UNIVERSITY OF AMSTERDAM

UvA-DARE (Digital Academic Repository)

Abstract and concrete type theories

Uemura, T.

Publication date
2021
Document Version
Final published version

Link to publication

Citation for published version (APA):
Uemura, T. (2021). *Abstract and concrete type theories*. [Thesis, fully internal, Universiteit van Amsterdam]. Institute for Logic, Language and Computation.

General rights
It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations
If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: https://uba.uva.nl/en/contact, or a letter to: Library of the University of Amsterdam, Secretariat, P.O. Box 19185, 1000 GD Amsterdam, The Netherlands. You will be contacted as soon as possible.

UvA-DARE is a service provided by the library of the University of Amsterdam (https://dare.uva.nl)

Download date:16 Sept 2026

Abstract and Concrete Type Theories

Taichi Uemura

Abstract and Concrete Type Theories

ILLC Dissertation Series DS-2021-09

Institute for Logic, Language and Computation

For further information about ILLC-publications, please contact

Institute for Logic, Language and Computation
Universiteit van Amsterdam
Science Park 107
1098 XG Amsterdam
phone: +31-20-525 6051
e-mail: illc@uva.nl
homepage: http://www.illc.uva.nl/

The investigations were supported by the research programme "The Computational Content of Homotopy Type Theory" with project number 613.001.602, which is financed by the Netherlands Organisation for Scientific Research (NWO).

Copyright © 2021 by Taichi Uemura

Printed and bound by Ipskamp Printing.

ISBN: 978-94-6421-376-8

Abstract and Concrete Type Theories

Academisch Proefschrift

ter verkrijging van de graad van doctor
aan de Universiteit van Amsterdam
op gezag van de Rector Magnificus
prof. dr. ir. K.I.J. Maex
ten overstaan van een door het College voor Promoties ingestelde
commissie, in het openbaar te verdedigen in de Agnietenkapel
op vrijdag 9 juli 2021, te 16:00 uur

door

Taichi Uemura

geboren te Mie, Japan

Promotiecommisie

Promotor:	prof. dr. S.J.L. Smets	Universiteit van Amsterdam
Co-promotor:	dr. B. van den Berg	Universiteit van Amsterdam
	prof. dr. T. Streicher	Technische Universität Darmstadt
Overige leden:	dr. B. Afshari	Universiteit van Amsterdam
	prof. dr. S.M. Awodey	Carnegie Mellon University
	prof. dr. B. Löwe	Universiteit van Amsterdam
	dr. P.L. Lumsdaine	Stockholm University
	prof. dr. Y. Venema	Universiteit van Amsterdam

Faculteit der Natuurwetenschappen, Wiskunde en Informatica

Contents

Acknowledgments **ix**

1 Introduction **1**
1.1 Type theories . . . 1
1.1.1 Semantics of type theories . . . 2
1.1.2 General type theories . . . 3
1.2 Abstract type theories . . . 5
1.2.1 Functorial semantics . . . 5
1.2.2 Type theories as categories . . . 5
1.2.3 Syntactic presentations of type theories . . . 6
1.2.4 Summary and related work . . . 7
1.3 Homotopy type theory . . . 8
1.3.1 Realizability models of homotopy type theory . . . 9
1.3.2 Impredicative universe . . . 9
1.3.3 Propositional resizing . . . 10
1.3.4 Church's Thesis . . . 10
1.3.5 Related work . . . 11
1.4 Higher dimensional type theories . . . 12
1.4.1 Coherence problems . . . 12
1.4.2 Solutions to coherence problems . . . 13
1.4.3 General coherence problems and ∞-type theories . . . 13
1.4.4 Related work . . . 14
1.5 Summary of contributions . . . 15
1.5.1 Origin of the material . . . 16

2 Preliminaries **19**
2.1 Foundations . . . 19
2.2 Type theory . . . 19
2.3 Category theory . . . 23

2.3.1 Higher categories . . . 25
2.3.2 Presheaves . . . 25
2.3.3 Compactly generated categories . . . 26
2.3.4 Exponentiable arrows . . . 28

3 Categories with representable maps 31
3.1 Natural models of type theory . . . 33
3.1.1 Discrete fibrations . . . 34
3.1.2 Modeling type theory . . . 36
3.1.3 Properties of representable maps of discrete fibrations . . . 39
3.2 Type theories . . . 41

4 Second-order generalized algebraic theories 47
4.1 Running example . . . 48
4.2 Syntax of SOGATs . . . 52
4.2.1 Signatures . . . 52
4.2.2 Expressions . . . 54
4.2.3 Substitutions . . . 57
4.2.4 Instantiations . . . 57
4.3 Inference rules of SOGATs . . . 58
4.3.1 Declarations . . . 58
4.3.2 Judgments . . . 62
4.3.3 Inference rules . . . 66
4.3.4 Derivations . . . 69
4.3.5 Well-formedness conditions . . . 71
4.4 Properties of derivations . . . 71
4.4.1 Stability under substitutions . . . 71
4.4.2 Stability under instantiations . . . 76
4.4.3 Contextual completeness . . . 77
4.5 Second-order generalized algebraic theories . . . 81
4.5.1 Well-ordered presentation . . . 81
4.5.2 Finitary pretheories . . . 83
4.6 Examples of SOGATs . . . 83
4.6.1 Martin-Löf type theory . . . 84
4.6.2 Extensions of Martin-Löf type theory . . . 89
4.6.3 Cubical type theory . . . 94
4.6.4 Second-order algebraic theories . . . 103
4.6.5 Generalized algebraic theories . . . 105
4.6.6 Bauer et al.'s general type theories . . . 106
4.7 Theories over a SOGAT . . . 107
4.8 Semantics of SOGATs . . . 112
4.8.1 Syntactic categories . . . 112
4.8.2 Interpretations . . . 114

4.8.3 Functorial semantics 128
4.8.4 The internal SOGAT of a CwR 133

5 The theory of type theories **137**
5.1 Theories and models 138
5.1.1 The internal language at work 139
5.1.2 Democratic models 141
5.1.3 The theory-model correspondence 142
5.2 The category of models of a type theory 143
5.2.1 Presentability of the category of models 143
5.2.2 The universal property of the category of models 146
5.3 The category of type theories 148
5.3.1 Slice type theories 149
5.3.2 Presentability of the category of type theories 149
5.4 The theory-model correspondence 150
5.4.1 The initial model 151
5.4.2 Syntactic models generated by compact theories 154
5.4.3 The equivalence of theories and democratic models 156

6 ∞-type theories **159**
6.1 The theory of ∞-type theories 162
6.1.1 Univalent representable maps 164
6.1.2 The representable map classifier 166
6.2 Type-theoretic structures 166
6.2.1 Finitely complete $(\infty, 1)$-categories 167
6.2.2 Locally cartesian closed $(\infty, 1)$-categories 171
6.2.3 ∞-type theories 172
6.3 Coherence problems 173
6.3.1 Coherence for comprehension categories 175
6.3.2 Coherence for finitely complete categories 182
6.3.3 Coherence for finitely complete $(\infty, 1)$-categories 183

7 Models of cubical type theory **189**
7.1 A general construction of a model of CTT 191
7.1.1 Axioms for modeling CTT 192
7.1.2 Fibrations 193
7.1.3 Type constructors 194
7.1.4 Universes 196
7.1.5 Higher inductive types 197
7.1.6 Discrete types 198
7.2 Internal cubical models 199
7.2.1 Internal presheaves 200
7.2.2 Lifting universes 201

7.2.3 Intervals . . . 203
7.2.4 Locally decidable propositions . . . 204
7.2.5 Cube categories . . . 206
7.2.6 W-types with reductions . . . 207
7.2.7 Constant and codiscrete presheaves . . . 208

8 Cubical assembly models of type theory 211
8.1 Assemblies . . . 212
8.2 Impredicative universe . . . 214
8.3 Failure of propositional resizing . . . 215
8.3.1 Uniform assemblies . . . 217
8.3.2 The counterexample . . . 219
8.4 Markov's Principle . . . 219
8.5 Church's Thesis . . . 220
8.5.1 Failure of Church's Thesis in internal cubical models . . . 221
8.5.2 Null types . . . 223
8.5.3 Church's Thesis in null types . . . 229

Bibliography 233

Acronym 249

Notation 251

Index 257

Samenvatting 261

Abstract 263

Acknowledgments

I would like to thank my supervisors Benno van den Berg and Thomas Streicher. Without their help this thesis would never have been completed. They made a lot of helpful comments on the earlier draft of the thesis to make it more readable and accessible to a broader audience.

Benno has been my main supervisor since I started my PhD. I really appreciate all his suggestions for my research, feedback on my papers and talks, and support for other activities.

I would like to thank Sonja Smets for being my promotor. I would also like to thank the members of the committee, Bahareh Afshari, Steve Awodey, Benedikt Löwe, Peter Lumsdaine, and Yde Venema.

I am also grateful to my collaborators, Andrew Swan, John Bourke, and Hoang Kim Nguyen. Working with them has been pleasant and inspiring experiences.

Over the years, I have benefited, both directly and indirectly, from interactions with people in the HoTT community and the wider logic, computer science, and category theory communities: Thorsten Altenkirch, Nathanael Arkor, Nick Bezhanishvili, Martin Bidlingmaier, Kenta Cho, Martijn den Besten, Jonas Frey, Dan Frumin, Soichiro Fujii, Daniel Gratzer, Chris Kapulkin, Frederik Lauridsen, Chaitanya Leena Subramaniam, Tadeusz Litak, Robert Passman, Kazuhiko Sakaguchi, Genki Sato, Michael Shulman, Bas Spitters, Niels van der Weide, and Norihiro Yamada.

Chapter 1

Introduction

In this thesis, we study *abstract* and *concrete type theories*. We propose an abstract notion of a type theory while we develop concrete syntactic presentations of type theories. We establish general results on abstract type theories while we study concrete models of a specific type theory to obtain consistency and independence results.

1.1 Type theories

A *type theory* is a formal system used as a foundation of mathematics, a basis for programming languages, and a basis for computer proof assistants. As a foundation of mathematics or a basis for computer proof assistants, a type theory checks the well-formedness of a mathematical expression and the correctness of a proof. As a basis for programming languages, a type theory checks the specification of a program.

The job of a type theory is to derive *judgments*. For example, most type theories can speak about a judgment of the form $a : A$ which is officially read as "*term* a has *type* A" but informally read as "a is an element of set A", "program a satisfies specification A", or "a is a proof of proposition A", according to the usage of the type theory. If a type theory derives a judgment $a : A$, then we understand that element a of set A is constructed, program a satisfying specification A is defined, or proposition A is proved. The judgments that a type theory can derive are controlled by *inference rules* associated to the type theory. A type theory checks whether a judgment can be derived using the inference rules.

A type theory may deal with other forms of judgments. In a *dependent* type theory, a type expression may contain term expressions as subexpressions, and thus one needs another form of a judgment A type meaning that A is a well-formed type expression. Some complex type theories such as two-level type theory [3, 7] and cubical type theory [41] have more forms of judgments. In this thesis, we understand type theories in a broad sense: a type theory is a formal system

specified by a grammar for building judgments and by a set of inference rules for deriving judgments. Here the form of a judgment is not restricted to the traditional $a : A$ but can be an arbitrary formal expression.

1.1.1 Semantics of type theories

For a type theory to be reliable, one has to prove that it does not derive wrong judgments. For example, as a foundation of mathematics or a basis for proof assistants, a type theory must not derive a contradiction. The correctness of a type theory could be proved by a syntactic argument, but the *semantic* approach is often more powerful. For example, if we construct a *model* of a type theory in which the judgment expressing a contradiction is interpreted as the empty set, then we immediately conclude that the type theory does not derive a contradiction.

The semantic approach to the study of a type theory has another advantage. We can use derivations in the type theory for constructing and reasoning about elements in a model of the type theory. Precisely, given a model of a type theory, we extend the type theory by adjoining elements in the model as constants. This extended theory, called the *internal language* of the model, admits the interpretation in the model, and thus any derivable judgment over the internal language gives rise to an element in the model or proves some property on an element in the model. This is particularly useful when the model comes from a *category*. Sometimes we have to draw a huge diagram to explain the construction of a certain morphism in a category, but a few lines will be enough in the internal language of the category.

The construction of the internal language is formulated as part of the correspondence between *theories* and *models*. By a *model* of a type theory, we mean a mathematical structure that interprets inference rules of the type theory. We also consider a *theory* over a type theory which would need to be explained. A type theory is specified by a grammar and a set of inference rules, but one can adjoin some symbols as atoms and judgments as axioms. By a *theory* over a type theory, we mean a set of symbols and judgments to be adjoined to the type theory. For example, first-order logic is a type theory consisting of the usual inference rules for logical connectives and quantifiers, and then a first-order theory is a theory over first-order logic since it is a set of operator and relation symbols and axioms. The internal language of a model of a type theory is a theory over the type theory consisting of elements in the model. For a good notion of a model of a type theory, we can conversely construct a model of the type theory from a theory over the type theory. This model usually consists of derivable judgments over the theory and is called the *syntactic model* generated by the theory. In the ideal situation, the construction of internal languages and the construction of syntactic models are mutually inverses, and thus theories and models are equivalent.

1.1.2 General type theories

The correspondence between theories and models is one of the most fundamental results in the semantics of type theories, and a lot of examples are found in the literature: simply typed lambda calculi and cartesian closed categories [106]; first-order theories and hyperdoctrines [150]; generalized algebraic theories and contextual categories [35]; Martin-Löf theories and locally cartesian closed categories [149]; and more [148, 46, 120, 16]. These results are proved individually but share the same idea: the syntactic model generated by a theory consists of derivable judgments over the theory; the internal language of a model consists of elements in the model. Thus, it is natural to ask if one can define a general notion of a type theory and establish the correspondence between theories and models uniformly for a wide range of type theories.

However, it is hard to say even what a general type theory is exactly, because the grammar and inference rules of a type theory are extremely flexible. We would not be able to establish the theory-model correspondence for a type theory with wild inference rules, and thus we have to limit a class of type theories. Furthermore, there can be several notions of a model of a type theory, and we have to choose one particular notion of a model to state the general theory-model correspondence. For example, display map categories [166], clans [94], categories with attributes [35], and categories with families [51] are all considered as models of a dependent type theory. Among these notions of a model, *categories with families* are the closest to the syntax of the type theory and considered as a canonical notion of a model of the type theory.

Hence, we would like to find a class of type theories such that for each type theory in the class, we can define a canonical notion of a model based on categories with families and establish the correspondence between theories and models. Some approaches to general type theories are proposed in the literature.

Logical frameworks

A successful approach to general type theories is *logical frameworks* [75, 133]. A logical framework is a type theory such that theories over it represent type theories. The idea is summarized in the slogan "*judgments-as-types*" [75]: a type in a logical framework expresses a judgment and a term expresses a derivation. Some logical frameworks are sufficiently expressive to define a wide range of type theories and logics including first-order and higher-order logic [75] and fragments of Martin-Löf type theory [133].

The semantics of type theories defined in a logical framework, however, has not received much attention. Since a logical framework is a type theory, one can consider models of a logical framework, but what we want is a notion of a model of a theory over a logical framework. One could define a model of a theory over a logical framework as a model of the logical framework equipped with additional

structure to interpret the theory, but this does not coincide with the usual notion of a model of a type theory because the interpretation of the logical framework is not part of the usual notion of a model of a type theory.

We also note that in the logical framework approach, we have to prove the *adequacy* of a logical framework presentation of a type theory. Suppose that we have a target type theory presented by a set of inference rules and could define a theory over a logical framework representing the target type theory. Working with the theory over the logical framework, we can use the full power of the logical framework, and thus more inference rules are available than the target type theory. The *adequacy theorem* asserts that these extra inference rules are a conservative extension of the target type theory. The adequacy theorem could be proved for individual logical framework presentations, but it is hard to even state the general adequacy theorem asserting that any theory over the logical framework is adequate: for this statement to make sense, we need a general notion of a type theory outside the logical framework.

Bauer et al.'s general type theories

Recently, Bauer, Haselwarter, and Lumsdaine [20] proposed a general definition of a type theory based on a careful analysis on the properties of inference rules of reasonable type theories. A type theory in their sense is defined as a set of inference rules satisfying certain conditions and coincides with the traditional presentation of a type theory, and thus there is no need to prove an extra adequacy theorem. The semantics of type theories in their sense based on categories with families is also being developed [115]. However, the possible forms of judgments are fixed, and thus one cannot define in their style some important type theories such as polymorphic type theory [72, 143, 73], pure type systems [18], two-level type theory [7, 3], and cubical type theory [41, 5]

Type theories as essentially algebraic theories

Another approach to general type theories is to regard a type theory as a special kind of *essentially algebraic theory*. An essentially algebraic theory, originally introduced by Freyd [59], consists of sorts, operator symbols, and equational axioms like an ordinary many-sorted algebraic theory, but the operator symbols can be *partial* operators whose domains are defined by equations. An inference rule of a type theory can be regarded as an algebraic operator that takes some input judgments and returns an output judgment. One can apply an inference rule only when the input judgments match a certain pattern, and thus an inference rule is a partial operator whose domain is defined by equations between judgments. In this way, any type theory generates a certain essentially algebraic theory.

Garner [68] showed that the category of generalized algebraic theories [35] is monadic over a presheaf category from which we can extract partial operators

involved with the basic structure of type dependency. Voevodsky [178] explicitly described the essentially algebraic theory for the basic structure of type dependency. Isaev [87] proposed a class of essentially algebraic theories and illustrated how a wide range of type theories are presented by essentially algebraic theories in that class. However, his definition has the same limitation as Bauer et al.'s: the possible forms of judgments are fixed.

1.2 Abstract type theories

The first topic of this thesis is a general definition of a type theory for which we establish the correspondence between theories and models. The idea of our definition of a type theory comes from Lawvere's *functorial semantics* of algebraic theories [108] and its variants [123, 1].

1.2.1 Functorial semantics

Lawvere's innovation in the semantics of algebraic theories is to replace theories by structured *categories* and models by structure-preserving *functors*. A (many-sorted) algebraic theory is usually presented by a set of sorts, a set of operator symbols, and a set of equational axioms. Given an equational presentation of an algebraic theory, we can construct a category with finite products such that arrows are represented by terms over the operator symbols and two parallel arrows are equal if they are represented by terms provably equal over the equational axioms. Finite products are used for representing the arity of a term: a term of sort B over the variables $\mathtt{x}_1 : A_1, \ldots, \mathtt{x}_n : A_n$ gives rise to an arrow $A_1 \times \cdots \times A_n \to B$ in the associated category with finite products. A model of an algebraic theory consists of a set for each sort symbol and an operator for each operator symbol such that the equational axioms hold. By the inductive definitions of terms and provable equality, any model of an algebraic theory gives rise to a functor preserving finite products from the associated category with finite products to the category of sets. Conversely, for any functor preserving finite products from the associated category with finite products to the category of sets, we have a model of the algebraic theory. Therefore, the study of models of an algebraic theory is equivalent to the study of functors preserving finite products. We may now *define* an algebraic theory to be a category with finite products and a model of an algebraic theory to be a functor valued in sets preserving finite products.

1.2.2 Type theories as categories

In the spirit of functorial semantics, we define a type theory to be a category equipped with certain structure and a model of a type theory to be a structure-preserving functor. Intuitively, arrows in such a category will be represented

by derivations in a type theory. The first attempt is to define a type theory to be a category with finite limits and some additional structure, because a wide range of type theories are special essentially algebraic theories [87] and essentially algebraic theories are identified with categories with finite limits [2]. However, the definition of a type theory as an essentially algebraic theory given in [87] is tied to a particular syntactic presentation, and it is not clear what the corresponding categorical structure would be.

To discover the missing categorical structure, we take a close look at models of a type theory. The notion of a model of a type theory we have in mind is categories with families, but we work with an equivalent notion, *natural models* introduced by Awodey [12]. The key observation is that a natural model of a type theory is a diagram in a presheaf category in which some maps are specified to be *representable* in the sense of Grothendieck. Thus, for models of a type theory to be functors, some arrows in the domain should be marked as "representable". We call such a category equipped with a class of "representable maps" a *category with representable maps*. It turns out that the operators in the essentially algebraic theory for a type theory [68, 178, 87] are obtained through the right adjoint of the pullback functor along a representable map. We now define a type theory to be a category with representable maps and a model of a type theory to be a functor to a presheaf category that carries representable maps in the type theory to representable maps of presheaves.

To state the correspondence between theories and models, we also have to introduce a notion of a theory over a type theory. Confusingly, a theory over a type theory is also defined as a *functor* but valued in sets rather than presheaves. Intuitively, a functor valued in sets assigns a set of symbols or axioms to each object in the type theory and thus matches our informal definition of a theory over a type theory: it consists of symbols and axioms.

We have defined a type theory, a model of a type theory, and a theory over a type theory in terms of categories and functors. Then the correspondence between theories and models is constructed using pure category theory, and no complicated syntactic argument is necessary. This correspondence induces an equivalence between the class of all theories and a class of models. We also give a characterization of that class of models.

1.2.3 Syntactic presentations of type theories

A type theory is now a certain structured category, but how is it related to the traditional presentation of a type theory? We certainly need a precise definition of a syntactic presentation such that any syntactic presentation induces a type theory and any type theory is induced by some syntactic presentation.

Since our type theories are certain structured categories, it is not difficult to design a *logical framework* corresponding to the categorical structure. Indeed, in the earlier paper [169], the author introduced a logical framework to define

syntactic presentations of type theories. However, like the usual logical framework presentation [75], *adequacy* is still a theorem that needs to be proved for each presentation of a type theory separately.

In this thesis, we give an alternative syntactic presentation of a type theory, extending the general definition of a type theory given by Bauer, Haselwarter, and Lumsdaine [20]. They defined a type theory as a set of inference rules which coincides with the traditional presentation of a type theory, and thus there is no need to prove an extra adequacy theorem. We modify their definition to fit our categorical notion of a type theory. It turns out that the new syntactic presentation of a type theory looks essentially the same as the logical framework presentation given in [169], but the expressive power is limited to eliminate the need for an extra adequacy theorem.

1.2.4 Summary and related work

We define a general type theory to be a certain structured category and a model of a type theory to be a structure-preserving functor. With this abstract definition, the correspondence between theories and models is established for any type theory. We also provide syntactic presentations of type theories to connect our abstract type theories with the traditional presentations of type theories.

There have been a lot of notions of a model of a type theory such as comprehension categories [90], display map categories [166], categories with families [51], tribes [94], and natural models [12]. All of them are models of a particular type theory. If we extend the type theory with some type constructors such as Π-types, then we have to invent another notion of a model, say natural models with Π-types. In contrast, our notion of a model is a model of a general type theory. The notion of a model of a particular type theory such as the type theory with Π-types is obtained as a special instance of the general notion.

Isaev [87] also proposed a definition of a general type theory and a notion of a model of a general type theory. In his definition, a type theory is a special essentially algebraic theory and a model of a type theory is just a model of the underlying essentially algebraic theory. His notion of a model is equivalent to a generalization of a contextual category [35], while our notion of a model is a generalization of a natural model [12] or, equivalently, category with families [51]. We consider that a natural model is closer than a contextual category to our informal definition of a model of a type theory as an interpretation of inference rules, because a contextual category also interprets the length of a context, but inference rules are usually insensitive to the length of a context. We also note that since contextual categories are equivalent to generalized algebraic theories [35], models of a type theory in Isaev's sense are equivalent to theories over a type theory in our sense.

The idea of using categorical structure to represent inference rules of a type theory also appears in the rule framework of Capriotti [33]. In his framework, a

type constructor is defined as an object in a certain category with families. A type constructor defined in his framework is not necessarily semantically well-behaved. Any type constructor defined in our framework is, in contrast, semantically well-behaved as we always have the correspondence between theories and models.

1.3 Homotopy type theory

Let us turn our attention to a specific type theory. The development of *homotopy type theory* [172] is an important factor in the growing interest in type theory. Homotopy type theory is based on Martin-Löf's *intensional type theory* [124] in which the identity between two elements a_1 and a_2 of a type A is represented by a type $\mathsf{Id}(a_1, a_2)$. A term of type $\mathsf{Id}(a_1, a_2)$ is considered as a witness to the identity between a_1 and a_2. There can be different terms of type $\mathsf{Id}(a_1, a_2)$, and thus $\mathsf{Id}(a_1, a_2)$ behaves like the set of morphisms from a_1 to a_2 in a groupoid or the space of *paths* from a_1 to a_2 in a space, rather than the proposition that a_1 and a_2 are equal. Indeed, Martin-Löf's intensional type theory admits interpretations in groupoids [81] and in spaces (Kan complexes, to be precise) [102]. More generally, Martin-Löf's intensional type theory is closely related to frameworks for abstract homotopy theory such as model categories [15, 8, 154] and $(\infty, 1)$-categories [103, 99].

Such a homotopy-theoretic interpretation of Martin-Löf's intensional type theory allows us to prove theorems in abstract homotopy theory inside the type theory, but since the type theory also admits a set-theoretic interpretation, we can only prove theorems that hold in both homotopy-theoretic and set-theoretic settings. To obtain truly homotopy-theoretic results, we have to extend the type theory with axioms that characterize homotopy-theoretic interpretations. Voevodsky's *univalence axiom*, which was formerly called *universe extensionality* in the groupoid interpretation [81], asserts that paths between two types are equivalent to invertible maps between the types. This conflicts with a set-theoretic interpretation and makes the type theory homotopy-theoretic. Postulating the univalence axiom is not enough to do homotopy theory inside the type theory because the type constructors in Martin-Löf's type theory create only discrete spaces. *Higher inductive types* are an extension of inductive types to allow us to construct types representing spaces such as the circle, the sphere, and the torus. With univalence and higher inductive types, we can state and prove theorems in homotopy theory inside the type theory. The basic example is the calculation of the fundamental group of the circle [112].

The univalence axiom and higher inductive types are thus the main theme in the study of homotopy type theory. In this thesis, we study the univalence axiom and higher inductive types in relation to other axioms and concepts in type theory including *impredicative universes*, the *propositional resizing axiom*, and *Church's Thesis*. We primarily aim to construct models of univalence with additional

structures and axioms, but some higher inductive types are also obtained through the construction. The main tool we use is *realizability* models of a type theory.

1.3.1 Realizability models of homotopy type theory

Realizability is a technique of constructing models of logics and type theories. It was originally introduced by Kleene [105] as an interpretation of Heyting arithmetic. The idea is to assign a set of "realizers" to each logical formula, and then the validity of a formula is verified by constructing a realizer for the formula. Given a notion of realizability, one can construct a category whose objects are called *assemblies* that has enough structure to interpret Martin-Löf's type theory [176], and we refer to this model as the *assembly model*.

The interpretation of Martin-Löf's type theory in assemblies is set-theoretic rather than homotopy-theoretic and does not validate the univalence axiom. To obtain a model of univalence, we *internalize* an existing construction of models of univalence in the assembly model. One problem is that the assembly model does not satisfy classical axioms such as the law of excluded middle and the axiom of choice, and thus we have to choose a construction of models that is valid in intuitionistic or constructive logic. Voevodsky's construction of the simplicial set model [102] is known to be non-constructive [24]. On the other hand, the construction of *cubical* set models [22, 23, 41] is described in constructive metalogic. The work by Orton and Pitts [134, 135] and Licata et al. [111] formally shows that the construction of cubical models can be internalized in an extension of Martin-Löf type theory. Applying the cubical construction to the assembly model, we have a model of univalence which we refer to as the *cubical assembly model*. In this thesis, we study the cubical assembly model to obtain consistency and independence results.

1.3.2 Impredicative universe

A feature of the assembly model is that it has an *impredicative universe*. By a *universe* in a type theory, we mean a type whose elements are types. A universe $\mathtt{u}$ is *impredicative* if for an *arbitrary* type A and for any type family $B : A \to \mathtt{u}$, the type $\prod_{\mathtt{x}:A} B(\mathtt{x})$ of functions that assign an element of $B(\mathtt{x})$ to each element $\mathtt{x}$ of A belongs to $\mathtt{u}$. An impredicative universe allows us an impredicative definition of a type. For example, we may define the type of polymorphic endofunctions $\prod_{\mathtt{A}:\mathtt{u}} \mathtt{A} \to \mathtt{A}$ as a type in $\mathtt{u}$, but then what we are defining belongs to the range of the quantification $\prod_{\mathtt{A}:\mathtt{u}}$. It is known that the assembly model has an impredicative universe [84, 114].

An impredicative universe is useful for representing inductive types in polymorphic type theory [73]. For example, the type of natural numbers is represented

by

$$\prod_{\mathtt{A:u}} (\mathtt{A} \to \mathtt{A}) \to (\mathtt{A} \to \mathtt{A})$$

whose elements are so-called Church numerals. In the context of homotopy type theory, Shulman [155] proposed the use of polymorphic encoding to represent *higher* inductive types, and Awodey, Frey, and Speight [13] and Speight [156] studied a refinement of polymorphic encoding using the full expressive power of dependent types.

We show that the cubical assembly model has an impredicative universe to justify the use of polymorphic encoding in homotopy type theory. By the method of Orton and Pitts [134, 135] and Licata et al. [111], we can transform the impredicative universe in the assembly model into a univalent universe in the cubical assembly model. This new universe is shown to be impredicative, and thus we have a universe that is both univalent and impredicative.

1.3.3 Propositional resizing

In homotopy type theory, a type is said to be a *proposition* if any two elements of the type are identical. For two nested universes $\mathsf{u} : \widehat{\mathsf{u}}$, we have a function from the universe $\mathsf{Prop}(\mathsf{u})$ of propositions in u to the universe $\mathsf{Prop}(\widehat{\mathsf{u}})$ of propositions in $\widehat{\mathsf{u}}$. The *propositional resizing axiom* [172, Section 3.5] asserts that this function $\mathsf{Prop}(\mathsf{u}) \to \mathsf{Prop}(\widehat{\mathsf{u}})$ is an equivalence for any nested universes $\mathsf{u} : \widehat{\mathsf{u}}$.

The propositional resizing axiom allows us an impredicative definition of a proposition. For example, the type $\prod_{\mathtt{P}:\mathsf{Prop}(\mathsf{u})} \mathtt{P} \to \mathtt{P}$ usually belongs to a larger universe, but using the propositional resizing axiom, we may replace it by an equivalent proposition that belongs to u. Propositional resizing is consistent with univalence, which follows from the consistency of the law of excluded middle with univalence [101].

In this thesis, we show the *independence* of the propositional resizing axiom from Martin-Löf's type theory with a univalent universe. Precisely, we show that for the impredicative universe u in the cubical assembly model, the function $\mathsf{Prop}(\mathsf{u}) \to \mathsf{Prop}(\widehat{\mathsf{u}})$ is not an equivalence for any larger universe $\widehat{\mathsf{u}}$.

1.3.4 Church's Thesis

Another important aspect of realizability is that it provides a model of *Constructive Recursive Mathematics*, a form of constructivism in which the validity of a proposition is justified by the existence of an *algorithm* or *recursive function*. We can choose recursive functions as realizers, and then the assembly model satisfies a proposition if it is realized by some recursive function. One of the defining characteristics of Constructive Recursive Mathematics is *Church's Thesis*, the principle asserting that any function on natural numbers is computable by some Turing

machine. The assembly model based on recursive realizability indeed satisfies Church's Thesis [176].

In this thesis, we construct a model of univalence that also satisfies Church's Thesis to show the consistency of Church's Thesis with univalence. One might expect that Church's Thesis holds in the cubical assembly model based on recursive realizability, but this is not the case. The homotopy-theoretic interpretation of the statement of Church's Thesis is quite different from the set-theoretic interpretation, and the existence of a realizer no longer validates Church's Thesis under the homotopy-theoretic interpretation.

To obtain a model of univalence and Church's Thesis, we use the theory of *modalities* in homotopy type theory developed by Rijke, Shulman, and Spitters [146]. Modalities in this context are more restrictive than those studied in modal logic and close to closure operators in topos theory [119]. The fundamental result of Rijke et al. is that given a family of propositions and a model of univalence, one can construct a new model of univalence consisting of those objects in the original model that "believe that the given propositions are all true" so that the given propositions are "forced" to be true in the new model. Applying their result to the statement of Church's Thesis and the cubical assembly model, we obtain a model of univalence and Church's Thesis consisting of those objects in the cubical assembly model that "believe that any function on natural numbers is computable by some Turing machine". Our final task is to show that this new model interprets the type of contradiction as the empty cubical assembly, and then Church's Thesis and univalence do not derive a contradiction.

1.3.5 Related work

Our realizability model relies on the cubical method developed by Orton and Pitts [134, 135] and Licata et al. [111]. Their method is based on Coquand's idea of using the internal type theory of a topos to formulate composition structure [43]. For categorical accounts of cubical methods, see [66, 147, 60, 10]. There are other approaches to realizability models of univalence. Van den Berg [174] took a globular approach and constructed a model of an impredicative and univalent universe. Although his model is truncated, his impredicative universe furthermore admits propositional resizing. Simplicial approaches are more difficult than cubical approaches, since Voevodsky's construction of the simplicial model [102] is non-constructive [24]. See [157] for an early attempt to a simplicial realizability model. Recent development of constructive simplicial models [76, 67, 63, 64, 175] would be the key to a simplicial realizability model.

The propositional resizing axiom was introduced by Voevodsky [177]. The law of excluded middle implies propositional resizing [172, Exercise 3.10]. The consistency of the law of excluded middle with univalence had been a folklore, and Kapulkin and Lumsdaine [101] recently gave a full proof. Consequently, propositional resizing is also consistent with univalence. To the author's knowledge,

our cubical assembly model is the first model of univalence that does not admit propositional resizing. De Jong and Escardó [49] studied propositional resizing in relation to principles in order theory such as the existence of a certain complete poset, Zorn's lemma, Tarski's greatest fixed point theorem, and Pataraia's lemma.

Church's Thesis has extensively been studied in constructive mathematics; see [e.g. 21, 167] for standard textbooks. To the author's knowledge, our cubical assembly model is the first model of univalence that satisfies Church's Thesis. The consistency of Church's Thesis with Martin-Löf's intensional type theory without univalence under the propositions-as-types interpretation was conjectured by Maietti and Sambin [122]. Since the interpretation of Church's Thesis there is different from the interpretation in univalent type theory, our result does not prove Maietti and Sambin's conjecture. In fact, their formulation of Church's Thesis is inconsistent with univalence, because their formulation is inconsistent with function extensionality and univalence implies function extensionality [172, Section 4.9]. Ishihara et al. [88] proved the consistency of Church's Thesis with a variant of Martin-Löf's intensional type theory. Recently, a proof of Maietti and Sambin's original conjecture was announced by Yamada [182].

1.4 Higher dimensional type theories

The final topic of this thesis is a higher-dimensional generalization of a type theory which we call an *∞-type theory*. ∞-type theories are a novel approach to *coherence problems* in (higher) categorical semantics of type theories.

1.4.1 Coherence problems

The idea of *categorical semantics* of type theories is to interpret types as objects and terms as morphisms in a category. This interpretation works very well for simple type theories [106], but a naive interpretation of a *dependent* type theory in a category causes a *coherence problem*, a mismatch between levels of equality. In a dependent type theory, a type expression can contain a term expression as a subexpression, and thus equality between terms can induce equality between types. Therefore, equality between types is a fundamental concept in a dependent type theory. On the other hand, it is not a good idea to speak about equality between objects in a category because it is not invariant under categorical equivalences. The correct notion of identity between objects in a category is isomorphisms. Hence, to interpret a dependent type theory in a category, we have to justify in some way interpreting equations between types as isomorphisms between objects.

Coherence problems become much more serious in *higher* categorical semantics of type theories. One of the ultimate goals of homotopy type theory [172] is

to interpret type theories in structured $(\infty, 1)$-categories, but the interpretation is far from obvious. Even the simplest coherence problem of interpreting Martin-Löf's intensional identity types in finitely complete $(\infty, 1)$-categories is open [100]. In an $(\infty, 1)$-category, besides equality between objects, equality between morphisms is also too strict, and the correct notion of identity between morphisms is homotopies. We thus have to justify interpreting equations between terms as homotopies between morphisms.

There is a stronger form of a coherence problem. The usual coherence problem is the problem of interpreting a type theory in "non-split" models where the notion of equality is weaker than the equality in the type theory. Once such an interpretation is justified, we would get the *internal language* of a non-split model. In some cases, one can conversely construct a non-split model of the type theory from a theory over the type theory. Then one may ask if this correspondence between theories and non-split models is an equivalence in some sense. For example, Clairambault and Dybjer [39, 40] established the biequivalence between theories over Martin-Löf extensional type theory and locally cartesian closed categories, and Kapulkin and Lumsdaine [100] conjectured that theories over Martin-Löf intensional type theory are equivalent to locally cartesian closed $(\infty, 1)$-categories in a suitable sense.

1.4.2 Solutions to coherence problems

There has been two approaches to coherence problems. Curien [47] and Hofmann [78] gave solutions to the coherence problem in Seely's interpretation of Martin-Löf's type theory in locally cartesian closed categories [149]. A comparison of their approaches is found in [48].

Curien [47] solved the coherence problem on the syntactic side. He modified the type theory by introducing a weaker notion of equality between types which behaves like isomorphisms between types. It is straightforward to interpret the modified type theory in categories. His key result is that any two proofs of equality between types in the modified type theory receives the same interpretation. This justifies assigning a unique isomorphism to each equation between types in the original type theory.

Hofmann [78], in contrast, took a semantic approach. His idea is to replace a non-split model by a split model that is equivalent to the non-split model in some sense. The interpretation of the type theory in a non-split model is justified by interpreting the type theory in the split model and passing through the equivalence.

1.4.3 General coherence problems and ∞-type theories

These approaches to coherence problems work for several type theories, but it is not easy to formulate general coherence problems and determine for which class

of type theories coherence theorems hold. A difficulty is that the notion of a non-split model is not formulated in the language of type theories and models of a type theory. The notion of a model of a type theory explained in Section 1.2 is a split model since it is a generalization of a natural model [12], and there is no obvious way to define a general notion of a non-split model of a type theory. To formulate a general coherence problem, we would like to speak about both type theories and non-split models in the same language.

In this thesis, we introduce a higher dimensional generalization of a type theory called an ∞-*type theory* to give a precise and unified formulation of a coherence problem. Let us call an ordinary type theory a 1-type theory for emphasis. Intuitively, an ∞-type theory is a kind of type theory where the notion of equality is replaced by homotopies. In this sense, an ∞-type theory is a higher dimensional extension of Curien's modified type theory, though ∞-type theories are defined as certain structured $(\infty, 1)$-categories and syntactic presentations of ∞-type theories have not been developed.

∞-type theories provide a precise and uniform formulation of general coherence problems. The problem in the usual formulation of a coherence problem is that the notion of a non-split model is not formulated in the language of type theories and models of a type theory. It turns out that non-split models of a 1-type theory are often naturally regarded as models of an ∞-type theory. Then the coherence problem is the problem of interpreting the 1-type theory in models of the ∞-type theory. Because 1-type theories are special ∞-type theories, general coherence problems are now formulated in the language of ∞-type theories and related concepts, which allows us to uniformly treat various coherence problems in both categorical and $(\infty, 1)$-categorical semantics of type theories.

Of course, this is just a reformulation of the coherence problem, and we still need some techniques to solve the problem, but this reformulation helps strengthen a coherence result to the correspondence between theories and non-split models. We demonstrate that if a certain key lemma is proved, then one can systematically establish the correspondence between theories and non-split models. We also discuss that both Curien's and Hofmann's approaches to coherence problems can be seen as proofs of the key lemma. The argument given there is quite general and works for both categorical and $(\infty, 1)$-categorical coherence problems. As an application to the $(\infty, 1)$-categorical semantics of type theories, we sketch a positive solution to the internal language conjectures for finitely complete and for locally cartesian closed $(\infty, 1)$-categories given by Kapulkin and Lumsdaine [100].

1.4.4 Related work

There are further approaches to coherence problems. Lumsdaine and Warren [116] introduced a new splitting technique that applies to homotopy theoretic models for which Hofmann's splitting does not work. Bidlingmaier [25] considered

interpreting a type theory in the category of all locally cartesian closed categories instead of one locally cartesian closed category. Bocquet [27] introduced a notion of higher congruence to tackle coherence and conservativity problems. The idea of higher congruence seems to be related to our ∞-type theories, but we leave it as future work.

Kapulkin and Lumsdaine [100] conjectured that the type theory with intensional identity types (and Π-types) provides internal languages for $(\infty, 1)$-categories with finite limits (and pushforwards). Kapulkin and Szumiło [103] partly solved this conjecture by giving a certain equivalence between comprehension categories with intensional identity types and finitely complete $(\infty, 1)$-categories, but a full proof of the equivalence of theories over the type theory and such comprehension categories is left open. One of the ultimate goals of homotopy type theory is to show that univalent type theory provides internal languages for yet-to-be-defined elementary $(\infty, 1)$-toposes; see [96, 153, 140] for proposed definitions of an elementary $(\infty, 1)$-topos. We expect that ∞-type theories are also useful for formulating and solving the internal language conjecture for elementary $(\infty, 1)$-toposes.

