\![A]\!]^X, [\![B]\!]^X, [\![a]\!]^X)\\
[\![\mathbf{inr}(A,B,b)]\!]^X &:= \mathrm{inrStr}(X, [\![A]\!]^X, [\![B]\!]^X, [\![b]\!]^X)\\
[\![\mathbf{match}(A,B,P,d_{\mathbf{inl}},d_{\mathbf{inr}},u)]\!]^X &:= \mathrm{matchStr}(X, [\![A]\!]^X, [\![B]\!]^X,\\
&\quad [\![P]\!]^{\mathrm{SumStr}(X,[\![A]\!]^X,[\![B]\!]^X)}, [\![d_{\mathbf{inl}}]\!]^{([\![A]\!]^X)}, [\![d_{\mathbf{inr}}]\!]^{([\![B]\!]^X)}, [\![u]\!]^X)\\
[\![\lambda(A,B,u)]\!]^X &:= \lambda\text{-Str}(X, [\![A]\!]^X, [\![B]\!]^{([\![A]\!]^X)}, [\![u]\!]^{([\![A]\!]^X)})\\
[\![\mathbf{app}(A,B,f,a)]\!]^X &:= \mathrm{app}(X, [\![A]\!]^X, [\![B]\!]^{([\![A]\!]^X)}, [\![f]\!]^X, [\![a]\!]^X)\\
[\![\mathbf{pair}(A,B,a,b)]\!]^X &:= \mathrm{pairStr}(X, [\![A]\!]^X, [\![B]\!]^{([\![A]\!]^X)}, [\![a]\!]^X, [\![b]\!]^X)\\
[\![\mathbf{pr}_1(A,B,u)]\!]^X &:= \mathrm{pr}_1\text{-Str}(X, [\![A]\!]^X, [\![B]\!]^{([\![A]\!]^X)}, [\![u]\!]^X)\\
[\![\mathbf{pr}_2(A,B,u)]\!]^X &:= \mathrm{pr}_2\text{-Str}(X, [\![A]\!]^X, [\![B]\!]^{([\![A]\!]^X)}, [\![u]\!]^X)\\
[\![\mathbf{refl}(A,a)]\!]^X &:= \mathrm{reflStr}(X, [\![A]\!]^X, [\![a]\!]^X)\\
[\![\mathbf{J}(A,P,d_{\mathbf{refl}},u,v,p)]\!]^X &:= \mathrm{jStr}(X, [\![A]\!]^X, [\![P]\!]^{\mathsf{Id}([\![A]\!]^X)}, [\![d_{\mathbf{refl}}]\!]^{([\![A]\!]^X)},\\
&\qquad [\![u]\!]^X, [\![v]\!]^X, [\![p]\!]^X)\\
[\![0_i]\!]^X &:= \mathrm{emptyStr}(i, X)\\
[\![1_i]\!]^X &:= \mathrm{unitStr}(i, X)\\
[\![\mathbf{n}_i]\!]^X &:= \mathrm{natStr}(i, X)\\
[\![\mathbf{u}_i]\!]^X &:= \text{u-Str}(i, X)\\
[\![a +_i b]\!]^X &:= \mathrm{sumStr}(X, [\![a]\!]^X, [\![b]\!]^X)\\
[\![\pi_i(a,b)]\!]^X &:= \pi\text{-Str}(X, [\![a]\!]^X, [\![b]\!]^{\mathrm{ElStr}(X,[\![a]\!]^X)})\\
[\![\sigma_i(a,b)]\!]^X &:= \sigma\text{-Str}(X, [\![a]\!]^X, [\![b]\!]^{\mathrm{ElStr}(X,[\![a]\!]^X)})\\
[\![\mathbf{id}_i(a,u,v)]\!]^X &:= \mathrm{idStr}(X, [\![a]\!]^X, [\![u]\!]^X, [\![v]\!]^X).
\end{aligned}
$$

We make the following observation about the partial interpretation just defined. They follow from the definition of the various structure.

Lemma 4.1.4. *For $X \in \mathrm{Ob}_{\mathcal{C}}(n)$ we have*

- *if $[\![A]\!]^X_{\mathsf{Ty}} \downarrow$ then $[\![A]\!]^X_{\mathsf{Ty}} \in \mathrm{Ob}_{\mathcal{C}}(n+1)$ and $\mathrm{ft}([\![A]\!]^X_{\mathsf{Ty}}) = X$,*
- *if $[\![u]\!]^X_{\mathsf{Tm}} \downarrow$ then $[\![u]\!]^X_{\mathsf{Tm}}$ is a term morphism*

$$[\![u]\!]^X_{\mathsf{Tm}} : X \to \mathrm{cod}([\![u]\!]^X_{\mathsf{Tm}}).$$

□

We can extend our interpretation to raw contexts and raw context morphisms. Observe that these reflect the deduction rules for contexts and context morphisms Definition 2.3.5.

Definition 4.1.5. We define two function

$$[\![-]\!]_{\mathsf{Ctx}} : \mathrm{Ctx}(n) \to \mathrm{Partial}\,\mathrm{Ob}_{\mathcal{C}}(n)$$
$$[\![-]\!]^{-,-}_{\mathsf{CtxMor}} : \mathrm{CtxMor}(m,n) \times \mathrm{Ob}_{\mathcal{C}}(m) \times \mathrm{Ob}_{\mathcal{C}}(n) \to \mathrm{Partial}\,\mathrm{Mor}_{\mathcal{C}}$$

by structural induction on raw syntax. Again we will often omit the subscripts.

$$\begin{aligned}
[\![\diamond]\!] &:= \mathrm{pt} \\
[\![\Gamma, A]\!] &:= [\![A]\!]^{[\![\Gamma]\!]} \\
[\![!]\!]^{X,\mathrm{pt}} &:= \mathrm{ptmor}_X \\
[\![\delta, t]\!]^{X,Y} &:= \mathrm{q}([\![\delta]\!]^{X,\mathrm{ft}\,Y}_{\mathsf{CtxMor}}, Y) \circ [\![t]\!]^X_{\mathsf{Tm}}
\end{aligned}$$

Remark 4.1.6. Reading the partiality in q and composition, $[\![\delta, t]\!]^{X,Y}$ is defined precisely when

- $[\![\delta]\!]^{X,\mathrm{ft}\,Y}_{\mathsf{CtxMor}} \downarrow$ and $\mathrm{cod}([\![\delta]\!]^{X,\mathrm{ft}\,Y}) = \mathrm{ft}\,Y$,
- $[\![t]\!]^X_{\mathsf{Tm}} \downarrow$ and $\mathrm{cod}([\![t]\!]^X) = ([\![\delta]\!]^{X,\mathrm{ft}\,Y})^* Y$.

The following is immediate.

Lemma 4.1.7. *If $[\![\delta]\!]^{X,Y}_{\mathsf{CtxMor}} \downarrow$ then $[\![\delta]\!]^{X,Y}_{\mathsf{CtxMor}} : X \to Y$.* □

4.2 Totality

The key observation about the partial interpretation is that if the input is *well-formed* then the output is defined. In order to show this, we first need lemmas that deal with the interpretation of weakening and substituting as these are present in various rules. This is a common first step, see for instance [Hof97, Proposition 2.5.3 and Lemma 2.5.5].

Weakening

We need the result about weakening first as it is used in the case for substitution.

Lemma 4.2.1. *For appropriate input:*

- $\mathrm{w}_k([\![A]\!]^X_{\mathsf{Ty}}, W) \leq [\![\mathrm{wTy}_k(A)]\!]^{\mathrm{w}_k(X,W)}_{\mathsf{Ty}}$,
- $\mathrm{w}_{k+1}([\![u]\!]^X_{\mathsf{Tm}}, W) \leq [\![\mathrm{wTm}_k(u)]\!]^{\mathrm{w}_k(X,W)}_{\mathsf{Tm}}$,
- $\mathrm{w}_{k+1}(\mathrm{cod}([\![u]\!]^X_{\mathsf{Tm}}), W) \leq \mathrm{cod}([\![\mathrm{wTm}_k(u)]\!]^{\mathrm{w}_k(X,W)}_{\mathsf{Tm}})$.

Proof. One starts by showing that if both sides are indeed defined, the first two equalities hold. This is done by a mutual induction on raw syntax.

Only then, we deal with the implication part of $\leq$ which is done again by a mutual induction on raw syntax for the first two, while the third is a direct consequence of the second and the observation in Definition 3.2.5. This approach allows us to break up the mutual induction in two steps.

Both proofs are straightforward apart from the variable case and making sure the induction hypotheses is used correctly. Any difficulty in the variable cases is taken care of by Lemma 3.2.15. □

Remark 4.2.2. We note that in the current version of the formalization we start by proving the case $k = 0$ first and build up the remaining cases only up to $k \leq 3$. However, the presented proof is more general.

Due to the definition of syntactic substitution we need to know how to interpret $\mathrm{wMor+}(-)$.

Corollary 4.2.3. *For appropriate input*

- $[\![\delta]\!]^{X,Y} \circ \mathrm{p}_Z \leq [\![\mathrm{wMor}_0(\delta)]\!]^{Z,Y}$,
- $\mathrm{q}^k([\![\delta]\!]^{X,Y}, W) \leq [\![\mathrm{wMor+}^k(\delta)]\!]^{([\![\delta]\!]^{X,Y})^{*k}W,W}$.

Proof. One needs the first one result in order to show the second one, due to the definition of $\mathrm{wMor+}(\delta)$, but both are straightforward. □

As a consequence we find that the interpretation of identity morphisms are as one would expect.

Corollary 4.2.4. *For $X \in \mathrm{Ob}_{\mathcal{C}}(n)$ we have*

$$[\![\mathrm{id}_n]\!]^{X,X} \simeq \mathrm{id}_X.$$

Proof. By induction on n. The base case follows, as the interpretation of ! is always defined. The step case follows by the definition of id_{n+1} and the previous corollary. □

In fact, we can do better and extend this to the context morphisms used for syntactic term substitution. It is formalized only up to the point necessary for our results, but can be easily extended.

Corollary 4.2.5. *For appropriate input we have*

$$[\![\mathrm{id}_n, t_0, \ldots, t_{p-1}]\!]^{\mathrm{ft}^p\, Y, Y}_{\mathsf{CtxMor}} \geq \mathrm{q}^{p-1}([\![t_0]\!]^{\mathrm{ft}^p\, Y}_{\mathsf{Tm}}, Y) \circ \cdots \circ \mathrm{q}^0([\![t_{p-1}]\!]^{\mathrm{ft}^p\, Y}_{\mathsf{Tm}}, Y\left[[\![t_0]\!]^{\mathrm{ft}^p\, Y}_{\mathsf{Tm}}, \ldots, [\![t_{p-2}]\!]^{\mathrm{ft}^p\, Y}_{\mathsf{Tm}}\right]).$$

Proof. One builds it up starting at $p = 1$ and using the equations satisfied by q in Definition 3.1.5. □

Substitution

We are now able to link semantic and syntactic substitution.