The coherence problem of interpreting a type theory in certain *presentable* $(\infty, 1)$-categories is solved by first replacing a presentable $(\infty, 1)$-category by a well-behaved model category [69, 151] and then applying the 1-categorical coherence theorem of Lumsdaine and Warren [116]. Since the syntactic $(\infty, 1)$-category of a type theory should not be presentable, this coherence result does not imply the stronger correspondence between theories and presentable $(\infty, 1)$-categories.

1.5 Summary of contributions

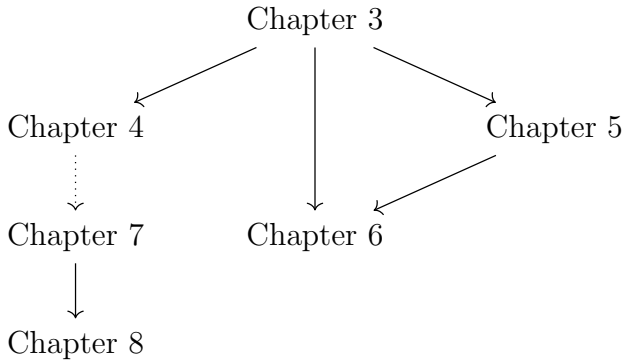
We introduce *categories with representable maps* as a general definition of type theories (Chapter 3) and establish the correspondence between theories and models for any type theory (Chapter 5). We also introduce syntactic presentations of categories with representable maps called *second-order generalized algebraic theories* (Chapter 4). Our notion of a type theory covers a wide range of existing and future type theories including Martin-Löf type theory [124, 133] and cubical type theory [41]. Our results formally justify the syntactic construction of models and the use of internal languages for these type theories.

We further generalize the notion of a type theory to a higher dimensional one which we call an *∞-type theory* (Chapter 6). ∞-type theories provide a precise and unified formulation of general *coherence problems* in both categorical and $(\infty, 1)$-categorical semantics of type theories. As an application, we describe a positive solution to Kapulkin and Lumsdaine's internal language conjectures for finitely complete $(\infty, 1)$-categories and for locally cartesian closed $(\infty, 1)$-categories [100].

Let us turn our attention to a specific type theory, *homotopy type theory*. We

study *realizability* models of homotopy type theory to obtain consistency and independence results (Chapters 7 and 8). Chapter 7 is preliminary to Chapter 8 and is devoted to reviewing the construction of cubical models given by Orton and Pitts [134, 135] and Licata et al. [111]. In Chapter 8, we show several results including the consistency of an *impredicative universe* with the univalence axiom, the unprovability of the *propositional resizing axiom* over Martin-Löf type theory plus the univalence axiom, and the consistency of *Church's Thesis* and *Markov's Principle* with the univalence axiom.

The essential dependency between chapters is as follows.



In Chapter 7, we use the syntactic presentation of cubical type theory described in Section 4.6.3, but the reader need not read all the details of Chapter 4 to understand the results in Chapters 7 and 8.

1.5.1 Origin of the material

Some chapters in this thesis are based on previous and on-going work as follows. In the case when the work is co-authored, all authors contributed equally. Chapters 3 and 5 are based on:

[169] T. Uemura. *A General Framework for the Semantics of Type Theory*. 2019. arXiv: `1904.04097v2`.

The proof of the results in Chapter 5 is the version given in [130]. Chapter 4 is new but closely related to [169, Section 5]. This chapter also contains an alternative proof of the results in:

[171] T. Uemura. *The Universal Exponentiable Arrow*. 2020. arXiv: `2001.09940v1`.

Chapter 6 is based on the unpublished manuscript:

[130] H. K. Nguyen and T. Uemura. *∞-type theories*. in preparation.

Chapters 7 and 8 are based on:

[170] T. Uemura. "Cubical Assemblies, a Univalent and Impredicative Universe and a Failure of Propositional Resizing". In: *24th International Conference on Types for Proofs and Programs (TYPES 2018)*. Ed. by P. Dybjer, J. E. Santo, and L. Pinto. Vol. 130. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019, 7:1–7:20. DOI: 10.4230/LIPIcs.TYPES.2018.7

[163] A. W. Swan and T. Uemura. *On Church's Thesis in Cubical Assemblies*. 2019. arXiv: 1905.03014v1.

Chapter 2

Preliminaries

In this chapter we fix some notations and terminologies and recall general facts used throughout the thesis.

2.1 Foundations

Unless otherwise mentioned, we freely use classical axioms such as the law of excluded middle and the axiom of choice. We also assume that there exist as many Grothendieck universes as we want. For category-theoretic results, just one or two Grothendieck universes will suffice. For constructing some models of a type theory with a countable chain of universes, we will need countably many Grothendieck universes.

2.2 Type theory

In this section, we remind the reader about basic concepts in the syntax and semantics of dependent type theory to motivate the definition of a model of a type theory given in Chapter 3 and the syntactic presentation of a type theory given in Chapter 4. See [e.g. 124, 125, 133, 160, 18, 79] for more information about dependent type theory.

A type theory is specified by a grammar, possible forms of judgments, and inference rules. Given a grammar, the set of *expressions* is determined. A traditional dependent type theory has the following four *judgment forms*

- $_$ type expressing that something is a *type*;
- $_ : A$ expressing that something is a *term* of type A;
- $A_1 \equiv A_2$ type expressing that two types A_1 and A_2 are *judgmentally equal*;

- $a_1 \equiv a_2 : A$ expressing that two terms a_1 and a_2 of type A are *judgmentally equal*,

where A, A_1, A_2, a_1, and a_2 are expressions and $_$ is a "hole". More complex type theories such as cubical type theory [41] have more judgment forms explained in later chapters. A *context* is a list of the form

$$\mathtt{x}_1 : A_1, \ldots, \mathtt{x}_n : A_n$$

where $\mathtt{x}_1, \ldots, \mathtt{x}_n$ are distinct variables and $A_1, \ldots, A_n$ are expressions. A *judgment* consists of a context, a judgment form, and, when the judgment form contains a hole, an expression filling the hole. For example, a judgment in a traditional dependent type theory is one of the following

$$\Gamma \vdash A\ \mathsf{type} \qquad \Gamma \vdash a : A \qquad \Gamma \vdash A_1 \equiv A_2\ \mathsf{type} \qquad \Gamma \vdash a_1 \equiv a_2 : A$$

where Γ is a context. Inference rules determine the set of *derivable judgments*. A *type* over a context Γ is an expression A such that the judgment $\Gamma \vdash A\ \mathsf{type}$ is derivable. A *term* of a type A over a context Γ is an expression a such that the judgment $\Gamma \vdash a : A$ is derivable. For contexts Γ and $\Delta = (\mathtt{y}_1 : B_1, \ldots, \mathtt{y}_n : B_n)$, a *substitution* $\Gamma \to \Delta$ is a list of terms denoted by

$$(\mathtt{y}_1 := b_1, \ldots, \mathtt{y}_n := b_n)$$

such that $\Gamma \vdash b_i : B_i \cdot (\mathtt{y}_1 := b_1, \ldots, \mathtt{y}_{i-1} := b_{i-1})$ is derivable, where $- \cdot (\mathtt{y}_1 := b_1, \ldots, \mathtt{y}_n := b_n)$ denotes the *action* of the substitution. For a type $\Gamma \vdash A\ \mathsf{type}$, the *context comprehension* is the context $\Gamma, \mathtt{x} : A$ with a fresh variable $\mathtt{x}$. We have a canonical substitution $(\Gamma, \mathtt{x} : A) \to \Gamma$ and a canonical term $\Gamma, \mathtt{x} : A \vdash \mathtt{x} : A$.

A *model* of a traditional dependent type theory should have the following structure to interpret components of the type theory:

- a category $\mathcal{C}$ to interpret contexts and substitutions;
- for any object $\Gamma \in \mathcal{C}$, a set $U(\Gamma)$ to interpret types over a context;
- for any object $\Gamma \in \mathcal{C}$ and any element $A \in U(\Gamma)$, a set $E(\Gamma, A)$ to interpret terms in a type over a context;
- action of morphisms in $\mathcal{C}$ on U and E to interpret substitution;
- for any object $\Gamma \in \mathcal{C}$ and any element $A \in U(\Gamma)$, an object $\{A\} \in \mathcal{C}$, a morphism $\mathrm{p} : \{A\} \to \Gamma$, and an element $\mathrm{q} \in E(\{A\}, A \cdot \mathrm{p})$ to interpret context comprehension.

These data are part of a *category with families* of Dybjer [51]. From the meaning of substitutions, it is required that the sections of $\{A\} \to \Gamma$ bijectively correspond to the elements of $E(\Gamma, A)$. Judgmental equalities $\Gamma \vdash A_1 \equiv A_2\ \mathsf{type}$ and $\Gamma \vdash$

$a_1 \equiv a_2 : A$ are interpreted as equalities in U and E, respectively. A model of a type theory with various *type constructors* is defined by adding algebraic operators between U and E corresponding to inference rules. We note that type constructors are usually preserved by substitution, and thus the corresponding algebraic operators must commute with the action of morphisms in $\mathcal{C}$. We recall some basic type constructors.

Dependent function types *Dependent function types*, *dependent product types*, or Π-*types* have the following rule for forming a Π-type

$$\frac{\Gamma \vdash A\ \mathsf{type} \qquad \Gamma, \mathtt{x} : A \vdash B\ \mathsf{type}}{\Gamma \vdash \Pi(A, \langle \mathtt{x} \rangle B)\ \mathsf{type}},$$

and rules expressing that the terms of $\Pi(A, \langle \mathtt{x} \rangle B)$ bijectively correspond to the terms of B under the assumption $\mathtt{x} : A$, schematically written as

$$\frac{\Gamma, \mathtt{x} : A \vdash b : B}{\Gamma \vdash \lambda(\langle \mathtt{x} \rangle b) : \Pi(A, \langle \mathtt{x} \rangle B)}.$$

Here the notation $\langle \mathtt{x} \rangle B$ expresses that the variable $\mathtt{x}$ is *bound* in the whole expression $\Pi(A, \langle \mathtt{x} \rangle B)$. The corresponding algebraic operators on a model are:

- a map Π that assigns an element $\Pi(\Gamma, A, B) \in U(\Gamma)$ to each triple (Γ, A, B) consisting of an object $\Gamma \in \mathcal{C}$ and elements $A \in U(\Gamma)$ and $B \in U(\{A\})$; and
- a bijection $\lambda : E(\{A\}, B) \cong E(\Gamma, \Pi(\Gamma, A, B))$ for any Γ, A, and B.

Dependent pair types *Dependent pair types*, *dependent sum types*, or Σ-*types* have the formation rule

$$\frac{\Gamma \vdash A\ \mathsf{type} \qquad \Gamma, \mathtt{x} : A \vdash B\ \mathsf{type}}{\Gamma \vdash \Sigma(A, \langle \mathtt{x} \rangle B)\ \mathsf{type}},$$

and elements of $\Sigma(A, \langle \mathtt{x} \rangle B)$ are pairs.

$$\frac{\Gamma \vdash a : A \qquad \Gamma \vdash b : B \cdot (\mathtt{x} := a)}{\Gamma \vdash (a, b) : \Sigma(A, \langle \mathtt{x} \rangle B)}$$

The corresponding algebraic operators on a model are:

- a map Σ that assigns an element $\Sigma(\Gamma, A, B) \in U(\Gamma)$ to each triple (Γ, A, B) consisting of an object $\Gamma \in \mathcal{C}$ and elements $A \in U(\Gamma)$ and $B \in U(\{A\})$; and
- a bijection $\sum_{a \in E(\Gamma, A)} E(\Gamma, B \cdot a) \cong E(\Gamma, \Sigma(\Gamma, A, B))$ for any Γ, A, and B, where $B \cdot a$ is the right action of the morphism $\Gamma \to \{A\}$ corresponding to the element $a \in E(\Gamma, A)$.

Identity types There are two kinds of *identity types* and both have the same formation rule.

$$\frac{\Gamma \vdash A\ \mathsf{type} \qquad \Gamma \vdash a_1 : A \qquad \Gamma \vdash a_2 : A}{\Gamma \vdash \mathsf{Id}(A, a_1, a_2)\ \mathsf{type}}$$

The corresponding algebraic operator on a model is a map Id that assigns an element $\mathsf{Id}(\Gamma, A, a_1, a_2) \in U(\Gamma)$ for each tuple (Γ, A, a_1, a_2) consisting of $\Gamma \in \mathcal{C}$, $A \in U(\Gamma)$, and $a_1, a_2 \in E(\Gamma, A)$. *Extensional identity types* are inhabited if and only if a_1 and a_2 are judgmentally equal.

$$\frac{\Gamma \vdash a_1 \equiv a_2 : A}{\Gamma \vdash \mathsf{refl} : \mathsf{Id}(A, a_1, a_2)}$$

The corresponding algebraic operator on a model is a bijection $\{* \mid a_1 = a_2\} \cong E(\Gamma, \mathsf{Id}(\Gamma, A, a_1, a_2))$ for any Γ, A, a_1, and a_2. *Intensional identity types* are defined as an *inductive type family* and has the following introduction, elimination, and equality rules.

$$\frac{\Gamma \vdash a : A}{\Gamma \vdash \mathsf{refl}(a) : \mathsf{Id}(A, a, a)}$$

$$\frac{\begin{array}{c}\Gamma, \mathtt{x}_1 : A, \mathtt{x}_2 : A, \mathtt{y} : \mathsf{Id}(A, \mathtt{x}_1, \mathtt{x}_2) \vdash C\ \mathsf{type} \\ \Gamma, \mathtt{x} : A \vdash c : C \cdot (\mathtt{x}_1 := \mathtt{x}, \mathtt{x}_2 := \mathtt{x}, \mathtt{y} := \mathsf{refl}(\mathtt{x})) \\ \Gamma \vdash a_1 : A \qquad \Gamma \vdash a_2 : A \qquad \Gamma \vdash p : \mathsf{Id}(A, a_1, a_2)\end{array}}{\Gamma \vdash \mathsf{elim}_{\mathsf{Id}}(\langle \mathtt{x}_1 \mathtt{x}_2 \mathtt{y} \rangle C, \langle \mathtt{x} \rangle c, p) : C \cdot (\mathtt{x}_1 := a_1, \mathtt{x}_2 := a_2, \mathtt{y} := p)}$$

$$\frac{\begin{array}{c}\Gamma, \mathtt{x}_1 : A, \mathtt{x}_2 : A, \mathtt{y} : \mathsf{Id}(A, \mathtt{x}_1, \mathtt{x}_2) \vdash C\ \mathsf{type} \\ \Gamma, \mathtt{x} : A \vdash c : C \cdot (\mathtt{x}_1 := \mathtt{x}, \mathtt{x}_2 := \mathtt{x}, \mathtt{y} := \mathsf{refl}(\mathtt{x})) \qquad \Gamma \vdash a : A\end{array}}{\Gamma \vdash \mathsf{elim}_{\mathsf{Id}}(\langle \mathtt{x}_1 \mathtt{x}_2 \mathtt{y} \rangle C, \langle \mathtt{x} \rangle c, \mathsf{refl}(a)) \equiv c \cdot (\mathtt{x} := a)}$$

These rules express that $\mathsf{Id}(A, -, -)$ is the type family on $(\mathtt{x}_1 : A, \mathtt{x}_2 : A)$ freely generated by $\mathsf{refl}(-)$. One can write down the corresponding algebraic operators on a model, but we omit them.

Inductive types An *inductive type* is a type freely generated by a set of *constructors*. The simplest inductive type is the *empty type* which is the inductive type with no constructor and has the following rules.

$$\frac{}{\Gamma \vdash 0\ \mathsf{type}} \qquad\qquad \frac{\Gamma, \mathtt{x} : 0 \vdash C\ \mathsf{type} \qquad \Gamma \vdash a : 0}{\Gamma \vdash \mathsf{elim}_0(\langle \mathtt{x} \rangle C, a) : C \cdot (\mathtt{x} := a)}$$

A more interesting inductive type is the *natural numbers type* which has the following rule.

$$\frac{}{\Gamma \vdash \mathsf{N}\ \mathsf{type}} \qquad \frac{}{\Gamma \vdash \mathsf{zero} : \mathsf{N}} \qquad \frac{\Gamma \vdash n : \mathsf{N}}{\Gamma \vdash \mathsf{succ}(n) : \mathsf{N}}$$

$$\frac{\begin{array}{c}\Gamma, \mathtt{n} : \mathsf{N} \vdash C\ \mathsf{type} \\ \Gamma \vdash c_0 : C \cdot (\mathtt{n} := \mathsf{zero}) \qquad \Gamma, \mathtt{n} : \mathsf{N}, \mathtt{x} : C \vdash c_1 : C \cdot (\mathtt{n} := \mathsf{succ}(\mathtt{n})) \qquad \Gamma \vdash n : \mathsf{N}\end{array}}{\Gamma \vdash \mathsf{elim}_{\mathsf{N}}(\langle \mathtt{n} \rangle C, c_0, \langle \mathtt{nx} \rangle c_1, n) : C \cdot (\mathtt{n} := n)}$$

$$\frac{\begin{array}{c}\Gamma, \mathtt{n} : \mathsf{N} \vdash C\ \mathsf{type} \\ \Gamma \vdash c_0 : C \cdot (\mathtt{n} := \mathsf{zero}) \qquad \Gamma, \mathtt{n} : \mathsf{N}, \mathtt{x} : C \vdash c_1 : C \cdot (\mathtt{n} := \mathsf{succ}(\mathtt{n}))\end{array}}{\Gamma \vdash \mathsf{elim}_{\mathsf{N}}(\langle \mathtt{n} \rangle C, c_0, \langle \mathtt{nx} \rangle c_1, \mathsf{zero}) \equiv c_0}$$

$$\frac{\begin{array}{c}\Gamma, \mathtt{n} : \mathsf{N} \vdash C\ \mathsf{type} \\ \Gamma \vdash c_0 : C \cdot (\mathtt{n} := \mathsf{zero}) \qquad \Gamma, \mathtt{n} : \mathsf{N}, \mathtt{x} : C \vdash c_1 : C \cdot (\mathtt{n} := \mathsf{succ}(\mathtt{n})) \qquad \Gamma \vdash n : \mathsf{N}\end{array}}{\Gamma \vdash \mathsf{elim}_{\mathsf{N}}(\langle \mathtt{n} \rangle C, c_0, \langle \mathtt{nx} \rangle c_1, \mathsf{succ}(n)) \equiv c_1 \cdot (\mathtt{n} := n, \mathtt{x} := \mathsf{elim}_{\mathsf{N}}(\langle \mathtt{n} \rangle C, c_0, \langle \mathtt{nx} \rangle c_1, n))}$$

It is straightforward to write down the corresponding algebraic operators on a model.

2.3 Category theory

We assume that the reader is familiar with basic concepts of category theory such as equivalences, adjunctions, and (co)limits. Standard references are [118, 11, 109, 144].

2.3.1. Definition. A *category* consists of a set $\mathcal{C}$ of *objects*, a set denoted by $\mathcal{C}(x, y)$ or $\mathrm{Hom}(x, y)$ for any pair of objects $x, y \in \mathcal{C}$ whose elements are called *arrows*, *morphisms*, or *maps* from x to y, an arrow $\mathrm{id}_x \in \mathcal{C}(x, x)$ for any object $x \in \mathcal{C}$ called the *identity on* x and an operator $\circ : \mathcal{C}(x_2, x_3) \times \mathcal{C}(x_1, x_2) \to \mathcal{C}(x_1, x_3)$ for any triple of objects $x_1, x_2, x_3 \in \mathcal{C}$ called the *composition* satisfying the associativity law and the unit law.

We fix some notations for constructions of categories.

- **Set** denotes the category of small sets, that is, sets in the first universe, and maps between them.
- For a category $\mathbf{C}$ of small sets with some structure, we write $\widehat{\mathbf{C}}$ for the category of sets in the next universe with that structure. For example, $\widehat{\mathbf{Set}}$ is the category of sets in the next universe.

- For two categories $\mathcal{C}$ and $\mathcal{D}$, we write $\mathbf{Fun}(\mathcal{C}, \mathcal{D})$ or $\mathcal{D}^{\mathcal{C}}$ for the category of functors from $\mathcal{C}$ to $\mathcal{D}$ and natural transformations between them.
- $\mathcal{C}^{\to}$ denotes the category of arrows and commutative squares in $\mathcal{C}$.
- For two categories $\mathcal{C}$ and $\mathcal{D}$ with finite limits, we write $\mathbf{Lex}(\mathcal{C}, \mathcal{D}) \subset \mathbf{Fun}(\mathcal{C}, \mathcal{D})$ for the full subcategory spanned by the left exact functors, that is, functors preserving finite limits.
- For a category $\mathcal{C}$, we write $\mathrm{k}(\mathcal{C})$ for the largest groupoid contained in $\mathcal{C}$, that is, the subcategory of $\mathcal{C}$ spanned by all the objects and invertible arrows.

We remind the reader about miscellaneous concepts and lemmas.

2.3.2. Definition (Beck-Chevalley condition). Let

$$\begin{array}{ccc} \mathcal{C}_1 & \xrightarrow{G} & \mathcal{C}_2 \\ {\scriptstyle F_1}\downarrow & & \downarrow{\scriptstyle F_2} \\ \mathcal{D}_1 & \xrightarrow[H]{} & \mathcal{D}_2 \end{array}$$

be a commutative (up to natural isomorphism) square of categories and suppose that F_1 and F_2 has right adjoints F_1^* and F_2^*, respectively. We say this square satisfies the *Beck-Chevalley condition* if the natural transformation

$$\begin{array}{ccccccc} & & \mathcal{C}_1 & \xrightarrow{G} & \mathcal{C}_2 & = & \mathcal{C}_2 \\ & {\scriptstyle F_1^*}\nearrow & \Downarrow{\scriptstyle \varepsilon_1} & \searrow{\scriptstyle F_1} & & \Downarrow{\scriptstyle \eta_2} & \nearrow{\scriptstyle F_2^*} \\ \mathcal{D}_1 & = & \mathcal{D}_1 & & \xrightarrow[H]{} & \mathcal{D}_2 & \end{array}$$

is invertible, where ε_1 is the counit of $F_1 \dashv F_1^*$ and η_2 is the unit of $F_2 \dashv F_2^*$.

2.3.3. Lemma (Two-pullbacks lemma). *For a commutative diagram in a category*

$$\begin{array}{ccccc} x_1 & \longrightarrow & x_2 & \longrightarrow & x_3 \\ \downarrow & & \downarrow & & \downarrow \\ y_1 & \longrightarrow & y_2 & \longrightarrow & y_3 \end{array}$$

in which the right square is a pullback, the left square is a pullback if and only if the outer square is a pullback.

We fix some notations.

- For an object x in a category $\mathcal{C}$ with finite product, we write $\Delta_x : x \to x \times x$ for the diagonal arrow, that is, the arrow whose projections on both sides are the identities.
- We write $\mathrm{inl} : x \to x + y$ and $\mathrm{inr} : y \to x + y$ for the coproduct inclusions.

2.3.1 Higher categories

For $0 \leq i \leq n \leq \infty$, by an (n, i)-*category*, we mean a weak higher category in which m-cells are invertible for $m > i$ and m-cells are trivial for $m > n$. We mostly work with the following cases:

- a $(1, 1)$-category is nothing but an ordinary category;
- a $(2, 1)$-category is a category weakly enriched over groupoids;
- a $(2, 2)$-category is what is called a bicategory in the literature;
- an $(\infty, 1)$-category is a category weakly enriched over ∞-groupoids.

We understand category-theoretic concepts in a maximally weak sense. For example, by a functor between $(2, 2)$-categories we mean a map preserving composition up to coherent isomorphism, which is called a pseudo-functor in the literature, even when the domain and codomain are strict $(2, 2)$-categories.

For concreteness, we will work with *quasicategories* [117, 95, 38] as models for $(\infty, 1)$-categories. By a $(2, 1)$-category we mean a 2-category in the sense of Lurie [117], that is, an $(\infty, 1)$-category satisfying certain triviality conditions on higher cells. A category weakly enriched over groupoids is regarded as a $(2, 1)$-category in this sense via the Duskin nerve [50]. The reason for this choice of definition of a $(2, 1)$-category is to apply powerful theorems in $(\infty, 1)$-category theory such as the adjoint functor theorem to $(2, 1)$-categories and generalize some results in the thesis to $(\infty, 1)$-categorical analogues.

We fix some notations.

- Unless otherwise mentioned, we view **Cat** as a $(2, 1)$-category where the 2-cells are the natural isomorphisms.
- **Space** denotes the $(\infty, 1)$-category of spaces.

2.3.2 Presheaves

2.3.4. Definition. Let $\mathcal{C}$ be a category. A *presheaf over* $\mathcal{C}$ consists of a set $A(x)$ for any object $x \in \mathcal{C}$ whose elements are called *sections over* x and an operator $\cdot : A(x) \times \mathcal{C}(x', x) \to A(x')$ for any pair of objects $x', x \in \mathcal{C}$ called the *right action of* $\mathcal{C}$ compatible with the identity and composition of $\mathcal{C}$. Since the right action corresponds to a map $\mathcal{C}(x', x) \to \mathbf{Set}(A(x), A(x'))$, a presheaf is equivalent to a functor $\mathcal{C}^{\mathrm{op}} \to \mathbf{Set}$.

For an object $x \in \mathcal{C}$, we write $\mathrm{Y}_{\mathcal{C}}\, x$ for the presheaf defined by $(\mathrm{Y}_{\mathcal{C}}\, x)(x') = \mathcal{C}(x', x)$. A presheaf is called *representable* if it is isomorphic to $\mathrm{Y}_{\mathcal{C}}\, x$ for some object x.

2.3.5. Theorem (The Yoneda Lemma). *For any presheaf A over $\mathcal{C}$ and any object $x \in \mathcal{C}$, the map*

$$\mathrm{Hom}(\mathrm{Y}_{\mathcal{C}}\, x, A) \ni a \mapsto a(\mathrm{id}_x) \in A(x)$$

is invertible.

By the Yoneda Lemma, we identify a section $a \in A(x)$ with the corresponding map $\mathrm{Y}_{\mathcal{C}}\, x \to A$. The assignment $x \mapsto \mathrm{Y}_{\mathcal{C}}\, x$ is part of a functor $\mathrm{Y}_{\mathcal{C}} : \mathcal{C} \to \mathbf{Fun}(\mathcal{C}^{\mathrm{op}}, \mathbf{Set})$ which we refer to as the *Yoneda embedding*.

2.3.3 Compactly generated categories

We review the theory of *compactly generated categories*, also known as *locally finitely presentable categories*. The notion of a locally finitely presentable category was introduced by Gabriel and Ulmer [62], and the standard textbook is [2]. Although we only explain the 1-categorical case, analogous results hold for *compactly generated* $(n,1)$*-categories* for an arbitrary $1 \leq n \leq \infty$ [117], and we need the case when $n = 2$ in order to construct certain compactly generated $(2,1)$-categories.

2.3.6. Definition. A category Ξ is said to be *filtered* if, for any finite diagram in Ξ, there exists a cocone over it. By a *filtered colimit* we mean a colimit indexed over a filtered category.

2.3.7. Definition. Let $\mathcal{X}$ be a category with filtered colimits. An object $A \in \mathcal{X}$ is *compact* if the functor $\mathcal{X}(A, -) : \mathcal{X} \to \mathbf{Set}$ preserves filtered colimits.

2.3.8. Definition. Let $\mathcal{X}$ be a category. We say a small full subcategory $\mathcal{C} \subset \mathcal{X}$ is a *strong generator* if the functor

$$\mathcal{C}^{\bullet} : \mathcal{X} \ni A \mapsto \mathcal{X}(-, A)|_{\mathcal{C}} \in \mathbf{Fun}(\mathcal{C}^{\mathrm{op}}, \mathbf{Set})$$

is faithful and conservative.

2.3.9. Definition. A category is *compactly generated* if it is cocomplete and has a strong generator consisting of compact objects.

2.3.10. Theorem (Representation Theorem). *Let $\mathcal{X}$ be a compactly generated category. Let $\mathcal{C} \subset \mathcal{X}$ be the full subcategory spanned by the compact objects.*

1. *$\mathcal{C}$ is essentially small and closed under finite colimits.*
2. *The functor $\mathcal{C}^{\bullet} : \mathcal{X} \to \mathbf{Fun}(\mathcal{C}^{\mathrm{op}}, \mathbf{Set})$ is fully faithful and its replete image is $\mathbf{Lex}(\mathcal{C}^{\mathrm{op}}, \mathbf{Set})$.*

3. *$\mathcal{X}$ is the cocompletion of $\mathcal{C}$ under filtered colimits.*

4. *$\mathcal{X}$ is the ω-free cocompletion of $\mathcal{C}$, that is, it is the initial cocomplete category $\mathcal{X}$ equipped with a functor $\mathcal{C} \to \mathcal{X}$ preserving finite colimits.*

Consequently, we have an equivalence $\mathcal{X} \simeq \mathbf{Lex}(\mathcal{C}^{\mathrm{op}}, \mathbf{Set})$.

2.3.11. Corollary. *Any compactly generated category has small limits.*

Let $\mathbf{Pr}^{\mathrm{R}}_{\omega} \subset \widehat{\mathbf{Cat}}$ denote the subcategory spanned by the compactly generated categories and functors preserving limits and filtered colimits.

2.3.12. Theorem (Limit Theorem). *$\mathbf{Pr}^{\mathrm{R}}_{\omega} \subset \widehat{\mathbf{Cat}}$ is closed under small limits and cotensors with small categories.*

The Limit Theorem is particularly useful for constructing new compactly generated categories out of old ones. We first prepare basic building blocks.

2.3.13. Example. **Set** is a compactly generated category. **Cat** is a compactly generated $(2,1)$-category.

2.3.14. Example. Let **LAdj** denote the category of left adjoints: the objects are the left adjoints $F : \mathcal{C} \to \mathcal{D}$ between small categories; the morphisms are the commutative squares satisfying the Beck-Chevalley condition. **LAdj** is a compactly generated $(2,1)$-category and the forgetful functor $\mathbf{LAdj} \to \mathbf{Cat}^{\to}$ preserves limits and filtered colimits and is conservative.

We can now construct various compactly generated $(2,1)$-categories of structures defined by adjoints.

2.3.15. Example. Let Ξ be a small category. We write $\mathbf{Lex}^{(\Xi)}$ for the $(2,1)$-category of small categories with limits of shape Ξ and functors preserving those limits. This $(2,1)$-category fits into the pullback

$$\begin{array}{ccc} \mathbf{Lex}^{(\Xi)} & \longrightarrow & \mathbf{LAdj} \\ \downarrow & \lrcorner & \downarrow \\ \mathbf{Cat} & \xrightarrow[\mathcal{C} \mapsto (\Delta : \mathcal{C} \to \mathcal{C}^{\Xi})]{} & \mathbf{Cat}^{\to}, \end{array}$$

and thus $\mathbf{Lex}^{(\Xi)}$ is a compactly generated $(2,1)$-category and the forgetful functor $\mathbf{Lex}^{(\Xi)} \to \mathbf{Cat}$ preserves limits filtered colimits and is conservative. The $(2,1)$-category **Lex** of small categories with finite limits and functors preserving finite limits is the wide pullback of $\mathbf{Lex}^{(\Xi)}$'s over **Cat** for all finite categories Ξ, and thus **Lex** is compactly generated and the forgetful functor $\mathbf{Lex} \to \mathbf{Cat}$ preserves limits and filtered colimits and is conservative.

2.3.16. Theorem (Adjoint Functor Theorem). *For a functor $F : \mathcal{X} \to \mathcal{Y}$ between compactly generated categories, the following are equivalent:*

1. *F has a left adjoint preserving compact objects;*
2. *F preserves limits and filtered colimits.*

The Adjoint Functor Theorem ensures that any functor constructed inside $\mathbf{Pr}^{\mathrm{R}}_{\omega}$ has a left adjoint. A consequence is that compactly generated categories admit *free constructions*. For example, the forgetful functor $\mathbf{Lex} \to \mathbf{Cat}$ has a left adjoint which assigns the free category with finite limits to each category.

2.3.4 Exponentiable arrows

Exponentiable arrows plays a crucial role in this thesis. In this section, we review the definition and basic properties of exponentiable arrows. Again we describe in the 1-categorical context, but the results also hold in the $(\infty, 1)$-categorical context.

2.3.17. Definition. Let $\mathcal{C}$ be a category with finite products. We say an object $x \in \mathcal{C}$ is *exponentiable* if the product functor $(- \times x) : \mathcal{C} \to \mathcal{C}$ has a right adjoint.

2.3.18. Notation. Let $u : y \to x$ be an arrow in a category $\mathcal{C}$ with finite limits. We write $u_! : \mathcal{C}/y \to \mathcal{C}/x$ for the functor defined by the composite with u and $u^* : \mathcal{C}/x \to \mathcal{C}/y$ for the right adjoint of $u_!$ defined by the pullback along u. When x is the final object, use the notations $y_!$ and y^* instead of $u_!$ and u^*, respectively.

2.3.19. Proposition (Niefield [131]). *Let $\mathcal{C}$ be a category with finite limits. For an arrow $u : y \to x$ in $\mathcal{C}$, the following are equivalent:*

1. *u is an exponentiable object of $\mathcal{C}/x$;*
2. *the pullback functor $u^* : \mathcal{C}/x \to \mathcal{C}/y$ has a right adjoint;*
3. *the fiber product functor $(- \times_x y) : \mathcal{C}/x \to \mathcal{C}$ has a right adjoint;*

Proof:
Although a proof is found in [131], we provide an alternative proof that also works in the $(\infty, 1)$-categorical context. The product by u in $\mathcal{C}/x$ is the composite

$$\mathcal{C}/x \xrightarrow{u^*} \mathcal{C}/y \xrightarrow{u_!} \mathcal{C}/x.$$

Since $u_!$ has the right adjoint u^*, Item 2 implies Item 1. The fiber product functor $(- \times_x y) : \mathcal{C}/x \to \mathcal{C}$ is the composite

$$\mathcal{C}/x \xrightarrow{(-\times u)} \mathcal{C}/x \xrightarrow{x_!} \mathcal{C}.$$

Since $x_!$ has the right adjoint x^*, Item 1 implies Item 3. To show that Item 3 implies Item 2, suppose that the functor $(- \times_x y)$ has a right adjoint $\mathrm{P}_u : \mathcal{C} \to \mathcal{C}/x$ with unit η. The right adjoint of u^* is defined by the pullback

$$\begin{array}{ccc} \bullet & \longrightarrow & \mathrm{P}_u(z) \\ \downarrow & \lrcorner & \downarrow \\ x & \xrightarrow[\eta_x]{} & \mathrm{P}_u(y) \end{array}$$

for an object $z \in \mathcal{C}/y$. □

2.3.20. Definition. We say an arrow in a category with finite limits is *exponentiable* if it satisfies the equivalent conditions of Proposition 2.3.19.

2.3.21. Definition. We say a class of arrows in a category with finite limits is *pullback-stable* if it is closed under identities, composition, and pullbacks.

2.3.22. Proposition (Niefield [131]). *The class of exponentiable arrows in a category $\mathcal{C}$ with finite limits is pullback-stable.*

Proof:
Let

$$\begin{array}{ccc} y' & \longrightarrow & y \\ {\scriptstyle u'}\downarrow & \lrcorner & \downarrow{\scriptstyle u} \\ x' & \xrightarrow[v]{} & x \end{array}$$

be a pullback in $\mathcal{C}$. Observe that the fiber product functor $(- \times_{x'} y')$ is the composite

$$\mathcal{C}/x' \xrightarrow{v_!} \mathcal{C}/x \xrightarrow{(- \times_x y)} \mathcal{C}.$$

Since $v_!$ has the right adjoint v^*, if $(- \times_x y)$ has a right adjoint, so does $(- \times_{x'} y')$. □

Let $u : y \to x$ is an exponentiable arrow in a category $\mathcal{C}$ with finite limits. The right adjoint of the pullback functor u^* is denoted by u_* and called the *pushforward along* u. An exponentiable arrow is viewed as a *polynomial from* 1 *to* 1 in the sense of Weber [181]. The *polynomial functor* P_u *associated to* u is the composite

$$\mathcal{C} \xrightarrow{y^*} \mathcal{C}/y \xrightarrow{u_*} \mathcal{C}/x \xrightarrow{x_!} \mathcal{C}.$$

By definition, we may view P_u as a functor $\mathcal{C} \to \mathcal{C}/x$ which is nothing but the right adjoint of $(- \times_x y) : \mathcal{C}/x \to \mathcal{C}$. We refer the reader to [65] for information about polynomials and polynomial functors. Here we recall that polynomials can

be *composed*: given two exponentiable arrows $u_1 : y_1 \to x_1$ and $u_2 : y_2 \to x_2$, we have an exponentiable arrow $u_1 \otimes u_2$ characterized by the natural isomorphism $\mathrm{P}_{u_1 \otimes u_2} \cong \mathrm{P}_{u_1} \circ \mathrm{P}_{u_2}$. Concretely, $u_1 \otimes u_2$ is defined as follows. The codomain of $u_1 \otimes u_2$ is $\mathrm{P}_{u_1}(x_2)$. The domain of $u_1 \otimes u_2$ is the pullback

$$\begin{array}{ccc} \mathrm{dom}(u_1 \otimes u_2) & \longrightarrow & y_2 \\ \downarrow & \lrcorner & \downarrow{\scriptstyle u_2} \\ \mathrm{P}_{u_1}(x_2) \times_{x_1} y_1 & \xrightarrow[\varepsilon_{x_2}]{} & x_2, \end{array}$$

where ε is the counit of the adjunction $(- \times_{x_1} y_1) \dashv \mathrm{P}_{u_1}$. Then $u_1 \otimes u_2$ is the composite

$$\mathrm{dom}(u_1 \otimes u_2) \longrightarrow \mathrm{P}_{u_1}(x_2) \times_{x_1} y_1 \longrightarrow \mathrm{P}_{u_1}(x_2) = \mathrm{cod}(u_1 \otimes u_2)$$

which is exponentiable by the stability under composition and pullbacks.

Chapter 3

Categories with representable maps

In this chapter, we introduce the notion of a *category with representable maps (CwR)* as an abstract notion of a type theory. This notion covers a wide range of syntactically presented type theories as we will see in Chapter 4 and has nice semantic properties proved in Chapter 5.

The idea comes from Lawvere's *functorial semantics* of algebraic theories [108] and its variants [123, 1]. In functorial semantics, *theories* are identified with structured *categories*, and *models* of a theory are identified with structure-preserving *functors* from the structured category corresponding to the theory. For example, consider the theory of groups whose models are of course groups. A group is a set A equipped with three operators satisfying certain equational axioms. The operators form the following diagram of sets.

$$\begin{array}{ccccc} 1 & \xrightarrow{1} & A & \xleftarrow{\cdot} & A \times A \\ & & {\scriptstyle (-)^{-1}}\uparrow & & \\ & & A & & \end{array} \tag{3.1}$$

In category theory, a diagram is nothing but a functor, and thus a group would be a functor $F : \mathcal{C}_{\mathrm{group}} \to \mathbf{Set}$ from a suitable category $\mathcal{C}_{\mathrm{group}}$. What structure should $\mathcal{C}_{\mathrm{group}}$ have? The category $\mathcal{C}_{\mathrm{group}}$ should contain a diagram

$$\begin{array}{ccccc} 1 & \xrightarrow{1} & x_0 & \xleftarrow{\cdot} & x_0 \times x_0 \\ & & {\scriptstyle (-)^{-1}}\uparrow & & \\ & & x_0 & & \end{array} \tag{3.2}$$

so that Eq. (3.1) is the image of Eq. (3.2) by the functor $F : \mathcal{C}_{\mathrm{group}} \to \mathbf{Set}$. For Eq. (3.2) to make sense, the category $\mathcal{C}_{\mathrm{group}}$ should have finite products. For Eq. (3.2) to be sent to Eq. (3.1) by F, the functor F should preserve finite products. Equational axioms are also explained diagrammatically. For example,

associativity is equivalent to that the following diagram commutes.

$$\begin{array}{ccc} A \times A \times A & \xrightarrow{A\times\cdot} & A \times A \\ {\scriptstyle \cdot\times A}\downarrow & & \downarrow{\scriptstyle \cdot} \\ A \times A & \xrightarrow[\cdot]{} & A \end{array}$$

Therefore, the corresponding diagram in $\mathcal{C}_{\mathrm{group}}$

$$\begin{array}{ccc} x_0 \times x_0 \times x_0 & \xrightarrow{x_0\times\cdot} & x_0 \times x_0 \\ {\scriptstyle \cdot\times x_0}\downarrow & & \downarrow{\scriptstyle \cdot} \\ x_0 \times x_0 & \xrightarrow[\cdot]{} & x_0 \end{array}$$

should commute. The diagrams corresponding to the other group axioms should also commute in $\mathcal{C}_{\mathrm{group}}$. The category $\mathcal{C}_{\mathrm{group}}$ should not contain other non-trivial arrows nor satisfy other non-trivial equations: otherwise functors $\mathcal{C}_{\mathrm{group}} \to \mathbf{Set}$ preserving finite products would correspond to groups with extra structure or axioms. Thus, we conclude that our category $\mathcal{C}_{\mathrm{group}}$ is the category with finite products *presented* by the object x_0 and arrows in Eq. (3.2) as generators and the commutative diagrams corresponding to the group axioms as relations.