Lemma 4.2.6. *For $k \leq 3$ and appropriate input*

- $([\![\delta]\!]^{X,Y})^{*k+1}[\![A]\!]^W \leq [\![A[\mathrm{wMor}{+}^k(\delta)]]\!]^{([\![\delta]\!]^{X,Y})^{*k}W}$,
- $([\![\delta]\!]^{X,Y})^{*k+1}[\![u]\!]^W \leq [\![u[\mathrm{wMor}{+}^k(\delta)]]\!]^{([\![\delta]\!]^{X,Y})^{*k}W}$,
- $([\![\delta]\!]^{X,Y})^{*k+1}(\mathrm{cod}([\![u]\!]^W)) \leq \mathrm{cod}([\![u[\mathrm{wMor}{+}^k(\delta)]]\!]^{([\![\delta]\!]^{X,Y})^{*k}W})$.

Proof. One starts by showing the case $k = 0$ which is done in a similar fashion to the case for weakening. The variable cases are again the most difficult and taken care of by Lemma 3.2.13. The cases for $k > 0$ are built up from this, using Corollary 4.2.3. □

The case for term substitution also agrees between syntax and semantics.

Corollary 4.2.7. *For $p \leq 3$ and appropriate input we have*

- $[\![A]\!]^X_{\mathsf{Ty}}\left[[\![t_0]\!]^{\mathrm{ft}^p\, X}, \ldots, [\![t_{p-1}]\!]^{\mathrm{ft}^p\, X}\right] \leq [\![A[t_0, \ldots, t_{p-1}]]\!]^{\mathrm{ft}^p\, X}$,
- $[\![u]\!]^X_{\mathsf{Tm}}\left[[\![t_0]\!]^{\mathrm{ft}^p\, X}, \ldots, [\![t_{p-1}]\!]^{\mathrm{ft}^p\, X}\right] \leq [\![u[t_0, \ldots, t_{p-1}]]\!]^{\mathrm{ft}^p\, X}$.

Proof. Using Lemma 4.2.6 and Corollary 4.2.5. □

Additionally, we can also show the link between syntactic and semantic composition.

Corollary 4.2.8. *For appropriate input we have*

$$[\![\delta]\!]^{X,Y} \circ [\![\theta]\!]^{Z,X} \leq [\![\delta[\theta]]\!]^{Z,Y}.$$ □

Well-formed syntax

We will now show the promised result that the interpretation of well-formed syntax is defined and moreover that it preserves judgmental equality. This will finish up all the preparation to prove the initiality theorem.

Theorem 4.2.9. *For a context* $\Gamma \in \mathrm{Ctx}(n)$ *such that* $[\![\Gamma]\!] \downarrow$ *the following hold:*

- *If* $\Gamma \vdash A$ *then* $[\![A]\!]^{[\![\Gamma]\!]} \downarrow$.
- *If* $\Gamma \vdash u : A$ *then* $[\![u]\!]^{[\![\Gamma]\!]} \downarrow$.
- *If* $\Gamma \vdash u : A$, $[\![A]\!]^{[\![\Gamma]\!]} \downarrow$ *and* $[\![u]\!]^{[\![\Gamma]\!]} \downarrow$ *then* $\mathrm{cod}([\![u]\!]^{[\![\Gamma]\!]}) = [\![A]\!]^{[\![\Gamma]\!]}$.
- *If* $\Gamma \vdash A \equiv A'$, $[\![A]\!]^{[\![\Gamma]\!]} \downarrow$ *and* $[\![A']\!]^{[\![\Gamma]\!]} \downarrow$ *then*
$$[\![A]\!]^{[\![\Gamma]\!]} = [\![A']\!]^{[\![\Gamma]\!]}\,.$$
- *If* $\Gamma \vdash u \equiv u' : A$, $[\![u]\!]^{[\![\Gamma]\!]} \downarrow$ *and* $[\![u']\!]^{[\![\Gamma]\!]} \downarrow$ *then*
$$[\![u]\!]^{[\![\Gamma]\!]} = [\![u']\!]^{[\![\Gamma]\!]}\,.$$

Proof. All are proven simultaneously by induction on the relevant derivation. We note that the statements are presented in such a way that the various induction hypotheses can be used accordingly. □

The above can again be extended to the interpretation of well-formed contexts and context morphisms. Both are done by a straightforward structural induction and using an appropriate presupposition from Lemma 2.3.15.

Corollary 4.2.10. *For* $\Gamma, \Gamma' \in \mathrm{Ctx}(n)$ *we have*

- *if* $\vdash \Gamma$, *then* $[\![\Gamma]\!] \downarrow$.
- *if* $\vdash \Gamma \equiv \Gamma'$, *then* $[\![\Gamma]\!] = [\![\Gamma']\!]$.

Corollary 4.2.11. *For* $\delta, \delta' \in \mathrm{CtxMor}(m, n)$ *and derivable contexts* $\Delta \in \mathrm{Ctx}(m)$ *and* $\Delta \in \mathrm{Ctx}(n)$ *we have*

- *if* $\Delta \vdash \delta : \Gamma$, *then* $[\![\delta]\!]^{[\![\Delta]\!],[\![\Gamma]\!]} \downarrow$,
- *if* $\Delta \vdash \delta \equiv \delta' : \Gamma$, *then* $[\![\delta]\!]^{[\![\Delta]\!],[\![\Gamma]\!]} = [\![\delta']\!]^{[\![\Delta]\!],[\![\Gamma]\!]}$.

4.3 The proof of the initiality theorem

With the results from the previous section, we can define a contextual morphism $\mathcal{C}(\mathrm{MLTT}) \to \mathcal{C}$. This finishes the first step in the proof of Theorem 4.0.1.

Lemma 4.3.1. *There exists a contextual morphism* $\mathcal{C}(\mathrm{MLTT}) \to \mathcal{C}$.

Proof. The functor acts on objects by

$$[\Gamma] \mapsto [\![\Gamma]\!]_{\mathsf{Ctx}}\,.$$

This is well-defined by Corollary 4.2.10. It acts on morphism by

$$[\delta] \mapsto [\![\delta]\!]^{[\![\mathrm{dom}(\delta)]\!],[\![\mathrm{cod}(\delta)]\!]}_{\mathsf{CtxMor}}\,.$$

This is well-defined by Corollary 4.2.11 and Corollary 4.2.10. It remains to show that all structure is preserved on the nose. Some of the basic structure has already been taken care of in the previous section. The remaining structure, including the ones coming from the logical rules, follows readily. This relies on the general shape of the additional structure on $\mathcal{C}(\mathrm{MLTT})$ defined in Theorem 3.3.11. For example, given an object $[\Gamma]$ the functor just defined acts on the Nat-type structure on $\mathcal{C}(\mathrm{MLTT})$ by

$$\mathrm{NatStr}([\Gamma]) = [\Gamma, \mathbf{N}] \mapsto [\![\Gamma, \mathbf{N}]\!]_{\mathsf{Ctx}} = [\![\mathbf{N}]\!]^{[\![\Gamma]\!]_{\mathsf{Ctx}}}_{\mathsf{Ty}} = \mathrm{NatStr}([\![\Gamma]\!]_{\mathsf{Ctx}}).$$

Similarly, if $[\mathrm{id}, u] : [\Gamma] \to [\Gamma, \mathbf{N}]$ is a term morphism in $\mathcal{C}(\mathrm{MLTT})$ we find

$$\begin{aligned}\mathrm{sucStr}([\Gamma], [\mathrm{id}, u]) = [\mathrm{id}, \mathbf{suc}(u)] &\mapsto [\![\mathrm{id}, \mathbf{suc}(u)]\!]^{[\![\Gamma]\!],[\![\Gamma,\mathbf{N}]\!]}_{\mathsf{CtxMor}} \\ &= [\![\mathbf{suc}(u)]\!]^{[\![\Gamma]\!]_{\mathsf{Ctx}}}_{\mathsf{Tm}} \\ &= \mathrm{sucStr}([\![\Gamma]\!]_{\mathsf{Ctx}}, [\![u]\!]^{[\![\Gamma]\!]_{\mathsf{Ctx}}}_{\mathsf{Tm}}) \\ &= \mathrm{sucStr}([\![\Gamma]\!]_{\mathsf{Ctx}}, [\![\mathrm{id}, u]\!]^{[\![\Gamma]\!],[\![\Gamma,\mathbf{N}]\!]}_{\mathsf{CtxMor}})\end{aligned}$$

using Corollary 4.2.5. We refer to the formalization for the details. □

We will now continue with the second step. This only requires the following lemma.

Lemma 4.3.2. *For any two contextual morphism* $F, G : \mathcal{C}(\mathrm{MLTT}) \to \mathcal{C}$ *and an object* $[\Gamma]$ *such that* $F([\Gamma]) = G([\Gamma])$ *we have:*

- *If* $\Gamma \vdash A$ *then*

$$F([\Gamma, A]) = G([\Gamma, A]).$$

- *If* $\Gamma \vdash u : A$ *then*
$$F([\mathrm{id}, u]) = G([\mathrm{id}, u]).$$
where $[\mathrm{id}, u] : [\Gamma] \to [\Gamma, A]$ *is as in Example 3.2.4.*

Proof. By a mutual induction on the relevant derivation and noting that the full structure on $\mathcal{C}(\mathrm{MLTT})$ is precisely given by objects and morphisms of this kind, hence F and G preserve them on the nose and the induction hypotheses takes care of the rest. We highlight two examples. For the **N**-FORM we find

$$\begin{aligned} F([\Gamma, \mathbf{N}]) &= F(\mathrm{NatStr}([\Gamma])) \\ &= \mathrm{NatStr}(F([\Gamma])) \\ &= \mathrm{NatStr}(G([\Gamma])) \\ &= G(\mathrm{NatStr}([\Gamma])) \\ &= G([\Gamma, \mathbf{N}]). \end{aligned}$$

For **suc**-INTRO we find

$$\begin{aligned} F([\mathrm{id}, \mathbf{suc}(u)]) &= F(\mathrm{sucStr}([\Gamma], [\mathrm{id}, u])) \\ &= \mathrm{sucStr}(F([\Gamma]), F([\mathrm{id}, u])) \\ &= \mathrm{sucStr}(G([\Gamma]), G([\mathrm{id}, u])) \\ &= G(\mathrm{sucStr}([\Gamma], [\mathrm{id}, u])) \\ &= G([\mathrm{id}, \mathbf{suc}(u)]). \end{aligned}$$

□

The uniqueness now follows without too much effort.