In the same way, any (many-sorted) algebraic structure can be regarded as a functor $\mathcal{C} \to \mathbf{Set}$ preserving finite products for some category $\mathcal{C}$ with finite products, because an algebraic structure is a diagram of sets described in the language of finite products. Thus, algebraic theories are identified with categories with finite products, and models of an algebraic theory are identified with functors preserving finite products. The usual specification of an algebraic theory by operators and axioms is now considered as a *presentation* of the corresponding category with finite products: operators are generators and axioms are relations.

Similarly, we will identify type theories with certain structured categories and models of a type theory with structure-preserving functors. We begin by determining the language for describing models of a type theory. The notion of a model of a type theory we have in mind is a *natural model* introduced by Awodey [12], which is equivalent to a category with families of Dybjer [51]. The key observation is that in the natural model semantics, a wide range of type constructors such as Π-types and Σ-types are described in the language of *representable maps* of presheaves, *finite limits*, and *pushforwards along representable maps*. In the spirit of functorial semantics, we identify type theories with categories equipped with a class of arrows called representable maps, finite limits, and pushforwards along representable maps so that models of a type theory are identified with functors preserving these structures. We call such a category equipped with a class of representable maps a *category with representable maps (CwR)*. The traditional specification by inference rules is now considered as a *presentation* of a CwR.

We review natural models of type theory in Section 3.1. A minor change from the original work by Awodey is that we work with discrete fibrations instead of

presheaves, which makes the algebraic nature of natural models more apparent. In Section 3.2 we introduce the notion of a CwR and define a type theory to be just a CwR. A model of a type theory is then defined to be a structure-preserving functor into a category of discrete fibrations.

3.0.1. Remark. Although the notion of a CwR covers a wide range of type theories including Martin-Löf type theory [124, 133], univalent type theory [172], and cubical type theory [41], it cannot cover some important type theories. Characteristics of type theories considered in this thesis are that contexts are single-layered and that assumptions in a context can be used at any time, any number of times. Type theories with "dual-contexts" [e.g. 136, 111, 152] are not of this sort. Substructural type theories such as linear logic [71] are out of scope since assumptions can be used a limited number of times. Type theories with modality presented in [26] are not covered since assumptions can be used only under some restriction on contexts. The elegant framework of Licata, Shulman, and Riley [113] covers a lot of substructural and modal simple type theories, though its dependently-typed extension [110] has not been finished. Compared to their framework, our notion of a type theory is limited but produces nice semantic results proved in Chapter 5.

3.1 Natural models of type theory

We review natural models of type theory [12, 129] which are defined to be *representable maps* of presheaves.

3.1.1. Definition ([Stacks, Tag 0023]). We say a map $f : B \to A$ of presheaves over a category $\mathcal{C}$ is *representable* if, for any object $x \in \mathcal{C}$ and any section $a : \mathrm{Y}\, x \to A$, the presheaf a^*B is representable.

3.1.2. Definition ([12]). A *natural model* consists of a category $\mathcal{C}$ with a final object and a representable map $\partial : E \to U$ of presheaves over $\mathcal{C}$.

3.1.3. Remark. The fact that a representable map of presheaves models dependent type theory was also observed by Fiore [54] independently of Awodey.

3.1.4. Remark. For a presheaf A over a category $\mathcal{C}$, the following are *not* equivalent in general:

1. the presheaf A is representable;
2. the final projection $A \to 1$ is representable.

When $\mathcal{C}$ has a final object, Item 2 is equivalent to that the presheaf A is representable by some object $x \in \mathcal{C}$ and $\mathcal{C}$ has products with x. Therefore, if $\mathcal{C}$ has finite products, then Items 1 and 2 are equivalent.

Presheaves are known to be equivalent to *discrete fibrations*. We prefer to work with discrete fibrations instead of presheaves because the representability of a map of presheaves is characterized algebraically in terms of discrete fibrations (Proposition 3.1.11).We recall the definition and basic properties of discrete fibrations in Section 3.1.1. In Section 3.1.2, we review how natural models of Awodey can model type theories. In Section 3.1.3, we study representable maps of discrete fibrations in more detail.

3.1.1 Discrete fibrations

3.1.5. Definition. A functor $p : A \to \mathcal{C}$ is a *discrete fibration* if the square

$$\begin{array}{ccc} A^{\to} & \xrightarrow{\mathrm{cod}} & A \\ {\scriptstyle p^{\to}}\downarrow & & \downarrow{\scriptstyle p} \\ \mathcal{C}^{\to} & \xrightarrow[\mathrm{cod}]{} & \mathcal{C} \end{array}$$

is a strict pullback of categories. We write $\mathbf{DFib} \subset \mathbf{Cat}^{\to}$ for the full subcategory spanned by the discrete fibrations. For a category $\mathcal{C}$, by a *discrete fibration over* $\mathcal{C}$, we mean a discrete fibration of the form $A \to \mathcal{C}$. We write $\mathbf{DFib}_{\mathcal{C}} \subset \mathbf{Cat}/\mathcal{C}$ for the full subcategory spanned by the discrete fibrations over $\mathcal{C}$.

The following are immediate from the definition.

3.1.6. Proposition. *Discrete fibrations are closed under identities, composition, and strict pullbacks along arbitrary functors.* □

3.1.7. Proposition. *Let $f : B \to A$ and $p : A \to \mathcal{C}$ be functors and suppose that p is a discrete fibration. Then $p \circ f$ is a discrete fibration if and only if f is.*

Proof:
By the two-pullbacks lemma. □

The following elementary characterization of discrete fibrations might be more familiar.

3.1.8. Proposition. *For a functor $p : A \to \mathcal{C}$, the following are equivalent:*

1. *p is a discrete fibration;*
2. *for any object $a \in A$, the functor $p/a : A/a \to \mathcal{C}/p(a)$ is an isomorphism;*
3. *for any object $a \in A$, any object $x' \in \mathcal{C}$ and any arrow $u : x' \to p(a)$, there exists a unique pair (a', f) consisting of an object $a' \in A$ and an arrow*

$f : a' \to a$ such that $p(a') = x'$ and $p(f) = u$.

$$\begin{array}{ccc} \exists! a' & \overset{\exists! f}{\dashrightarrow} & a \\ & & \\ x' & \xrightarrow[u]{} & p(a) \end{array} \qquad \begin{array}{c} A \\ \downarrow{\scriptstyle p} \\ \mathcal{C} \end{array}$$

We write $a \cdot u = a'$ and call it the right action of u on a.

Proof:
Items 1 and 2 are equivalent by definition since A/a is the fiber of $A^{\to}$ over a. Clearly, Item 2 implies Item 3. Item 3 is equivalent to that the functor $p/a : A/a \to \mathcal{C}/p(a)$ is bijective on objects for any object $a \in A$. This also implies that p/a is fully faithful as follows. Let $f_1 : a_1 \to a$ and $f_2 : a_2 \to a$ be objects of A/a and $u : p(a_1) \to p(a_2)$ an arrow in $\mathcal{C}/p(a)$. By assumption, we have a unique arrow $g : a_1' \to a_2$ over u. Then the composite $f_2 \circ g : a_1' \to a_2$ is an arrow over $p(f_1) : p(a_1) \to p(a)$, and thus $a_1' = a_1$ and $f_1 = f_2 \circ g$. Hence, g is a unique arrow $a_1 \to a_2$ in A/a such that $p(g) = u$. □

3.1.9. Corollary. *For any category $\mathcal{C}$ and object $x \in \mathcal{C}$, the forgetful functor $\mathcal{C}/x \to \mathcal{C}$ is a discrete fibration.*

Proof:
Because the functor $(\mathcal{C}/x)/u \to \mathcal{C}/y$ is an isomorphism for any object $u : y \to x$ of $\mathcal{C}/x$. □

Let $\mathcal{C}$ be a category. The presheaves over $\mathcal{C}$ and the discrete fibrations over $\mathcal{C}$ are equivalent as follows. For a presheaf $A : \mathcal{C}^{\mathrm{op}} \to \mathbf{Set}$, the *category of elements* $\int_{\mathcal{C}} A$ is defined to be the strict pullback

$$\begin{array}{ccc} \int_{\mathcal{C}} A & \longrightarrow & \mathbf{Fun}(\mathcal{C}^{\mathrm{op}}, \mathbf{Set})/A \\ \downarrow & \lrcorner & \downarrow \\ \mathcal{C} & \xrightarrow[\mathrm{Y}_{\mathcal{C}}]{} & \mathbf{Fun}(\mathcal{C}^{\mathrm{op}}, \mathbf{Set}). \end{array}$$

By construction, the functor $\int_{\mathcal{C}} A \to \mathcal{C}$ is a discrete fibration.

3.1.10. Proposition. *For any category $\mathcal{C}$, the construction $A \mapsto \int_{\mathcal{C}} A$ is part of an equivalence of categories*

$$\int_{\mathcal{C}} : \mathbf{Fun}(\mathcal{C}^{\mathrm{op}}, \mathbf{Set}) \simeq \mathbf{DFib}_{\mathcal{C}}.$$

Proof:
For a discrete fibration $p : A \to \mathcal{C}$, we have the presheaf

$$\mathcal{C}^{\mathrm{op}} \in x \mapsto \mathbf{DFib}_{\mathcal{C}}(\mathcal{C}/x, A) \in \mathbf{Set},$$

which gives an inverse of $\int_{\mathcal{C}}$. □

Under the equivalence $\mathbf{Fun}(\mathcal{C}^{\mathrm{op}}, \mathbf{Set}) \simeq \mathbf{DFib}_{\mathcal{C}}$, the representable presheaf $\mathrm{Y}_{\mathcal{C}}\, x$ corresponds to the discrete fibration $\mathcal{C}/x$. We thus say a discrete fibration over $\mathcal{C}$ is *representable* if it is isomorphic to some $\mathcal{C}/x$ and call a map of discrete fibrations over $\mathcal{C}$ of the form $a : \mathcal{C}/x \to A$ a *section over* x.

We prefer to work with discrete fibrations instead of presheaves because of the following algebraic characterization of representability of a map of presheaves.

3.1.11. Proposition. *A map $f : B \to A$ of presheaves over a category $\mathcal{C}$ is representable if and only if the functor $\int_{\mathcal{C}} f : \int_{\mathcal{C}} B \to \int_{\mathcal{C}} A$ has a right adjoint.*

Proof:
For an object $(x, a) \in \int_{\mathcal{C}} A$, the comma category $\left(\int_{\mathcal{C}} f \downarrow (x, a)\right)$ is equivalent to the category of commutative squares of the form

$$\begin{array}{ccc} \mathrm{Y}\, y & \xrightarrow{\;b\;} & B \\ {\scriptstyle \mathrm{Y}\, u}\downarrow & & \downarrow{\scriptstyle f} \\ \mathrm{Y}\, x & \xrightarrow[\;a\;]{} & A, \end{array}$$

and this square is a final object in $\left(\int_{\mathcal{C}} f \downarrow (x, a)\right)$ if and only if it is a pullback. Thus, $\left(\int_{\mathcal{C}} f \downarrow (x, a)\right)$ has a final object if and only if the presheaf $a^* B$ is representable. □

3.1.12. Definition. Let $\mathcal{C}$ be a category. We say a map $f : A \to B$ of discrete fibrations over $\mathcal{C}$ is *representable* if it has a right adjoint seen as a functor.

3.1.2 Modeling type theory

A representable map $f : B \to A$ of discrete fibrations over $\mathcal{C}$ is considered as a model of dependent type theory. We think of objects $\Gamma \in \mathcal{C}$ as *contexts*, sections $a : \mathcal{C}/\Gamma \to A$ as *types over* Γ, and sections $b : \mathcal{C}/\Gamma \to B$ as *terms over* Γ. The type of a term $b : \mathcal{C}/\Gamma \to B$ is given by $f(b)$. A morphism $u : \Gamma' \to \Gamma$ in $\mathcal{C}$ corresponds to a *substitution*, and the right action $a \cdot u$ and $b \cdot u$ on sections of A and B represents the action of the substitution. The representability of f is used for modeling *context comprehension*. For a type $a : \mathcal{C}/\Gamma \to A$, the pullback $a^* B$ is a representable discrete fibration as f is representable. We write $\{a\}$ for

the representing object and call it the *context comprehension of a (with respect to f)*. By definition, $\{a\}$ fits into the following pullback square.

$$\begin{array}{ccc} \mathcal{C}/\{a\} & \xrightarrow{\mathrm{q}(a)} & B \\ {\scriptstyle \mathrm{p}(a)}\downarrow & \lrcorner & \downarrow{\scriptstyle f} \\ \mathcal{C}/\Gamma & \xrightarrow[a]{} & A \end{array}$$

As Awodey [12] noted, natural models are equivalent to categories with families of Dybjer [51]. An advantage of natural models is that type constructors such as Π-types and identity types are described inside the category of discrete fibration over the base category. Since the action of a substitution is represented by the right action of a morphism in the base category, everything constructed inside the category of discrete fibrations automatically becomes stable under substitutions. This simplifies definitions and constructions of type constructors on natural models.

We observe that type constructors are modeled by maps of discrete fibrations whose domains and codomains are built out of $f : B \to A$ using finite limits and pushforwards along f. For example, extensional identity types are modeled by a pullback of the form

$$\begin{array}{ccc} B & \overset{\mathsf{refl}}{\cdots\cdots\rightarrow} & B \\ {\scriptstyle \Delta}\downarrow & \lrcorner & \downarrow{\scriptstyle f} \\ B \times_A B & \underset{\mathsf{Id}}{\cdots\cdots\rightarrow} & A. \end{array}$$

The map Id sends a triple (a, b_1, b_2) consisting of a type $a : \mathcal{C}/\Gamma \to A$ and terms $b_1, b_2 : \mathcal{C}/\Gamma \to B$ of the type a to a type $\mathsf{Id}(a, b_1, b_2) : \mathcal{C}/\Gamma \to A$ and models the formation rule for identity types. The map refl sends a term $b : \mathcal{C}/\Gamma \to B$ of the type a to a term of the type $\mathsf{Id}(a, b, b)$ and models the introduction rule for identity types. Since the square is a pullback, for any term c of $\mathsf{Id}(a, b_1, b_2)$, the terms b_1 and b_2 must be equal and c is $\mathsf{refl}(b_1)$, and thus Id and refl model extensional identity types.

Type constructors with variable binding are modeled using pushforwards along f. For example, Π-types are modeled by a pullback of the form

$$\begin{array}{ccc} \mathrm{P}_f(B) & \overset{\lambda}{\cdots\cdots\rightarrow} & B \\ {\scriptstyle \mathrm{P}_f(f)}\downarrow & \lrcorner & \downarrow{\scriptstyle f} \\ \mathrm{P}_f(A) & \underset{\Pi}{\cdots\cdots\rightarrow} & A, \end{array}$$

where $\mathrm{P}_f = A_! f_* B^*$ is the polynomial functor. From the definition of the polynomial functor, a section $\mathcal{C}/\Gamma \to \mathrm{P}_f(A)$ correspond to a pair (a_1, a_2) consisting of a type $a_1 : \mathcal{C}/\Gamma \to A$ and a map $a_2 : \mathcal{C}/\Gamma \times_A B \to A$. By the representability of f,

we have $\mathcal{C}/\Gamma \times_A B \cong \mathcal{C}/\{a_1\}$, and thus a_2 is a type over $\{a_1\}$. Then the map Π models the formation rule for Π-types. Similarly, a section of $\mathrm{P}_f(B)$ corresponds to a pair consisting of a type $a : \mathcal{C}/\Gamma \to A$ and a term $b : \mathcal{C}/\{a\} \to B$, and the map λ models the introduction rule for Π-types. Since the square is a pullback, the terms of $\Pi(a_1, a_2)$ correspond to the sections of $\mathrm{P}_f(f) : \mathrm{P}_f(B) \to \mathrm{P}_f(A)$ over $(a_1, a_2) : \mathcal{C}/\Gamma \to \mathrm{P}_f(A)$ which correspond to the terms $\mathcal{C}/\{a_1\} \to B$ of a_2 by the definition of P_f. Therefore, Π and λ model Π-types.

Σ-types are also explained in terms of polynomial functors. Σ-types are modeled by a pullback of the form

$$\begin{array}{ccc} \mathrm{dom}(f \otimes f) & \overset{\mathsf{pair}}{\dashrightarrow} & B \\ {\scriptstyle f\otimes f}\downarrow & \lrcorner & \downarrow{\scriptstyle f} \\ \mathrm{cod}(f \otimes f) & \underset{\Sigma}{\dashrightarrow} & A \end{array} \tag{3.3}$$

where $\otimes$ is the composition of polynomials. From the concrete definition of $f \otimes f$ [65], one can see that $\mathrm{cod}(f \otimes f) \cong \mathrm{P}_f(A)$ and that a section of $f \otimes f$ over a section $(a_1, a_2) : \mathcal{C}/\Gamma \to \mathrm{P}_f(A)$ corresponds to a pair (b_1, b_2) consisting of a section $b_1 : \mathcal{C}/\Gamma \to B$ over a_1 and a section $b_2 : \mathcal{C}/\Gamma \to B$ over $a_2 \circ b_1'$ where b_1' is the induced section of $\mathcal{C}/\{a_1\} \to \mathcal{C}/\Gamma$. The condition that Eq. (3.3) is a pullback expresses that the elements of $\Sigma(a_1, a_2)$ are precisely the pairs (b_1, b_2) consisting of an element b_1 of a_1 and an element b_2 of $b_2 \cdot a_1$. The pullback square (3.3) can be viewed as part of a polynomial pseudo-monad [14], but it is beyond the scope of this thesis.

Inductive types are also modeled by maps between discrete fibrations. We explain the simplest case, the empty type, that is, the inductive type without any term constructor. The formation rule for the empty type is simply modeled by a global section $\mathsf{0} : \mathcal{C} \to A$. The elimination rule for the empty type is modeled by a section of the form

$$\begin{array}{ccc} & & \mathrm{P}_f(B) \\ & {\scriptstyle \mathsf{elim}_0}\nearrow & \downarrow{\scriptstyle \mathrm{P}_f(A)} \\ \mathsf{0}^*\mathrm{P}_f(A) & \longrightarrow & \mathrm{P}_f(A) \\ \downarrow & \lrcorner & \downarrow \\ \mathcal{C} & \underset{\mathsf{0}}{\longrightarrow} & A. \end{array}$$

Indeed, a section $\mathcal{C}/\Gamma \to \mathsf{0}^*\mathrm{P}_f(A)$ corresponds to a type $a : \mathcal{C}/\Gamma \times \{\mathsf{0}\} \to A$ and the map elim_0 sends such a type a to a term $\mathcal{C}/\Gamma \times \{\mathsf{0}\} \to B$ of a. Similarly, other inductive types such as the natural number type and intensional identity types are modeled by maps between discrete fibrations.

All the type theories considered in the original work by Awodey [12] are modeled by a single representable map of presheaves and some other maps. Extending natural models by more than one representable maps allows us to model more

complex type theories. *Cubical type theory* [41] is a motivating example. This type theory has a formal interval $\mathbb{I}$ by which contexts can be extended, but $\mathbb{I}$ is not considered as a type. Thus, the interval is modeled by another discrete fibration $\mathbb{I}$ such that the final projection $\mathbb{I} \to 1$ is a representable map, rather than a global section $\mathbb{I} : \mathcal{C} \to A$. Since the base category $\mathcal{C}$ is required to have a final object, the representability of $\mathbb{I} \to 1$ is equivalent to that the discrete fibration $\mathbb{I}$ is representable and $\mathcal{C}$ has products by $\mathbb{I}$ (Remark 3.1.4). Then the context comprehension by $\mathbb{I}$ is modeled by the product by $\mathbb{I}$, which coincides with the interpretation of the interval in the presheaf semantics of cubical type theory explained in [41]. Another component of cubical type theory is the face lattice or cofibrant propositions, and it is modeled by an additional representable map that is also a monomorphism.

3.1.3 Properties of representable maps of discrete fibrations

Representable maps of discrete fibrations are closed under several operations [129, Theorem 3.3.14]. The following closure properties are relevant for our purpose.

3.1.13. Proposition. *Let $\mathcal{C}$ be a category. Then the class of representable maps of discrete fibrations over $\mathcal{C}$ is closed under identities, composition, and pullbacks along arbitrary maps.* □

Pullbacks and right adjoints of representable maps nicely interact.

3.1.14. Lemma. *Let*

$$\begin{array}{ccc} A_1 & \xrightarrow{F'} & A_2 \\ {\scriptstyle p_1}\downarrow & & \downarrow{\scriptstyle p_2} \\ \mathcal{C}_1 & \xrightarrow[F]{} & \mathcal{C}_2 \end{array}$$

be a commutative square of categories in which p_1 and p_2 are discrete fibrations and F has a right adjoint G. Then the square is a pullback if and only if F' has a right adjoint G' over G.

Proof:
This follows from general facts about fibred adjunctions [77, Chapter 3]. Here we give an elementary account. Let η and ε denote the unit and counit, respectively, of the adjunction $F \dashv G$. Suppose that the square is a pullback. Then A_1 is the category of pairs (x_1, a_2) consisting of an object $x_1 \in \mathcal{C}_1$ and an object $a_2 \in A_2$ over $F(x_1)$. Given an object $a_2 \in A_2$ over an object $x_2 \in \mathcal{C}_2$, we define an object

$G'(a_2) \in A_1$ over $G(x_2)$ by the unique lift

$$G'(a_2) \dashrightarrow a_2$$

$$F(G(x_2)) \xrightarrow[\varepsilon]{} x_2$$

and then G' is a right adjoint of F' over G. Conversely, if G' is a right adjoint of F' over G, we can construct an inverse of the functor $A_1 \to F^*A_2$: given an object $(x_1, a_2) \in F^*A_2$, we have an object $a_1 \in A_1$ over x_1 by the unique lift

$$a_1 \dashrightarrow G'(a_2)$$

$$x_1 \xrightarrow[\eta]{} G(F(x_1)).$$

□

3.1.15. Proposition. *Let*

$$\begin{array}{ccc} B_1 & \xrightarrow{h} & B_2 \\ {\scriptstyle f_1}\downarrow & & \downarrow{\scriptstyle f_2} \\ A_1 & \xrightarrow[g]{} & B_2 \end{array}$$

be a commutative square of discrete fibrations over a category $\mathcal{C}$. Suppose that f_1 and f_2 are representable. Then this square is a pullback if and only if it satisfies the Beck-Chevalley condition.

Proof:
By Proposition 3.1.7, the functors g and h are discrete fibrations. Then use Lemma 3.1.14. □

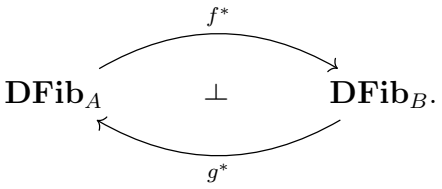
It is known that in the category $\mathbf{DFib}_{\mathcal{C}}$ every map is exponentiable, but the pushforward along a representable map is defined algebraically.

3.1.16. Lemma. *For any discrete fibration A over a category $\mathcal{C}$, the equivalence $(\mathbf{Cat}/\mathcal{C})/A \simeq \mathbf{Cat}/A$ is restricted to an equivalence $(\mathbf{DFib}_{\mathcal{C}})/A \simeq \mathbf{DFib}_A$.*

Proof:
By Proposition 3.1.7. □

3.1.17. Proposition. *Let $f : B \to A$ be a representable map of discrete fibrations over a category $\mathcal{C}$. The pushforward along f is given by the pullback along the right adjoint $A \to B$ of f.*

Proof:
Let $g : A \to B$ denote the right adjoint of f. Then the pullbacks along f and g determine an adjunction



By Lemma 3.1.16, g^* determines the right adjoint of the pullback functor $f^* : \mathbf{DFib}_{\mathcal{C}}/A \to \mathbf{DFib}_{\mathcal{C}}/B$. □

3.1.18. Corollary. *Let $f : B \to A$ be a representable map of discrete fibrations over a category $\mathcal{C}$. Then the pushforward functor $f_* : \mathbf{DFib}_{\mathcal{C}}/B \to \mathbf{DFib}_{\mathcal{C}}/A$ has a right adjoint.*

Proof:
This is because the pullback functor $g^* : \mathbf{DFib}_B \to \mathbf{DFib}_A$ has a right adjoint, where g is the right adjoint of f. □

3.2 Type theories

We have seen in Section 3.1.2 that models of a type theory are described using the following concepts:

- representable maps
- finite limits;
- pushforwards along representable maps.

In the spirit of functorial semantics, we identify a type theory with a category equipped with these structures and a model of the type theory with a structure-preserving functor. Axiomatizing properties of representable maps in the category of discrete fibrations over a category in relation to models of a type theory, we obtain the following notion.

3.2.1. Definition. A *category with representable maps (CwR)* is a category $\mathcal{C}$ with finite limits equipped with a pullback-stable class $\mathrm{R}_{\mathcal{C}}$ of exponentiable arrows. Arrows in $\mathrm{R}_{\mathcal{C}}$ are called *representable maps*. A *morphism of CwRs $\mathcal{C} \to \mathcal{D}$* is a functor $F : \mathcal{C} \to \mathcal{D}$ preserving finite limits, representable maps, and pushforwards along representable maps. We write **CwR** for the $(2, 1)$-category of CwRs, morphisms of CwRs, and natural isomorphisms.

3.2.2. Example. For any category $\mathcal{C}$, the category $\mathbf{DFib}_{\mathcal{C}}$ of discrete fibrations over $\mathcal{C}$ is a CwR in which a map is representable if it is representable in the sense of Definition 3.1.12 by Propositions 3.1.13 and 3.1.17.

3.2.3. Definition. A *type theory* is a small CwR. A *morphism of type theories* is a morphism of CwRs. We write $\mathbf{TT}$ for the $(2,1)$-category of type theories which is nothing but $\mathbf{CwR}$.

3.2.4. Definition. Let $\mathcal{T}$ be a type theory. A *model* $\mathcal{M}$ *of* $\mathcal{T}$ consists of the following data:

- a category $\mathcal{M}(\diamond)$ with a final object;
- a morphism of CwRs $\mathcal{M} : \mathcal{T} \to \mathbf{DFib}_{\mathcal{M}(\diamond)}$.

3.2.5. Definition. Let $\mathcal{T}$ be a type theory and $\mathcal{M}$ and $\mathcal{N}$ models of $\mathcal{T}$. A *morphism* $F : \mathcal{M} \to \mathcal{N}$ *of models of* $\mathcal{T}$ consists of the following data:

- a functor $F_\diamond : \mathcal{M}(\diamond) \to \mathcal{N}(\diamond)$ preserving final objects;
- for each object $x \in \mathcal{T}$, a map of discrete fibrations $F_x : \mathcal{M}(x) \to \mathcal{N}(x)$ over $F_\diamond$

satisfying the following conditions:

1. for any arrow $u : x \to y$ in $\mathcal{T}$, the square

$$\begin{array}{ccc} \mathcal{M}(x) & \xrightarrow{F_x} & \mathcal{N}(x) \\ {\scriptstyle \mathcal{M}(u)}\downarrow & & \downarrow{\scriptstyle \mathcal{N}(u)} \\ \mathcal{M}(y) & \xrightarrow[F_y]{} & \mathcal{N}(y) \end{array} \tag{3.4}$$

   commutes;

2. when u is a representable map in $\mathcal{T}$, Eq. (3.4) satisfies the Beck-Chevalley condition.

Note that since the right adjoint of a representable map models context comprehension, the Beck-Chevalley condition for Eq. (3.4) means that a morphism between models of a type theory is required to preserve context comprehension (up to canonical isomorphism).

Examples of type theories are specified by certain *universal properties*, and there are two ways of proving the existence of type theories with universal properties. One way is to use the presentability of the $(2,1)$-category of type theories proved in Section 5.3. Intuitively, the definition of a CwR is essentially algebraic in the $(2,1)$-categorical sense, and thus we can construct suitable *free CwRs* analogously to the construction of free algebras such as free groups, free lattices, and free categories. The other way is to construct a CwR from a syntactic presentation of a type theory explained in Chapter 4.

3.2.6. Example. Let $\mathbb{D}$ denote the free CwR generated by a representable map $\partial : E \to U$. The universal property of $\mathbb{D}$ asserts that a morphism of CwRs $F : \mathbb{D} \to \mathcal{C}$ is completely determined by the image of the representable map ∂. Therefore, a model of $\mathbb{D}$ is equivalent to the following structure:

- a category $\mathcal{M}(\diamond)$ with a final object;
- a representable map $\mathcal{M}(\partial) : \mathcal{M}(E) \to \mathcal{M}(U)$ of discrete fibrations over $\mathcal{M}(\diamond)$,

that is, a natural model. Thus, $\mathbb{D}$ is considered as the dependent type theory without any type constructors.

3.2.7. Example. Let $\mathbb{D}^{\Pi}$ denote the free CwR generated by a representable map $\partial : E \to U$ and a pullback of the form

$$\begin{array}{ccc} \mathrm{P}_{\partial}(E) & \overset{\lambda}{\dashrightarrow} & E \\ {\scriptstyle \mathrm{P}_{\partial}(\partial)}\downarrow & \lrcorner & \downarrow{\scriptstyle \partial} \\ \mathrm{P}_{\partial}(U) & \underset{\Pi}{\dashrightarrow} & U. \end{array}$$

Then a model of $\mathbb{D}^{\Pi}$ is a natural model $(\mathcal{M}(\diamond), \mathcal{M}(\partial) : \mathcal{M}(E) \to \mathcal{M}(U))$ equipped with a pullback of the form

$$\begin{array}{ccc} \mathrm{P}_{\mathcal{M}(\partial)}(\mathcal{M}(E)) & \overset{\mathcal{M}(\lambda)}{\dashrightarrow} & \mathcal{M}(E) \\ {\scriptstyle \mathrm{P}_{\mathcal{M}(\partial)}(\mathcal{M}(\partial))}\downarrow & \lrcorner & \downarrow{\scriptstyle \mathcal{M}(\partial)} \\ \mathrm{P}_{\mathcal{M}(\partial)}(\mathcal{M}(U)) & \underset{\mathcal{M}(\Pi)}{\dashrightarrow} & \mathcal{M}(U). \end{array}$$

Hence, $\mathbb{D}^{\Pi}$ is the dependent type theory with Π-types.

3.2.8. Example. We would need a lot of pages to write down the universal property of the CwR that represents cubical type theory [41]. Here we only describe the fundamental structure. Cubical type theory is defined to be the free CwR generated by the following data:

- a representable map $E \to U$;
- a representable monomorphism $\mathsf{true} \to \mathsf{Cof}$;
- an object $\mathbb{I}$ such that the final projection $\mathbb{I} \to 1$ is a representable map;
- a bunch of arrows and equations representing components of cubical type theory.

We will give a more detailed syntactic presentation of cubical type theory in Section 4.6.3.

The guiding principle of presenting type theories as CwRs is to express *judgment forms as objects*. For example, the object U of $\mathbb{D}$ corresponds to the judgment form $(\vdash _\ \mathsf{type})$ meaning that something is a type, and the object $E \in \mathbb{D}/U$ corresponds to the family of judgment forms $(\vdash _ : \mathtt{A})$ indexed over types $\mathtt{A}$ meaning that something is an element of $\mathtt{A}$. We express a family of judgment forms as a representable map when contexts in the type theory can be extended by those judgment forms. For example, in the ordinary syntax of dependent type theory, a context is a list of term variables $\mathtt{x} : \mathtt{A}$ but does not contain type variables $\mathtt{x}\ \mathsf{type}$ (unless some form of polymorphism is introduced). Therefore, the arrow $\partial : E \to U$ in $\mathbb{D}$ is a representable map while the projection $U \to 1$ is not. We use finite limits to create equality judgment forms. For example, the diagonal arrow $U \to U \times U$ seen as an object of $\mathbb{D}/U \times U$ corresponds to the family of judgment forms $(\vdash \mathtt{A}_1 \equiv \mathtt{A}_2\ \mathsf{type})$ indexed over pairs of types $(\mathtt{A}_1, \mathtt{A}_2)$ meaning that $\mathtt{A}_1$ and $\mathtt{A}_2$ are equal. The pushforward along ∂ creates hypothetical judgment forms. For example, the object $\mathrm{P}_\partial(U) \in \mathbb{D}/U$ corresponds to the family of judgment forms $(\mathtt{x} : \mathtt{A} \vdash _\ \mathsf{type})$ indexed over types $\mathtt{A}$ meaning that something is a type under the hypothesis that $\mathtt{x}$ is an element of $\mathtt{A}$.

Since objects are judgment forms, arrows are transformations from judgments to judgments, that is, *inference rules*. For example, the arrow $\Pi : \mathrm{P}_\partial(U) \to U$ in $\mathbb{D}^\Pi$ corresponds to the formation rule for Π-types: it takes types $(\vdash \mathtt{A}\ \mathsf{type})$ and $(\mathtt{x} : \mathtt{A} \vdash \mathtt{B}\ \mathsf{type})$ and returns a type $(\vdash \Pi(\mathtt{A}, \mathtt{B})\ \mathsf{type})$.

3.2.9. Remark. The judgment-forms-as-objects principle is essentially the same as the *judgments-as-types* principle in the context of logical framework [75]. One change is that in our style, the judgment forms $(\vdash _\ \mathsf{type})$ and $(\vdash _ : \mathtt{A})$ are distinguished based on whether they can be put in a context.

In the next few chapters, we will develop a theory of type theories. Chapter 4 is devoted to establishing a connection between CwRs and syntactic presentations of type theories. We introduce syntactic counterparts of CwRs called *second-order generalized algebraic theories (SOGATs)*. We associate to each SOGAT a CwR called the *syntactic CwR* and show that Definition 3.2.4 gives us the correct notion of a model of the SOGAT. Syntactic CwRs are also a major source of examples of CwRs. In Chapter 5 we develop a theory of type theories in a purely categorical way. We will not use the results of Chapter 4 except for giving examples. The main result is the *theory-model correspondence*: we introduce a notion of a *theory over a type theory* which is roughly an extension of the type theory by constants; given a theory over a type theory, we construct a model of the type theory called the *syntactic model*; given a model of a type theory, we construct a theory over the type theory called the *internal language*; the syntactic model construction and the internal language construction are in adjunction and induce an equivalence

between the category of theories over a type theory and a full subcategory of the category of models of the type theory.

There are some remarks on the definition of a CwR.

3.2.10. Remark. The definition of a CwR might remind the reader about similar concepts familiar to categorical type theorists such as contextual categories [35], C-systems [179], display map categories [166], and clans [94]. These are also categories equipped with a pullback-stable class of arrows. However, the notion of a CwR and the other notions come from different motivations. While the others are considered as models of a type theory, CwRs are considered as type theories themselves.

3.2.11. Remark. We do not require that representable maps in a CwR are closed under pushforwards since this closure property fails in $\mathbf{DFib}_{\mathcal{C}}$ unless the base category $\mathcal{C}$ is locally cartesian closed.

3.2.12. Remark. The pushforward along a representable map is used for representing variable binding, and this is essentially the same as the use of Π-types to represent variable binding in logical frameworks [75, 133]. Since logical frameworks usually have all Π-types, one might think that CwRs are less expressive than logical frameworks and that categorical presentations of type theories should be locally cartesian closed, that is, have all pushforwards. It is certainly true that CwRs are less expressive than logical frameworks, but, as we will see in Section 4.6, restricted forms of Π-types are sufficient to define a wide range of practical type theories, because variable binding in a type theory only occurs at the "first-order" level, and we do not need higher-order variable binding. Of course, it is possible to consider type constructors with higher-order variable binding such as W-types in the absence of Π-types, and Gratzer and Sterling [74] proposed the use of locally cartesian closed categories for presenting type theories to deal with such higher-order operators. However, the main use of W-types is to internalize the construction of inductive types in a type theory, and thus it is natural to assume that the type theory already has Π-types to internalize some concepts. Then higher-order variable binding can be performed by a combination of first-order variable binding and Π-types.

A more technical reason for not requiring a CwR to be locally cartesian closed is that allowing arbitrary pushforwards causes extra difficulty in the study of models of type theories. An important feature of our notion of a model of a type theory (Definition 3.2.4) is that the $(2,1)$-category of models of a type theory is compactly generated, as proved in Section 5.2.1. Compactly generated categories have good properties such as (co)completeness and the special adjoint functor theorem. If we defined a type theory to be a locally cartesian closed category and a model of a type theory to be a functor preserving finite limits and arbitrary pushforwards, then we would not get this feature because functors preserving pushforwards need not form a compactly generated category. The pushforward

along a representable map of discrete fibrations is special because it is defined by a pullback (Proposition 3.1.17), and this will be the key to the compact generation of models.

Chapter 4

Second-order generalized algebraic theories

In this chapter, we introduce *second-order generalized algebraic theories (SOGATs)* for several reasons. In short, SOGATs are:

1. syntactic counterparts of CwRs;
2. extension of the general definition of dependent type theories given by Bauer, Haselwarter, and Lumsdaine [20] to allow user-definable judgment forms;
3. second-order extension of *generalized algebraic theories* of Cartmell [35];
4. dependently-typed extension of *second-order algebraic theories* of Fiore and Mahmoud [57].

The first aspect is the most important in this thesis. In [169] the author introduced a logical framework to give syntactic counterparts of CwRs, but it is not satisfactory. When defining a type theory as a theory over a logical framework, one has to prove the *adequacy* of the definition, that is, the theory over the logical framework derives the same judgments as the original type theory does. Although adequacy can systematically be proved for each individual type theory syntactically [75] or semantically [79], the adequacy of all logical framework presentations cannot be proved nor even stated unless we have a general notion of a type theory outside the logical framework. What we really want is a general notion of a syntactic presentation of a type theory such that a wide range of existing type theories are verified to be instances of that notion by just unfolding the definition. Bauer, Haselwarter, and Lumsdaine [20] have made a careful analysis on inference rules in traditional presentations of type theories and proposed a class of syntactic presentations of type theories that fits our requirement. A restriction on their type theories is that the forms of judgments are fixed, and thus we cannot define some complex type theories such as cubical type theory [41]. We thus modify

their definition of type theories to allow users to declare new judgment forms. It turns out that our syntactic presentations of type theories are considered as an extension of algebraic theories by dependent types and variable binding. We thus choose the name "second-order generalized algebraic theory" combining the dependently-typed extension and the second-order extension of algebraic theories found in the literature.

4.1 Running example

To explain the syntax and inference rules of SOGATs, we use the *dependent type theory with* Π*-types* as a running example of a type theory. It has two judgment forms:

- U, for which a judgment $A : U$ expresses that A is a type in the type theory;
- $E(A)$ for any $A : U$, for which a judgment $a : E(A)$ expresses that a is an element of A.

The judgment $A : U$ is traditionally written like A type and the judgment $a : E(A)$ is traditionally written like $a : A$. Since we allow users to declare their own judgment forms, every construction of a judgment form should be explicit. The ordinary presentation of dependent type theory also contains judgment forms $A_1 \equiv A_2 : U$ and $a_1 \equiv a_2 : E(A)$ expressing type equality and term equality, respectively, but we consider that these equality judgment forms are automatically generated from U and $E(A)$, respectively. Π*-types* are specified by the inference rules listed in Fig. 4.1 where we omit the congruence rules for Π, λ, and @. These rules are actually *templates* of inference rules. For example, the first rule template

$$\frac{\vdash \mathtt{A} : U \qquad \mathtt{x} : E(\mathtt{A}) \vdash \mathtt{B} : U}{\vdash \Pi(\mathtt{A}, \mathtt{B}) : U}$$

induces the rule

$$\frac{\Gamma \vdash A : U \qquad \Gamma, \mathtt{x} : E(A) \vdash B : U}{\Gamma \vdash \Pi(A, \langle \mathtt{x} \rangle B) : U}$$

for any context Γ and expressions A and B, where the notation $\langle \mathtt{x} \rangle B$ means that the variable $\mathtt{x}$ is bound.