Corollary 4.3.3. *Any two contextual morphisms* $F, G : \mathcal{C}(\mathrm{MLTT}) \to \mathcal{C}$ *are equal.*

Proof. By Lemma 3.3.14 we know it is enough to show that F and G agree on objects and morphisms. This is done by structural induction on contexts and context morphisms. Objects works without any effort by Lemma 4.3.2. For morphisms, we can factor

$$[\delta, t] = \mathrm{q}(\mathrm{p}_{\mathrm{cod}(\delta,t)} \circ [\delta, t], [\mathrm{cod}(\delta), A]) \circ \mathrm{s}_{[\delta,t]} = \mathrm{q}([\delta], [\mathrm{cod}(\delta), A]) \circ \mathrm{s}_{[\delta,t]}$$

by the basic properties of a contextual category and the syntactic equality

$$(\mathrm{wMor}_0(\mathrm{id}))[\delta, t] = \mathrm{id}[\delta] = \delta.$$

Since F and G preserve composition, the q operation and objects, the left side is taken care of by the induction hypotheses. Recall from Lemma 3.1.7 that by definition $\mathrm{s}_{[\delta,t]} = [\mathrm{id}, t]$ and as such the result follows from Lemma 4.3.2. □

Remark 4.3.4. The proofs above are straightforward, but our current formalization fails to type check with a reasonable amount of RAM. This is again due to the complexity of $d_{\mathbf{refl}}$. The solution we have implemented to deal with the term model, Remark 3.3.12, has been sufficient for these files as well. We will discuss this further once we discus the formalization in the next chapter.

Combining Lemma 4.3.1 and Corollary 4.3.3 results in the proof of the initiality theorem for MLTT: Theorem 4.0.1.

5. Formalization

In this chapter we will discuss various parts of our formalization. We will discuss how much of Agda we have used and give a general overview of the files and how the reader can run the formalization on their own system. We will also touch upon the performance and in particular highlight the current issue with memory use.

5.1 Agda

The current formalization runs on the development version of Agda, to be precise version 2.6.1-0c79dd4 (February 29, 2020). In terms of built-in modules we use:

- Agda.primitive,
- Agda.builtin.Nat,
- Agda.builtin.List,
- Agda.builtin.Bool,
- Agda.builtin.Unit,
- Agda.builtin.String,
- Agda.builtin.Reflection.

We expect only the last of these to require some explanation. Reflection is a way to generate and type check Agda code automatically[1]. This is particular useful for defining function on inductively defined data types that share a common pattern among most if not all of the inductive clauses. An example is our heuristic definition of weakening Definition 2.2.3 and substitution Definition 2.2.12. At the moment we use reflection only in the development of syntax, but it is expected to also be useful in other parts of the formalization.

[1]Documentation on reflection is available at https://Agda.readthedocs.io/en/v2.6.1/language/reflection.html.

It is worth noting that one can force Agda to print code generated by reflection by adjusting the verbosity levels.

The options we use are prop and rewriting. The option prop turns on the hierarchy of universes of strict propositions as mentioned in the introduction.[1] Recall that this is based on [GCST19]. The option rewriting allows the user to implement new computation rules into Agda. We use it to add the computation rule for quotients. Built-in Agda features we use are for example implicit arguments and generalized variables.

We postulate a number of additional axioms.

- Function extensionality,
- Propositional extensionality for types of sProp,
- The formation, introduction, elimination and computation rules for quotient types,
- An operation erase that for P : sProp, takes any element of P and produces a generic element of P (disregarding any computational content from the original element). Under the hood, this is implemented using an unsound assumption, discussed in the following section.

Finally, it is worth noting that for the development we consider the equality type to be sProp-valued. This type is equivalent to the truncation of the ordinary identity type (by propositional extensionality). This can be understood as the formalization only requiring the *mere* existence of equalities involved. However, formally verifying that the formalization does not rely on UIP is left for future investigation.

The erase function

As mentioned above, erase has been a tool for erasing computational content which would otherwise cause memory issues during type checking. Precisely, given P : sProp and and $x : P$ it produces an element $\mathsf{erase}(x) : P$, but with the property that the type checker will no longer inspect the element x in $\mathsf{erase}(x)$. The goal is to keep the *propositional* content, but erase the *computational* content. Note that $\mathsf{erase}(x) = x$ even *judgmentally* by the fact that sProp is a universe of *strict* propositions.

Moreover, the use of erase is in line with the philosophy behind a universe of propositions: inhabitants of propositions should not have computational

[1] Documentation can be found https://Agda.readthedocs.io/en/v2.6.1/language/prop.html.

meaning i.e. propositions are *proof-irrelevant*. In other words, once a proposition is shown to be true, the particular inhabitant should not matter. This philosophy is emphasized by the implementation of sProp in Agda, where inhabitants of propositions are shown by an underscore during development instead of their actual value.

An operation of this type is clearly sound, but to achieve the desired computational result it is implemented by postulating a generic element for any proposition. It should be clear that the postulation of an inhabitant of any proposition produces an inconsistent system. However, the author is confident that the particular implementation of erase and the use of it is safe and justified, although not formally verified. This is similar to the use of type-in-type in the UnitMath library.

To be precise about where we use it: erase is used only on the witness of the derivability of a context or context morphism when constructing objects or morphisms of the term model. This turns out to save enough memory to allow Agda to type check the entire formalization.

However, the author still has concerns regarding the need for such an erase function which demands future investigation both by having a closer look at the implementation of our formalization, discussions with the developers of Agda and writers of [GCST19]. Sadly, this was beyond the time available for this licentiate thesis.

Even with the inclusion of erase, the minimum RAM a station is required to ensure the entire formalization can be type checked is 25 GB. This has also been the limit of the RAM available to the author. It is expected that if enough additional RAM is available, the erase function is no longer required. However, one of the goals of the formalization was to ensure that it can be type checked on most personal computers. For that reason the aim for future development is more towards reducing the total amount of RAM required, and thus making erase no longer necessary, rather than finding the correct amount of RAM needed to type check the current version (without erase).

5.2 Outline of the files

In this section we will go through all the files in our repository and briefly mention their content.

- *common.agda*: In this file we define most of the basic machinery such as equality, dependent pairs and finite sets and show some of their properties. We set the built-in equality of Agda to be sProp-valued. This allows us to use rewrite with it, which enables some syntactic sugar in the

development of the syntax.[1] This should not be confused with the option rewriting. This is also the file in which we define the partiality monad and set up the do-notation which is common for the use of monads. For details about this presentation of partial operations we refer again to [Kna18].

- *typetheory.agda*: Here we define the inductive data types of raw type and term expressions. When someone is interested in including/excluding certain constructors, this is the place to start.

- *reflection.agda*: A technical file to set up the machinery of reflection. Noticeable functions that are already produced here, are the various ap-... functions for type and term constructions.

- *syntx.agda*: Here we define contexts and contexts morphisms and the various operations on syntax. This is also the place we show the syntactic equalities. Some of the parts in this file are generated by reflection. We also use the syntactic sugar provided by rewrite at various places. This could be down without it.

- *rules.agda*: Here we define judgments, deduction rules and prove the admissible rules. We have chosen derivability to be sProp-valued. An alternative would be to define them to be Set-valued and truncate them only later.

- *contextualcat.agda*: Here we define contextual categories, the core structure they possess and verify the semantic equalities. We also define all the additional structure a contextual category can have related to the logical rules.

- *contextualcatmor.agda*: We define the notion of contextual morphisms between fully structured contextual categories and show two such morphisms are equal if they agree as functors.

- *quotients.agda*: Here we define our quotient types. We use pathover to deal with fibred paths over proof-irrelevant equalities. As such, this file also includes various results about pathover and shows that the quotients are effective.

- *termmodel-common.agda*: Here we define the objects and morphisms of the term model, followed by a couple of helper functions. This is also the place we postulate (privately) a generic element of any type

[1] Documentation on rewrite is available at https://agda.readthedocs.io/en/v2.6.1/language/with-abstraction.html#with-rewrite.

in sProp and define the erase function. We use this to then erase the inhabitant showing a context or context morphism is derivable when building objects or morphisms. We could have waited with this until the file termmodel - id.agda, but choice to be uniform in our development of the term model instead.

- *termmodel-synccat.agda*: In this file we define the term model and show that it satisfies the basic structure of a contextual category. Afterwards, we give a list of lemmas that allows the additional structure on the term model to be defined in a uniform way. In particular, we set up specialized quotient eliminators.

- *termmodel-*.agda*: Replacing the * with any of the type constructor gives the file in which the additional structure related to this type is defined.

- *termmodel-id.agda* and *termmodel-id2.agda*: We highlight these particular files for the explicit need of erase. Even with erase, we are forced to split up this definition into two files.

- *termmodel.agda*: Here we group all of the constructors together and show that the term model is fully structured.

- *partialinterpretation.agda*: We define the partially interpretation function. The use of the partiality monad and do-notation allows one to read this definition intuitively.

- *interpretationweakening.agda*: Here we deal with the interpretation of weakening and show that the interpretation matches with the semantics.

- *interpretationsubstitution.agda*: Here we deal with the interpretation of substitution and show that the interpretation matches with the semantics.

- *totality.agda*: Combining the previous files we manage to show that the partial interpretation function is defined on well-formed syntax.

- *initiality-existence.agda*: We define the functor out of the term model into any other fully structured contextual category.

- *initiality-uniqueness.agda*: We show that any two functors from the term model into any other fully structured contextual category are equal.

5.3 On running the formalization yourself

We will now describe the steps required for anyone interested in running the formalization on their own station. Be aware that to guarantee the entire formalization can be type checked, the system is required to have at least 25 GB of RAM available. The bottle neck is the file initiality - uniqueness.agda.

- First, one needs to install the development version of Agda. Instruction can be found at the bottom of https://agda.readthedocs.io/en/v2.6.1/getting-started/installation.html. The formalization runs on version 2.6.1-0c79dd4 (February 29, 2020).
- Then, one should clone the github repository of our formalization. The repository is found at https://github.com/guillaumebrunerie/initiality and the particular commit this thesis is based is commit 17c2477 (March 27, 2020).
- When running Agda on the formalization via terminal, it is recommended to run it including the options:

  +RTS -M25G -RTS

  This ensures that Agda will not use more than 25 GB of RAM. In case the user works on a station that has more RAM, change the number 25 accordingly. If these restrictions are not used, we have experienced that Agda can crash the system.