The dependent type theory with Π-types will be presented by a SOGAT shown in Fig. 4.2. This is read as a list of *symbols* and *axioms* with which rule templates are associated. The first two symbols, U and E, introduce judgment forms U and $E(-)$, respectively. When presenting a type theory as a SOGAT, *types* in the SOGAT represent *judgment forms* in the type theory. The difference between Type and type is that a type symbol is to be a judgment form by which contexts can be extended. The next three symbols, Π, λ, and @, correspond to the first three

$$\frac{\vdash \mathtt{A} : U \qquad \mathtt{x} : E(\mathtt{A}) \vdash \mathtt{B} : U}{\vdash \Pi(\mathtt{A}, \mathtt{B}) : U}$$

$$\frac{\vdash \mathtt{A} : U \qquad \mathtt{x} : E(\mathtt{A}) \vdash \mathtt{B} : U \qquad \mathtt{x} : E(\mathtt{A}) \vdash \mathtt{b} : E(\mathtt{B}(\mathtt{x}))}{\vdash \lambda(\mathtt{A}, \mathtt{B}, \mathtt{b}) : E(\Pi(\mathtt{A}, \mathtt{B}))}$$

$$\frac{\vdash \mathtt{A} : U \qquad \mathtt{x} : E(\mathtt{A}) \vdash \mathtt{B} : U \qquad \vdash \mathtt{f} : E(\Pi(\mathtt{A}, \mathtt{B})) \qquad \vdash \mathtt{a} : E(\mathtt{A})}{\vdash @(\mathtt{A}, \mathtt{B}, \mathtt{f}, \mathtt{a}) : E(\mathtt{B}(\mathtt{a}))}$$

$$\frac{\vdash \mathtt{A} : U \qquad \mathtt{x} : E(\mathtt{A}) \vdash \mathtt{B} : U \qquad \mathtt{x} : E(\mathtt{A}) \vdash \mathtt{b} : E(\mathtt{B}(\mathtt{x})) \qquad \vdash \mathtt{a} : E(\mathtt{A})}{\vdash @(\mathtt{A}, \mathtt{B}, \lambda(\mathtt{A}, \mathtt{B}, \mathtt{b}), \mathtt{a}) \equiv \mathtt{b}(\mathtt{a}) : E(\mathtt{B}(\mathtt{a}))}$$

$$\frac{\vdash \mathtt{A} : U \qquad \mathtt{x} : E(\mathtt{A}) \vdash \mathtt{B} : U \qquad \vdash \mathtt{f} : E(\Pi(\mathtt{A}, \mathtt{B}))}{\vdash \lambda(\mathtt{A}, \mathtt{B}, \langle \mathtt{x} \rangle @(\mathtt{A}, \mathtt{B}, \mathtt{f}, \mathtt{x})) \equiv \mathtt{f} : E(\Pi(\mathtt{A}, \mathtt{B}))}$$

Figure 4.1: Π-types

rule templates in Fig. 4.1. We note that a *unique* rule template is associated with each symbol, and thus SOGATs are designed to satisfy the *tightness* condition of Bauer, Haselwarter, and Lumsdaine [20] which is one of the requirements for acceptable type theories in their sense. We also note that there is no need to add the congruence rules for Π, λ, and @ because they can automatically be generated from the types of these symbols. The last two entries are axioms corresponding to the last two rule templates in Fig. 4.1.

We take a closer look at associated rule templates. For example, consider the symbol Π.

$$\Pi : (\mathtt{A} : () \to U, \mathtt{B} : (\mathtt{x} : E(\mathtt{A})) \to U) \Rightarrow U \tag{4.1}$$

The left side of $\Rightarrow$ is called an *environment* and is a list of *metavariables* equipped with certain typing data. In Eq. (4.1), A and B are metavariables. The environment associated with a symbol is to be the *premises* of a rule template. We note that a *unique* type is assigned to each metavariable, expressing the *tightness* of the rule template in the sense of Bauer, Haselwarter, and Lumsdaine [20]. The right side of $\Rightarrow$ is the return type of Π and corresponds to the judgment form of the *conclusion* of the rule template. Next, consider the metavariable B.

$$\mathtt{B} : (\mathtt{x} : E(\mathtt{A})) \to U$$

The left side of $\to$ is called a *context* and is a list of *variables* equipped with certain typing data. Contexts over a SOGAT are to be the ordinary contexts over a type theory.

$$
\begin{aligned}
&U : () \Rightarrow \mathsf{Type} \\
&E : (\mathtt{A} : () \to U) \Rightarrow \mathsf{type} \\
&\Pi : (\mathtt{A} : () \to U, \mathtt{B} : (\mathtt{x} : E(\mathtt{A})) \to U) \Rightarrow U \\
&\lambda : (\mathtt{A} : () \to U, \mathtt{B} : (\mathtt{x} : E(\mathtt{A})) \to U, \mathtt{b} : (\mathtt{x} : E(\mathtt{A})) \to E(\mathtt{B}(\mathtt{x}))) \Rightarrow E(\Pi(\mathtt{A}, \mathtt{B})) \\
&@ : (\mathtt{A} : () \to U, \mathtt{B} : (\mathtt{x} : E(\mathtt{A})) \to U, \mathtt{f} : () \to E(\Pi(\mathtt{A}, \mathtt{B})), \mathtt{a} : () \to E(\mathtt{A})) \\
&\qquad \Rightarrow E(\mathtt{B}(\mathtt{a})) \\
&_ : (\mathtt{A} : () \to U, \mathtt{B} : (\mathtt{x} : E(\mathtt{A})) \to U, \mathtt{b} : (\mathtt{x} : E(\mathtt{A})) \to E(\mathtt{B}(\mathtt{x})), \mathtt{a} : () \to E(\mathtt{A})) \\
&\qquad \Rightarrow @(\mathtt{A}, \mathtt{B}, \lambda(\mathtt{A}, \mathtt{B}, \mathtt{b}), \mathtt{a}) \equiv \mathtt{b}(\mathtt{a}) : E(\mathtt{B}(\mathtt{a})) \\
&_ : (\mathtt{A} : () \to U, \mathtt{B} : (\mathtt{x} : E(\mathtt{A})) \to U, \mathtt{f} : () \to E(\Pi(\mathtt{A}, \mathtt{B}))) \\
&\qquad \Rightarrow \lambda(\mathtt{A}, \mathtt{B}, \langle \mathtt{x} \rangle @(\mathtt{A}, \mathtt{B}, \mathtt{f}, \mathtt{x})) \equiv \mathtt{f} : E(\Pi(\mathtt{A}, \mathtt{B}))
\end{aligned}
$$

Figure 4.2: SOGAT for Π-types

Equation (4.1) contains essentially the same information as the first rule template of Fig. 4.1, but from Eq. (4.1) it is more apparent that Π is an "algebraic operator" that takes two arguments `A` and `B`. The types of `A` and `B` are more complex than those in ordinary algebraic theories. First, SOGATs are *dependently-typed* as the metavariable `A` appears in the type of `B`. Second, symbols in SOGATs may *bind variables*. The context associated with each metavariable in Eq. (4.1) expresses the variables bound by Π.

Because of this complexity, the definition of a SOGAT takes a few steps. We introduce a notion of a *symbol signature* in Section 4.2 which is a set of symbols equipped with certain data on *variable bindings*. For example, the symbol signature for dependent type theory with Π-types will be defined as in Fig. 4.3. Compared to Fig. 4.2, this symbol signature does not tell us the type of each symbol or metavariable, but specifies the number of arguments and bound variables. A symbol signature thus has enough information to build a set of *expressions*. In Section 4.3, we introduce a notion of a *pretheory* which is a signature equipped with typing data. It is called a *pre*theory because the type of each symbol is not necessarily *well-formed*. The notion of well-formedness only makes sense after a set of *derivations* is determined. We associate a set of *inference rules* with each pretheory and then build a set of derivations. We see in Section 4.4 basic properties of derivations including *stability under substitutions* and *contextual completeness*. A SOGAT is defined in Section 4.5 and should be a well-formed pretheory, but we further require a SOGAT to be *well-ordered* and *finitary* for technical convenience. These conditions are satisfied by practical pretheories and ensure that every SOGAT can be decomposed into small pieces so that the analysis of SOGATs becomes much easier. In Section 4.6 we give examples of SOGATs including Martin-Löf type theory [124, 125] and cubical type theory [41]. We

$$
\begin{aligned}
U &: () \Rightarrow \mathsf{Type} \\
E &: (\mathtt{A} : ()) \Rightarrow \mathsf{type} \\
\Pi &: (\mathtt{A} : (), \mathtt{B} : (\mathtt{x})) \Rightarrow \mathsf{term} \\
\lambda &: (\mathtt{A} : (), \mathtt{B} : (\mathtt{x}), \mathtt{b} : (\mathtt{x})) \Rightarrow \mathsf{term} \\
@ &: (\mathtt{A} : (), \mathtt{B} : (\mathtt{x}), \mathtt{f} : (), \mathtt{a} : ()) \Rightarrow \mathsf{term}
\end{aligned}
$$

Figure 4.3: Symbol signature for Π-types

also compare SOGATs to second-order algebraic theories, generalized algebraic theories, and general type theories of Bauer et al. In Section 4.7, we study environments over a SOGAT in detail which will play a central role in functorial semantics of SOGATs. Section 4.8 is devoted to developing semantics of SOGATs valued in CwRs.

4.1.1. Remark. The presentation in Fig. 4.2 might remind the reader about logical framework encodings of type theories [75, 133, 132, 61]. In a logical framework, the Π-type constructor is declared as a constant of the function type

$$(\mathtt{A} : U) \texttt{->} (E(\mathtt{A}) \texttt{->} U) \texttt{->} U.$$

The arrows "$\to$" and "$\Rightarrow$" in Fig. 4.2 are not function types and cannot be nested. Therefore, we are not allowed to write a higher-order operator like

$$((\mathtt{A} \to \mathtt{B}) \to \mathtt{C}) \Rightarrow \mathtt{D}$$

in a SOGAT. In general, an operator in a SOGAT is of the form

$$S : (\mathtt{X}_1 : \Gamma_1 \to e_1, \ldots, \mathtt{X}_n : \Gamma_n \to e_n) \Rightarrow e,$$

where Γ_i is a list of typed variables, and thus S is a "second-order" operator. SOGATs might thus sound more restrictive than logical framework encodings, but we will see in Section 4.6 that a bunch of examples of type constructors are second-order operators. We also emphasize that a wide range of existing type theories are verified to be instances of SOGATs by just unfolding the definition so that we do not need extra adequacy theorems to justify using SOGATs as syntactic presentations of type theories.

4.1.2. Remark. A similar restriction on the order of operators appears in Kaposi and Kovács's type theory for specifying higher inductive-inductive types [97, 98]. Their purpose of the restriction is to enforce strict positivity. Their type theory and SOGATs should be related, but we leave it as future work.

4.1.3. Remark. In Sections 4.2 to 4.4, we will work in a constructive metalogic to leave open the possibility of formalizing the results in a proof assistant and developing a new proof assistant based on SOGATs. From Section 4.5, we will use classical axioms to choose certain well-orderings. Perhaps we could keep working constructively by switching from well-orderings to well-founded relations following Bauer, Haselwarter, and Lumsdaine [20], but we mainly use well-orderings in the categorical study of SOGATs where classical axioms are already assumed, and thus we would not benefit from doing this constructively. We also crucially use the impredicativity of the internal language of a topos in the semantics of SOGATs.

4.2 Syntax of SOGATs

We introduce three kinds of signatures, *variable signatures*, *metavariable signatures*, and *symbol signatures*. A *signature* is a set whose elements are called variables, symbols, etc. such that, for each element, certain data such as the number of arguments and the number of bound variables for each argument are specified, but no typing data are specified. Given a symbol signature, a metavariable signature, and a variable signature, we build a set of *expressions*. We discuss *substitution properties* on variables and metavariables.

4.2.1 Signatures

We view a signature as a set equipped with some data for each element.

4.2.1. Definition. Let A be a (possibly large) set. A *signature valued in A* or *A-signature* is a family valued in A, that is, a set s equipped with a map $\mathrm{el}_s : s \to A$. We write $(x : a) \in s$ to mean that $x \in s$ and $\mathrm{el}_s\, x = a$. A finite A-signature is then denoted by a finite sequence of the form

$$(x_1 : a_1, \ldots, x_n : a_n)$$

though the ordering is not relevant.

4.2.2. Definition. A *variable signature* is a signature valued in the singleton $\{0\}$. Therefore, a variable signature is equivalent to just a set. Elements of a variable signature are called *variables*.

4.2.3. Definition. A *metavariable signature* is a signature valued in variable signatures. Elements of a metavariable signature are called *metavariables*.

4.2.4. Definition. By *syntactic classes*, we mean the formal symbols Type, type, Prop, prop, and term.

4.2.5. Definition. A *symbol signature* is a signature valued in pairs (μ, c) consisting of a metavariable signature μ and a syntactic class c. For a symbol signature Σ, we write $(S : \mu \Rightarrow c) \in \Sigma$ instead of $(S : (\mu, c)) \in \Sigma$. Elements of a symbol signature is called *symbols*. A symbol $S : \mu \Rightarrow c$ is said to be:

- a *type symbol* when $c = \mathsf{Type}$;
- a *representable type symbol* when $c = \mathsf{type}$;
- a *proposition symbol* when $c = \mathsf{Prop}$;
- a *representable proposition symbol* when $c = \mathsf{prop}$;
- a *term symbol* when $c = \mathsf{term}$.

4.2.6. Example. The signature for the dependent type theory with Π-types is defined as follows.

$$
\begin{aligned}
&U : () \Rightarrow \mathsf{Type} \\
&E : (\mathtt{A} : ()) \Rightarrow \mathsf{type} \\
&\Pi : (\mathtt{A} : (), \mathtt{B} : (\mathtt{x})) \Rightarrow \mathsf{term} \\
&\lambda : (\mathtt{A} : (), \mathtt{B} : (\mathtt{x}), \mathtt{b} : (\mathtt{x})) \Rightarrow \mathsf{term} \\
&@ : (\mathtt{A} : (), \mathtt{B} : (\mathtt{x}), \mathtt{f} : (), \mathtt{a} : ()) \Rightarrow \mathsf{term}
\end{aligned}
$$

In a symbol signature, we declare judgment forms as type symbols. Type and term constructors in the target type theory are declared as term symbols in a symbol signature. A representable type symbol $S : \mu \Rightarrow \mathsf{type}$ is to be a judgment form by which contexts can be extended. For example, contexts of the dependent type theory with Π-types can be extended by a variable of some type $\mathtt{x} : E(A)$ but not by a type variable $\mathtt{x} : U$, and thus only E should be a representable type symbol. The variable signature γ of a metavariable $(\mathtt{X} : \gamma) \in \mu$ for a symbol $S : \mu \Rightarrow c$ is understood as the variables bound by the operator S.

4.2.7. Remark. The motivation for including Prop and prop in syntactic classes comes from cubical type theory [41]. For the signature for cubical type theory, we will add symbols

$$
\begin{aligned}
&\mathsf{Cof} : () \Rightarrow \mathsf{Type} \\
&\mathsf{true} : (\mathtt{P} : ()) \Rightarrow \mathsf{prop}
\end{aligned}
$$

to deal with cofibrant propositions; see Section 4.6.3 for details. $\mathsf{true}(\mathtt{P})$ is a judgment form but also a judgment by itself meaning that the cofibrant proposition $\mathtt{P}$ is true.

The category of signatures

4.2.8. Definition. Let s_1 and s_2 be signatures valued in a set A. A *morphism* $r : s_1 \to s_2$ *of signatures* is a map $r : s_1 \to s_2$ between the underlying sets such that $\mathrm{el}_{s_1} = \mathrm{el}_{s_2} \circ r$.

The signatures valued in A and the morphisms of signatures form a category $\mathbf{Sig}_A$. Alternatively, it is defined to be the pullback

$$\begin{array}{ccc} \mathbf{Sig}_A & \longrightarrow & \widehat{\mathbf{Set}}/A \\ \downarrow & \lrcorner & \downarrow{\scriptstyle \mathrm{dom}} \\ \mathbf{Set} & \hookrightarrow & \widehat{\mathbf{Set}}. \end{array}$$

$\mathbf{Sig}_A$ shares some nice properties with slice categories, for example:

4.2.9. Proposition. *For any (possibly large) set A, the category $\mathbf{Sig}_A$ has colimits and the functor $\mathbf{Sig}_A \to \mathbf{Set}$ is conservative and preserves colimits.* □

4.2.2 Expressions

4.2.10. Definition. Let Σ be a symbol signature and μ a metavariable signature. We simultaneously define an inductive family $\mathrm{Expr}_{\Sigma,\mu}(\gamma, c)$ indexed over pairs of a variable signature γ and a syntactic class c, a notion of a *substitution*, and a notion of an *instantiation* as follows.

- For variable signatures γ and δ, a *substitution f of δ in γ over Σ and μ*, written $\Sigma, \mu \vdash f : \gamma \to \delta$, is a map $f : \delta \to \mathrm{Expr}_{\Sigma,\mu}(\gamma, \mathsf{term})$. We write $\mathrm{Subst}_{\Sigma,\mu}(\gamma, \delta)$ for the set of substitutions $\Sigma, \mu \vdash f : \gamma \to \delta$.
- For a variable signature γ and a metavariable signature ν, an *instantiation I of ν in μ over Σ and γ*, written $\Sigma, \mu \vdash I : \gamma \Rightarrow \nu$, is a dependent map $I : \prod_{(\Upsilon:\delta)\in\nu} \mathrm{Expr}_{\Sigma,\mu}(\gamma + \delta, \mathsf{term})$. We write $\mathrm{Inst}_{\Sigma,\mu}(\gamma, \nu)$ for the set of instantiations $\Sigma, \mu \vdash I : \gamma \Rightarrow \nu$.
- $\mathsf{var}(\mathtt{x}) \in \mathrm{Expr}_{\Sigma,\mu}(\gamma, \mathsf{term})$ for any variable $\mathtt{x} \in \gamma$.
- $\mathsf{mvar}(\mathtt{X}, f) \in \mathrm{Expr}_{\Sigma,\mu}(\gamma, \mathsf{term})$ for any metavariable $(\mathtt{X} : \delta) \in \mu$ and any substitution $\Sigma, \mu \vdash f : \gamma \to \delta$.
- $\mathsf{sym}(S, I) \in \mathrm{Expr}_{\Sigma,\mu}(\gamma, c)$ for any symbol $(S : \nu \Rightarrow c)$ and any instantiation $\Sigma, \mu \vdash I : \gamma \Rightarrow \nu$.
- $\mathsf{coe}_{\mathsf{type}}(A) \in \mathrm{Expr}_{\Sigma,\mu}(\gamma, \mathsf{Type})$ for any $A \in \mathrm{Expr}_{\Sigma,\mu}(\gamma, \mathsf{type})$.
- $\mathsf{coe}_{\mathsf{prop}}(p) \in \mathrm{Expr}_{\Sigma,\mu}(\gamma, \mathsf{Prop})$ for any $p \in \mathrm{Expr}_{\Sigma,\mu}(\gamma, \mathsf{prop})$.

- $\mathsf{eq}(K, a_1, a_2) \in \mathrm{Expr}_{\Sigma,\mu}(\gamma, \mathsf{Prop})$ for any $K \in \mathrm{Expr}_{\Sigma,\mu}(\gamma, \mathsf{Type})$ and any $a_1, a_2 \in \mathrm{Expr}_{\Sigma,\mu}(\gamma, \mathsf{term})$.

An element of $\mathrm{Expr}_{\Sigma,\mu}(\gamma, c)$ is called an *expression over* Σ, μ, *and* γ. An expression is called

- a *type expression* when $c = \mathsf{Type}$;
- a *representable type expression* when $c = \mathsf{type}$;
- a *proposition expression* when $c = \mathsf{Prop}$;
- a *representable proposition expression* when $c = \mathsf{prop}$;
- a *term expression* when $c = \mathsf{term}$;
- a *sort expression* when $c \in \{\mathsf{Type}, \mathsf{type}, \mathsf{Prop}, \mathsf{prop}\}$.

We will omit the subscripts $_\Sigma$ and $_\mu$ when they are clear from the context.

4.2.11. Notation. We introduce the following notations.

$$\begin{aligned}
\mathtt{x} &= \mathsf{var}(\mathtt{x}) \\
\mathtt{X}(f) &= \mathsf{mvar}(\mathtt{X}, f) \\
\mathtt{X} &= \mathtt{X}() && (\text{when } \mathtt{X} : ()) \\
S(I) &= \mathsf{sym}(S, I) \\
S &= S() && (\text{when } S : () \Rightarrow c) \\
A &= \mathsf{coe}_{\mathsf{type}}(A) \\
p &= \mathsf{coe}_{\mathsf{prop}}(p) \\
(a_1 \equiv a_2 : K) &= \mathsf{eq}(K, a_1, a_2)
\end{aligned}$$

4.2.12. Notation. A substitution $\Sigma, \mu \vdash f : \gamma \to \delta$ is written as a key-value list

$$(\mathtt{y}_1 := f_1, \ldots, \mathtt{y}_n := f_n)$$

when $\delta = (\mathtt{y}_1, \ldots, \mathtt{y}_n)$ or as a list

$$(f_1, \ldots, f_n)$$

when the order of the variables $\mathtt{y}_1, \ldots, \mathtt{y}_n$ is clear from the context. We use a similar notation for an instantiation.

4.2.13. Notation. We write $\langle \mathtt{x}_1, \ldots, \mathtt{x}_n \rangle e$ to emphasize that e is an expression over a variable signature of the form $\gamma + (\mathtt{x}_1, \ldots, \mathtt{x}_n)$. This notation mainly

appears in the application of a symbol to mean that the variables $\mathtt{x}_1, \ldots, \mathtt{x}_n$ are bound. For example, we have a term expression

$$\Pi(\mathtt{A}, \langle \mathtt{x} \rangle \mathtt{B}(\mathtt{a}(\mathtt{x})))$$

over the symbol signature for the dependent type theory with Π-types (Example 4.2.6) and the metavariable signature $(\mathtt{A} : (), \mathtt{B} : (\mathtt{x}), \mathtt{a} : (\mathtt{x}))$. We identify a metavariable $\mathtt{X} : (\mathtt{x}_1, \ldots, \mathtt{x}_n)$ with the expression $\langle \mathtt{x}_1, \ldots, \mathtt{x}_n \rangle \mathtt{X}(\mathtt{x}_1, \ldots, \mathtt{x}_n)$. For example, we write $\Pi(\mathtt{A}, \mathtt{B})$ instead of $\Pi(\mathtt{A}, \langle \mathtt{x} \rangle \mathtt{B}(\mathtt{x}))$ for metavariables $(\mathtt{A} : (), \mathtt{B} : (\mathtt{x}))$.

The action of morphisms of signatures

Variables signatures, metavariable signatures, and symbol signatures act on expressions:

- for any morphism $r : \gamma \to \gamma'$ of variable signatures, we have a map $(r \cdot -) : \mathrm{Expr}_{\Sigma,\mu}(\gamma, c) \to \mathrm{Expr}_{\Sigma,\mu}(\gamma', c)$;
- for any morphism $r : \mu \to \mu'$ of metavariable signatures, we have a map $(r \cdot -) : \mathrm{Expr}_{\Sigma,\mu}(\gamma, c) \to \mathrm{Expr}_{\Sigma,\mu'}(\gamma, c)$;
- for any morphism $r : \Sigma \to \Sigma'$ of symbol signatures, we have a map $(r \cdot -) : \mathrm{Expr}_{\Sigma,\mu}(\gamma, c) \to \mathrm{Expr}_{\Sigma',\mu}(\gamma, c)$.

The definitions of the action of a morphism of metavariable signatures and the action of a morphism of symbol signatures are straightforward by induction on expressions. The action of a morphism of variable signatures is also defined by induction, but we need little care in the case of $\mathsf{sym}(S, I)$. We define the expression $r \cdot \mathsf{sym}(S, I)$ to be $\mathsf{sym}(S, r \cdot I)$ where $\Sigma, \mu \vdash r \cdot I : \gamma' \Rightarrow \nu$ is the instantiation defined by $(r \cdot I)(\mathtt{Y} : \delta) = (r + \delta) \cdot I(\mathtt{Y})$, but $r + \delta$ is a morphism different from r. We thus have to generalize r and construct a map $(- \cdot e) : \forall \gamma'.\mathbf{Sig}(\gamma, \gamma') \to \mathrm{Expr}_{\Sigma,\mu}(\gamma', c)$ by induction on $e \in \mathrm{Expr}_{\Sigma,\mu}(\gamma, c)$.

4.2.14. Proposition. *For any monomorphism r of variable (metavariable or symbol) signatures, the map $(r \cdot -)$ is monic.*

Proof:
Straightforward. □

We thus regard $\mathrm{Expr}_{\Sigma',\mu'}(\gamma', c)$ as a subset of $\mathrm{Expr}_{\Sigma,\mu}(\gamma, c)$ when $\Sigma' \subset \Sigma$, $\mu' \subset \mu$ and $\gamma' \subset \gamma$.

4.2.3 Substitutions

Substitutions act on expressions. Let $e \in \mathrm{Expr}_{\Sigma,\mu}(\gamma, c)$ be an expression. We define by induction on e a map $(e \cdot -) : \forall\gamma'.\mathrm{Subst}_{\Sigma,\mu}(\gamma', \gamma) \to \mathrm{Expr}_{\Sigma,\mu}(\gamma', c)$. We only describe two selected cases, and the other cases are straightforward.

In the case of $\mathsf{var}(\mathtt{x})$, we simply define $\mathsf{var}(\mathtt{x}) \cdot f = f(\mathtt{x})$.

In the case of $\mathsf{sym}(S, I)$, we define $\mathsf{sym}(S, I) \cdot f = \mathsf{sym}(S, I \cdot f)$ where $\Sigma, \mu \vdash I \cdot f : \gamma' \Rightarrow \nu$ is the instantiation defined by $(I \cdot f)(\mathtt{Y} : \delta) = I(\mathtt{Y}) \cdot (f + \delta)$ and $\Sigma, \mu \vdash f + \delta : \gamma' + \delta \to \gamma + \delta$ is the substitution defined by $(f + \delta)(\mathrm{inl}\,\mathtt{x}) = \mathrm{inl} \cdot f(\mathtt{x})$ and $(f + \delta)(\mathrm{inr}\,\mathtt{y}) = \mathsf{var}(\mathrm{inr}\,\mathtt{y})$.

Variable signatures and substitutions form a category. The identity substitution $\Sigma, \mu \vdash \mathrm{id}_\gamma : \gamma \to \gamma$ is defined by $\mathrm{id}_\gamma(\mathtt{x}) = \mathsf{var}(\mathtt{x})$. The composition $g \circ f$ of two substitutions $\Sigma, \mu \vdash g : \gamma_2 \to \gamma_3$ and $\Sigma, \mu \vdash f : \gamma_1 \to \gamma_2$ is defined by $(g \circ f)(\mathtt{x}) = g(\mathtt{x}) \cdot f$.

4.2.15. Proposition. *For any expression $e \in \mathrm{Expr}_{\Sigma,\mu}(\gamma, c)$, the following hold:*

1. $e \cdot \mathrm{id}_\gamma = e$;
2. $e \cdot (g \circ f) = (e \cdot g) \cdot f$ *for any composable substitutions g and f.*

Proof:
By induction on e. □

Proposition 4.2.15 implies that $\mathrm{Expr}_{\Sigma,\mu}(-, \mathsf{term})$ is part of a monad on the category of variable signatures, that is, on the category of sets: the unit is $\mathsf{var} : \gamma \to \mathrm{Expr}_{\Sigma,\mu}(\gamma, \mathsf{term})$; the Kleisli extension of a map $f : \delta \to \mathrm{Expr}_{\Sigma,\mu}(\gamma, \mathsf{term})$ is the action $(- \cdot f) : \mathrm{Expr}_{\Sigma,\mu}(\delta, \mathsf{term}) \to \mathrm{Expr}_{\Sigma,\mu}(\gamma, \mathsf{term})$. Then a substitution is nothing but a morphism in the Kleisli category.

4.2.4 Instantiations

Instantiations act on expressions. Let $e \in \mathrm{Expr}_{\Sigma,\mu}(\gamma, c)$ be an expression. We define by induction on e a map $(e \cdot -) : \forall\gamma'\mu'.\mathrm{Inst}_{\Sigma,\mu'}(\gamma', \mu) \to \mathrm{Expr}_{\Sigma,\mu'}(\gamma' + \gamma, c)$. We only describe selected two cases, and the other cases are straightforward.

In the case of $\mathsf{mvar}(\mathtt{X}, f)$, we define $\mathsf{mvar}(\mathtt{X}, f) \cdot I = I(\mathtt{X}) \cdot (f \cdot I)$ where $\Sigma, \mu' \vdash f \cdot I : \gamma' + \gamma \to \gamma' + \delta$ is the substitution defined by $(f \cdot I)(\mathrm{inl}\,\mathtt{x}') = \mathsf{var}(\mathrm{inl}\,\mathtt{x}')$ and $(f \cdot I)(\mathrm{inr}\,\mathtt{y}) = f(\mathtt{y}) \cdot I$.

In the case of $\mathsf{sym}(S, I')$, we define $\mathsf{sym}(S, I') \cdot I = \mathsf{sym}(S, I' \circ I)$ where $\Sigma, \mu \vdash I' \circ I : \gamma' + \gamma \Rightarrow \nu$ is the instantiation defined by $(I' \circ I)(\mathtt{Y} : \delta) = I'(\mathtt{Y}) \cdot I$. Strictly speaking, $I'(\mathtt{Y}) \cdot I$ is an expression over $\gamma' + (\gamma + \delta)$ while $(I' \circ I)(\mathtt{Y} : \delta)$ is expected to be an expression over $(\gamma' + \gamma) + \delta$, and thus we have to insert the action of the canonical isomorphism $\gamma' + (\gamma + \delta) \cong (\gamma' + \gamma) + \delta$. For readability, we will not explicitly write the action of such a canonical isomorphism.

Metavariable signatures and instantiations form a category-like structure. We define the identity instantiation $\Sigma, \mu \vdash \mathrm{id}_\mu : 0 \Rightarrow \mu$ by $\mathrm{id}_\mu(\mathtt{X} : \gamma) = \mathsf{mvar}(\mathtt{X}, \mathrm{id}_\gamma)$. The composition $\Sigma, \mu_1 \vdash J \circ I : \gamma_1 + \gamma_2 \Rightarrow \mu_3$ of two instantiations $\Sigma, \mu_2 \vdash J : \gamma_2 \Rightarrow \mu_3$ and $\Sigma, \mu_1 \vdash I : \gamma_1 \Rightarrow \mu_2$ is defined in the previous paragraph.

4.2.16. Proposition. *For any expression $e \in \mathrm{Expr}_{\Sigma,\mu}(\gamma, c)$, the following hold:*

1. $e \cdot \mathrm{id}_\mu = e$;
2. $e \cdot (J \circ I) = (e \cdot J) \cdot I$ *for any composable instantiations* J *and* I;
3. $(e \cdot f) \cdot I = (e \cdot I) \cdot (f \cdot I)$ *for any substitution* $\Sigma, \mu \vdash f : \delta \to \gamma$ *and any instantiation* $\Sigma, \mu' \vdash I : \gamma' \Rightarrow \mu$;
4. $e \cdot (I \cdot f) = (e \cdot I) \cdot (f + \gamma)$ *for any instantiation* $\Sigma, \mu' \vdash I : \gamma' \Rightarrow \mu$ *and any substitution* $\Sigma, \mu \vdash f : \delta' \to \gamma'$.

Proof:
By induction on e. □

4.3 Inference rules of SOGATs

We introduce in Section 4.3.1 *contexts*, *environments*, and *pretheories* which are variable, metavariable, and symbol signatures, respectively, equipped with certain typing data. These are special kinds of *declarations* by which we mean signatures equipped with extra data. In Section 4.3.2, we introduce a notion of a *judgment*. For a pair consisting of a pretheory and an environment, we list a set of *inference rules* in Section 4.3.3 and build a set of *derivations* in Section 4.3.4. Then the set of *derivable judgments* is determined, and we introduce the *well-formedness conditions* in Section 4.3.5.

4.3.1 Declarations

We introduce a notion of a *declaration* which is similar to a signature and is a set of symbols, variables, etc. such that certain *typing* data is specified for each element. As special cases, we introduce *contexts*, *environments*, and *pretheories*. These are not defined as signatures because the possible values of type assignment depend on the set of symbols or variables. For example, the type of the symbol Π in the running example

$$\Pi : (\mathtt{A} : () \to U, \mathtt{B} : (\mathtt{x} : E(\mathtt{A})) \to U) \Rightarrow U$$

makes sense only in the presence of the symbols U and E. In general, the type of a symbol can depend on the whole symbol signature. A SOGAT will also contain

axioms to which proposition expressions are assigned instead of type expressions. Thus, the assignment should also depend on whether an entry is a symbol or an axiom. We can formulate this sort of dependency as follows.

4.3.1. Definition. Let A be a set and $B : \mathbf{Sig}_A \times A \to \mathbf{Set}$ a functor preserving monomorphisms where we regard the set A as a discrete category. An (A, B)-*declaration* d is an A-signature d equipped with a dependent map $\mathrm{val}_d : \prod_{x \in d} B(d, \mathrm{el}_d\, x)$. We write $(x : b) \in d$ to mean that $x \in d$ and $\mathrm{val}_d\, x = b$. A finite declaration is then written as a list of the form

$$(x_1 : b_1, \ldots, x_n : b_n).$$

Although this notation is similar to that of a finite signature, it expresses more complex data since the set that b_i belongs to depends on $\mathrm{el}\, x_i$ and the whole signature $(x_1, \ldots, x_n)$. A *morphism* $d_1 \to d_2$ *of* (A, B)*-declarations* is a morphism $r : d_1 \to d_2$ of A-signatures such that $r \cdot (\mathrm{val}_{d_1}\, x) = \mathrm{val}_{d_2}(r(x))$ for any $x \in d_1$.

Any pair (f, g) of a map $f : A \to A'$ and a natural transformation $g : B \Rightarrow B' \circ \big(\mathbf{Sig}_f \times f\big) : \mathbf{Sig}_A \times A \to \mathbf{Set}$ acts on declarations: for any (A, B)-declaration d, we have the (A', B')-declaration $(f, g) \cdot d$ whose underlying A'-signature is $f \cdot d$ and $\mathrm{val}_{(f,g)\cdot d}\, x = g(\mathrm{val}_d\, x)$.

For an A-signature s_0, by a *relative* (A, B)*-declaration over* s_0, we mean an $(A, B(s_0 + -_1, -_2))$-declaration. For an (A, B)-declaration d_1 and a relative (A, B)-declaration d_2 over d_1, the *extension* $d_1 \,.\, d_2$ is the (A, B)-declaration whose underlying A-signature is $d_1 + d_2$ and $\mathrm{val}_{d_1 . d_2}(\mathrm{inl}\, x_1) = \mathrm{inl} \cdot (\mathrm{val}_{d_1}\, x_1)$ and $\mathrm{val}_{d_1 . d_2}(\mathrm{inr}\, x_2) = \mathrm{val}_{d_2}\, x_2$.

4.3.2. Definition. Let d be an (A, B)-declaration. A *subdeclaration of* d is a subsignature $d' \subset d$ such that there exists a dependent map $\mathrm{val}_{d'} : \prod_{x \in d'} B(d', \mathrm{el}_d\, x)$ such that $r \cdot (\mathrm{val}_{d'}\, x) = \mathrm{val}_d(x)$ where $r : d' \to d$ is the inclusion. Since B preserves monomorphisms, such a map $\mathrm{val}_{d'}$ is unique. By definition, any subdeclaration d' of d is an (A, B)-declaration and the inclusion $d' \to d$ is a morphism of declarations.

Contexts

4.3.3. Definition. Let Σ be a symbol signature and μ a metavariable signature. A *context over* Σ *and* μ is an $(A_{\mathsf{ctx}}, B_{\mathsf{ctx}})$-declaration, where A_{ctx} and B_{ctx} are defined as follows.

- A_{ctx} is the two-element set $\{\mathsf{term}, \mathsf{proof}\}$. For an A_{ctx}-signature Γ, we write $\Gamma|_{\mathsf{term}}$ for the variable signature $\{\mathsf{x} \mid (\mathsf{x} : \mathsf{term}) \in \Gamma\}$. An element of the form $(H : \mathsf{proof}) \in \Gamma$ is called a *hypothesis*.
- $B_{\mathsf{ctx}}(\Gamma, \mathsf{term}) = \mathrm{Expr}_{\Sigma,\mu}(\Gamma|_{\mathsf{term}}, \mathsf{type})$.

- $B_{\mathsf{ctx}}(\Gamma, \mathsf{proof}) = \mathrm{Expr}_{\Sigma,\mu}(\Gamma|_{\mathsf{term}}, \mathsf{prop})$.

Any morphism of A_{ctx}-signatures $\Gamma_1 \to \Gamma_2$ induces a morphism of variable signatures $\Gamma_1|_{\mathsf{term}} \to \Gamma_2|_{\mathsf{term}}$, and thus B_{ctx} is a functor preserving monomorphisms.

A finite context is written as a list

$$(x_1 : e_1, \ldots, x_n : e_n)$$

such that each entry $(x_i : e_i)$ is either of the following forms:

- $(\mathtt{x} : A)$ where $\mathtt{x}$ is a variable and A is a representable type expression over the variables of the context;
- $(H : p)$ where H is a hypothesis and p is a representable proposition expression over the variables of the context.

Notice that hypotheses are never stored in expressions. The use of hypotheses is motivated by cubical type theory [41] in which proofs of cofibrant propositions are never stored in expressions.

Substitutions and instantiations act on (relative) contexts. For a substitution of the form $\Sigma, \mu \vdash f : \Gamma'|_{\mathsf{term}} \to \Gamma|_{\mathsf{term}}$, the action of $f + \mathrm{id}$ induces a natural transformation $(- \cdot f) : B_{\mathsf{ctx}}(\Gamma + -_1, -_2) \Rightarrow B_{\mathsf{ctx}}(\Gamma' + -_1, -_2)$. We thus have the relative context $\Delta \cdot f$ over Γ' for any relative context Δ over Γ. For an instantiation of the form $\Sigma, \mu' \vdash I : \Gamma'|_{\mathsf{term}} \Rightarrow \mu$, the action of I induces a natural transformation $(- \cdot I) : B_{\mathsf{ctx},\mu}(-_1, -_2) \Rightarrow B_{\mathsf{ctx},\mu'}(\Gamma' + -_1, -_2)$. We thus have the relative context $\Gamma \cdot I$ over Γ' for any context Γ.

Environments

4.3.4. Definition. Let Σ be a symbol signature. An *environment over* Σ is an $(A_{\mathsf{env}}, B_{\mathsf{env}})$-declaration, where A_{env} and B_{env} are defined as follows.

- A_{env} is the set of pairs (γ, c) consisting of an A_{ctx}-signature γ and $c \in \{\mathsf{term}, \mathsf{proof}\}$. For an A_{env}-signature Φ, we write $\Phi|_{\mathsf{term}}$ for the metavariable signature $\{(\mathtt{X} : \gamma|_{\mathsf{term}}) \mid (\mathtt{X} : (\gamma, \mathsf{term})) \in \Phi\}$. An element of the form $(H : (\gamma, \mathsf{proof})) \in \Phi$ is called an *assumption*.
- $B_{\mathsf{env}}(\Phi, (\gamma, \mathsf{term}))$ is the set of pairs (Γ, K) denoted by $\Gamma \to K$ consisting of a context Γ over Σ and $\Phi|_{\mathsf{term}}$ whose underlying A_{ctx}-signature is γ and $K \in \mathrm{Expr}_{\Sigma,\Phi|_{\mathsf{term}}}(\gamma|_{\mathsf{term}}, \mathsf{Type})$.
- $B_{\mathsf{env}}(\Phi, (\gamma, \mathsf{proof}))$ is the set of pairs (Γ, P) denoted by $\Gamma \to P$ consisting of a context Γ over Σ and $\Phi|_{\mathsf{term}}$ whose underlying A_{ctx}-signature is γ and $P \in \mathrm{Expr}_{\Sigma,\Phi|_{\mathsf{term}}}(\gamma|_{\mathsf{term}}, \mathsf{Prop})$.

Any morphism of A_{env}-signatures $\Phi_1 \to \Phi_2$ induces a morphism of metavariable signatures $\Phi_1|_{\mathsf{term}} \to \Phi_2|_{\mathsf{term}}$, and thus B_{env} is a functor preserving monomorphisms.

A finite environment is written as a list

$$(x_1 : \Gamma_1 \to e_1, \ldots, x_n : \Gamma_n \to e_n)$$

such that each entry $(x_i : \Gamma_i \to e_i)$ is either of the following forms:

- $(\mathtt{X} : \Gamma \to K)$ where $\mathtt{X}$ is a metavariable, Γ is a context over the metavariables of the environment, and K is a type expression over $\Gamma|_{\mathsf{term}}$;
- $(H : \Gamma \to P)$ where H is an assumption, Γ is a context over the metavariables of the environment, and P is a proposition expression over $\Gamma|_{\mathsf{term}}$.

Instantiations act on relative environments. Let I be an instantiation of the form $\Sigma, \Phi'|_{\mathsf{term}} \vdash I : \Gamma'|_{\mathsf{term}} \Rightarrow \Phi|_{\mathsf{term}}$. The map $(\gamma, c) \mapsto (\Gamma' + \gamma, c)$ acts on A_{env}-signatures: for an A_{env}-signature Φ, we have the A_{env}-signature $(\Gamma' + \Phi) = \{(x : (\Gamma' + \gamma, c)) \mid (x : (\gamma, c)) \in \Phi\}$. Extending I by $(\mathtt{X} : (\gamma, c)) \mapsto \mathtt{X}(\mathrm{id}_{\Gamma'+\gamma})$, we have an instantiation $\Sigma, (\Phi' + (\Gamma' + \Phi))|_{\mathsf{term}} \vdash I +_{\Gamma'} \Phi : \Gamma'|_{\mathsf{term}} \Rightarrow (\Phi + \Phi)|_{\mathsf{term}}$. Then the action of $I +_{\Gamma'} \Phi$ induces a natural transformation

$$(- \cdot I) : B_{\mathsf{env}}(\Phi + -_1, (-_2, -_3)) \to B_{\mathsf{env}}(\Phi' + (\Gamma' + -_1), (\Gamma' + -_2, -_3)).$$

We thus have a relative environment $\Psi \cdot I$ over Φ' for any relative environment Ψ over Φ.

Pretheories

4.3.5. Definition. A *pretheory* is an $(A_{\mathsf{preth}}, B_{\mathsf{preth}})$-declaration, where A_{preth} and B_{preth} are defined as follows.

- A_{preth} is the set of pairs (μ, c) where μ is an A_{env}-signature and c is either a syntactic class or the formal symbol proof. For an A_{preth}-signature T, we write $T|_{\mathsf{expr}}$ for the symbol signature $\{(S : \mu|_{\mathsf{term}} \Rightarrow c) \mid (S : (\mu, c)) \in T, c \neq \mathsf{proof}\}$. An element of the form $(H : (\mu, \mathsf{proof})) \in T$ is called an *axiom*.
- $B_{\mathsf{preth}}(T, (\mu, c))$ with $c \in \{\mathsf{Type}, \mathsf{type}, \mathsf{Prop}, \mathsf{prop}\}$ is the set of environments over $T|_{\mathsf{expr}}$ whose underlying A_{env}-signature is μ.
- $B_{\mathsf{preth}}(T, (\mu, \mathsf{term}))$ is the set of pairs (Φ, K) denoted by $\Phi \Rightarrow K$ consisting of an environment Φ over $T|_{\mathsf{expr}}$ whose underlying A_{env}-signature is μ and $K \in \mathrm{Expr}_{T|_{\mathsf{expr}}, \mu|_{\mathsf{term}}}(0, \mathsf{Type})$.
- $B_{\mathsf{preth}}(T, (\mu, \mathsf{proof}))$ is the set of pairs (Φ, P) denoted by $\Phi \Rightarrow P$ consisting of an environment Φ over $T|_{\mathsf{expr}}$ whose underlying A_{env}-signature is μ and $P \in \mathrm{Expr}_{T|_{\mathsf{expr}}, \mu|_{\mathsf{term}}}(0, \mathsf{Prop})$.

Any morphism of A_{preth}-signatures $T_1 \to T_2$ induces a morphism of symbol signatures $T_1|_{\mathsf{expr}} \to T_2|_{\mathsf{expr}}$, and thus B_{preth} is a functor preserving monomorphisms.

A finite pretheory is written as a list

$$(x_1 : \Phi_1 \Rightarrow e_1, \ldots, x_n : \Phi_n \Rightarrow e_n)$$

such that each entry $(x_i : \Phi_i \Rightarrow e_i)$ is either of the following forms:

- $(S : \Phi \Rightarrow c)$ where S is a sort symbol, $c \in \{\mathsf{Type}, \mathsf{type}, \mathsf{Prop}, \mathsf{prop}\}$, and Φ is an environment over the symbols of the pretheory;
- $(S : \Phi \Rightarrow K)$ where S is a term symbol, Φ is an environment over the symbols of the pretheory, and K is a type expression over $\Phi|_{\mathsf{term}}$;
- $(H : \Phi \Rightarrow P)$ where H is an axiom, Φ is an environment over the symbols of the pretheory, and P is a proposition expression over $\Phi|_{\mathsf{term}}$.

The name of a hypothesis, assumption, or axiom is often irrelevant and in that case we write $(_ : \ldots)$ instead of giving an explicit name $(H : \ldots)$.

4.3.6. Example. The dependent type theory with Π-types is defined by the following pretheory.

$$\begin{aligned}
&U : () \Rightarrow \mathsf{Type}\\
&E : (\mathtt{A} : () \to U) \Rightarrow \mathsf{type}\\
&\Pi : (\mathtt{A} : () \to U, \mathtt{B} : (\mathtt{x} : E(\mathtt{A})) \to U) \Rightarrow U\\
&\lambda : (\mathtt{A} : () \to U, \mathtt{B} : (\mathtt{x} : E(\mathtt{A})) \to U, \mathtt{b} : (\mathtt{x} : E(\mathtt{A})) \to E(\mathtt{B}(\mathtt{x}))) \Rightarrow E(\Pi(\mathtt{A}, \mathtt{B}))\\
&@ : (\mathtt{A} : () \to U, \mathtt{B} : (\mathtt{x} : E(\mathtt{A})) \to U, \mathtt{f} : () \to E(\Pi(\mathtt{A}, \mathtt{B})), \mathtt{a} : () \to E(\mathtt{A}))\\
&\qquad \Rightarrow E(\mathtt{B}(\mathtt{a}))\\
&_ : (\mathtt{A} : () \to U, \mathtt{B} : (\mathtt{x} : E(\mathtt{A})) \to U, \mathtt{b} : (\mathtt{x} : E(\mathtt{A})) \to E(\mathtt{B}(\mathtt{x})), \mathtt{a} : () \to E(\mathtt{A}))\\
&\qquad \Rightarrow @(\mathtt{A}, \mathtt{B}, \lambda(\mathtt{A}, \mathtt{B}, \mathtt{b}), \mathtt{a}) \equiv \mathtt{b}(\mathtt{a}) : E(\mathtt{B}(\mathtt{a}))\\
&_ : (\mathtt{A} : () \to U, \mathtt{B} : (\mathtt{x} : E(\mathtt{A})) \to U, \mathtt{f} : () \to E(\Pi(\mathtt{A}, \mathtt{B})))\\
&\qquad \Rightarrow \lambda(\mathtt{A}, \mathtt{B}, \langle \mathtt{x} \rangle @(\mathtt{A}, \mathtt{B}, \mathtt{f}, \mathtt{x})) \equiv \mathtt{f} : E(\Pi(\mathtt{A}, \mathtt{B}))
\end{aligned}$$

4.3.2 Judgments

4.3.7. Definition. Let Σ be a symbol signature, μ a metavariable signature and γ a variable signature.

- A *type judgment head over* Σ, μ, *and* γ is a pair (a, K) of a term expression a and a type expression K over Σ, μ, and γ. We write $a : K$ to mean that (a, K) is a type judgment head.

- A *proposition judgment head over* Σ, μ, *and* γ is a proposition expression P over Σ, μ, and γ.
- A *judgment head* is either a type judgment head or a proposition judgment head.

Substitutions and instantiations act on judgment heads by the action of them on expressions.

4.3.8. Definition. Let Σ be a symbol signature and μ a metavariable signature. A *judgment over* Σ *and* μ is a pair $(\Gamma, \mathcal{H})$ consisting of a context Γ over Σ and μ and a judgment head $\mathcal{H}$ over Σ, μ, and $\Gamma|_{\mathsf{term}}$. We write $\Gamma \to \mathcal{H}$ to mean that $(\Gamma, \mathcal{H})$ is a judgment.

Instantiations act on judgments. Suppose that we are given an instantiation of the form $\Sigma, \mu' \vdash I : \Gamma'|_{\mathsf{term}} \Rightarrow \mu$ and a judgment $\Gamma \to \mathcal{H}$ over Σ and μ. Then $\Gamma \cdot I$ is a relative context over Γ', and we have the extended context $\Gamma' \,.\, (\Gamma \cdot I)$. Then $\Gamma' \,.\, (\Gamma \cdot I) \to \mathcal{H} \cdot I$ is a judgment over Σ and μ'.

4.3.9. Remark. By definition, the possible judgments are either

- $\Gamma \to a : K$ for a term expression a and a type expression K, or
- $\Gamma \to P$ for a proposition expression P.

We do not include judgments for the well-formedness of sort expressions like $\Gamma \to K\ \mathsf{Type}$ in ordinary dependent type theory. We do not need such a judgment because, since any type expression is of the form $S(I)$ for a type symbol S and an instantiation I, the well-formedness of the type expression is reduced to the well-formedness of the instantiation I. There is also a reason for explicitly excluding such a judgment. Recall that when defining a type theory as a SOGAT, a type expression K over the SOGAT represents a judgment form in the type theory. Then the judgment $\Gamma \to K\ \mathsf{Type}$ would mean that K is a well-formed judgment form, but the usual presentation of a type theory does not have such a judgment on a judgment form. If a SOGAT contained a judgment that does not appear in a type theory in the real world, we would have to provide some adequacy theorem, but one of motivations for SOGATs is to avoid this situation.

4.3.10. Example. Consider the pretheory for the dependent type theory with Π-types (Example 4.3.6). Observe that the only possible type expressions are either U or $E(A)$ for a term expression A and that the only possible proposition expressions are either $(A_1 \equiv A_2 : U)$ for term expressions A_1 and A_2 or $(a_1 \equiv a_2 : E(A))$ for term expression A, a_1, and a_2. Therefore, the only possible judgment heads are

$$A : U \qquad a : E(A) \qquad A_1 \equiv A_2 : U \qquad a_1 \equiv a_2 : E(A)$$

which exactly match the possible judgment heads in the usual presentation of the dependent type theory with Π-types.

Pseudo-judgments

We will introduce the following notations:

- $\Gamma \to f : \Delta$ meaning that a substitution f is well-formed;
- $\Gamma \to I : \Phi$ meaning that an instantiation I is well-formed;
- $\Gamma \to e\ \mathsf{ok}$ meaning that a sort expression e is well-formed.

These are all families of judgments, but we treat them as if they were single judgments so that the action of substitutions and instantiations is defined as expected. For example, we have $(\Gamma \to f : \Delta) \cdot g = (\Gamma' \to f \circ g : \Delta)$ for another substitution $g : \Gamma'|_{\mathsf{term}} \to \Gamma|_{\mathsf{term}}$. We call a certain kind of family of judgments a *pseudo-judgment*.

4.3.11. Definition. Let Σ be a symbol signature and μ a metavariable signature. For a context Γ over Σ and μ, a *pseudo-judgment over* Γ is a family of judgments of the form $\{\Gamma \,.\, \Delta_\xi \to \mathcal{H}_\xi\}_{\xi \in \Xi}$. A pseudo-judgment over Γ is often written in the form $\Gamma \to \mathcal{J}$ for some notation $\mathcal{J}$.

Substitutions act on pseudo-judgments. For a pseudo-judgment $\{\Gamma \,.\, \Delta_\xi \to \mathcal{H}_\xi\}_{\xi \in \Xi}$ and a substitution $f : \Gamma'|_{\mathsf{term}} \to \Gamma|_{\mathsf{term}}$, we have the pseudo-judgment $\{\Gamma' \,.\, (\Delta_\xi \cdot f) \to \mathcal{H}_\xi \cdot f\}_{\xi \in \Xi}$.

4.3.12. Notation. Let Γ and Δ be contexts over Σ and μ. For a substitution $\Sigma, \mu \vdash f : \Gamma|_{\mathsf{term}} \to \Delta|_{\mathsf{term}}$, we write

$$\Gamma \to f : \Delta$$

for the pseudo-judgment consisting of:

- $\Gamma \to f(\mathtt{x}) : A \cdot f$ for any variable $(\mathtt{x} : A) \in \Delta$;
- $\Gamma \to p \cdot f$ for any hypothesis $(H : p) \in \Delta$.

For two parallel substitutions f_1 and f_2, we write

$$\Gamma \to f_1 \equiv f_2 : \Delta$$

for the pseudo-judgment consisting of $\Gamma \to f_1(\mathtt{x}) \equiv f_2(\mathtt{x}) : A \cdot f_1$ for any variable $(\mathtt{x} : A) \in \Delta$.

For another substitution $g : \Gamma'|_{\mathsf{term}} \to \Gamma|_{\mathsf{term}}$, we have

$$(\Gamma \to f : \Delta) \cdot g = (\Gamma' \to f \circ g : \Delta)$$
$$(\Gamma \to f_1 \equiv f_2 : \Delta) \cdot g = (\Gamma' \to f_1 \circ g \equiv f_2 \circ g : \Delta).$$

4.3.13. NOTATION. Let Σ be a symbol signature, Φ and Ψ environments over Σ and Γ a context over Σ and $\Phi|_{\mathsf{term}}$. For an instantiation $\Sigma, \Phi|_{\mathsf{term}} \vdash I : \Gamma|_{\mathsf{term}} \Rightarrow \Psi|_{\mathsf{term}}$, we write

$$\Gamma \to I : \Psi$$

for the pseudo-judgment consisting of:

- $\Gamma \,.\, (\Delta \cdot I) \to I(\mathtt{X}) : K \cdot I$ for any metavariable $(\mathtt{X} : \Delta \to K) \in \Psi$;
- $\Gamma \,.\, (\Delta \cdot I) \to P \cdot I$ for any assumption $(H : \Delta \to P) \in \Psi$.

For two parallel instantiations I_1 and I_2, we write

$$\Gamma \to I_1 \equiv I_2 : \Psi$$

for the pseudo-judgment consisting of $\Gamma \,.\, (\Delta \cdot I_1) \to I_1(\mathtt{X}) \equiv I_2(\mathtt{X}) : K \cdot I_1$ for any metavariable $(\mathtt{X} : \Delta \to K) \in \Psi$.

For a substitution $f : \Gamma' \to \Gamma$, we have

$$(\Gamma \to I : \Psi) \cdot f = (\Gamma' \to I \cdot f : \Psi)$$
$$(\Gamma \to I_1 \equiv I_2 : \Psi) \cdot f = (\Gamma' \to I_1 \cdot f \equiv I_2 \cdot f : \Psi).$$

4.3.14. NOTATION. Let T be a pretheory, μ a metavariable signature, and Γ a context over $T|_{\mathsf{expr}}$ and μ. For a syntactic class $c \in \{\mathsf{Type}, \mathsf{type}, \mathsf{Prop}, \mathsf{prop}\}$ and an expression $e \in \mathrm{Expr}_{T|_{\mathsf{expr}}, \mu}(\Gamma|_{\mathsf{term}}, c)$, we inductively define a pseudo-judgment

$$\Gamma \to e \ \mathsf{ok}$$

as follows:

- $\Gamma \to \mathsf{sym}(S, I)\ \mathsf{ok}$ is $\Gamma \to I : \Psi$ for a symbol $(S : \Psi \Rightarrow c) \in T$ and an instantiation $T|_{\mathsf{expr}}, \mu \vdash I : \Gamma|_{\mathsf{term}} \Rightarrow \Psi|_{\mathsf{term}}$;
- $\Gamma \to \mathsf{coe}_{\mathsf{type}}(A)\ \mathsf{ok}$ is $\Gamma \to A\ \mathsf{ok}$ for a representable type expression A;
- $\Gamma \to \mathsf{coe}_{\mathsf{prop}}(p)\ \mathsf{ok}$ is $\Gamma \to p\ \mathsf{ok}$ for a representable proposition expression p;
- $\Gamma \to \mathsf{eq}(K, a_1, a_2)\ \mathsf{ok}$ is the two judgments $\Gamma \to a_1 : K$ and $\Gamma \to a_2 : K$ for a type expression K and term expressions a_1 and a_2.

Alternatively, we may write

$$\Gamma \to e\ \mathsf{Type} \qquad \Gamma \to e\ \mathsf{type} \qquad \Gamma \to e\ \mathsf{Prop} \qquad \Gamma \to e\ \mathsf{prop}$$

to emphasize that e is a type, representable type, proposition, and representable proposition expression, respectively.

For a substitution $f : \Gamma' \to \Gamma$, we have

$$(\Gamma \to e\ \mathsf{ok}) \cdot f = (\Gamma' \to (e \cdot f)\ \mathsf{ok}).$$

Presuppositions

4.3.15. Definition. The *presuppositions* of a judgment $\Gamma \to \mathcal{H}$ is defined as follows:

- $\Gamma \to K\ \mathsf{ok}$ when $\mathcal{H}$ is a type judgment head $(a : K)$;
- $\Gamma \to P\ \mathsf{ok}$ when $\mathcal{H}$ is a proposition judgment head P.

4.3.3 Inference rules

Let T be a pretheory and Φ an environment over $T|_{\mathsf{expr}}$. We list up the *inference rules associated with T and Φ*. Each rule is of the form

$$\mathrm{R}\ \frac{\Gamma \to \mathcal{J}}{\Gamma \to \mathcal{H}}$$

where R is the name of the rule, Γ is a context, $\mathcal{J}$ is a pseudo-judgment over Γ, and $\mathcal{H}$ is a judgment head over Γ. By the definition of a pseudo-judgment, it is also written in the form

$$\mathrm{R}\ \frac{\{\Gamma\ .\ \Delta_\xi \to \mathcal{H}_\xi\}_{\xi \in \Xi}}{\Gamma \to \mathcal{H}}.$$

The upper judgments $(\Gamma\ .\ \Delta_\xi \to \mathcal{H}_\xi)$'s are called the *premises* and the lower judgment $\Gamma \to \mathcal{H}$ is called the *conclusion*.

4.3.16. Remark. We note that all the rules except the symmetry and transitivity rules for $\equiv$ are *strictly presuppositive* in the sense that the presuppositions of the conclusion are included in the premises, in contrast to the formulation by Bauer, Haselwarter, and Lumsdaine [20] where the presuppositions of the conclusion are required to be derivable over the premises. This change does not affect the notion of derivability when certain well-foundedness/well-orderedness conditions are satisfied. The strict presuppositivity is convenient for some inductive proofs: we will use the induction hypotheses on presuppositions in the proof of Proposition 4.4.9.

Variables and hypotheses

We add the following rules to introduce variables and hypotheses:

$$\textsc{Var}(\mathtt{x})\ \frac{\Gamma \to A\ \mathsf{ok}}{\Gamma \to \mathtt{x} : A}$$

for any variable $(\mathtt{x} : A) \in \Gamma$;

$$\textsc{Hyp}(H)\ \frac{\Gamma \to p\ \mathsf{ok}}{\Gamma \to p}$$

for any hypothesis $(H : p) \in \Gamma$.

Metavariables and assumptions

We add the following rules to introduce metavariables and assumptions:

$$\textsc{MVar}(\mathtt{X}, f)\ \frac{\Gamma \to f : \Delta \qquad \Gamma \to K \cdot f\ \mathsf{ok}}{\Gamma \to \mathtt{X}(f) : K \cdot f}$$

for any metavariable $(\mathtt{X} : \Delta \to K) \in \Phi$ and any substitution $f : \Gamma|_{\mathsf{term}} \to \Delta|_{\mathsf{term}}$;

$$\textsc{Asm}(H, f)\ \frac{\Gamma \to f : \Delta \qquad \Gamma \to P \cdot f\ \mathsf{ok}}{\Gamma \to P \cdot f}$$

for any assumption $(H : \Delta \to P) \in \Phi$ and any substitution $f : \Gamma|_{\mathsf{term}} \to \Delta|_{\mathsf{term}}$. For a metavariable $(\mathtt{X} : \Delta \to K) \in \Phi$, we also add the congruence rule

$$\equiv\text{-}\textsc{MVar}(\mathtt{X}, f_1, f_2)\ \frac{\Gamma \to f_1 \equiv f_2 : \Delta \qquad \Gamma \to (\mathtt{X}(f_1) \equiv \mathtt{X}(f_2) : K \cdot f_1)\ \mathsf{ok}}{\Gamma \to \mathtt{X}(f_1) \equiv \mathtt{X}(f_2) : K \cdot f_1}$$

for any substitutions $f_1, f_2 : \Gamma|_{\mathsf{term}} \to \Delta|_{\mathsf{term}}$.

Symbols and axioms

We add the following rules to introduce term symbols and axioms:

$$\textsc{Sym}(S, I)\ \frac{\Gamma \to I : \Psi \qquad \Gamma \to K \cdot I\ \mathsf{ok}}{\Gamma \to S(I) : K \cdot I}$$

for any term symbol $(S : \Psi \Rightarrow K) \in T$ and any instantiations $I : \Gamma|_{\mathsf{term}} \Rightarrow \Psi|_{\mathsf{term}}$;

$$\textsc{Axiom}(H, I)\ \frac{\Gamma \to I : \Psi \qquad \Gamma \to P \cdot I\ \mathsf{ok}}{\Gamma \to P \cdot I}$$

for any axiom $(H : \Psi \Rightarrow P) \in T$ and any instantiation $I : \Gamma|_{\mathsf{term}} \Rightarrow \Psi|_{\mathsf{term}}$. For a term symbol $(S : \Psi \Rightarrow K) \in T$, we also add the congruence rule

$$\equiv\text{-}\textsc{Sym}(S, I_1, I_2)\ \frac{\Gamma \to I_1 \equiv I_2 : \Psi \qquad \Gamma \to (S(I_1) \equiv S(I_2) : K \cdot I_1)\ \mathsf{ok}}{\Gamma \to S(I_1) \equiv S(I_2) : K \cdot I_1}$$

for any instantiations $I_1, I_2 : \Gamma|_{\mathsf{term}} \Rightarrow \Psi|_{\mathsf{term}}$. For sort symbols, we add the following conversion rules:

$$\textsc{TypeConv}(S, I_1, I_2, a)\ \frac{\Gamma \to I_1 \equiv I_2 : \Psi \qquad \Gamma \to a : S(I_1) \qquad \Gamma \to S(I_2)\ \mathsf{ok}}{\Gamma \to a : S(I_2)}$$

$$\textsc{TypeConvEq}(S, I_1, I_2, a_1, a_2)\ \frac{\begin{array}{c}\Gamma \to I_1 \equiv I_2 : \Psi \\ \Gamma \to a_1 \equiv a_2 : S(I_1) \qquad \Gamma \to (a_1 \equiv a_2 : S(I_2))\ \mathsf{ok}\end{array}}{\Gamma \to a_1 \equiv a_2 : S(I_2)}$$

for any (representable) type symbol $(S : \Psi \Rightarrow c) \in T$, any instantiations $I_1, I_2 : \Gamma|_{\mathsf{term}} \Rightarrow \Psi|_{\mathsf{term}}$, and any term expressions a, a_1, a_2;

$$\textsc{PropConv}(S, I_1, I_2)\ \frac{\Gamma \to I_1 \equiv I_2 : \Psi \qquad \Gamma \to S(I_1) \qquad \Gamma \to S(I_2)\ \mathsf{ok}}{\Gamma \to S(I_2)}$$

for any (representable) proposition symbol $(S : \Psi \Rightarrow c) \in T$ and any instantiations $I_1, I_2 : \Gamma|_{\mathsf{term}} \Rightarrow \Psi|_{\mathsf{term}}$.

4.3.17. Example. We describe some of the rules associated with the symbols and axioms of the dependent type theory with Π-types (Example 4.3.6). Since the symbol U takes no argument, the conversion rules are reduced to trivial ones.

$$\frac{\Gamma \to A : U}{\Gamma \to A : U} \qquad \frac{\Gamma \to A_1 \equiv A_2 : U \qquad \Gamma \to A_1 : U \qquad \Gamma \to A_2 : U}{\Gamma \to A_1 \equiv A_2 : U}$$

The usual presentation of dependent type theory does not contain these trivial rules, but these rules do not affect the notion of derivability. For the symbol E, we have the conversion rules.

$$\frac{\Gamma \to A_1 \equiv A_2 : U \qquad \Gamma \to a : A_1 \qquad \Gamma \to A_2 : U}{\Gamma \to a : E(A_2)}$$

$$\frac{\begin{array}{c}\Gamma \to A_1 \equiv A_2 : U \\ \Gamma \to a_1 \equiv a_2 : E(A_1) \qquad \Gamma \to a_1 : E(A_2) \qquad \Gamma \to a_2 : E(A_2)\end{array}}{\Gamma \to a_1 \equiv a_2 : E(A_2)}$$

For the symbol Π, we have the formation rule and the congruence rule.

$$\frac{\Gamma \to A : U \qquad \Gamma \,.\, (\mathtt{x} : E(A)) \to B : U}{\Gamma \to \Pi(A, \langle \mathtt{x} \rangle B) : U}$$

$$\frac{\begin{array}{c}\Gamma \to A_1 \equiv A_2 : U \qquad \Gamma \,.\, (\mathtt{x} : E(A_1)) \to B_1 \equiv B_2 : U \\ \Gamma \to \Pi(A_1, \langle \mathtt{x} \rangle B_1) : U \qquad \Gamma \to \Pi(A_2, \langle \mathtt{x} \rangle B_2) : U\end{array}}{\Gamma \to \Pi(A_1, \langle \mathtt{x} \rangle B_1) \equiv \Pi(A_2, \langle \mathtt{x} \rangle B_2) : U}$$

For the symbol λ, we have the introduction rule and the congruence rule.

$$\frac{\begin{array}{c}\Gamma \to A : U \\ \Gamma \,.\, (\mathtt{x} : E(A)) \to B : U \qquad \Gamma \,.\, (\mathtt{x} : E(A)) \to b : E(B) \qquad \Gamma \to \Pi(A, \langle \mathtt{x} \rangle B) : U\end{array}}{\Gamma \to \lambda(A, \langle \mathtt{x} \rangle B, \langle \mathtt{x} \rangle b) : E(\Pi(A, \langle \mathtt{x} \rangle B))}$$

$$\frac{\begin{array}{c}\Gamma \to A_1 \equiv A_2 : U \qquad \Gamma \,.\, (\mathtt{x} : E(A)) \to B_1 \equiv B_2 : U \\ \Gamma \,.\, (\mathtt{x} : E(A)) \to b_1 \equiv b_2 : E(B_1) \qquad \Gamma \to \lambda(A_1, \langle \mathtt{x} \rangle B_1, \langle \mathtt{x} \rangle b_1) : E(\Pi(A_1, \langle \mathtt{x} \rangle B_1)) \\ \Gamma \to \lambda(A_2, \langle \mathtt{x} \rangle B_2, \langle \mathtt{x} \rangle b_2) : E(\Pi(A_1, \langle \mathtt{x} \rangle B_1))\end{array}}{\Gamma \to \lambda(A_1, \langle \mathtt{x} \rangle B_1, \langle \mathtt{x} \rangle b_1) \equiv \lambda(A_2, \langle \mathtt{x} \rangle B_2, \langle \mathtt{x} \rangle b_2) : E(\Pi(A_1, \langle \mathtt{x} \rangle B_1))}$$

One might think that the presupposition $\Gamma \to \Pi(A, \langle \mathtt{x} \rangle B) : U$ in the premises of the introduction rule is redundant because it is derivable from the other premises by the Π-formation rule. However, there is no reason for a general presupposition to be derivable from premises. We thus require to provide a derivation of the presuppositions whenever we use a symbol. That redundancy does not affect the notion of derivability for reasonable type theories.

Equalities

We add the following rules to make $\equiv$ an equivalence relation:

$$\textsc{Refl}(K, a)\ \frac{\Gamma \to a : K}{\Gamma \to a \equiv a : K} \qquad \textsc{Smtry}(K, a_1, a_2)\ \frac{\Gamma \to a_1 \equiv a_2 : K}{\Gamma \to a_2 \equiv a_1 : K}$$

$$\textsc{Trans}(K, a_1, a_2, a_3)\ \frac{\Gamma \to a_1 \equiv a_2 : K \qquad \Gamma \to a_2 \equiv a_3 : K}{\Gamma \to a_1 \equiv a_3 : K}$$

for any type expression K and term expressions a, a_1, a_2, a_3.

Substitutions (optional)

The following substitution rules are optional.

$$\textsc{Subst}(f, \mathcal{H})\ \frac{\Gamma \to f : \Delta \qquad \Delta \to \mathcal{H}}{\Gamma \to \mathcal{H} \cdot f}$$

$$\equiv\text{-}\textsc{Subst}(f_1, f_2, a, K)\ \frac{\begin{array}{c}\Gamma \to f_1 : \Delta \\ \Gamma \to f_2 : \Delta \qquad \Gamma \to f_1 \equiv f_2 : \Delta \qquad \Delta \to a : K\end{array}}{\Gamma \to a \cdot f_1 \equiv a \cdot f_2 : K \cdot f_1}$$

In Section 4.4.1 we will see that these rules are admissible over the other rules in the sense that given derivations of the premises, one can construct a derivation of the conclusion. Therefore, the substitution rules do not affect the set of derivable judgments. In this thesis, we consider the version without the substitution rules.

4.3.4 Derivations

Having defined the set of inference rules, we obtain the set of *derivations*. A notion of a derivation is defined in a more general context.

4.3.18. Definition. Let A be a set. A *closure rule on* A is a pair $(\mathscr{P}, \mathcal{J})$ consisting of a family $\mathscr{P}$ of elements of A called the *premises* and an element

$\mathcal{J} \in A$ called the *conclusion*. A *closure system on* A is a family $\mathscr{F}$ of closure rules on A. We write

$$\mathrm{R}\ \frac{\mathscr{P}}{\mathcal{J}}$$

to mean that R is a closure rule in $\mathscr{F}$ with premises $\mathscr{P}$ and conclusion $\mathcal{J}$. Any map $f : A \to B$ acts on closure rules and closure systems component-wise.

4.3.19. Definition. Let $\mathscr{F}$ be a closure system on a set A and $\mathscr{P}$ a family of elements of A. We define an inductive family $\mathrm{Deriv}_{\mathscr{F}}(\mathscr{P}, \mathcal{J})$ indexed over the elements $\mathcal{J}$ of A as follows:

- $\mathsf{prem}(\mathcal{J}) \in \mathrm{Deriv}_{\mathscr{F}}(\mathscr{P}, \mathcal{J})$ for any $\mathcal{J} \in \mathscr{P}$;
- $\mathsf{app}(\mathrm{R}, D) \in \mathrm{Deriv}_{\mathscr{F}}(\mathscr{P}, \mathcal{J})$ for any rule $\mathrm{R}\ \dfrac{\mathscr{Q}}{\mathcal{J}}$ in $\mathscr{F}$ and any family of derivations $D \in \prod_{\mathcal{K} \in \mathscr{Q}} \mathrm{Deriv}_{\mathscr{F}}(\mathscr{P}, \mathcal{K})$.

Elements of $\mathrm{Deriv}_{\mathscr{F}}(\mathscr{P}, \mathcal{J})$ are called *derivations over* $\mathscr{F}$ *from* $\mathscr{P}$ *to* $\mathcal{J}$. For a family $\mathscr{Q}$ of elements of A, we define $\mathrm{Deriv}_{\mathscr{F}}(\mathscr{P}, \mathscr{Q}) = \prod_{\mathcal{J} \in \mathscr{Q}} \mathrm{Deriv}_{\mathscr{F}}(\mathscr{P}, \mathcal{J})$. We say $\mathcal{J}$ is *derivable over* $\mathscr{F}$ *from* $\mathscr{P}$ if there exists a derivation from $\mathscr{P}$ to $\mathcal{J}$. When $\mathcal{J}$ is derivable from the empty family, we simply say $\mathcal{J}$ is *derivable over* $\mathscr{F}$ and write $\mathscr{F} \vdash \mathcal{J}$. For a family $\mathscr{Q}$ of elements of A, we write $\mathscr{F} \vdash \mathscr{Q}$ when $\mathscr{F} \vdash \mathcal{J}$ for all $\mathcal{J} \in \mathscr{Q}$.

A derivation can be seen as a labeled tree: $\mathsf{prem}(\mathcal{J})$ is a leaf labeled by $\mathcal{J}$; and $\mathsf{app}\big(\mathrm{R}, \{D_{\mathcal{K}}\}_{\mathcal{K} \in \mathscr{Q}}\big)$ is a node labeled by R with branches $\{D_{\mathcal{K}}\}_{\mathcal{K} \in \mathscr{Q}}$.

We have some operations on derivations. Any rule $\mathrm{R}\ \dfrac{\mathscr{P}}{\mathcal{J}}$ in $\mathscr{F}$ is identified with the derivation $\mathsf{app}\big(\mathrm{R}, \{\mathsf{prem}(\mathcal{J})\}_{\mathcal{J} \in \mathscr{P}}\big) \in \mathrm{Deriv}_{\mathscr{F}}(\mathscr{P}, \mathcal{J})$. For a derivation $D \in \mathrm{Deriv}_{\mathscr{F}}(\mathscr{P}, \mathcal{J})$ and a family of derivations $D' \in \mathrm{Deriv}_{\mathscr{F}}(\mathscr{P}', \mathscr{P})$, we have the *composition* $D \circ D' \in \mathrm{Deriv}_{\mathscr{F}}(\mathscr{P}', \mathcal{J})$ defined by $\mathsf{prem}(\mathcal{J}) \circ D' = D'_{\mathcal{J}}$ and $\mathsf{app}(\mathrm{R}, D) \circ D' = \mathsf{app}(\mathrm{R}, \{D_{\mathcal{K}} \circ D'\}_{\mathcal{K}})$. Let $\mathscr{F}$ be a closure system on a set A and $\mathscr{F}'$ a closure system on a set A'. Any map f that sends elements of A to those of A' and rules $\mathrm{R}\ \dfrac{\mathscr{P}}{\mathcal{J}}$ in $\mathscr{F}$ to derivations $f(\mathrm{R}) \in \mathrm{Deriv}_{\mathscr{P}'}(f \cdot \mathscr{P}, f(\mathcal{J}))$ is extended to a map $(f \cdot -) : \forall \mathscr{P} \mathcal{J}.\mathrm{Deriv}_{\mathscr{F}}(\mathscr{P}, \mathcal{J}) \to \mathrm{Deriv}_{\mathscr{F}'}(f \cdot \mathscr{P}, f(\mathcal{J}))$ defined by $f \cdot \mathsf{prem}(\mathcal{J}) = \mathsf{prem}(f(\mathcal{J}))$ and $f \cdot \mathsf{app}(\mathrm{R}, D) = f(\mathrm{R}) \circ \{f \cdot D_{\mathcal{K}}\}_{\mathcal{K}}$.

4.3.20. Definition. Let T be a pretheory and Φ an environment over $T|_{\mathsf{expr}}$. We write $\mathrm{Deriv}_{T,\Phi}$ and $T, \Phi \vdash$ for $\mathrm{Deriv}_{\mathscr{F}}$ and $\mathscr{F} \vdash$, respectively, with $\mathscr{F}$ the family of inference rules associated with T and Φ.

4.3.5 Well-formedness conditions

4.3.21. Definition. Let T be a pretheory, Φ an environment over $T|_{\mathsf{expr}}$ and Γ a context over $T|_{\mathsf{expr}}$ and $\Phi|_{\mathsf{term}}$. We say Γ is *well-formed*, written $T, \Phi \vdash \Gamma\ \mathsf{ok}$, if $T, \Phi \vdash \Gamma \to e\ \mathsf{ok}$ for any $(x : e) \in \Gamma$.

4.3.22. Definition. Let T be a pretheory and Φ an environment over $T|_{\mathsf{expr}}$. We say Φ is *well-formed*, written $T \vdash \Phi\ \mathsf{ok}$, if $T, \Phi \vdash \Gamma\ \mathsf{ok}$ and $T, \Phi \vdash \Gamma \to e\ \mathsf{ok}$ for any $(x : \Gamma \to e) \in \Phi$.

4.3.23. Definition. We say a pretheory T is *well-formed* if the following conditions are satisfied:

1. $T \vdash \Phi\ \mathsf{ok}$ for any sort symbol $(S : \Phi \Rightarrow c) \in T$;
2. $T \vdash \Phi\ \mathsf{ok}$ and $T, \Phi \vdash () \to e\ \mathsf{ok}$ for any term symbol or axiom $(x : \Phi \Rightarrow K) \in T$.

4.4 Properties of derivations

Before defining SOGATs, we study basic properties of the derivations associated with a pretheory and an environment. We show the stability under substitutions in Section 4.4.1 and the stability under instantiations in Section 4.4.2. We show in Section 4.4.3 one of the most fundamental properties, *contextual completeness* which might be more familiar under the name *functional completeness* or *deduction theorem*.

4.4.1 Stability under substitutions

We show that the substitution rules are admissible. Following Bauer, Haselwarter, and Lumsdaine [20], we prove stronger statements (Propositions 4.4.5 and 4.4.7). Let T be a pretheory and Φ an environment over $T|_{\mathsf{term}}$. We first observe the stability under the action of morphisms of contexts.

4.4.1. Proposition. *The inference rules are stable under the action of morphisms of contexts: for any morphism of contexts $r : \Gamma \to \Gamma'$ and any inference rule of the form*

$$\mathrm{R}\ \frac{\Gamma \to \mathcal{J}}{\Gamma \to \mathcal{H}}$$

where $\mathcal{J}$ is a pseudo-judgment over Γ, we have an inference rule

$$r \cdot \mathrm{R}\ \frac{\Gamma' \to r \cdot \mathcal{J}}{\Gamma' \to r \cdot \mathcal{H}}.$$

Proof:
Immediate from the definition of inference rules. □

4.4.2. Proposition. *Any morphism of contexts $r : \Gamma \to \Gamma'$ acts on derivations as*

$$(r \cdot -) : \mathrm{Deriv}_{T,\Phi}(0, \Gamma \to \mathcal{H}) \to \mathrm{Deriv}_{T,\Phi}(0, \Gamma' \to r \cdot \mathcal{H}).$$

Proof:
We construct by induction on a derivation $D \in \mathrm{Deriv}_{T,\Phi}(0, \Gamma \to \mathcal{H})$ a family of derivations $\prod_{r:\Gamma\to\Gamma'} \mathrm{Deriv}_{T,\Phi}(0, \Gamma' \to r \cdot \mathcal{H})$. Since the premises of D is empty, D must be the application of a rule

$$\mathrm{R}\ \frac{\Gamma \to \mathcal{J}}{\Gamma \to \mathcal{H}}.$$

By Proposition 4.4.1, the rule R is mapped to a rule

$$r \cdot \mathrm{R}\ \frac{\Gamma' \to r \cdot \mathcal{J}}{\Gamma' \to r \cdot \mathcal{H}}.$$

Each premise of R is of the form $\Gamma \,.\, \Delta_\xi \to \mathcal{H}_\xi$, and the corresponding premise of $r \cdot \mathrm{R}$ is $\Gamma' \,.\, (r \cdot \Delta_\xi) \to r \cdot \mathcal{H}_\xi$. By the induction hypothesis with the morphism $r + \mathrm{id} : \Gamma \,.\, \Delta_\xi \to \Gamma' \,.\, (r \cdot \Delta_\xi)$, any derivation D_ξ of $\Gamma \,.\, \Delta_\xi \to \mathcal{H}_\xi$ is transformed into a derivation $r \cdot D_\xi$ of $\Gamma' \,.\, (r \cdot \Delta_\xi) \to r \cdot \mathcal{H}_\xi$. Applying the rule $r \cdot \mathrm{R}$ we have a derivation $r \cdot D$ of $\Gamma' \to r \cdot \mathcal{H}$. □

4.4.3. Corollary (Weakening). *If $T, \Phi \vdash \Gamma_1 . \Gamma_2 \to \mathcal{H}$, then $T, \Phi \vdash \Gamma_1 . \Delta . \Gamma_2 \to \mathcal{H}$ for any relative context Δ over Γ_1.*

Proof:
Apply Proposition 4.4.2 for the inclusion $\Gamma_1 \,.\, \Gamma_2 \to \Gamma_1 \,.\, \Delta \,.\, \Gamma_2$. □

We now prove the stability under substitution: if $T, \Phi \vdash \Gamma \to \mathcal{H}$ and $T, \Phi \vdash \Gamma' \to f : \Gamma$ for a substitution f, then $T, \Phi \vdash \Gamma' \to \mathcal{H} \cdot f$. Following Bauer, Haselwarter, and Lumsdaine [20], we strengthen the statement for technical reasons. A problem is that the assumption $T, \Phi \vdash \Gamma' \to f : \Gamma$ is too strong because Γ' is not necessarily well-formed. Suppose that we are trying to prove the substitution property by induction on the derivation and that a rule contains a premise of the form $\Gamma \,.\, \Delta \to \mathcal{H}'$. We would like to use the induction hypothesis for the substitution $f + \mathrm{id} : \Gamma' \,.\, (\Delta \cdot f) \to \Gamma \,.\, \Delta$. Unfortunately, we do not have a derivation of $\Gamma' \,.\, (\Delta \cdot f) \to f + \mathrm{id} : \Gamma \,.\, \Delta$. We would have to for example derive $\Gamma' \,.\, (\Delta \cdot f) \to \mathtt{x} : A \cdot f$ for any variable $(\mathtt{x} : A) \in \Delta$, but there is no obvious way to do that. However, this type checking looks trivial because $(\mathtt{x} : A \cdot f) \in \Delta \cdot f$,

and thus skipping it would not cause any problem. We thus prove the stability under substitution in the form that part of the type checking in $\Gamma' \to f : \Gamma$ can be skipped.

4.4.4. Definition. Let Γ' and Γ be contexts and $\Gamma_0 \subset \Gamma$ a decidable subset of variables and hypotheses of Γ. A *Γ_0-trivial substitution f of Γ in Γ'* consists of the following data:

- a morphism $f : \Gamma_0 \to \Gamma'$ of A_{ctx}-signatures;
- a substitution $f : \Gamma'|_{\mathsf{term}} \to \Gamma|_{\mathsf{term}} \setminus \Gamma_0$,

which naturally give rise to a substitution $f : \Gamma'|_{\mathsf{term}} \to \Gamma|_{\mathsf{term}}$, satisfying that $(f(x) : e \cdot f) \in \Gamma'$ for any variable or hypothesis $(x : e) \in \Gamma_0$. For a Γ_0-trivial substitution f of Γ in Γ', we write $\Gamma' \to f : \Gamma \setminus \Gamma_0$ for the following pseudo-judgment:

- $\Gamma' \to f(\mathtt{x}) : A \cdot f$ for any variable $(\mathtt{x} : A) \in \Gamma \setminus \Gamma_0$;
- $\Gamma' \to p \cdot f$ for any hypothesis $(H : p) \in \Gamma \setminus \Gamma_0$.