  Similarly, if the user is planning to work with Emacs, it is recommended to include the following line as a custom-set-variables in their .emacs file:

  '(agda2-program-args (quote ("+RTS" "-M25G" "-RTS"))).
- Now, one should be ready to let Agda type check the formalization either via terminal or opening Emacs. It should be pointed out not to immediately run Agda on initiality - existence.agda or initiality - uniqueness.agda. Agda will then rightfully type check all of the other files they depends on first, but it will not garbage collect the memory each time it is finished with a particular file. However, once a particular file has been successfully checked by Agda separately beforehand, it does not need to check it again when starting a file that depends on it.

  Because of this we advice the reader to type check the file termmodel - synccat.agda first, which should take at most 5 minutes. After this the files termmodel - id.agda and termmodel - id2.agda can be checked in

that order, taking about 35 minutes and 10 minutes, respectively. Then, termmodel.agda and totality.agda can be checked, both taking roughly 10 minutes. Finally, one can check initiality - existence.agda and initiality - uniqueness.agda. The former takes about 10 minutes, while the latter takes at most 40 minutes.

6. Future Directions

We end this thesis by listing a few directions of potential future research related to this thesis.

Formalize results depending on initiality

There are already a number of results in the literature that rely on the initiality theorem. Now that there is a formalized proof of initiality, it might be more tempting to formalize any one of these results as well. A prime example would be specific models of interest such as constructing cubical sets as a contextual category.

Extending or changing the type theory

We currently have a proof for the initiality theorem for the system we called MLTT. However, it is not uncommon to be interested in slightly different systems, either having different type constructors or different axioms. For instance, one might want to add W-types and univalence or remove certain η rules.

It would be a nice feature if the formalization has a way to 'turn off' certain type constructors and allow for easily adding new ones. At the moment, the first can be done by going through all the files and commenting out all the occurrences of a particular constructor, while the second can be done by adding the type constructor to the file typetheory.agda and going through the rest of the files to add the required extensions. However, the moment one file is altered, any other file that depends on it will start to rightfully complain when type checking. So, when adding additional constructors to the formalization, one can no longer run the final initiality file until the change has been pushed through in its entirety.

Initiality is a result that is well-suited with the philosophy of building up the type theory one constructor at a time. If this could be reflected in the way the files are set up, e.g. even when extending the type theory the initiality file still type-checks for the smaller theory, that would be a great improvement. In the same vein, one would like each of the constructors to have its own file, which includes all the steps in the proof of initiality for that particular constructor.

The author is currently unaware if it is possible to set up anything of that kind in a proof assistant like Agda.

Automatization

A lot of the constructions are independent of a given type constructor, modulo maybe a few places where one needs to address its particular characteristics. It would be excellent if, whenever one adds a new constructor to the theory, most if not all of the generic code that is required is generated automatically. As mentioned in the previous chapter, we are currently applying a feature of Agda called reflection for some of the definition and lemmas about syntax. This has been working out excellently and it would be interesting to see how far one can go using this.

Changing the categorical semantics

For our categorical semantics, we have used contextual categories. However, in the literature various other categorical structures have been studied for this purpose. It would be an interesting investigation to observe benefits and downsides of proving the initiality theorem for any of these other semantics by formalizing them.

As mentioned in the introduction, in the current project from which our formalization stems, a second formalization is being developed by Peter LeFanu Lumsdaine and Anders Mörtberg. They have chosen COQ as a proof assistant and category with attributes as categorical semantics. It will be interesting to observe the kind of differences and similarities they encounter.

References

[AAD07] 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 (MFPS XXIII). 23

[AL19] Benedikt Ahrens and Peter LeFanu Lumsdaine, *Displayed categories*, Log. Methods Comput. Sci. **15** (2019), no. 1, Paper No. 20, 18. 17

[ALV18] Benedikt Ahrens, Peter LeFanu Lumsdaine, and Vladimir Voevodsky, *Categorical structures for type theory in univalent foundations*, Log. Methods Comput. Sci. **14** (2018), no. 3, Paper No. 18, 18. 16

[Bar12] Bruno Barras, *Semantical investigations in intuitionistic set theory and type theories with inductive families*, Habilitation thesis, Université Paris **7** (2012).

[Ber18] Benno van den Berg, *Path categories and propositional identity types*, ACM Transactions on Computational Logic (TOCL) **19** (2018), no. 2, 1–32. 52

[BGL+16] Andrej Bauer, Jason Gross, Peter LeFanu Lumsdaine, Mike Shulman, Matthieu Sozeau, and Bas Spitters, *The HoTT Library: A formalization of homotopy type theory in Coq*, preprint (2016), arXiv:1610.04591. 13

[Car86] John Cartmell, *Generalised algebraic theories and contextual categories*, Ann. Pure Appl. Logic **32** (1986), no. 3, 209–243. 49, 50

[Cas14] Simon Castellan, *Dependent type theory as the initial category with families*, Internship Report, Chalmers University of Technology (2014). 16

[CGH14] Pierre-Louis Curien, Richard Garner, and Martin Hofmann, *Revisiting the categorical interpretation of dependent type theory*, Theoretical Computer Science **546** (2014), 99–119. 75

[CH88] Thierry Coquand and Gérard Huet, *The Calculus of Constructions*, Information and Computation **76** (1988), no. 2, 95–120. 19

[Cur93] Pierre-Louis Curien, *Substitution up to isomorphism*, Fund. Inform. **19** (1993), no. 1–2, 51–85. 75

[dB73] Nicolaas Govert de Bruijn, *AUTOMATH, a language for mathematics*, Les Presses de l'Université de Montréal, Montreal, Que., 1973, Séminaire de Mathématiques Supérieures, No. 52 (Été 1971). 13

[GCST19] Gaëtan Gilbert, Jesper Cockx, Matthieu Sozeau, and Nicolas Tabareau, *Definitional Proof-Irrelevance without K*, Proceedings of the ACM on Programming Languages (2019), 1–28. 19, 86, 87

[Hof95] Martin Hofmann, *On the interpretation of type theory in locally Cartesian closed categories*, Computer science logic (Kazimierz, 1994), Lecture Notes in Comput. Sci., vol. 933, Springer, Berlin, 1995, pp. 427–441.