We will sometimes use the notation $\Gamma' \to f : \Gamma \setminus \Gamma_0$ for a not necessarily Γ_0-trivial substitution f.

4.4.5. Proposition (Substitution). *Let f be a Γ_0-trivial substitution of Γ in Γ'. For any derivations of $\Gamma' \to f : \Gamma \setminus \Gamma_0$ and $\Gamma \to \mathcal{H}$, we have a derivation of $\Gamma' \to \mathcal{H} \cdot f$.*

Proof:
We construct by induction on a derivation $D \in \mathrm{Deriv}_{T,\Phi}(0, \Gamma \to \mathcal{H})$ a map

$$\forall f. \mathrm{Deriv}_{T,\Phi}(0, \Gamma' \to f : \Gamma \setminus \Gamma_0) \to \mathrm{Deriv}_{T,\Phi}(0, \Gamma' \to \mathcal{H} \cdot f).$$

Since the premises of D is empty, D must be the application of a rule R $\dfrac{\Gamma \to \mathcal{J}}{\Gamma \to \mathcal{H}}$. Each premise of R is of the form $\Gamma \,.\, \Delta_\xi \to \mathcal{H}_\xi$. Extending f by the identity on Δ_ξ, we obtain a $\Gamma_0 + \Delta_\xi$-trivial substitution $f + \mathrm{id}$ of $\Gamma \,.\, \Delta_\xi$ in $\Gamma \,.\, (\Delta_\xi \cdot f)$. The pseudo-judgment $\Gamma' \,.\, (\Delta_\xi \cdot f) \to f + \mathrm{id} : \Gamma \,.\, \Delta_\xi \setminus \Gamma_0 + \Delta_\xi$ is then obtained from $\Gamma' \to f : \Gamma \setminus \Gamma_0$ by the weakening by $\Delta_\xi \cdot f$. Thus, by Corollary 4.4.3, any derivation of $\Gamma' \to f : \Gamma \setminus \Gamma_0$ is transformed into a derivation of $\Gamma' \,.\, (\Delta_\xi \cdot f) \to f + \mathrm{id} : \Gamma \,.\, \Delta_\xi \setminus \Gamma_0 + \Delta_\xi$. Applying the induction hypotheses to $(f + \mathrm{id})$'s, we can transform a derivation of $\Gamma \to \mathcal{J}$ to a derivation of $\Gamma' \to \mathcal{J} \cdot f$. Hence, all we have to do is to construct a derivation

$$\frac{\Gamma' \to \mathcal{J} \cdot f}{\Gamma' \to \mathcal{H} \cdot f}$$

for each rule R $\dfrac{\Gamma \to \mathcal{J}}{\Gamma \to \mathcal{H}}$. The cases other than Var and Hyp are immediate by checking that the rules are stable under substitutions.

Variables For a variable $(\mathtt{x} : A) \in \Gamma$, we construct a derivation

$$\frac{\Gamma' \to A \cdot f\ \mathsf{ok}}{\Gamma' \to f(\mathtt{x}) : A \cdot f}.$$

There are two cases: $(\mathtt{x} : A) \in \Gamma_0$; or $(\mathtt{x} : A) \in \Gamma \setminus \Gamma_0$. In the former case, by the Γ_0-triviality of f, the value $f(\mathtt{x})$ is a variable and $(f(\mathtt{x}) : A \cdot f) \in \Gamma'$. Then the rule $\mathrm{Var}(f(\mathtt{x}))$ gives us a derivation from $\Gamma' \to A \cdot f\ \mathsf{ok}$ to $\Gamma' \to f(\mathtt{x}) : A \cdot f$. In the latter case, $\Gamma' \to f(\mathtt{x}) : A \cdot f$ is a member of the family of judgments $\Gamma' \to f : \Gamma \setminus \Gamma_0$ which has a derivation by assumption.

Hypotheses For a hypothesis $(H : p) \in \Gamma$, we construct a derivation

$$\frac{\Gamma' \to p \cdot f\ \mathsf{ok}}{\Gamma' \to p \cdot f}.$$

This is constructed by case analysis on $(H : p) \in \Gamma_0$ in the same way as the case of Var. □

We show that the results of judgmentally equal substitutions are judgmentally equal.

4.4.6. Definition. Let Γ' and Γ be contexts and $\Gamma_0 \subset \Gamma$ a decidable subset. *Jointly Γ_0-trivial substitutions (f_1, f_2) of Γ in Γ'* consists of the following data:

- a morphism $f : \Gamma_0 \to \Gamma'$ of A_{ctx}-signatures;
- two substitutions $f_1, f_2 : \Gamma'|_{\mathsf{term}} \to \Gamma|_{\mathsf{term}} \setminus \Gamma_0$,

which naturally give rise to two substitutions $f_1, f_2 : \Gamma'|_{\mathsf{term}} \to \Gamma|_{\mathsf{term}}$, satisfying that for any variable or hypothesis $(x : e) \in \Gamma_0$, either $(f(x) : e \cdot f_1) \in \Gamma'$ or $(f(x) : e \cdot f_2) \in \Gamma'$. For jointly Γ_0-trivial substitutions (f_1, f_2) of Γ in Γ', we write $\Gamma' \to f_1 \equiv f_2 : \Gamma \setminus \Gamma_0$ for the pseudo-judgment consisting of $\Gamma' \to f_1(\mathtt{x}) \equiv f_2(\mathtt{x}) : A \cdot f_1$ for any variable $(\mathtt{x} : A) \in \Gamma \setminus \Gamma_0$.

4.4.7. Proposition (Equality substitution). *Let (f_1, f_2) be jointly Γ_0-trivial substitutions of Γ in Γ'. Given derivations of $\Gamma' \to f_1 : \Gamma \setminus \Gamma_0$, $\Gamma' \to f_2 : \Gamma \setminus \Gamma_0$, $\Gamma' \to f_1 \equiv f_2 : \Gamma \setminus \Gamma_0$ and $\Gamma \to \mathcal{H}$, we have derivations of $\Gamma' \to \mathcal{H} \cdot f_1$ and $\Gamma' \to \mathcal{H} \cdot f_2$ and, when $\mathcal{H} = (a : K)$, derivations of $\Gamma' \to a \cdot f_1 \equiv a \cdot f_2 : K \cdot f_1$ and $\Gamma' \to a \cdot f_2 \equiv a \cdot f_1 : K \cdot f_2$.*

Proof:
By induction on the derivation of $\Gamma \to \mathcal{H}$. Suppose that the derivation ends with a rule

$$\mathrm{R}\ \frac{\{\Gamma\,.\,\Delta_\xi \to \mathcal{H}_\xi\}_{\xi \in \Xi}}{\Gamma \to \mathcal{H}}.$$

The pair $(f_1 + \mathrm{id}, f_2 + \mathrm{id})$ determines jointly $(\Gamma_0 + \Delta_\xi)$-trivial substitutions of $\Gamma \,.\, \Delta_\xi$ in $\Gamma' \,.\, (\Delta_\xi \cdot f_i)$ for both $i = 1, 2$. By weakening, we derive $\Gamma' \,.\, (\Delta_\xi \cdot f_i) \to f_1 + \mathrm{id} : \Gamma \,.\, \Delta_\xi \setminus \Gamma_0 + \Delta_\xi$, $\Gamma' \,.\, (\Delta_\xi \cdot f_i) \to f_2 + \mathrm{id} : \Gamma \,.\, \Delta_\xi \setminus \Gamma_0 + \Delta_\xi$, and $\Gamma' \,.\, (\Delta_\xi \cdot f_i) \to f_1 + \mathrm{id} \equiv f_2 + \mathrm{id} : \Gamma \,.\, \Delta_\xi \setminus \Gamma_0 + \Delta_\xi$. Then, by the induction hypotheses, we derive $\Gamma' \,.\, (\Delta_\xi \cdot f_i) \to \mathcal{H}_\xi \cdot f_1$ and $\Gamma' \,.\, (\Delta_\xi \cdot f_i) \to \mathcal{H}_\xi \cdot f_2$ and, when $\mathcal{H}_\xi = (a_\xi : K_\xi)$, derive $\Gamma' \,.\, (\Delta_\xi \cdot f_i) \to a_\xi \cdot f_1 \equiv a_\xi \cdot f_2 : K_\xi \cdot f_1$ and $\Gamma' \,.\, (\Delta_\xi \cdot f_i) \to a_\xi \cdot f_2 \equiv a_\xi \cdot f_1 : K_\xi \cdot f_2$. The goal is to derive $\Gamma' \to \mathcal{H} \cdot f_1$ and $\Gamma' \to \mathcal{H} \cdot f_2$ and, when $\mathcal{H} = (a : K)$, to derive $\Gamma' \to a \cdot f_1 \equiv a \cdot f_2 : K \cdot f_1$ and $\Gamma' \to a \cdot f_2 \equiv a \cdot f_1 : K \cdot f_2$. We proceed by case analysis on the rule R. The case when R is a rule other than Hyp such that the conclusion is a proposition judgment is straightforward by the stability under substitutions.

Variables Suppose that R is $\mathrm{Var}(\mathtt{x})$ for a variable $(\mathtt{x} : A) \in \Gamma$. We first recall that A must be of the form $S(I)$ for a representable type symbol $S : \Psi \Rightarrow \mathsf{type}$ and an instantiation I and that $\Gamma \to A\ \mathsf{ok}$ is an abbreviation of $\Gamma \to I : \Psi$. Then, by the induction hypotheses, we derive $\Gamma' \to I \cdot f_i : \Psi$, $\Gamma' \to I \cdot f_1 \equiv I \cdot f_2 : \Psi$, and $\Gamma' \to I \cdot f_2 \equiv I \cdot f_1 : \Psi$. Hence, the rule $\mathrm{TypeConv}(S, I_1, I_2, a)$ justifies the use of conversion of the form

$$\frac{\Gamma' \to a : A \cdot f_1}{\Gamma' \to a : A \cdot f_2} \qquad \frac{\Gamma' \to a : A \cdot f_2}{\Gamma' \to a : A \cdot f_1}.$$

There are two cases, $\mathtt{x} \in \Gamma_0$ or $\mathtt{x} \in \Gamma \setminus \Gamma_0$. In the latter case, $\Gamma' \to f_1(\mathtt{x}) : A \cdot f_1$, $\Gamma' \to f_2(\mathtt{x}) : A \cdot f_2$, and $\Gamma' \to f_1(\mathtt{x}) \equiv f_2(\mathtt{x}) : A \cdot f_1$ are members of $\Gamma' \to f_1 : \Gamma \setminus \Gamma_0$, $\Gamma' \to f_2 : \Gamma \setminus \Gamma_0$, and $\Gamma' \to f_1 \equiv f_2 : \Gamma \setminus \Gamma_0$, respectively. To derive $\Gamma' \to f_2(\mathtt{x}) \equiv f_1(\mathtt{x}) : A \cdot f_2$, we first derive $\Gamma' \to f_1(\mathtt{x}) \equiv f_2(\mathtt{x}) : A \cdot f_2$ by conversion and then apply the rule $\mathrm{Smtry}(A \cdot f_2, f_1(\mathtt{x}), f_2(\mathtt{x}))$. In the former case, there are two cases, $(f(\mathtt{x}) : A \cdot f_1) \in \Gamma'$ or $(f(\mathtt{x}) : A \cdot f_2) \in \Gamma'$. In both cases, we derive $\Gamma' \to f(\mathtt{x}) : A \cdot f_1$ and $\Gamma' \to f(\mathtt{x}) : A \cdot f_2$ by the rule $\mathrm{Var}(f(\mathtt{x}))$ and conversion. Then, by the rule $\mathrm{Refl}(A \cdot f_i, f(\mathtt{x}))$, we derive $\Gamma' \to f(\mathtt{x}) \equiv f(\mathtt{x}) : A \cdot f_i$.

Hypotheses The case when R is $\mathrm{Hyp}(H)$ is proved by case analysis on $H \in \Gamma_0$ in the same way as the case of Var.

Metavariables Suppose that R is $\mathrm{MVar}(\mathtt{X}, g)$ for a metavariable $(\mathtt{X} : \Delta \to K) \in \Phi$ and a substitution g. Since $\mathtt{X}(g) \cdot f_i = \mathtt{X}(g \circ f_i)$ and $(K \cdot g) \cdot f_i = K \cdot (g \circ f_i)$, we derive $\Gamma' \to \mathtt{X}(g) \cdot f_i : (K \cdot g) \cdot f_i$ by the rule $\mathrm{MVar}(\mathtt{X}, g \circ f_i)$ from the induction hypotheses. Similarly, we apply the rule $\equiv$-MVar to derive $\Gamma' \to \mathtt{X}(g) \cdot f_1 \equiv \mathtt{X}(g) \cdot f_2 : (K \cdot g) \cdot f_1$ and $\Gamma' \to \mathtt{X}(g) \cdot f_2 \equiv \mathtt{X}(g) \cdot f_1 : (K \cdot g) \cdot f_2$.

Term symbols The case when R is Sym is done by the rules Sym and $\equiv$-Sym in the same way as the case of MVar.

Sort symbols Suppose that R is $\mathrm{TypeConv}(S, I_1, I_2, a)$ for a (representable) type symbol $(S : \Psi \Rightarrow c)$, instantiations I_1, I_2, and a term expression a. Since $S(I) \cdot f = S(I \cdot f)$, we derive $\Gamma' \to a \cdot f_i : S(I_2) \cdot f_i$ by the rule $\mathrm{TypeConv}(S, I_1 \cdot f_i, I_2 \cdot f_i, a \cdot f_i)$ from the induction hypotheses. We apply the rule TypeConvEq to derive the equalities. □

4.4.2 Stability under instantiations

We show that derivations are stable under instantiations.

4.4.8. Proposition. *For any derivations of* $T, \Phi' \vdash \Gamma' \to I : \Phi$ *and of* $T, \Phi \vdash \Gamma \to \mathcal{H}$*, we have a derivation of* $T, \Phi' \vdash \Gamma' \,.\, (\Gamma \cdot I) \to \mathcal{H} \cdot I$*.*

Proof:
By induction on a derivation D of $\Gamma \to \mathcal{H}$. The cases other than MVar, ≡-MVar or Asm are straightforward by checking that the rules are stable under instantiations.

Metavariables Suppose that D ends with the rule $\mathrm{MVar}(\mathtt{X}, f)$ for a metavariable $(\mathtt{X} : \Delta \to K) \in \Phi$ and a substitution f. By the induction hypotheses, we derive $\Gamma' \,.\, (\Gamma \cdot I) \to f \cdot I : \Gamma' \,.\, (\Delta \cdot I) \setminus \Gamma'$. By assumption, we also derive $\Gamma' \,.\, (\Delta \cdot I) \to I(\mathtt{X}) : K \cdot I$. Then, by substitution, we derive $\Gamma' \,.\, (\Gamma \cdot I) \to \mathtt{X}(f) \cdot I : (K \cdot f) \cdot I$.

Congruence Suppose that D ends with the rule $\equiv\text{-}\mathrm{MVar}(\mathtt{X}, f_1, f_2)$ for a metavariable $(\mathtt{X} : \Delta \to K) \in \Phi$ and two substitutions f_1 and f_2. Again by the induction hypotheses and substitution, we derive $\Gamma' . (\Gamma \cdot I) \to \mathtt{X}(f_1) \cdot I \equiv \mathtt{X}(I_2) \cdot I : (K \cdot f_1) \cdot I$.

Assumptions The case of Asm is analogous to the case of MVar. □

To see the action of judgmentally equal instantiations, we introduce a notation. For two instantiations $\Sigma, \mu' \vdash I_1, I_2 : \gamma' \Rightarrow \mu$ and a context Γ over Σ and μ, we write $\Gamma \cdot I_?$ for an arbitrary mixture of $\Gamma \cdot I_1$ and $\Gamma \cdot I_2$. That is, $\Gamma \cdot I_?$ denotes a relative context over γ' whose underlying signature is the same as Γ and, for any variable or hypothesis $(x : e) \in \Gamma$, either $(x : e \cdot I_1) \in \Gamma \cdot I_?$ or $(x : e \cdot I_2) \in \Gamma \cdot I_?$.

4.4.9. Proposition. *Given derivations of* $T, \Phi' \vdash \Gamma' \to I_1 : \Phi$*,* $T, \Phi' \vdash \Gamma' \to I_2 : \Phi$*,* $T, \Phi' \vdash \Gamma' \to I_1 \equiv I_2 : \Phi$*, and* $T, \Phi \vdash \Gamma \to \mathcal{H}$ *and an arbitrary choice of* $\Gamma \cdot I_?$*, we have derivations of* $T, \Phi' \vdash \Gamma' \,.\, (\Gamma \cdot I_?) \to \mathcal{H} \cdot I_1$ *and* $T, \Phi' \vdash \Gamma' \,.\, (\Gamma \cdot I_?) \to \mathcal{H} \cdot I_2$ *and, when* $\mathcal{H} = (a : K)$*, derivations of* $T, \Phi' \vdash \Gamma' \,.\, (\Gamma \cdot I_?) \to a \cdot I_1 \equiv a \cdot I_2 : K \cdot I_1$ *and* $T, \Phi' \vdash \Gamma' \,.\, (\Gamma \cdot I_?) \to a \cdot I_2 \equiv a \cdot I_1 : K \cdot I_2$*.*

Proof:
By induction on derivation D of $\Gamma \to \mathcal{H}$. Suppose that D ends with a rule $\mathrm{R}\ \dfrac{\{\Gamma \,.\, \Delta_\xi \to \mathcal{H}_\xi\}_{\xi\in\Xi}}{\Gamma \to \mathcal{H}}$. By the induction hypotheses, we have derivations of $\Gamma' \,.\, (\Gamma \,.\, \Delta_\xi) \cdot I_? \to \mathcal{H}_\xi \cdot I_1$ and $\Gamma' \,.\, (\Gamma \,.\, \Delta_\xi) \cdot I_? \to \mathcal{H}_\xi \cdot I_2$ and, when $\mathcal{H}_\xi = (a_\xi : K_\xi)$, derivations of $\Gamma' \,.\, (\Gamma \,.\, \Delta_\xi) \cdot I_? \to a_\xi \cdot I_1 \equiv a_\xi \cdot I_2 : K_\xi \cdot I_1$ and $\Gamma' \,.\, (\Gamma \,.\, \Delta_\xi) \cdot I_? \to a_\xi \cdot I_2 \equiv a_\xi \cdot I_1 : K_\xi \cdot I_2$, for an arbitrary choice of $(\Gamma \,.\, \Delta_\xi) \cdot I_?$. In particular, we can choose $(x : e \cdot I_1) \in (\Gamma \,.\, \Delta_\xi) \cdot I_?$ for all $(x : e) \in \Delta_\xi$ or $(x : e \cdot I_2) \in (\Gamma \,.\, \Delta_\xi)$ for all $(x : e) \in \Delta_\xi$. Then the cases other than MVar, $\equiv$-MVar or Asm become straightforward: just apply R to derive $\Gamma' \,.\, (\Gamma \cdot I_?) \to \mathcal{H} \cdot I_i$; use congruence to derive the equalities.

Metavariables Suppose that R is $\mathrm{MVar}(\mathtt{X}, f)$ for a metavariable $(\mathtt{X} : \Delta \to K) \in \Phi$ and a substitution f. We first note that we can freely use conversion between $(K \cdot f) \cdot I_1$ and $(K \cdot f) \cdot I_2$ in the same way as the variable case of the proof of Proposition 4.4.7. Since $\mathtt{X}(f) \cdot I_i = I_i(\mathtt{X}) \cdot (f \cdot I_i)$ and $(K \cdot f) \cdot I_i = (K \cdot I_i) \cdot (f \cdot I_i)$, we derive $\Gamma' \,.\, (\Gamma \cdot I_?) \to \mathtt{X}(f) \cdot I_i : (K \cdot f) \cdot I_i$ from the induction hypotheses by the action of the substitution $f \cdot I_i$. For the equality $\Gamma' \,.\, (\Gamma \cdot I_?) \to \mathtt{X}(f) \cdot I_1 \equiv \mathtt{X}(f) \cdot I_2 : (K \cdot f) \cdot I_1$, we first derive $\Gamma' \,.\, (\Gamma \cdot I_?) \to I_1(\mathtt{X}) \cdot (f \cdot I_1) \equiv I_2(\mathtt{X}) \cdot (f \cdot I_1) : (K \cdot I_1) \cdot (f \cdot I_1)$ by assumption and substitution. By the induction hypotheses, we also derive $\Gamma' \,.\, (\Gamma \cdot I_?) \to f \cdot I_1 \equiv f \cdot I_2 : \Gamma' \,.\, (\Delta \cdot I_1) \setminus \Gamma'$. Then, by equality substitution, we derive $\Gamma' \,.\, (\Gamma \cdot I_?) \to I_2(\mathtt{X}) \cdot (f \cdot I_1) \equiv I_2(\mathtt{X}) \cdot (f \cdot I_2) : (K \cdot I_1) \cdot (f \cdot I_1)$. By transitivity, we derive $\Gamma' \,.\, (\Gamma \cdot I_?) \to I_1(\mathtt{X}) \cdot (f \cdot I_1) \equiv I_2(\mathtt{X}) \cdot (f \cdot I_2) : (K \cdot I_1) \cdot (f \cdot I_1)$, that is, $\Gamma' \,.\, (\Gamma \cdot I_?) \to \mathtt{X}(f) \cdot I_1 \equiv \mathtt{X}(f) \cdot I_2 : (K \cdot f) \cdot I_1$. For the other equality, use conversion and symmetry.

Congruence Suppose that R is $\equiv$-$\mathrm{MVar}(\mathtt{X}, f_1, f_2)$ for a metavariable $(\mathtt{X} : \Delta \to K) \in \Phi$ and substitutions f_1 and f_2. By the induction hypotheses, we derive $\Gamma' \,.\, (\Gamma \cdot I_?) \to f_1 \cdot I_i \equiv f_2 \cdot I_i : \Gamma' \,.\, (\Delta \cdot I_i) \setminus \Gamma'$. Then, by equality substitution, we derive $\Gamma' \,.\, (\Gamma \cdot I_?) \to I_i(\mathtt{X}) \cdot (f_1 \cdot I_i) \equiv I_i(\mathtt{X}) \cdot (f_2 \cdot I_i) : (K \cdot I_i) \cdot (f_1 \cdot I_i)$, that is, $\Gamma' \,.\, (\Gamma \cdot I_?) \to \mathtt{X}(f_1) \cdot I_i \equiv \mathtt{X}(f_2) \cdot I_i : (K \cdot f_1) \cdot I_i$.

Assumptions The case of Asm is analogous to the case of MVar. □

4.4.3 Contextual completeness

We show a form of *contextual completeness* in the sense of Hermida [77]. For a simply typed lambda calculus T, contextual completeness asserts that for any types A and B and any context Γ of T, we have a bijective correspondence between the set of terms $\Gamma, \mathtt{x} : A \vdash b : B$ defined in T and the set of terms $\Gamma \vdash b' : B$ defined in the extension of T by a constant $a : A$. This is a refinement

of *functional completeness* [106] which asserts that the terms $\Gamma \vdash b' : A$ defined in the extension of T by a constant $a : A$ bijectively correspond to the functions $\Gamma \vdash b : A \texttt{->} B$ defined in T. Functional completeness is essentially the same as the usual *deduction theorem*: proving a proposition B under an assumption A is equivalent to proving the implication $A \texttt{->} B$ without the assumption. In the presence of function types, contextual completeness and functional completeness are equivalent, but the former also makes sense in the absence of function types.

In the language of SOGATs, a metavariable $\mathtt{X} : () \to A$ plays the same role as a constant $a : A$. We thus formulate contextual completeness in that variables and metavariables with arity 0 are interchangeable and show that any SOGAT satisfies contextual completeness.

4.4.10. Definition. Let γ_0 be a variable signature. The *metavariable replacement* $\gamma_0^\dagger$ *of* γ_0 is the metavariable signature

$$\big\{\big(\mathtt{x}^\dagger : ()\big) \mid \mathtt{x} \in \gamma_0\big\}.$$

We have a canonical instantiation

$$(), () \vdash \Omega_{\gamma_0} : \gamma_0 \Rightarrow \gamma_0^\dagger$$

defined by $\Omega_{\gamma_0}\big(\mathtt{x}^\dagger\big) = \mathtt{x}$ and a canonical substitution

$$(), \gamma_0^\dagger \vdash \omega_{\gamma_0} : () \to \gamma_0$$

defined by $\omega_{\gamma_0}(\mathtt{x}) = \mathtt{x}^\dagger$.

4.4.11. Proposition. *1.* $(), () \vdash \omega_{\gamma_0} \cdot \Omega_{\gamma_0} : \gamma_0 \to \gamma_0$ *is the identity substitution.*

2. $(), \gamma_0^\dagger \vdash \Omega_{\gamma_0} \cdot \omega_{\gamma_0} : () \Rightarrow \gamma_0^\dagger$ *is the identity instantiation.*

Proof:
By definition. □

4.4.12. Definition. Let Σ be a symbol signature and μ a metavariable signature. For a context Γ_0 over Σ and μ, the *metavariable replacement* $\Gamma_0^\dagger$ *of* Γ_0 is the relative environment over μ

$$\big\{\big(x^\dagger : () \to e \cdot \omega_{\Gamma_0|_{\mathsf{term}}}\big) \mid (x : e) \in \Gamma\big\}.$$

Note that $\Gamma_0^\dagger|_{\mathsf{term}} = (\Gamma_0|_{\mathsf{term}})^\dagger$. We set $\Omega_{\Gamma_0} = \Omega_{\Gamma_0|_{\mathsf{term}}}$ and $\omega_{\Gamma_0} = \omega_{\Gamma_0|_{\mathsf{term}}}$.

4.4.13. THEOREM (Contextual completeness). *Let T be a pretheory, Φ an environment over $T|_{\mathsf{term}}$, and Γ_0 a context over $T|_{\mathsf{term}}$ and $\Phi|_{\mathsf{term}}$. The instantiation Ω_{Γ_0} and the substitution ω_{Γ_0} induce a bijective correspondence between the following sets of derivable judgments.*

$$\left\{T, \left(\Phi \,.\, \Gamma_0^\dagger\right) \vdash \Gamma_1 \to \mathcal{H}_1\right\} \cong \{T, \Phi \vdash \Gamma_0 \,.\, \Gamma_2 \to \mathcal{H}_2\}$$

Contextual completeness is proved by the following two lemmas.

4.4.14. LEMMA. *If $T, \left(\Phi \,.\, \Gamma_0^\dagger\right) \vdash \Gamma \to \mathcal{H}$, then $T, \Phi \vdash \Gamma_0 \,.\, (\Gamma \cdot \Omega_{\Gamma_0}) \to \mathcal{H} \cdot \Omega_{\Gamma_0}$.*

Proof:
By induction on derivation. It suffices to transform each rule

$$\mathrm{R}\ \frac{\Gamma \to \mathcal{J}}{\Gamma \to \mathcal{H}}$$

associated with T and $\Phi \,.\, \Gamma_0^\dagger$ into a derivation

$$\frac{\Gamma_0 \,.\, (\Gamma \cdot \Omega_{\Gamma_0}) \to \mathcal{J}}{\Gamma_0 \,.\, (\Gamma \cdot \Omega_{\Gamma_0}) \to \mathcal{H}}.$$

The cases other than MVar, $\equiv$-MVar, or Asm are straightforward.

Metavariable For a metavariable $(\mathtt{X} : \Delta \to K) \in \Phi \,.\, \Gamma_0^\dagger$ and a substitution $f : \Gamma \to \Delta$, we construct a derivation of the form

$$\frac{\Gamma_0 \,.\, (\Gamma \cdot \Omega_{\Gamma_0}) \to f \cdot \Omega_{\Gamma_0} : \Delta \qquad \Gamma_0 \,.\, (\Gamma \cdot \Omega_{\Gamma_0}) \to (K \cdot f) \cdot \Omega_{\Gamma_0}\ \mathsf{ok}}{\Gamma_0 \,.\, (\Gamma \cdot \Omega_{\Gamma_0}) \to \mathtt{X}(f) \cdot \Omega_{\Gamma_0} : (K \cdot f) \cdot \Omega_{\Gamma_0}}.$$

There are two cases: $\mathtt{X} \in \Phi$ or $\mathtt{X} \in \Gamma_0^\dagger$. In the former case, we just give the rule $\mathrm{MVar}(\mathtt{X}, f \cdot \Omega_{\Gamma_0})$. In the latter case, $(\mathtt{X} : \Delta \to K) = \left(\mathtt{x}^\dagger : 0 \to A \cdot \omega_{\Gamma_0}\right)$ for a variable $(\mathtt{x} : A) \in \Gamma_0$ and f is empty. Therefore, the rule $\mathrm{MVar}(\mathtt{X}, f)$ is of the form

$$\frac{\Gamma \to A \cdot \omega_{\Gamma_0}\ \mathsf{ok}}{\Gamma \to \mathtt{x}^\dagger : A \cdot \omega_{\Gamma_0}}$$

and then the goal is to give a derivation of the form

$$\frac{\Gamma_0 \,.\, (\Gamma \cdot \Omega_{\Gamma_0}) \to (A \cdot \omega_{\Gamma_0}) \cdot \Omega_{\Gamma_0}\ \mathsf{ok}}{\Gamma_0 \,.\, (\Gamma \cdot \Omega_{\Gamma_0}) \to \mathtt{x} : (A \cdot \omega_{\Gamma_0}) \cdot \Omega_{\Gamma_0}}.$$

Since $\omega_{\Gamma_0} \cdot \Omega_{\Gamma_0}$ is the identity substitution, the rule $\mathrm{Var}(\mathtt{x})$ applies.

Assumptions The case of Asm is analogous to the case of MVar.

Congruence Consider the case of $\equiv$-MVar$(\mathtt{X}, f_1, f_2)$ for a metavariable $(\mathtt{X} : \Delta \to K) \in \Phi \,.\, \Gamma_0^\dagger$ and substitutions $f_1, f_2 : \Gamma \to \Delta$. Again there are two cases and the case when $\mathtt{X} \in \Phi$ is straightforward. In the case when $\mathtt{X} \in \Gamma_0^\dagger$, we have $(\mathtt{X} : \Delta \to K) = \left(\mathtt{x}^\dagger : 0 \to A \cdot \omega_{\Gamma_0}\right)$ for a variable $(\mathtt{x} : A) \in \Gamma_0$ and f_1 and f_2 are empty. Then the rule MVar$(\mathtt{X}, f_1, f_2)$ is of the form

$$\frac{\Gamma \to \left(\mathtt{x}^\dagger \equiv \mathtt{x}^\dagger : A \cdot \omega_{\Gamma_0}\right)\ \mathsf{ok}}{\Gamma \to \mathtt{x}^\dagger \equiv \mathtt{x}^\dagger : A \cdot \omega_{\Gamma_0}}$$

and the goal is to give a derivation of the form

$$\frac{\Gamma_0 \,.\, (\Gamma \cdot \Omega_{\Gamma_0}) \to (\mathtt{x} \equiv \mathtt{x} : (A \cdot \omega_{\Gamma_0}) \cdot \Omega_{\Gamma_0})\ \mathsf{ok}}{\Gamma_0 \,.\, (\Gamma \cdot \Omega_{\Gamma_0}) \to \mathtt{x} \equiv \mathtt{x} : (A \cdot \omega_{\Gamma_0}) \cdot \Omega_{\Gamma_0}}.$$

This time the rule Refl$(A, \mathtt{x})$ applies. □

4.4.15. Lemma. *If* $T, \Phi \vdash \Gamma_0 \,.\, \Gamma \to \mathcal{H}$*, then* $T, \left(\Phi \,.\, \Gamma_0^\dagger\right) \vdash \Gamma \cdot \omega_{\Gamma_0} \to \mathcal{H} \cdot \omega_{\Gamma_0}$.

Proof:
By induction on derivation. It suffices to transform each rule of the form

$$\mathrm{R}\ \frac{\Gamma_0 \,.\, \Gamma \to \mathcal{J}}{\Gamma_0 \,.\, \Gamma \to \mathcal{H}}$$

associated with T and Φ into a derivation

$$\frac{\Gamma \cdot \omega_{\Gamma_0} \to \mathcal{J} \cdot \omega_{\Gamma_0}}{\Gamma \cdot \omega_{\Gamma_0} \to \mathcal{H} \cdot \omega_{\Gamma_0}}.$$

The cases other than Var or Hyp are straightforward.

Variables For a variable $(\mathtt{x} : A) \in \Gamma_0 \,.\, \Gamma$, we construct a derivation of the form

$$\frac{\Gamma \cdot \omega_{\Gamma_0} \to A \cdot \omega_{\Gamma_0}\ \mathsf{ok}}{\Gamma \cdot \omega_{\Gamma_0} \to \mathtt{x} \cdot \omega_{\Gamma_0} : A \cdot \omega_{\Gamma_0}}$$

There are two cases: $\mathtt{x} \in \Gamma_0$; or $\mathtt{x} \in \Gamma$. In the latter case, we just give the rule Var$(\mathtt{x})$. In the former case, the goal becomes a derivation of the form

$$\frac{\Gamma \cdot \omega_{\Gamma_0} \to A \cdot \omega_{\Gamma_0}\ \mathsf{ok}}{\Gamma \cdot \omega_{\Gamma_0} \to \mathtt{x}^\dagger : A \cdot \omega_{\Gamma_0}}$$

and thus the rule MVar$(\mathtt{x}^\dagger, 0)$ applies.

Hypotheses The case of Hyp is analogous to the case of Var. □

4.5 Second-order generalized algebraic theories

We will define a SOGAT to be a pretheory that is *well-ordered* and *finitary*. Both requirements are reasonable in the sense that most practical type theories have these properties. From a theoretical point of view, these properties ensure that any SOGAT can be decomposed into small pieces, making the analysis of SOGATs much easier.

4.5.1 Well-ordered presentation

4.5.1. Definition. Let T be a pretheory and Φ an environment over $T|_{\mathsf{expr}}$. We say a context Γ over $T|_{\mathsf{expr}}$ and $\Phi|_{\mathsf{term}}$ is *well-ordered over T and Φ* if there exists a well-ordering $<$ on variables and hypotheses satisfying the following conditions.

1. For any variable or hypothesis $(x : e) \in \Gamma$, the expression e is over $\Gamma_{<x}|_{\mathsf{term}}$, that is, e belongs to the subset $\mathrm{Expr}(\Gamma_{<x}|_{\mathsf{term}}, c) \subset \mathrm{Expr}(\Gamma|_{\mathsf{term}}, c)$. In particular, $\Gamma_{<x} \subset \Gamma$ is a subcontext.

2. For any variable or hypothesis $(x : e) \in \Gamma$, we have $T, \Phi \vdash \Gamma_{<x} \to e\ \mathsf{ok}$.

By definition and weakening, any well-ordered context is well-formed.

4.5.2. Definition. Let T be a pretheory. We say an environment Φ over $T|_{\mathsf{expr}}$ is *well-ordered over T* if there exists a well-ordering $<$ on metavariables and assumptions satisfying the following conditions.

1. For any metavariable or assumption $(x : \Gamma \to e) \in \Phi$, the context Γ and the expression e are over $\Phi_{<x}|_{\mathsf{term}}$. In particular, $\Phi_{<x} \subset \Phi$ is a subenvironment.

2. For any metavariable or assumption $(x : \Gamma \to e) \in \Phi$, the context Γ is well-ordered over T and $\Phi_{<x}$, and we have $T, \Phi_{<x} \vdash \Gamma \to e\ \mathsf{ok}$.

By definition and weakening, any well-ordered environment is well-formed.

4.5.3. Definition. We say a pretheory T is *well-ordered* if there exists a well-ordering $<$ on symbols and axioms satisfying the following conditions.

1. For any sort symbol $(S : \Phi \Rightarrow c) \in T$, the environment Φ is over $T_{<S}|_{\mathsf{expr}}$.

2. For any term symbol or axiom $(x : \Phi \Rightarrow e) \in T$, the environment Φ and the expression e are over $T_{<x}|_{\mathsf{expr}}$. In particular, $T_{<x} \subset T$ is a subpretheory for any symbol or axiom $x \in T$.

3. For any sort symbol $(S : \Phi \Rightarrow c) \in T$, the environment Φ is well-ordered over $T_{<S}$.

4. For any term symbol or axiom $(x : \Phi \Rightarrow e) \in T$, the environment Φ is well-ordered over $T_{<S}$, and we have $T_{<S}, \Phi \vdash () \to e\ \mathsf{ok}$.

By definition and weakening, any well-ordered pretheory is well-formed.

It is natural to ask if we can make any well-formed pretheory well-ordered. Unfortunately, it is not the case.

4.5.4. Counterexample. Let T be the following pretheory.

$$\begin{aligned}
A &: () \Rightarrow \mathsf{type} \\
a &: () \Rightarrow A \\
B &: (\mathtt{x} : A) \Rightarrow \mathsf{type} \\
b &: () \Rightarrow B(a) \\
b' &: () \Rightarrow B(a) \\
p &: (\mathtt{y} : B(a)) \Rightarrow \mathsf{prop} \\
H_1 &: (_ : (_ : p(b)) \to p(b'), \mathtt{X} : () \to A) \Rightarrow \mathtt{X} \equiv a : A \\
H_2 &: (\mathtt{X} : () \to A, \mathtt{Y} : () \to B(\mathtt{X}), _ : () \to p(\mathtt{Y})) \Rightarrow p(b')
\end{aligned}$$

Clearly A to H_1 are well-ordered. The axiom H_2 is not well-formed over A to H_1 because p is applied to $\mathtt{Y} : B(\mathtt{X})$ while p expects a term of $B(a)$. However, H_2 is well-formed over T as follows. The idea is that there can be a well-formed instantiation of the environment of H_2 even when the well-formedness of the environment is not known, so we can use the rule $\mathrm{Axiom}(H_2, I)$ before confirming the well-formedness of H_2. Let $\Phi = (\mathtt{X} : () \to A, \mathtt{Y} : () \to B(\mathtt{X}), _ : () \to p(\mathtt{Y}))$. It remains to derive $T, \Phi \vdash () \to p(\mathtt{Y})\ \mathsf{ok}$, that is, $T, \Phi \vdash () \to \mathtt{Y} : B(a)$. First, $I = (\mathtt{X} := a, \mathtt{Y} := b)$ is a well-formed instantiation of $(\mathtt{X} : () \to A, \mathtt{Y} : () \to B(\mathtt{X}), _ : () \to p(\mathtt{Y}))$ over the context $(_ : p(b))$. Then, by the rule $\mathrm{Axiom}(H_2, I)$ we derive $T, \Phi \vdash (_ : p(b)) \to p(b')$. By the rule $\mathrm{Axiom}(H_1, (\mathtt{X} := \mathtt{X}))$ we derive $T, \Phi \vdash () \to \mathtt{X} \equiv a : A$. Thus, by conversion, we derive $T, \Phi \vdash () \to \mathtt{Y} : B(a)$.

4.5.5. Remark. We expect that any well-formed pretheory is nevertheless "equivalent" to a well-ordered one. For example, we obtain a well-ordered pretheory T' from the pretheory T in the above counterexample by inserting a new axiom

$$H_{1.5} : (_ : () \to p(b)) \Rightarrow p(b')$$

between H_1 and H_2. Indeed, $H_{1.5}$ is well-formed over A to H_1, and we can derive $(\mathtt{x} : A) \to \mathtt{x} \equiv a : A$ from H_1 and $H_{1.5}$. Then H_2 becomes well-formed over A to $H_{1.5}$. Since $(_ : p(b)) \to p(b')$ was already derivable over T, the pretheories T and T' are equivalent in the sense that they derive the same judgments. In general, the same method as the *well-founded replacement* of Bauer, Haselwarter, and Lumsdaine [20, Section 6.5] would apply, but we leave details as future work.

4.5.2 Finitary pretheories

4.5.6. Definition. We say a context is *finite* if its set of variables and hypotheses is finite.

4.5.7. Definition. We say an environment Φ is *finitary* if for any metavariable or assumption $(x : \Gamma \to e) \in \Phi$, the context Γ is finite. We say Φ is *finite* if it is finitary and the set of metavariables and assumptions is finite.

4.5.8. Definition. We say a pretheory T is *finitary* if for any symbol or axiom $(x : \Phi \Rightarrow _) \in T$, the environment Φ is finite.

4.5.9. Proposition. *Let T be a finitary pretheory and Φ a finitary environment over $T|_{\mathsf{expr}}$. Then, any derivation over T and Φ is finite seen as a tree.*

Proof:
By the inductive definition of derivations, no derivation tree contains an infinite branch. When T and Φ, the set of premises of each rule is finite, and thus any derivation tree is finitely branching. Then, by König's Lemma, any derivation tree is finite. □

4.5.10. Definition. A *second-order generalized algebraic theory (SOGAT)* is a well-ordered finitary pretheory.

4.6 Examples of SOGATs

We give a bunch of examples of SOGATs. The fundamental example is the dependent type theory without any type constructor.

4.6.1. Example. We define the *basic dependent type theory (DTT)* to be the following SOGAT.

$$U : () \Rightarrow \mathsf{Type}$$
$$E : (\mathtt{A} : () \to U) \Rightarrow \mathsf{type}$$

We will omit the application of E and consider a term $A : U$ as a representable type. This would not be confusing unless we add another way to convert a term of U to a representable type. When working with DTT, we will call a term of U a *type*.

4.6.2. Notation. When defining a SOGAT, we will make some arguments of a symbol or metavariable implicit by marking with $[-]$. For example, if we introduce a symbol like

$$S : ([\mathtt{A} : () \to U], \mathtt{a} : () \to \mathtt{A}) \Rightarrow U,$$

then the argument $\mathtt{A}$ is treated as implicit, that is, we simply write $S(\mathtt{a})$ instead of $S(\mathtt{A}, \mathtt{a})$ for terms $\mathtt{A} : U$ and $\mathtt{a} : \mathtt{A}$. This would not be confusing because the argument $\mathtt{A}$ is inferred from the type of $\mathtt{a}$.

Sometimes we will *define* a term over a SOGAT. When we say that we define a term

$$S : (x_1 : \Gamma_1 \to e_1, \ldots, x_n : \Gamma_n \to e_n) \Rightarrow K$$

followed by a definition

$$S(x_1, \ldots, x_n) \equiv e,$$

we mean that we add the following symbol and axiom.

$$\begin{aligned} S &: (x_1 : \Gamma_1 \to e_1, \ldots, x_n : \Gamma_n \to e_n) \Rightarrow K \\ _ &: (x_1 : \Gamma_1 \to e_1, \ldots, x_n : \Gamma_n \to e_n) \Rightarrow S(x_1, \ldots, x_n) \equiv e : K \end{aligned}$$

In contrast, when we say that we *declare* a symbol

$$S : \Phi \Rightarrow K,$$

we mean that we just add that symbol.

We mainly explain examples of type theories used in later chapters (Chapters 7 and 8). In Section 4.6.1, we present fragments of Martin-Löf type theory as SOGATs. In Section 4.6.2, we explain extensions of Martin-Löf type theory including the univalence axiom and higher inductive types. Section 4.6.3 is devoted to cubical type theory which is the subject of Chapters 7 and 8. In Sections 4.6.4 to 4.6.6, we discuss more general sources of examples of SOGATs. In Section 4.6.4, we show that SOGATs are a conservative extension of second-order algebraic theories. We compare SOGATs to generalized algebraic theories in Section 4.6.5. This time we do not claim that SOGATs are a conservative extension of generalized algebraic theories because of some technical details. In Section 4.6.6, we compare SOGATs to Bauer et al.'s type theories.

4.6.1 Martin-Löf type theory

Fragments of *Martin-Löf type theory* [124, 125, 133, 132] are defined as extensions of DTT. Components of Martin-Löf type theory such as Π-types, identity types and inductive types follow a common pattern. Each component has four kinds of rules:

- type formation rule;
- introduction rule;

- elimination rule;
- equality or computation rule.

When defining a type theory as a SOGAT, the first three rules are represented by symbols and the last one is represented by an axiom.

4.6.3. Example. We can extend DTT by *dependent product types (Π-types)* represented by the following symbols and axioms.

$$
\begin{aligned}
&\Pi : (\mathtt{A} : () \to U, \mathtt{B} : (\mathtt{x} : \mathtt{A}) \to U) \Rightarrow U \\
&\lambda : ([\mathtt{A} : () \to U], [\mathtt{B} : (\mathtt{x} : \mathtt{A}) \to U], \mathtt{b} : (\mathtt{x} : \mathtt{A}) \to \mathtt{B}(\mathtt{x})) \Rightarrow \Pi(\mathtt{A}, \mathtt{B}) \\
&@ : ([\mathtt{A} : () \to U], [\mathtt{B} : (\mathtt{x} : \mathtt{A}) \to U], \mathtt{f} : () \to \Pi(\mathtt{A}, \mathtt{B}), \mathtt{a} : () \to \mathtt{A}) \Rightarrow \mathtt{B}(\mathtt{a}) \\
&_ : (\mathtt{A} : () \to U, \mathtt{B} : (\mathtt{x} : \mathtt{A}) \to U, \mathtt{b} : (\mathtt{x} : \mathtt{A}) \to \mathtt{B}(\mathtt{x}), \mathtt{a} : () \to \mathtt{A}) \\
&\qquad \Rightarrow @(\lambda(\mathtt{b}), \mathtt{a}) \equiv \mathtt{b}(\mathtt{a}) : \mathtt{B}(\mathtt{a}) \\
&_ : (\mathtt{A} : () \to U, \mathtt{B} : (\mathtt{x} : \mathtt{A}) \to U, \mathtt{f} : () \to \Pi(\mathtt{A}, \mathtt{B})) \Rightarrow \lambda(\langle \mathtt{x} \rangle @(\mathtt{f}, \mathtt{x})) \equiv \mathtt{f} : \Pi(\mathtt{A}, \mathtt{B})
\end{aligned}
$$

The symbols Π, λ, and @ represent the formation, introduction, and elimination rules, respectively, for Π-types. The two axioms represent the β-equality rule and the η-equality rule. We introduce the following notations.

$$
\begin{aligned}
\prod_{\mathtt{x}:\mathtt{A}} \mathtt{B} &= \Pi(\mathtt{A}, \langle \mathtt{x} \rangle \mathtt{B}) \\
(\mathtt{x}_1 : \mathtt{A}_1, \ldots, \mathtt{x}_n : \mathtt{A}_n) \texttt{ -> } \mathtt{B} &= \prod_{\mathtt{x}_1:\mathtt{A}_1} \cdots \prod_{\mathtt{x}_n:\mathtt{A}_n} \mathtt{B} \\
\mathtt{A} \texttt{ -> } \mathtt{B} &= (\mathtt{x} : \mathtt{A}) \texttt{ -> } \mathtt{B} \qquad (\mathtt{x} \text{ does not freely occur in } \mathtt{B}) \\
\lambda \mathtt{x}.\mathtt{b} &= \lambda(\langle \mathtt{x} \rangle \mathtt{b}) \\
\lambda \mathtt{x}_1 \ldots \mathtt{x}_n.\mathtt{b} &= \lambda \mathtt{x}_1. \ldots . \lambda \mathtt{x}_n.\mathtt{b} \\
\mathtt{f}(\mathtt{a}) &= @(\mathtt{f}, \mathtt{a}) \\
\mathtt{f}(\mathtt{a}_1, \ldots, \mathtt{a}_n) &= \mathtt{f}(\mathtt{a}_1) \ldots (\mathtt{a}_n)
\end{aligned}
$$

4.6.4. Example. *Dependent sum types (Σ-types)* are declared in two ways, as *positive types* or as *negative types*. In both cases, we add the same type formation rule and introduction rule.

$$
\begin{aligned}
&\Sigma : (\mathtt{A} : () \to U, \mathtt{B} : (\mathtt{x} : \mathtt{A}) \to U) \Rightarrow U \\
&\mathsf{pair} : ([\mathtt{A} : () \to U], [\mathtt{B} : (\mathtt{x} : \mathtt{A}) \to U], \mathtt{a} : () \to \mathtt{A}, \mathtt{b} : () \to \mathtt{B}(\mathtt{a})) \Rightarrow \Sigma(\mathtt{A}, \mathtt{B})
\end{aligned}
$$

In the negative type formulation, we add the following elimination and equality

rules.

$$\begin{aligned}
&\mathsf{proj}_1 : ([\mathtt{A} : () \to U], [\mathtt{B} : (\mathtt{x} : \mathtt{A}) \to U], \mathtt{c} : () \to \Sigma(\mathtt{A}, \mathtt{B})) \Rightarrow \mathtt{A} \\
&\mathsf{proj}_2 : ([\mathtt{A} : () \to U], [\mathtt{B} : (\mathtt{x} : \mathtt{A}) \to U], \mathtt{c} : () \to \Sigma(\mathtt{A}, \mathtt{B})) \Rightarrow \mathtt{B}(\mathsf{proj}_1(\mathtt{c})) \\
&\quad _ : (\mathtt{A} : () \to U, \mathtt{B} : (\mathtt{x} : \mathtt{A}) \to U, \mathtt{a} : () \to \mathtt{A}, \mathtt{b} : () \to \mathtt{B}(\mathtt{a})) \\
&\qquad \Rightarrow \mathsf{proj}_1(\mathsf{pair}(\mathtt{a}, \mathtt{b})) \equiv \mathtt{a} : \mathtt{A} \\
&\quad _ : (\mathtt{A} : () \to U, \mathtt{B} : (\mathtt{x} : \mathtt{A}) \to U, \mathtt{a} : () \to \mathtt{A}, \mathtt{b} : () \to \mathtt{B}(\mathtt{a})) \\
&\qquad \Rightarrow \mathsf{proj}_2(\mathsf{pair}(\mathtt{a}, \mathtt{b})) \equiv \mathtt{b} : \mathtt{B}(\mathtt{a}) \\
&\quad _ : (\mathtt{A} : () \to U, \mathtt{B} : (\mathtt{x} : \mathtt{A}) \to U, \mathtt{c} : () \to \Sigma(\mathtt{A}, \mathtt{B})) \\
&\qquad \Rightarrow \mathsf{pair}(\mathsf{proj}_1(\mathtt{c}), \mathsf{proj}_2(\mathtt{c})) \equiv \mathtt{c} : \Sigma(\mathtt{A}, \mathtt{B})
\end{aligned}$$

In the positive type formulation, Σ-types are an *inductive type* and have the following elimination and equality rules.

$$\begin{aligned}
&\mathsf{elim}_\Sigma : ([\mathtt{A} : () \to U], [\mathtt{B} : (\mathtt{x} : \mathtt{A}) \to U], \mathtt{D} : (\mathtt{z} : \Sigma(\mathtt{A}, \mathtt{B})) \to U, \\
&\qquad \mathtt{d} : (\mathtt{x} : \mathtt{A}, \mathtt{y} : \mathtt{B}(\mathtt{x})) \to \mathtt{D}(\mathsf{pair}(\mathtt{x}, \mathtt{y})), \mathtt{c} : () \to \Sigma(\mathtt{A}, \mathtt{B})) \Rightarrow \mathtt{D}(\mathtt{c}) \\
&\quad _ : (\mathtt{A} : () \to U, \mathtt{B} : (\mathtt{x} : \mathtt{A}) \to U, \mathtt{D} : (\mathtt{z} : \Sigma(\mathtt{A}, \mathtt{B})) \to U, \\
&\qquad \mathtt{d} : (\mathtt{x} : \mathtt{A}, \mathtt{y} : \mathtt{B}(\mathtt{x})) \to \mathtt{D}(\mathsf{pair}(\mathtt{x}, \mathtt{y})), \mathtt{a} : () \to \mathtt{A}, \mathtt{b} : () \to \mathtt{B}(\mathtt{a})) \\
&\qquad \Rightarrow \mathsf{elim}_\Sigma(\mathtt{D}, \mathtt{d}, \mathsf{pair}(\mathtt{a}, \mathtt{b})) \equiv \mathtt{d}(\mathtt{a}, \mathtt{b}) : \mathtt{D}(\mathsf{pair}(\mathtt{a}, \mathtt{b}))
\end{aligned}$$

The negative Σ-type interprets the positive one by defining

$$\mathsf{elim}_\Sigma(\mathtt{D}, \mathtt{d}, \mathtt{c}) = \mathtt{d}(\mathsf{proj}_1(\mathtt{c}), \mathsf{proj}_2(\mathtt{c})).$$

The positive Σ-type can interpret the eliminators and the first two equalities of the negative Σ-type, but we cannot derive the last equality $\mathsf{pair}(\mathsf{proj}_1(\mathtt{c}), \mathsf{proj}_2(\mathtt{c})) \equiv \mathtt{c}$. In this thesis, we understand Σ-types in the negative type formulation. We introduce the following notations.

$$\begin{aligned}
\sum_{\mathtt{x}:\mathtt{A}} \mathtt{B} &= \Sigma(\mathtt{A}, \langle \mathtt{x} \rangle \mathtt{B}) \\
(\mathtt{x} : \mathtt{A}) \times \mathtt{B} &= \sum_{\mathtt{x}:\mathtt{A}} \mathtt{B} \\
\mathtt{A} \times \mathtt{B} &= (\mathtt{x} : \mathtt{A}) \times \mathtt{B} \qquad (\mathtt{x} \text{ does not freely occur in } \mathtt{B}) \\
(\mathtt{a}, \mathtt{b}) &= \mathsf{pair}(\mathtt{a}, \mathtt{b})
\end{aligned}$$

4.6.5. Example. *Identity types* are also declared as positive or negative types. In both cases, we add the same type formation rule and introduction rule.

$$\begin{aligned}
&\mathsf{Id} : ([\mathtt{A} : () \to U], \mathtt{a}_1 : () \to \mathtt{A}, \mathtt{a}_2 : () \to \mathtt{A}) \Rightarrow U \\
&\mathsf{refl} : ([\mathtt{A} : () \to U], \mathtt{a} : () \to \mathtt{A}) \Rightarrow \mathsf{Id}(\mathtt{a}, \mathtt{a})
\end{aligned}$$

In the negative type formulation, called *extensional identity types*, we add the following equality rules.

$$\begin{aligned}
_ &: (\mathtt{A} : () \to U, \mathtt{a}_1 : () \to \mathtt{A}, \mathtt{a}_2 : () \to \mathtt{A}, \mathtt{p} : () \to \mathsf{Id}(\mathtt{a}_1, \mathtt{a}_2)) \Rightarrow \mathtt{a}_1 \equiv \mathtt{a}_2 : \mathtt{A} \\
_ &: (\mathtt{A} : () \to U, \mathtt{a} : () \to \mathtt{A}, \mathtt{p} : () \to \mathsf{Id}(\mathtt{a}, \mathtt{a})) \Rightarrow \mathtt{p} \equiv \mathsf{refl}(\mathtt{a}) : \mathsf{Id}(\mathtt{a}, \mathtt{a})
\end{aligned}$$

In the positive type formulation, called *intensional identity types*, we add the following elimination and equality rules.

$$\begin{aligned}
\mathsf{elim}_{\mathsf{Id}} &: ([\mathtt{A} : () \to U], \mathtt{B} : ([\mathtt{x}_1 : \mathtt{A}], [\mathtt{x}_2 : \mathtt{A}], \mathtt{y} : \mathsf{Id}(\mathtt{x}_1, \mathtt{x}_2)) \to U, \\
&\quad \mathtt{b} : (\mathtt{x} : \mathtt{A}) \to \mathtt{B}(\mathsf{refl}(\mathtt{x})), [\mathtt{a}_1 : () \to \mathtt{A}], [\mathtt{a}_2 : () \to \mathtt{A}], \mathtt{p} : () \to \mathsf{Id}(\mathtt{a}_1, \mathtt{a}_2)) \\
&\quad \Rightarrow \mathtt{B}(\mathtt{p}) \\
_ &: (\mathtt{A} : () \to U, \mathtt{B} : ([\mathtt{x}_1 : \mathtt{A}], [\mathtt{x}_2 : \mathtt{A}], \mathtt{y} : \mathsf{Id}(\mathtt{x}_1, \mathtt{x}_2)) \to U, \mathtt{b} : (\mathtt{x} : \mathtt{A}) \to \mathtt{B}(\mathsf{refl}(\mathtt{x})), \\
&\quad \mathtt{a} : () \to \mathtt{A}) \Rightarrow \mathsf{elim}_{\mathsf{Id}}(\mathtt{B}, \mathtt{b}, \mathsf{refl}(\mathtt{a})) \equiv \mathtt{b}(\mathtt{a}) : \mathtt{B}(\mathsf{refl}(\mathtt{a}))
\end{aligned}$$

We introduce the following notation.

$$\mathtt{a}_1 == \mathtt{a}_2 = \mathsf{Id}(\mathtt{a}_1, \mathtt{a}_2)$$

4.6.6. Example. The (negative) *unit type* is declared as follows.

$$\begin{aligned}
1 &: () \Rightarrow U \\
* &: () \Rightarrow 1 \\
_ &: (\mathtt{a} : 1) \Rightarrow \mathtt{a} \equiv * : 1
\end{aligned}$$

4.6.7. Example. The *empty type* is declared as follows.

$$\begin{aligned}
0 &: () \Rightarrow U \\
\mathsf{elim}_0 &: (\mathtt{B} : (\mathtt{x} : 0) \to U, \mathtt{a} : () \to 0) \Rightarrow \mathtt{B}(\mathtt{a})
\end{aligned}$$

4.6.8. Example. *Binary coproduct types* are declared as follows.

$$\begin{aligned}
+ &: (\mathtt{A} : () \to U, \mathtt{B} : () \to U) \Rightarrow U \\
\mathsf{inl} &: ([\mathtt{A} : () \to U], [\mathtt{B} : () \to U], \mathtt{a} : () \to \mathtt{A}) \Rightarrow +(\mathtt{A}, \mathtt{B}) \\
\mathsf{inr} &: ([\mathtt{A} : () \to U], [\mathtt{B} : () \to U], \mathtt{b} : () \to \mathtt{B}) \Rightarrow +(\mathtt{A}, \mathtt{B}) \\
\mathsf{elim}_+ &: ([\mathtt{A} : () \to U], [\mathtt{B} : () \to U], \mathtt{D} : (\mathtt{z} : +(\mathtt{A}, \mathtt{B})) \to U, \mathtt{d}_1 : (\mathtt{x} : \mathtt{A}) \to \mathtt{D}(\mathsf{inl}(\mathtt{x})), \\
&\quad \mathtt{d}_2 : (\mathtt{y} : \mathtt{B}) \to \mathtt{D}(\mathsf{inr}(\mathtt{x})), \mathtt{c} : +(\mathtt{A}, \mathtt{B})) \Rightarrow \mathtt{D}(\mathtt{c}) \\
_ &: (\mathtt{A} : () \to U, \mathtt{B} : () \to U, \mathtt{D} : (\mathtt{z} : +(\mathtt{A}, \mathtt{B})) \to U, \mathtt{d}_1 : (\mathtt{x} : \mathtt{A}) \to \mathtt{D}(\mathsf{inl}(\mathtt{x})), \\
&\quad \mathtt{d}_2 : (\mathtt{y} : \mathtt{B}) \to \mathtt{D}(\mathsf{inr}(\mathtt{x})), \mathtt{a} : \mathtt{A}) \Rightarrow \mathsf{elim}_+(\mathtt{D}, \mathtt{d}_1, \mathtt{d}_2, \mathsf{inl}(\mathtt{a})) \equiv \mathtt{d}_1(\mathtt{a}) \\
_ &: (\mathtt{A} : () \to U, \mathtt{B} : () \to U, \mathtt{D} : (\mathtt{z} : +(\mathtt{A}, \mathtt{B})) \to U, \mathtt{d}_1 : (\mathtt{x} : \mathtt{A}) \to \mathtt{D}(\mathsf{inl}(\mathtt{x})), \\
&\quad \mathtt{d}_2 : (\mathtt{y} : \mathtt{B}) \to \mathtt{D}(\mathsf{inr}(\mathtt{x})), \mathtt{b} : \mathtt{B}) \Rightarrow \mathsf{elim}_+(\mathtt{D}, \mathtt{d}_1, \mathtt{d}_2, \mathsf{inr}(\mathtt{b})) \equiv \mathtt{d}_2(\mathtt{b})
\end{aligned}$$

We will write $\mathtt{A} + \mathtt{B}$ instead of $+(\mathtt{A}, \mathtt{B})$. We define 2 to be $1 + 1$ and refer to the constructors $\mathsf{inl}(*)$ and $\mathsf{inr}(*)$ as 0 and 1, respectively.

4.6.9. Example. The *natural numbers type* is declared as follows.

$$\begin{aligned}
\mathsf{N} &: () \Rightarrow U \\
\mathsf{zero} &: () \Rightarrow \mathsf{N} \\
\mathsf{succ} &: (\mathtt{n} : () \to \mathsf{N}) \Rightarrow \mathsf{N} \\
\mathsf{elim}_{\mathsf{N}} &: (\mathtt{A} : (\mathtt{x} : \mathsf{N}) \to U, \mathtt{a} : () \to \mathtt{A}(\mathsf{zero}), \mathtt{f} : ([\mathtt{x} : \mathsf{N}], \mathtt{y} : \mathtt{A}(\mathtt{x})) \to \mathtt{A}(\mathsf{succ}(\mathtt{x})), \\
&\qquad \mathtt{n} : () \to \mathsf{N}) \Rightarrow \mathtt{A}(\mathtt{n}) \\
_ &: (\mathtt{A} : (\mathtt{x} : \mathsf{N}) \to U, \mathtt{a} : () \to \mathtt{A}(\mathsf{zero}), \mathtt{f} : ([\mathtt{x} : \mathsf{N}], \mathtt{y} : \mathtt{A}(\mathtt{x})) \to \mathtt{A}(\mathsf{succ}(\mathtt{x}))) \\
&\qquad \Rightarrow \mathsf{elim}_{\mathsf{N}}(\mathtt{A}, \mathtt{a}, \mathtt{f}, \mathsf{zero}) \equiv \mathtt{a} : \mathtt{A}(\mathsf{zero}) \\
_ &: (\mathtt{A} : (\mathtt{x} : \mathsf{N}) \to U, \mathtt{a} : () \to \mathtt{A}(\mathsf{zero}), \mathtt{f} : ([\mathtt{x} : \mathsf{N}], \mathtt{y} : \mathtt{A}(\mathtt{x})) \to \mathtt{A}(\mathsf{succ}(\mathtt{x})), \\
&\qquad \mathtt{n} : () \to \mathsf{N}) \Rightarrow \mathsf{elim}_{\mathsf{N}}(\mathtt{A}, \mathtt{a}, \mathtt{f}, \mathsf{succ}(\mathtt{n})) \equiv \mathtt{f}(\mathsf{elim}_{\mathsf{N}}(\mathtt{A}, \mathtt{a}, \mathtt{f}, \mathtt{n})) : \mathtt{A}(\mathsf{succ}(\mathtt{n}))
\end{aligned}$$

4.6.10. Example. *W-types* allows us to *internally* define inductive types and are declared as follows in the presence of Π-types.

$$\begin{aligned}
\mathsf{W} &: (\mathtt{A} : () \to U, \mathtt{B} : (\mathtt{x} : \mathtt{A}) \to U) \Rightarrow U \\
\mathsf{sup} &: ([\mathtt{A} : () \to U], [\mathtt{B} : (\mathtt{x} : \mathtt{A}) \to U], \mathtt{a} : () \to \mathtt{A}, \mathtt{c} : () \to \mathtt{B}(\mathtt{a})\ \texttt{->}\ \mathsf{W}(\mathtt{A}, \mathtt{B})) \\
&\qquad \Rightarrow \mathsf{W}(\mathtt{A}, \mathtt{B}) \\
\mathsf{elim}_{\mathsf{W}} &: ([\mathtt{A} : () \to U], [\mathtt{B} : (\mathtt{x} : \mathtt{A}) \to U], \mathtt{D} : (\mathtt{z} : \mathsf{W}(\mathtt{A}, \mathtt{B})) \to U, \mathtt{f} : (\mathtt{x} : \mathtt{A}, \\
&\qquad \mathtt{z} : (\mathtt{y} : \mathtt{B}(\mathtt{x}))\ \texttt{->}\ \mathsf{W}(\mathtt{A}, \mathtt{B}), \mathtt{w} : (\mathtt{y} : \mathtt{B}(\mathtt{x}))\ \texttt{->}\ \mathtt{D}(\mathtt{z}(\mathtt{y}))) \to \mathtt{D}(\mathsf{sup}(\mathtt{x}, \mathtt{z})), \\
&\qquad \mathtt{c} : \mathsf{W}(\mathtt{A}, \mathtt{B})) \Rightarrow \mathtt{D}(\mathtt{c}) \\
_ &: (\mathtt{A} : () \to U, \mathtt{B} : (\mathtt{x} : \mathtt{A}) \to U, \mathtt{D} : (\mathtt{z} : \mathsf{W}(\mathtt{A}, \mathtt{B})) \to U, \mathtt{f} : (\mathtt{x} : \mathtt{A}, \\
&\qquad \mathtt{z} : (\mathtt{y} : \mathtt{B}(\mathtt{x}))\ \texttt{->}\ \mathsf{W}(\mathtt{A}, \mathtt{B}), \mathtt{w} : (\mathtt{y} : \mathtt{B}(\mathtt{x}))\ \texttt{->}\ \mathtt{D}(\mathtt{z}(\mathtt{y}))) \to \mathtt{D}(\mathsf{sup}(\mathtt{x}, \mathtt{z})), \\
&\qquad \mathtt{a} : () \to \mathtt{A}, \mathtt{c} : () \to \mathtt{B}(\mathtt{a})\ \texttt{->}\ \mathsf{W}(\mathtt{A}, \mathtt{B})) \\
&\qquad \Rightarrow \mathsf{elim}_{\mathsf{W}}(\mathtt{D}, \mathtt{f}, \mathsf{sup}(\mathtt{a}, \mathtt{c})) \equiv \mathtt{f}(\mathtt{a}, \mathtt{c}, \lambda \mathtt{y}.\mathsf{elim}_{\mathsf{W}}(\mathtt{D}, \mathtt{f}, \mathtt{c}(\mathtt{y}))) : \mathtt{D}(\mathsf{sup}(\mathtt{a}, \mathtt{c}))
\end{aligned}$$

Notice that we need Π-types (`->`) to introduce the elimination rule, and thus W-types do not make sense in the absence of Π-types. The requirement of Π-types is natural because the job of W-types is to construct inductive types inside a type theory and thus the type theory is already assumed to be strong enough to internalize some concepts. Without Π-types, almost nothing can be internalized.

4.6.11. Example. A *universe* (à la Tarski) is declared as the following symbols.

$$\begin{aligned}
\mathsf{u} &: () \Rightarrow U \\
\mathsf{el}_{\mathsf{u}} &: (\mathtt{A} : () \to \mathsf{u}) \Rightarrow U
\end{aligned}$$

We usually require a universe to be closed under type formations. For example, to make u closed under Π-types, we add the following.

$$\begin{aligned}
\Pi_{\mathsf{u}} &: (\mathtt{A} : () \to \mathsf{u}, \mathtt{B} : (\mathtt{x} : \mathsf{el}_{\mathsf{u}}(\mathtt{A})) \to \mathsf{u}) \Rightarrow \mathsf{u} \\
_ &: (\mathtt{A} : () \to \mathsf{u}, \mathtt{B} : (\mathtt{x} : \mathsf{el}_{\mathsf{u}}(\mathtt{A})) \to \mathsf{u}) \Rightarrow \mathsf{el}_{\mathsf{u}}(\Pi_{\mathsf{u}}(\mathtt{A}, \mathtt{B})) \equiv \Pi(\mathsf{el}_{\mathsf{u}}(\mathtt{A}), \langle \mathtt{x} \rangle \mathsf{el}_{\mathsf{u}}(\mathtt{B}(\mathtt{x}))) : U
\end{aligned}$$

In some models of a type theory with a universe, equalities of elements of U like $\mathsf{el_u}(\Pi_\mathsf{u}(\mathtt{A},\mathtt{B})) \equiv \Pi(\mathsf{el_u}(\mathtt{A}), \langle \mathtt{x} \rangle \mathsf{el_u}(\mathtt{B}(\mathtt{x})))$ hold only up to isomorphism but not on the nose. We thus consider weaker closure conditions for universes. We say a universe u is *weakly closed* under a certain type constructor if u has the type formation rule restricted to types from u and the same introduction, elimination and equality rules. For example, to make u weakly closed under Π-types, we add the following.

$$
\begin{aligned}
&\Pi_\mathsf{u} : (\mathtt{A} : () \to \mathsf{u}, \mathtt{B} : (\mathtt{x} : \mathsf{el_u}(\mathtt{A})) \to \mathsf{u}) \Rightarrow \mathsf{u} \\
&\lambda_\mathsf{u} : ([\mathtt{A} : () \to \mathsf{u}], [\mathtt{B} : (\mathtt{x} : \mathsf{el_u}(\mathtt{A})) \to \mathsf{u}], \mathtt{b} : (\mathtt{x} : \mathsf{el_u}(\mathtt{A})) \to \mathsf{el_u}(\mathtt{B}(\mathtt{x}))) \\
&\qquad \Rightarrow \mathsf{el_u}(\Pi_\mathsf{u}(\mathtt{A},\mathtt{B})) \\
&@_\mathsf{u} : ([\mathtt{A} : () \to \mathsf{u}], [\mathtt{B} : (\mathtt{x} : \mathsf{el_u}(\mathtt{A})) \to \mathsf{u}], \mathtt{f} : () \to \mathsf{el_u}(\Pi_\mathsf{u}(\mathtt{A},\mathtt{B})), \mathtt{a} : () \to \mathsf{el_u}(\mathtt{A})) \\
&\qquad \Rightarrow \mathsf{el_u}(\mathtt{B}(\mathtt{a})) \\
&_ : (\mathtt{A} : () \to \mathsf{u}, \mathtt{B} : (\mathtt{x} : \mathsf{el_u}(\mathtt{A})) \to \mathsf{u}, \mathtt{b} : (\mathtt{x} : \mathsf{el_u}(\mathtt{A})) \to \mathsf{el_u}(\mathtt{B}(\mathtt{x})), \mathtt{a} : () \to \mathsf{el_u}(\mathtt{A})) \\
&\qquad \Rightarrow @_\mathsf{u}(\lambda_\mathsf{u}(\mathtt{b}), \mathtt{a}) \equiv \mathtt{b}(\mathtt{a}) : \mathsf{el_u}(\mathtt{B}(\mathtt{a})) \\
&_ : (\mathtt{A} : () \to \mathsf{u}, \mathtt{B} : (\mathtt{x} : \mathsf{el_u}(\mathtt{A})) \to \mathsf{u}, \mathtt{f} : () \to \mathsf{el_u}(\Pi_\mathsf{u}(\mathtt{A},\mathtt{B}))) \\
&\qquad \Rightarrow \lambda_\mathsf{u}(\langle \mathtt{x} \rangle @_\mathsf{u}(\mathtt{f}, \mathtt{x})) \equiv \mathtt{f} : \mathsf{el_u}(\Pi_\mathsf{u}(\mathtt{A},\mathtt{B}))
\end{aligned}
$$

For readability, we often omit $\mathsf{el_u}$ and regard $\mathtt{A} : \mathsf{u}$ as an element of U. The subscript $_\mathsf{u}$ in Π_u, λ_u, and $@_\mathsf{u}$ is also omitted.

4.6.2 Extensions of Martin-Löf type theory

There are a variety of extensions of Martin-Löf type theory including the *univalence axiom* and *higher inductive types*, the main features of *homotopy type theory* [172].

Well-behaved finite coproducts

Finite coproducts are not necessarily "well-behaved". In category theory, finite coproducts are considered to be well-behaved when the following conditions are satisfied:

1. finite coproducts are *stable under pullbacks*, that is, finite coproducts are preserved by pullback functors;

2. binary coproducts are *disjoint*, that is, the coproduct inclusions are monomorphisms and the pullback of the coproduct inclusions is the initial object.

The first condition is satisfied in any type theory, because pullbacks correspond to substitutions and any type-theoretic construction is stable under substitutions. The other condition is not for free, and we need additional axioms. We note that when finite coproducts are stable under pullbacks, binary coproducts are disjoint

if and only if the pullback of the coproduct inclusions is the initial object [34, Lemma 2.13]. Therefore, we only need the following axiom.

4.6.12. Example. *Disjoint finite coproducts* are declared by the empty type and binary coproducts together with the following additional symbol [cf. 121].

$$_ : (\mathtt{A} : () \to U, \mathtt{B} : () \to U, \mathtt{a} : () \to \mathtt{A}, \mathtt{b} : () \to \mathtt{B}, \mathtt{p} : () \to \mathsf{inl}(\mathtt{a}) \equiv \mathsf{inr}(\mathtt{b})) \Rightarrow \mathsf{0}$$

We sometimes need stronger finite coproducts.

4.6.13. Example. *Strictly extensive finite coproducts* are declared as the empty type and binary coproducts together with the following additional elimination rules.

$$\begin{aligned}
&\mathsf{elim}_{\mathsf{0}}^{U} : (\mathtt{a} : () \to \mathsf{0}) \Rightarrow U \\
&\qquad _ : (\mathtt{A} : (\mathtt{x} : \mathsf{0}) \to U, \mathtt{a} : () \to \mathsf{0}) \Rightarrow \mathtt{A} \equiv \mathsf{elim}_{\mathsf{0}}^{U}(\mathtt{a}) : U \\
&\mathsf{elim}_{+}^{U} : ([\mathtt{A}_1 : () \to U], [\mathtt{A}_2 : () \to U], \mathtt{B}_1 : (\mathtt{x}_1 : \mathtt{A}_1) \to U, \mathtt{B}_2 : (\mathtt{x}_2 : \mathtt{A}_2) \to U, \\
&\qquad \mathtt{a} : () \to \mathtt{A}_1 + \mathtt{A}_2) \Rightarrow U \\
&\qquad _ : (\mathtt{A}_1 : () \to U, \mathtt{A}_2 : () \to U, \mathtt{B}_1 : (\mathtt{x}_1 : \mathtt{A}_1) \to U, \mathtt{B}_2 : (\mathtt{x}_2 : \mathtt{A}_2) \to U, \\
&\qquad \mathtt{a}_1 : () \to \mathtt{A}_1) \Rightarrow \mathsf{elim}_{+}^{U}(\mathtt{B}_1, \mathtt{B}_2, \mathsf{inl}(\mathtt{a}_1)) \equiv \mathtt{B}_1(\mathtt{a}_1) : U \\
&\qquad _ : (\mathtt{A}_1 : () \to U, \mathtt{A}_2 : () \to U, \mathtt{B}_1 : (\mathtt{x}_1 : \mathtt{A}_1) \to U, \mathtt{B}_2 : (\mathtt{x}_2 : \mathtt{A}_2) \to U, \\
&\qquad \mathtt{a}_2 : () \to \mathtt{A}_2) \Rightarrow \mathsf{elim}_{+}^{U}(\mathtt{B}_1, \mathtt{B}_2, \mathsf{inr}(\mathtt{a}_2)) \equiv \mathtt{B}_2(\mathtt{a}_2) : U \\
&\qquad _ : (\mathtt{A}_1 : () \to U, \mathtt{A}_2 : () \to U, \mathtt{B} : (\mathtt{x} : \mathtt{A}_1 + \mathtt{A}_2) \to U, \mathtt{a} : () \to \mathtt{A}_1 + \mathtt{A}_2) \\
&\qquad\quad \Rightarrow \mathtt{B}(\mathtt{a}) \equiv \mathsf{elim}_{+}^{U}(\langle \mathtt{x}_1 \rangle \mathtt{B}(\mathsf{inl}(\mathtt{x}_1)), \langle \mathtt{x}_2 \rangle \mathtt{B}(\mathsf{inr}(\mathtt{x}_2)), \mathtt{a}) : U
\end{aligned}$$

In short, these rules allow us to define a type family $(\mathtt{z} : \mathtt{A} + \mathtt{B}) \to U$ by case analysis on $\mathtt{z}$. In category theory, a coproduct $\coprod_{\xi \in \Xi} x_\xi$ in $\mathcal{C}$ is said to be extensive if we have an equivalence of categories $\mathcal{C}/\coprod_{\xi \in \Xi} x_\xi \simeq \prod_{\xi \in \Xi} \mathcal{C}/x_\xi$ [34]. The above rules express that we have an isomorphism $\mathcal{C}/\coprod_{\xi \in \Xi} x_\xi \cong \prod_{\xi \in \Xi} \mathcal{C}/x_\xi$ instead of an equivalence, and thus we call them strictly extensive. Translating results on extensive coproducts [34] into type theory, we see that strictly extensive finite coproducts are disjoint in the presence of Σ-types and extensional identity types. The converse is almost true: for disjoint finite coproducts, one can construct the eliminators $\mathsf{elim}_{\mathsf{0}}^{U}$ and elim_{+}^{U}, but the equation rules only hold up to isomorphism.

4.6.14. Remark. In a type theory such that any type belongs to some universe, finite coproducts are strictly extensive. Since a type is now a term of a universe, the ordinary elimination rule works for type families.

Univalence axiom

Voevodsky's *univalence axiom* asserts that the identities between two types in a universe are equivalent to the equivalences between the types. Within Martin-Löf

type theory with Π-types, Σ-types, and intensional identity types, one can define a type $\mathsf{IsEquiv}(\mathtt{f})$ asserting that a function $\mathtt{f}$ is an equivalence.

$$\mathsf{IsEquiv} : ([\mathtt{A} : () \to U], [\mathtt{B} : () \to U], \mathtt{f} : () \to \mathtt{A} \texttt{ -> } \mathtt{B}) \Rightarrow U$$

See [172, Chapter 4] for various equivalent ways of defining the type $\mathsf{IsEquiv}(\mathtt{f})$. We then define the type of equivalences

$$\mathsf{Equiv}(\mathtt{A}, \mathtt{B}) = \sum_{\mathtt{f:A->B}} \mathsf{IsEquiv}(\mathtt{f}).$$

The identity functions are equivalences, and we can define a term

$$\mathsf{id} : (\mathtt{A} : () \to U) \Rightarrow \mathsf{Equiv}(\mathtt{A}, \mathtt{A})$$

such that the underlying function of $\mathsf{id}(\mathtt{A})$ is $\lambda \mathtt{x}.\mathtt{x}$. For a universe u, we define a function

$$\mathsf{IdToEquiv} : (\mathtt{A} : () \to \mathsf{u}, \mathtt{B} : () \to \mathsf{u}) \Rightarrow \mathsf{Id}(\mathtt{A}, \mathtt{B}) \texttt{ -> } \mathsf{Equiv}(\mathtt{A}, \mathtt{B})$$

sending $\mathsf{refl}(\mathtt{A})$ to $\mathsf{id}(\mathtt{A})$ by the elimination rule of intensional identity types. The *univalence axiom* is a new symbol of the form

$$\mathsf{ua} : (\mathtt{A} : () \to \mathsf{u}, \mathtt{B} : () \to \mathsf{u}) \Rightarrow \mathsf{IsEquiv}(\mathsf{IdToEquiv}(\mathtt{A}, \mathtt{B})).$$

We note that ua is a *symbol* but called the univalence *axiom*. This is because the type $\mathsf{IsEquiv}(\mathtt{f})$ is a proposition in a homotopical sense and thus ua is treated as an axiom in homotopy type theory.

Higher inductive types

Higher inductive types are an extension of inductive types to allow declaring *path constructors* as well as point constructors. We work with Martin-Löf type theory with a well-behaved equality-like type

$$\texttt{==} : ([\mathtt{A} : () \to U], \mathsf{a}_1 : () \to \mathtt{A}, \mathsf{a}_2 : () \to \mathtt{A}) \Rightarrow U.$$

For example, we may choose extensional or intensional identity types or path types of cubical type theory (Section 4.6.3).

4.6.15. Example. The *circle* is declared as a higher inductive type with the following formation rules and constructors.

$$\begin{aligned} \mathsf{Circle} &: () \Rightarrow U \\ \mathsf{base} &: () \Rightarrow \mathsf{Circle} \\ \mathsf{loop} &: () \Rightarrow \mathsf{base} \texttt{ == } \mathsf{base} \end{aligned}$$

The elimination rule varies according to the choice of ==, but the idea is that Circle is the initial type equipped with a point and a loop on the point, that is, one can construct a unique function Circle -> A out of a point a : A and a loop p : a == a. When working with extensional identity types, the type Circle is not interesting because loop is forced to be refl. When working with intensional identity types, there is no reason for loop forced to be refl, and we can even prove that loop is distinct from refl assuming the univalence axiom [172, Section 8.1].

4.6.16. Example. A type $\mathtt{A} : U$ is said to be a *proposition* if $(\mathtt{x}_1 : \mathtt{A}, \mathtt{x}_2 : \mathtt{A})$ -> $\mathtt{x}_1$ == $\mathtt{x}_2$ is inhabited. The *propositional truncation* builds the universal proposition under a given type.

$$\begin{aligned}
\mathsf{Trunc} &: (\mathtt{A} : () \to U) \Rightarrow U \\
\mathsf{trunc} &: ([\mathtt{A} : () \to U], \mathtt{a} : () \to \mathtt{A}) \Rightarrow \mathsf{Trunc}(\mathtt{A}) \\
\mathsf{sq} &: ([\mathtt{A} : () \to U], \mathtt{b}_1 : () \to \mathsf{Trunc}(\mathtt{A}), \mathtt{b}_2 : () \to \mathsf{Trunc}(\mathtt{A})) \Rightarrow \mathtt{b}_1 == \mathtt{b}_2
\end{aligned}$$

Trunc(A) is characterized up to equivalence as the proposition equipped with a function trunc : A -> Trunc(A) such that, for any proposition P, the function

$$\lambda \mathtt{f}.\mathtt{f} \circ \mathsf{trunc} : (\mathsf{Trunc}(\mathtt{A}) \to \mathtt{P}) \to (\mathtt{A} \to \mathtt{P})$$

is invertible. When working with extensional identity types, the propositional truncation is equivalent to *bracket types* [16] and *squash types* [126].