[Hof97] ———, *Extensional constructs in intensional type theory*, CPHC/BCS Distinguished Dissertations, Springer-Verlag London, Ltd., London, 1997. 16, 55, 56, 62, 63, 75, 78

[Jac99] Bart Jacobs, *Categorical logic and type theory*, Studies in Logic and the Foundations of Mathematics, vol. 141, North-Holland Publishing Co., Amsterdam, 1999.

[KL18] Krzysztof Kapulkin and Peter LeFanu Lumsdaine, *The homotopy theory of type theories*, Adv. Math. **337** (2018), 1–38.

[KL20] ———, *The simplicial model of univalent foundations (after Voevodsky)*, Journal of the European Mathematical Society (2020), to appear. 13, 62, 63, 72

[Kna18] Cory Knapp, *Partial functions and recursion in univalent type theory*, Ph.D. thesis, University of Birmingham, 2018. 76, 88

[Li15] Nuo Li, *Quotient types in type theory*, Ph.D. thesis, University of Nottingham, 2015. 19

[Luo94] Zhaohui Luo, *Computation and reasoning*, vol. 20, Oxford University Press New York, 1994. 40

[Mai99] Maria Emilia Maietti, *About effective quotients in constructive type theory*, Types for proofs and programs (Irsee, 1998), Lecture Notes in Comput. Sci., vol. 1657, Springer, Berlin, 1999, pp. 164–178.

[ML75] Per Martin-Löf, *An intuitionistic theory of types: predicative part*, Logic Colloquium '73 (Bristol, 1973), 1975, pp. 73–118. Studies in Logic and the Foundations of Mathematics, Vol. 80. 13, 17

[ML84] ———, *Intuitionistic type theory*, Studies in Proof Theory. Lecture Notes, vol. 1, Bibliopolis, Naples, 1984, Notes by Giovanni Sambin. 13, 17, 21

[NPS90] Bengt Nordström, Kent Petersson, and Jan M. Smith, *Programming in Martin-Löf's type theory, an introduction*, International Series of Monographs on Computer Science, vol. 7, The Clarendon Press, Oxford University Press, New York, 1990, An introduction. MR 1243882

[PV07] Erik Palmgren and Steve J. Vickers, *Partial horn logic and Cartesian categories*, Ann. Pure Appl. Logic **145** (2007), no. 3, 314–353.

[Str91] Thomas Streicher, *Semantics of type theory*, Progress in Theoretical Computer Science, Birkhäuser Basel, 1991. 14, 16, 49, 75

[Uem19] Taichi Uemura, *A General Framework for the Semantics of Type Theory*, preprint (2019), arXiv:1904.04097. 15

[UFP13] The Univalent Foundations Program, *Homotopy type theory — univalent foundations of mathematics*, The Univalent Foundations Program, Princeton, NJ; Institute for Advanced Study (IAS), Princeton, NJ, 2013. 13, 15, 17, 19, 38

[VAG+] Vladimir Voevodsky, Benedikt Ahrens, Daniel Grayson, et al., *UniMath — a computer-checked library of univalent mathematics*, available at https://github.com/UniMath/UniMath. 13

[Vel15] Niccolò Veltri, *Two set-based implementations of quotients in type theory*, Proceedings of the 14th Symposium on Programming Languages and Software Tools (SPLST'15), Tampere, Finland, October 9-10, 2015, CEUR Workshop Proceedings, vol. 1525, CEUR-WS.org, 2015, pp. 194–205. 19

[Voe06] Vladimir Voevodsky, *A very short note on the homotopy λ-calculus*, Unpublished note, https://www.math.ias.edu/~vladimir/Site3/Univalent_Foundations_files/Hlambda_short_current.pdf (2006). 13

[Voe09] ———, *Notes on type systems*, Unpublished notes, https://www.math.ias.edu/~dgrayson/Voevodsky-old-files/files/files-annotated/Dropbox/Unfinished_papers/Dynamic_logic/Stage_9_-2012_09_01/expressions_current.pdf (2009).

[Voe14] Vladimir Voevodsky, *C-system of a module over a monad on sets*, preprint (2014), arXiv:1407.3394. 49

[Voe15] Vladimir Voevodsky, *A C-system defined by a universe category*, Theory Appl. Categ. **30** (2015), Paper No. 37, 1181–1215. 49, 50

[Voe16] ———, *Subsystems and regular quotients of C-systems*, A panorama of mathematics: pure and applied, Contemp. Math., vol. 658, Amer. Math. Soc., Providence, RI, 2016, pp. 127–137. 49, 54, 55, 56, 58

[Yam17] Norihiro Yamada, *Categories with Dependence and Semantics of Dependent Types*, preprint (2017), arXiv:1704.04747. 16, 75

Index of Symbols

Index of Terms