4.6.17. Example. The *pushout* is declared as a higher inductive type with the following formation rules and constructors.

$$\begin{aligned}
\mathsf{PO} &: ([\mathtt{A} : () \to U], [\mathtt{B}_1 : () \to U], [\mathtt{B}_2 : () \to U], \mathtt{f}_1 : (\mathtt{x} : \mathtt{A}) \to \mathtt{B}_1, \\
&\quad \mathtt{f}_2 : (\mathtt{x} : \mathtt{A}) \to \mathtt{B}_2) \Rightarrow U \\
\mathsf{inl} &: ([\mathtt{A} : () \to U], [\mathtt{B}_1 : () \to U], [\mathtt{B}_2 : () \to U], [\mathtt{f}_1 : (\mathtt{x} : \mathtt{A}) \to \mathtt{B}_1], \\
&\quad [\mathtt{f}_2 : (\mathtt{x} : \mathtt{A}) \to \mathtt{B}_2], \mathtt{b}_1 : () \to \mathtt{B}_1) \Rightarrow \mathsf{PO}(\mathtt{f}_1, \mathtt{f}_2) \\
\mathsf{inr} &: ([\mathtt{A} : () \to U], [\mathtt{B}_1 : () \to U], [\mathtt{B}_2 : () \to U], [\mathtt{f}_1 : (\mathtt{x} : \mathtt{A}) \to \mathtt{B}_1], \\
&\quad [\mathtt{f}_2 : (\mathtt{x} : \mathtt{A}) \to \mathtt{B}_2], \mathtt{b}_2 : () \to \mathtt{B}_2) \Rightarrow \mathsf{PO}(\mathtt{f}_1, \mathtt{f}_2) \\
\mathsf{push} &: ([\mathtt{A} : () \to U], [\mathtt{B}_1 : () \to U], [\mathtt{B}_2 : () \to U], [\mathtt{f}_1 : (\mathtt{x} : \mathtt{A}) \to \mathtt{B}_1], \\
&\quad [\mathtt{f}_2 : (\mathtt{x} : \mathtt{A}) \to \mathtt{B}_2], \mathtt{a} : () \to \mathtt{A}) \Rightarrow \mathsf{inl}(\mathtt{f}_1(\mathtt{a})) == \mathsf{inr}(\mathtt{f}_2(\mathtt{a}))
\end{aligned}$$

4.6.18. Example. In a type theory with extensional identity types, we extend the notion of strictly extensive finite coproducts to strictly extensive finite colimits. *Strictly extensive finite colimits* are declared as strictly extensive finite

coproducts, pushouts, and the following additional elimination rules.

$$
\begin{aligned}
\mathsf{elim}^{U}_{\mathsf{PO}} : {} & ([\mathtt{A} : () \to U], [\mathtt{B}_1 : () \to U], [\mathtt{B}_2 : () \to U], [\mathtt{f}_1 : (\mathtt{x} : \mathtt{A}) \to \mathtt{B}_1], \\
& [\mathtt{f}_2 : (\mathtt{x} : \mathtt{A}) \to \mathtt{B}_2], \mathtt{C}_1 : (\mathtt{y}_1 : \mathtt{B}_1) \to U, \mathtt{C}_2 : (\mathtt{y}_2 : \mathtt{B}_2) \to U, \\
& _ : (\mathtt{x} : \mathtt{A}) \to \mathtt{C}_1(\mathtt{f}_1(\mathtt{x})) \equiv \mathtt{C}_2(\mathtt{f}_2(\mathtt{x})) : U, \mathtt{b} : \mathsf{PO}(\mathtt{f}_1, \mathtt{f}_2)) \Rightarrow U \\
_ : {} & (\mathtt{A} : () \to U, \mathtt{B}_1 : () \to U, \mathtt{B}_2 : () \to U, \mathtt{f}_1 : (\mathtt{x} : \mathtt{A}) \to \mathtt{B}_1, \mathtt{f}_2 : (\mathtt{x} : \mathtt{A}) \to \mathtt{B}_2, \\
& \mathtt{C}_1 : (\mathtt{y}_1 : \mathtt{B}_1) \to U, \mathtt{C}_2 : (\mathtt{y}_2 : \mathtt{B}_2) \to U, \\
& _ : (\mathtt{x} : \mathtt{A}) \to \mathtt{C}_1(\mathtt{f}_1(\mathtt{x})) \equiv \mathtt{C}_2(\mathtt{f}_2(\mathtt{x})) : U, \mathtt{b}_1 : \mathtt{B}_1) \\
& \Rightarrow \mathsf{elim}^{U}_{\mathsf{PO}}(\mathtt{C}_1, \mathtt{C}_2, \mathsf{inl}(\mathtt{b}_1)) \equiv \mathtt{C}_1(\mathtt{b}_1) : U \\
_ : {} & (\mathtt{A} : () \to U, \mathtt{B}_1 : () \to U, \mathtt{B}_2 : () \to U, \mathtt{f}_1 : (\mathtt{x} : \mathtt{A}) \to \mathtt{B}_1, \mathtt{f}_2 : (\mathtt{x} : \mathtt{A}) \to \mathtt{B}_2, \\
& \mathtt{C}_1 : (\mathtt{y}_1 : \mathtt{B}_1) \to U, \mathtt{C}_2 : (\mathtt{y}_2 : \mathtt{B}_2) \to U, \\
& _ : (\mathtt{x} : \mathtt{A}) \to \mathtt{C}_1(\mathtt{f}_1(\mathtt{x})) \equiv \mathtt{C}_2(\mathtt{f}_2(\mathtt{x})) : U, \mathtt{b}_2 : \mathtt{B}_2) \\
& \Rightarrow \mathsf{elim}^{U}_{\mathsf{PO}}(\mathtt{C}_1, \mathtt{C}_2, \mathsf{inr}(\mathtt{b}_2)) \equiv \mathtt{C}_2(\mathtt{b}_2) : U \\
_ : {} & (\mathtt{A} : () \to U, \mathtt{B}_1 : () \to U, \mathtt{B}_2 : () \to U, \mathtt{f}_1 : (\mathtt{x} : \mathtt{A}) \to \mathtt{B}_1, \mathtt{f}_2 : (\mathtt{x} : \mathtt{A}) \to \mathtt{B}_2, \\
& \mathtt{C} : (\mathtt{y} : \mathsf{PO}(\mathtt{f}_1, \mathtt{f}_2)) \to U, \mathtt{b} : \mathsf{PO}(\mathtt{f}_1, \mathtt{f}_2)) \\
& \Rightarrow \mathtt{C}(\mathtt{b}) \equiv \mathsf{elim}^{U}_{\mathsf{PO}}(\langle \mathtt{y}_1 \rangle \mathtt{C}(\mathsf{inl}(\mathtt{y}_1)), \langle \mathtt{y}_2 \rangle \mathtt{C}(\mathsf{inr}(\mathtt{y}_2)), \mathtt{b}) : U
\end{aligned}
$$

W-types with reductions

W-types with reductions [161] are a special form of higher inductive types in Martin-Löf type theory with Π-types, extensional identity types, and a universe Cof. We assume that any elements of any type in Cof are equal

$$_ : (\mathtt{P} : () \to \mathsf{Cof}, \mathtt{a}_1 : () \to \mathtt{P}, \mathtt{a}_2 : () \to \mathtt{P}) \Rightarrow \mathtt{a}_1 \equiv \mathtt{a}_2 : \mathtt{P},$$

so we treat a type in Cof as a proposition. The following are the formation rule and constructors of non-dependent W-types with reductions.

$$
\begin{aligned}
\mathsf{W} : {} & (\mathtt{A} : () \to U, \mathtt{B} : (\mathtt{x} : \mathtt{A}) \to U, \mathtt{P} : (\mathtt{x} : \mathtt{A}) \to \mathsf{Cof}, \mathtt{f} : (\mathtt{x} : \mathtt{A}, _ : \mathtt{P}(\mathtt{x})) \to \mathtt{B}(\mathtt{x})) \\
& \Rightarrow U \\
\mathsf{sup} : {} & ([\mathtt{A} : () \to U], [\mathtt{B} : (\mathtt{x} : \mathtt{A}) \to U], [\mathtt{P} : (\mathtt{x} : \mathtt{A}) \to \mathsf{Cof}], \\
& [\mathtt{f} : (\mathtt{x} : \mathtt{A}, _ : \mathtt{P}(\mathtt{x})) \to \mathtt{B}(\mathtt{x})], \mathtt{a} : () \to \mathtt{A}, \mathtt{c} : () \to \mathtt{B}(\mathtt{a}) \texttt{ -> } \mathsf{W}(\mathtt{A}, \mathtt{B}, \mathtt{P}, \mathtt{f})) \\
& \Rightarrow \mathsf{W}(\mathtt{A}, \mathtt{B}, \mathtt{P}, \mathtt{f}) \\
_ : {} & (\mathtt{A} : () \to U, \mathtt{B} : (\mathtt{x} : \mathtt{A}) \to U, \mathtt{P} : (\mathtt{x} : \mathtt{A}) \to \mathsf{Cof}, \mathtt{f} : (\mathtt{x} : \mathtt{A}, _ : \mathtt{P}(\mathtt{x})) \to \mathtt{B}(\mathtt{x}), \\
& \mathtt{a} : () \to \mathtt{A}, \mathtt{c} : () \to \mathtt{B}(\mathtt{a}) \texttt{ -> } \mathsf{W}(\mathtt{A}, \mathtt{B}, \mathtt{P}, \mathtt{f}), _ : () \to \mathtt{P}(\mathtt{a})) \\
& \Rightarrow \mathsf{sup}(\mathtt{a}, \mathtt{c}) \texttt{ == } \mathtt{c}(\mathtt{f}(\mathtt{a}))
\end{aligned}
$$

For the same reason as ordinary W-types, we need Π-types to formulate the elimination rule. The parameter $(\mathtt{A}, \mathtt{B}, \mathtt{P}, \mathtt{f})$ is called a *polynomial with reductions*.

4.6.19. Definition. Suppose that we are working with a type theory with strictly extensive finite coproducts. The *sum* of two polynomials with reductions $(\mathtt{A}_1, \mathtt{B}_1, \mathtt{P}_1, \mathtt{f}_1)$ and $(\mathtt{A}_2, \mathtt{B}_2, \mathtt{P}_2, \mathtt{f}_2)$ is defined to be the polynomial with reductions $(\mathtt{A}, \mathtt{B}, \mathtt{P}, \mathtt{f})$ where

- $\mathtt{A} \equiv \mathtt{A}_1 + \mathtt{A}_2$;
- $\mathtt{B}(\mathsf{inl}(\mathtt{x}_1)) \equiv \mathtt{B}_1(\mathtt{x}_1)$ and $\mathtt{B}(\mathsf{inr}(\mathtt{x}_2)) \equiv \mathtt{B}_2(\mathtt{x}_2)$;
- $\mathtt{P}(\mathsf{inl}(\mathtt{x}_1)) \equiv \mathtt{P}_1(\mathtt{x}_1)$ and $\mathtt{P}(\mathsf{inr}(\mathtt{x}_2)) \equiv \mathtt{P}_2(\mathtt{x}_2)$;
- $\mathtt{f}(\mathsf{inl}(\mathtt{x}_1)) \equiv \mathtt{f}_1(\mathtt{x}_1)$ and $\mathtt{f}(\mathsf{inr}(\mathtt{x}_2)) \equiv \mathtt{f}_2(\mathtt{x}_2)$.

Notice that we need the strict extensivity to define the type $\mathtt{B}$ by case analysis.

4.6.3 Cubical type theory

Cubical type theory (CTT) is an extension of Martin-Löf type theory with a builtin interval. We describe CTT in detail because it is the main motivating example of our notion of a type theory and because we study models of CTT in Chapters 7 and 8. Among a lot of variants of CTT [22, 23, 5, 6, 41, 134, 135, 36], we only describe CTT in the style of Cohen et al. [41] and its extension by higher inductive types [44] for concreteness.

The interval

The *interval* is declared as a representable type equipped with two *endpoints*.

$$\begin{aligned} \mathbb{I} &: () \Rightarrow \mathsf{type} \\ 0_{\mathbb{I}} &: () \Rightarrow \mathbb{I} \\ 1_{\mathbb{I}} &: () \Rightarrow \mathbb{I} \end{aligned}$$

We also add a De Morgan algebra structure [17] on $\mathbb{I}$ with $0_{\mathbb{I}}$ as the bottom element and $1_{\mathbb{I}}$ as the top element in the same say as the ordinary algebraic theory of De Morgan algebras.

In CTT, a type $A : U$ is considered as an abstract space and a term over the interval $(\mathtt{i} : \mathbb{I}) \to a : A$ is considered as a *line* or *path* in A. The endpoints of such a line is obtained by the substitution $a \cdot (\mathtt{i} := 0_{\mathbb{I}})$ and $a \cdot (\mathtt{i} := 1_{\mathbb{I}})$.

$$a \cdot (\mathtt{i} := 0_{\mathbb{I}}) \xrightarrow{a} a \cdot (\mathtt{i} := 1_{\mathbb{I}})$$

A term $(\mathtt{i}_1 : \mathbb{I}, \mathtt{i}_2 : \mathbb{I}) \to a : A$ depending on two interval variables expresses a

square in A. Substituting for $\mathtt{i}_1$ and $\mathtt{i}_2$, we can obtain edges of such a square.

$$\begin{array}{ccc} a \cdot (\mathtt{i}_1 := 0_{\mathbb{I}}, \mathtt{i}_2 := 0_{\mathbb{I}}) & \xrightarrow{a \cdot (\mathtt{i}_2 := 0_{\mathbb{I}})} & a \cdot (\mathtt{i}_1 := 1_{\mathbb{I}}, \mathtt{i}_2 := 0_{\mathbb{I}}) \\ \Big\downarrow{\scriptstyle a \cdot (\mathtt{i}_1 := 0_{\mathbb{I}})} & a & \Big\downarrow{\scriptstyle a \cdot (\mathtt{i}_1 := 1_{\mathbb{I}})} \\ a \cdot (\mathtt{i}_1 := 0_{\mathbb{I}}, \mathtt{i}_2 := 1_{\mathbb{I}}) & \xrightarrow[a \cdot (\mathtt{i}_2 := 1_{\mathbb{I}})]{} & a \cdot (\mathtt{i}_1 := 1_{\mathbb{I}}, \mathtt{i}_2 := 1_{\mathbb{I}}) \end{array}$$

In general, a term $a : A$ depending on n interval variables expresses an n-dimensional cube in A.

Cofibrations

Cofibrations are introduced to express more complex shapes than cubes. We first add the following symbols.

$$\begin{aligned} &\mathsf{Cof} : () \Rightarrow \mathsf{Type} \\ &\mathsf{true} : (\mathtt{P} : () \to \mathsf{Cof}) \Rightarrow \mathsf{prop} \end{aligned}$$

For readability, we omit the application of true and consider a term of Cof as a representable proposition. Cofibrations are required to be closed under logical connectives $\top$, $\bot$, $\wedge$ and $\vee$. For example, we add the following for conjunction.

$$\begin{aligned} &\wedge : (\mathtt{P} : () \to \mathsf{Cof}, \mathtt{Q} : () \to \mathsf{Cof}) \Rightarrow \mathsf{Cof} \\ &_ : (\mathtt{P} : () \to \mathsf{Cof}, \mathtt{Q} : () \to \mathsf{Cof}, _ : () \to \mathtt{P}, _ : () \to \mathtt{Q}) \Rightarrow \wedge(\mathtt{P}, \mathtt{Q}) \\ &_ : (\mathtt{P} : () \to \mathsf{Cof}, \mathtt{Q} : () \to \mathsf{Cof}, _ : () \to \wedge(\mathtt{P}, \mathtt{Q})) \Rightarrow \mathtt{P} \\ &_ : (\mathtt{P} : () \to \mathsf{Cof}, \mathtt{Q} : () \to \mathsf{Cof}, _ : () \to \wedge(\mathtt{P}, \mathtt{Q})) \Rightarrow \mathtt{Q} \end{aligned}$$

Only equalities with the endpoints are included.

$$\begin{aligned} &(== 0_{\mathbb{I}}) : (\mathtt{i} : () \to \mathbb{I}) \Rightarrow \mathsf{Cof} \\ &(== 1_{\mathbb{I}}) : (\mathtt{i} : () \to \mathbb{I}) \Rightarrow \mathsf{Cof} \end{aligned}$$

We also add the universal quantifier over the interval.

$$\begin{aligned} &\forall_{\mathbb{I}} : (\mathtt{P} : (\mathtt{i} : \mathbb{I}) \to \mathsf{Cof}) \Rightarrow \mathsf{Cof} \\ &_ : (\mathtt{P} : (\mathtt{i} : \mathbb{I}) \to \mathsf{Cof}, _ : (\mathtt{i} : \mathbb{I}) \to \mathtt{P}(\mathtt{i})) \Rightarrow \forall_{\mathbb{I}}(\mathtt{P}) \\ &_ : (\mathtt{P} : (\mathtt{i} : \mathbb{I}) \to \mathsf{Cof}, _ : () \to \forall_{\mathbb{I}}(\mathtt{P}), \mathtt{i} : () \to \mathbb{I}) \Rightarrow \mathtt{P}(\mathtt{i}) \end{aligned}$$

Note that in [41], the universal quantifier over the interval is defined by quantifier elimination. The bottom element and disjunction of Cof interact with components

of Martin-Löf type theory. For $\bot$, we add the following elimination and equality rules.

$$\begin{aligned}
&\mathsf{elim}^U_\bot : (_ : () \to \bot) \Rightarrow U \\
&\qquad _ : (\mathtt{A} : (_ : \bot) \to U, _ : () \to \bot) \Rightarrow \mathtt{A} \equiv \mathsf{elim}^U_\bot : U \\
&\mathsf{elim}^E_\bot : (\mathtt{A} : (_ : \bot) \to U, _ : () \to \bot) \Rightarrow \mathtt{A} \\
&\qquad _ : (\mathtt{A} : (_ : \bot) \to U, \mathtt{a} : (_ : \bot) \to \mathtt{A}, _ : () \to \bot) \Rightarrow \mathtt{a} \equiv \mathsf{elim}_\bot : \mathtt{A}
\end{aligned}$$

For $\vee$, we add the following elimination and equality rules.

$$\begin{aligned}
&\mathsf{elim}^U_\vee : ([\mathtt{P} : () \to \mathsf{Cof}], [\mathtt{Q} : () \to \mathsf{Cof}], \mathtt{A} : (_ : \mathtt{P}) \to U, \mathtt{B} : (_ : \mathtt{Q}) \to U, \\
&\qquad _ : (_ : \mathtt{P}, _ : \mathtt{Q}) \to \mathtt{A} \equiv \mathtt{B} : U, _ : () \to \vee(\mathtt{P}, \mathtt{Q})) \Rightarrow U \\
&\quad _ : (\mathtt{P} : () \to \mathsf{Cof}, \mathtt{Q} : () \to \mathsf{Cof}, \mathtt{A} : (_ : \mathtt{P}) \to U, \mathtt{B} : (_ : \mathtt{Q}) \to U, \\
&\qquad _ : (_ : \mathtt{P}, _ : \mathtt{Q}) \to \mathtt{A} \equiv \mathtt{B} : U, _ : () \to \mathtt{P}) \Rightarrow \mathsf{elim}^U_\vee(\mathtt{A}, \mathtt{B}) \equiv \mathtt{A} : U \\
&\quad _ : (\mathtt{P} : () \to \mathsf{Cof}, \mathtt{Q} : () \to \mathsf{Cof}, \mathtt{A} : (_ : \mathtt{P}) \to U, \mathtt{B} : (_ : \mathtt{Q}) \to U, \\
&\qquad _ : (_ : \mathtt{P}, _ : \mathtt{Q}) \to \mathtt{A} \equiv \mathtt{B} : U, _ : () \to \mathtt{Q}) \Rightarrow \mathsf{elim}^U_\vee(\mathtt{A}, \mathtt{B}) \equiv \mathtt{B} : U \\
&\quad _ : (\mathtt{P} : () \to \mathsf{Cof}, \mathtt{Q} : () \to \mathsf{Cof}, \mathtt{A} : (_ : \vee(\mathtt{P}, \mathtt{Q})) \to U, _ : () \to \vee(\mathtt{P}, \mathtt{Q})) \\
&\qquad \Rightarrow \mathtt{A} \equiv \mathsf{elim}^U_\vee(\mathtt{A}, \mathtt{A}) \\
&\mathsf{elim}^E_\vee : ([\mathtt{P} : () \to \mathsf{Cof}], [\mathtt{Q} : () \to \mathsf{Cof}], \mathtt{A} : (_ : \vee(\mathtt{P}, \mathtt{Q})) \to U, \mathtt{a} : (_ : \mathtt{P}) \to \mathtt{A}, \\
&\qquad \mathtt{b} : (_ : \mathtt{Q}) \to \mathtt{A}, _ : (_ : \mathtt{P}, _ : \mathtt{Q}) \to \mathtt{a} \equiv \mathtt{b} : \mathtt{A}, _ : () \to \vee(\mathtt{P}, \mathtt{Q})) \Rightarrow \mathtt{A} \\
&\quad _ : (\mathtt{P} : () \to \mathsf{Cof}, \mathtt{Q} : () \to \mathsf{Cof}, \mathtt{A} : (_ : \vee(\mathtt{P}, \mathtt{Q})) \to U, \mathtt{a} : (_ : \mathtt{P}) \to \mathtt{A}, \\
&\qquad \mathtt{b} : (_ : \mathtt{Q}) \to \mathtt{A}, _ : (_ : \mathtt{P}, _ : \mathtt{Q}) \to \mathtt{a} \equiv \mathtt{b} : \mathtt{A}, _ : () \to \mathtt{P}) \\
&\qquad \Rightarrow \mathsf{elim}^E_\vee(\mathtt{A}, \mathtt{a}, \mathtt{b}) \equiv \mathtt{a} : \mathtt{A} \\
&\quad _ : (\mathtt{P} : () \to \mathsf{Cof}, \mathtt{Q} : () \to \mathsf{Cof}, \mathtt{A} : (_ : \vee(\mathtt{P}, \mathtt{Q})) \to U, \mathtt{a} : (_ : \mathtt{P}) \to \mathtt{A}, \\
&\qquad \mathtt{b} : (_ : \mathtt{Q}) \to \mathtt{A}, _ : (_ : \mathtt{P}, _ : \mathtt{Q}) \to \mathtt{a} \equiv \mathtt{b} : \mathtt{A}, _ : () \to \mathtt{Q}) \\
&\qquad \Rightarrow \mathsf{elim}^E_\vee(\mathtt{A}, \mathtt{a}, \mathtt{b}) \equiv \mathtt{b} : \mathtt{A} \\
&\quad _ : (\mathtt{P} : () \to \mathsf{Cof}, \mathtt{Q} : () \to \mathsf{Cof}, \mathtt{A} : (_ : \vee(\mathtt{P}, \mathtt{Q})) \to U, \mathtt{a} : (_ : \vee(\mathtt{P}, \mathtt{Q})) \to \mathtt{A}, \\
&\qquad _ : () \to \vee(\mathtt{P}, \mathtt{Q})) \Rightarrow \mathtt{a} \equiv \mathsf{elim}^E_\vee(\mathtt{A}, \mathtt{a}, \mathtt{a}) : \mathtt{A}
\end{aligned}$$

With cofibrations, we can express a sub-polyhedra of a cube. For example, a term $(\mathtt{i}_1 : \mathbb{I}, \mathtt{i}_2 : \mathbb{I}, _ : (\mathtt{i}_1 == 0_\mathbb{I}) \vee (\mathtt{i}_2 == 0_\mathbb{I}) \vee (\mathtt{i}_2 == 1_\mathbb{I})) \to a : A$ is defined only on the edges $(\mathtt{i}_1 == 0_\mathbb{I})$, $(\mathtt{i}_2 == 0_\mathbb{I})$, and $(\mathtt{i}_2 == 1_\mathbb{I})$ and expresses the following "open box".

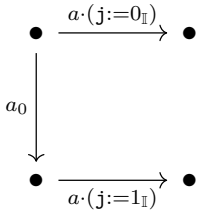
$$\begin{array}{ccc}
a \cdot (\mathtt{i}_1 := 0_\mathbb{I}, \mathtt{i}_2 := 0_\mathbb{I}) & \xrightarrow{a \cdot (\mathtt{i}_2 := 0_\mathbb{I})} & a \cdot (\mathtt{i}_1 := 1_\mathbb{I}, \mathtt{i}_2 := 0_\mathbb{I}) \\
\Big\downarrow{\scriptstyle a \cdot (\mathtt{i}_1 := 0_\mathbb{I})} & & \\
a \cdot (\mathtt{i}_1 := 0_\mathbb{I}, \mathtt{i}_2 := 1_\mathbb{I}) & \xrightarrow[a \cdot (\mathtt{i}_2 := 1_\mathbb{I})]{} & a \cdot (\mathtt{i}_1 := 1_\mathbb{I}, \mathtt{i}_2 := 1_\mathbb{I})
\end{array}$$

Composition

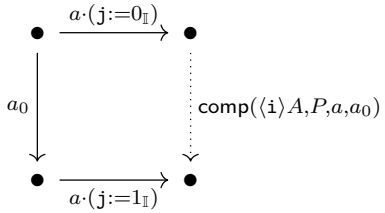
The *composition operation* is declared as follows.

$$\begin{aligned}
\mathsf{comp} : (\mathtt{A} : (\mathtt{i} : \mathbb{I}) \to U, \mathtt{P} : () \to \mathsf{Cof}, \mathtt{a} : (_ : \mathtt{P}, \mathtt{i} : \mathbb{I}) \to \mathtt{A}(\mathtt{i}), \mathtt{a}_0 : () \to \mathtt{A}(0_\mathbb{I}),\\
_ : (_ : \mathtt{P}) \to \mathtt{a}_0 \equiv \mathtt{a}(0_\mathbb{I}) : \mathtt{A}(0_\mathbb{I})) \Rightarrow \mathtt{A}(1_\mathbb{I})\\
_ : (\mathtt{A} : (\mathtt{i} : \mathbb{I}) \to U, \mathtt{P} : () \to \mathsf{Cof}, \mathtt{a} : (_ : \mathtt{P}, \mathtt{i} : \mathbb{I}) \to \mathtt{A}(\mathtt{i}), \mathtt{a}_0 : () \to \mathtt{A}(0_\mathbb{I}),\\
_ : (_ : \mathtt{P}) \to \mathtt{a}_0 \equiv \mathtt{a}(0_\mathbb{I}) : \mathtt{A}(0_\mathbb{I}), _ : () \to \mathtt{P})\\
\Rightarrow \mathsf{comp}(\mathtt{A}, \mathtt{P}, \mathtt{a}, \mathtt{a}_0) \equiv \mathtt{a}(1_\mathbb{I}) : \mathtt{A}(1_\mathbb{I})
\end{aligned}$$

Intuitively, it takes an "open box" as input and returns the "missing face". For example, let P be the cofibration $(\mathtt{j} == 0_\mathbb{I}) \vee (\mathtt{j} == 1_\mathbb{I})$ over the context $(\mathtt{j} : \mathbb{I})$ and consider the composition $\mathsf{comp}(\langle \mathtt{i} \rangle A, P, a, a_0)$ where $() \to A : U$, $(\mathtt{j} : \mathbb{I}, _ : P, \mathtt{i} : \mathbb{I}) \to a : A$, $(\mathtt{j} : \mathbb{I}) \to a_0 : A$, and $(\mathtt{j} : \mathbb{I}, _ : P) \to a_0 \equiv a \cdot (\mathtt{i} := 0_\mathbb{I}) : A$. By the definition of P, the term a is determined by the two lines $a \cdot (\mathtt{j} := 0_\mathbb{I})$ and $a \cdot (\mathtt{j} := 1_\mathbb{I})$. The condition that $a_0 \equiv a \cdot (\mathtt{i} := 0_\mathbb{I}) : A$ over $(_ : P)$ is equivalent to that $a_0 \cdot (\mathtt{j} := 0_\mathbb{I}) \equiv a \cdot (\mathtt{j} := 0_\mathbb{I}, \mathtt{i} := 0_\mathbb{I})$ and $a_0 \cdot (\mathtt{j} := 1_\mathbb{I}) \equiv a \cdot (\mathtt{j} := 1_\mathbb{I}, \mathtt{i} := 0_\mathbb{I})$. Thus, a and a_0 form the following "open box".



Then the composition $\mathsf{comp}(\langle \mathtt{i} \rangle A, P, a, a_0)$ with the condition $\mathsf{comp}(\langle \mathtt{i} \rangle A, P, a, a_0) \equiv a \cdot (\mathtt{i} := 1_\mathbb{I}) : A$ over $(_ : P)$ gives the "missing face" in this picture.



The composition can be decomposed into *homogeneous composition* and *transport*.

$$\begin{aligned}
\mathsf{hcomp} : (\mathtt{A} : () \to U, \mathtt{P} : () \to \mathsf{Cof}, \mathtt{a} : (_ : \mathtt{P}, \mathtt{i} : \mathbb{I}) \to \mathtt{A}, \mathtt{a}_0 : () \to \mathtt{A},\\
_ : (_ : \mathtt{P}) \to \mathtt{a}_0 \equiv \mathtt{a}(0_\mathbb{I}) : \mathtt{A}) \Rightarrow \mathtt{A}\\
_ : (\mathtt{A} : () \to U, \mathtt{P} : () \to \mathsf{Cof}, \mathtt{a} : (_ : \mathtt{P}, \mathtt{i} : \mathbb{I}) \to \mathtt{A}, \mathtt{a}_0 : () \to \mathtt{A},\\
_ : (_ : \mathtt{P}) \to \mathtt{a}_0 \equiv \mathtt{a}(0_\mathbb{I}) : \mathtt{A}, _ : () \to \mathtt{P}) \Rightarrow \mathsf{hcomp}(\mathtt{A}, \mathtt{P}, \mathtt{a}, \mathtt{a}_0) \equiv \mathtt{a}(1_\mathbb{I}) : \mathtt{A}\\
\mathsf{transp} : (\mathtt{A} : (\mathtt{i} : \mathbb{I}) \to U, \mathtt{P} : () \to \mathsf{Cof}, _ : (_ : \mathtt{P}, \mathtt{i} : \mathbb{I}) \to \mathtt{A}(\mathtt{i}) \equiv \mathtt{A}(0_\mathbb{I}) : U,
\end{aligned}$$

$$\begin{aligned}
&\mathtt{a}_0 : () \to \mathtt{A}(0_{\mathbb{I}})) \Rightarrow \mathtt{A}(1_{\mathbb{I}}) \\
_ &: (\mathtt{A} : (\mathtt{i} : \mathbb{I}) \to U, \mathtt{P} : () \to \mathsf{Cof}, _ : (_ : \mathtt{P}, \mathtt{i} : \mathbb{I}) \to \mathtt{A}(\mathtt{i}) \equiv \mathtt{A}(0_{\mathbb{I}}) : U, \\
&\mathtt{a}_0 : () \to \mathtt{A}(0_{\mathbb{I}}), _ : () \to \mathtt{P}) \Rightarrow \mathsf{transp}(\mathtt{A}, \mathtt{P}, \mathtt{a}_0) \equiv \mathtt{a}_0 : \mathtt{A}(0_{\mathbb{I}})
\end{aligned}$$

Indeed, we can define

$$\begin{aligned}
\mathsf{hcomp}(\mathtt{A}, \mathtt{P}, \mathtt{a}, \mathtt{a}_0) &= \mathsf{comp}(\langle \mathtt{i} \rangle \mathtt{A}, \mathtt{P}, \mathtt{a}, \mathtt{a}_0) \\
\mathsf{transp}(\mathtt{A}, \mathtt{P}, \mathtt{a}_0) &= \mathsf{comp}(\mathtt{A}, \mathtt{P}, \langle \mathtt{i} \rangle \mathtt{a}_0, \mathtt{a}_0).
\end{aligned}$$

Conversely, comp is definable from hcomp and transp [44]. This decomposition is relevant for defining higher inductive types.

Type constructors

We can add to CTT various type constructors like Π-types, Σ-types, and the natural numbers type. In addition to the usual formation, introduction, elimination, and equality rules, we add rules for interaction with compositions. For example, $\mathsf{comp}(\Sigma(\mathtt{A}, \mathtt{B}), \mathtt{P}, \mathtt{c}, \mathtt{c}_0)$ should be equal to $\mathsf{pair}(c_1, c_2)$ for suitable terms c_1 and c_2 defined using $\mathsf{comp}(\mathtt{A}, \ldots)$ and $\mathsf{comp}(\mathtt{B}, \ldots)$, respectively. See [41, Section 4.5] for details.

One new type constructor is *(dependent) path types* declared as follows.

$$\begin{aligned}
&\mathsf{Path} : (\mathtt{A} : (\mathtt{i} : \mathbb{I}) \to U, \mathtt{a}_0 : () \to \mathtt{A}(0_{\mathbb{I}}), \mathtt{a}_1 : () \to \mathtt{A}(1_{\mathbb{I}})) \Rightarrow U \\
&\quad \lambda_{\mathbb{I}} : ([\mathtt{A} : (\mathtt{i} : \mathbb{I}) \to U], \mathtt{a} : (\mathtt{i} : \mathbb{I}) \to \mathtt{A}(\mathtt{i})) \Rightarrow \mathsf{Path}(\mathtt{A}, \mathtt{a}(0_{\mathbb{I}}), \mathtt{a}(1_{\mathbb{I}})) \\
&\quad @_{\mathbb{I}} : ([\mathtt{A} : (\mathtt{i} : \mathbb{I}) \to U], [\mathtt{a}_0 : () \to \mathtt{A}(0_{\mathbb{I}})], [\mathtt{a}_1 : () \to \mathtt{A}(1_{\mathbb{I}})], \mathtt{p} : () \to \mathsf{Path}(\mathtt{A}, \mathtt{a}_0, \mathtt{a}_1), \\
&\qquad \mathtt{i} : () \to \mathbb{I}) \Rightarrow \mathtt{A}(\mathtt{i}) \\
&\quad _ : (\mathtt{A} : (\mathtt{i} : \mathbb{I}) \to U, \mathtt{a}_0 : () \to \mathtt{A}(0_{\mathbb{I}}), \mathtt{a}_1 : () \to \mathtt{A}(1_{\mathbb{I}}), \mathtt{p} : () \to \mathsf{Path}(\mathtt{A}, \mathtt{a}_0, \mathtt{a}_1)) \\
&\qquad \Rightarrow @_{\mathbb{I}}(\mathtt{p}, 0_{\mathbb{I}}) \equiv \mathtt{a}_0 : \mathtt{A}(0_{\mathbb{I}}) \\
&\quad _ : (\mathtt{A} : (\mathtt{i} : \mathbb{I}) \to U, \mathtt{a}_0 : () \to \mathtt{A}(0_{\mathbb{I}}), \mathtt{a}_1 : () \to \mathtt{A}(1_{\mathbb{I}}), \mathtt{p} : () \to \mathsf{Path}(\mathtt{A}, \mathtt{a}_0, \mathtt{a}_1)) \\
&\qquad \Rightarrow @_{\mathbb{I}}(\mathtt{p}, 1_{\mathbb{I}}) \equiv \mathtt{a}_1 : \mathtt{A}(1_{\mathbb{I}}) \\
&\quad _ : (\mathtt{A} : (\mathtt{i} : \mathbb{I}) \to U, \mathtt{a} : (\mathtt{i} : \mathbb{I}) \to \mathtt{A}(\mathtt{i}), \mathtt{i} : () \to \mathbb{I}) \Rightarrow @_{\mathbb{I}}(\lambda_{\mathbb{I}}(\mathtt{a}), \mathtt{i}) \equiv \mathtt{a}(\mathtt{i}) : \mathtt{A}(\mathtt{i}) \\
&\quad _ : (\mathtt{A} : (\mathtt{i} : \mathbb{I}) \to U, \mathtt{a}_0 : () \to \mathtt{A}(0_{\mathbb{I}}), \mathtt{a}_1 : () \to \mathtt{A}(1_{\mathbb{I}}), \mathtt{p} : () \to \mathsf{Path}(\mathtt{A}, \mathtt{a}_0, \mathtt{a}_1)) \\
&\qquad \Rightarrow \lambda_{\mathbb{I}}(\langle \mathtt{i} \rangle @_{\mathbb{I}}(\mathtt{p}, \mathtt{i})) \equiv \mathtt{p} : \mathsf{Path}(\mathtt{A}, \mathtt{a}_0, \mathtt{a}_1)
\end{aligned}$$

Path types play a similar role to intensional identity types and one can indeed emulate the constructor and eliminator of identity types using path types. However,

we only have the equality rule up to path. To fix this, a refinement is introduced.

$$
\begin{aligned}
\mathsf{Id} &: ([\mathtt{A} : () \to U], \mathtt{a}_0 : () \to \mathtt{A}, \mathtt{a}_1 : () \to \mathtt{A}) \Rightarrow U \\
\mathsf{mkId} &: ([\mathtt{A} : () \to U], \mathtt{a} : (\mathtt{i} : \mathbb{I}) \to \mathtt{A}, \mathtt{P} : () \to \mathsf{Cof}, _ : (_ : \mathtt{P}, \mathtt{i} : \mathbb{I}) \to \mathtt{a}(\mathtt{i}) \equiv \mathtt{a}(0_{\mathbb{I}})) \\
&\quad \Rightarrow \mathsf{Id}(\mathtt{a}(0_{\mathbb{I}}), \mathtt{a}(1_{\mathbb{I}})) \\
\mathsf{proj}_{\mathsf{path}} &: ([\mathtt{A} : () \to U], [\mathtt{a}_0 : () \to \mathtt{A}], [\mathtt{a}_1 : () \to \mathtt{A}], \mathtt{p} : () \to \mathsf{Id}(\mathtt{a}_0, \mathtt{a}_1), \mathtt{i} : () \to \mathbb{I}) \\
&\quad \Rightarrow \mathtt{A} \\
&\quad _ : (\mathtt{A} : () \to U, \mathtt{a}_0 : () \to \mathtt{A}, \mathtt{a}_1 : () \to \mathtt{A}, \mathtt{p} : () \to \mathsf{Id}(\mathtt{a}_0, \mathtt{a}_1)) \\
&\quad\quad \Rightarrow \mathsf{proj}_{\mathsf{path}}(\mathtt{p}, 0_{\mathbb{I}}) \equiv \mathtt{a}_0 : \mathtt{A} \\
&\quad _ : (\mathtt{A} : () \to U, \mathtt{a}_0 : () \to \mathtt{A}, \mathtt{a}_1 : () \to \mathtt{A}, \mathtt{p} : () \to \mathsf{Id}(\mathtt{a}_0, \mathtt{a}_1)) \\
&\quad\quad \Rightarrow \mathsf{proj}_{\mathsf{path}}(\mathtt{p}, 1_{\mathbb{I}}) \equiv \mathtt{a}_1 : \mathtt{A} \\
\mathsf{proj}_{\mathsf{cof}} &: ([\mathtt{A} : () \to U], [\mathtt{a}_0 : () \to \mathtt{A}], [\mathtt{a}_1 : () \to \mathtt{A}], \mathtt{p} : () \to \mathsf{Id}(\mathtt{a}_0, \mathtt{a}_1)) \Rightarrow \mathsf{Cof} \\
&\quad _ : (\mathtt{A} : () \to U, \mathtt{a}_0 : () \to \mathtt{A}, \mathtt{a}_1 : () \to \mathtt{A}, \mathtt{p} : () \to \mathsf{Id}(\mathtt{a}_0, \mathtt{a}_1), _ : () \to \mathsf{proj}_{\mathsf{cof}}(\mathtt{p}), \\
&\quad\quad \mathtt{i} : () \to \mathbb{I}) \Rightarrow \mathsf{proj}_{\mathsf{path}}(\mathtt{p}, \mathtt{i}) \equiv \mathtt{a}_0 \\
&\quad _ : (\mathtt{A} : () \to U, \mathtt{a} : (\mathtt{i} : \mathbb{I}) \to \mathtt{A}, \mathtt{P} : () \to \mathsf{Cof}, _ : (_ : \mathtt{P}, \mathtt{i} : \mathbb{I}) \to \mathtt{a}(\mathtt{i}) \equiv \mathtt{a}(0_{\mathbb{I}}), \\
&\quad\quad \mathtt{i} : () \to \mathbb{I}) \Rightarrow \mathsf{proj}_{\mathsf{path}}(\mathsf{mkId}(\mathtt{a}, \mathtt{P}), \mathtt{i}) \equiv \mathtt{a}(\mathtt{i}) : \mathtt{A} \\
&\quad _ : (\mathtt{A} : () \to U, \mathtt{a} : (\mathtt{i} : \mathbb{I}) \to \mathtt{A}, \mathtt{P} : () \to \mathsf{Cof}, _ : (_ : \mathtt{P}, \mathtt{i} : \mathbb{I}) \to \mathtt{a}(\mathtt{i}) \equiv \mathtt{a}(0_{\mathbb{I}})) \\
&\quad\quad \Rightarrow \mathsf{proj}_{\mathsf{cof}}(\mathsf{mkId}(\mathtt{a}, \mathtt{P})) \equiv \mathtt{P} : \mathsf{Cof} \\
&\quad _ : (\mathtt{A} : () \to U, \mathtt{a}_0 : () \to \mathtt{A}, \mathtt{a}_1 : () \to \mathtt{A}, \mathtt{p} : () \to \mathsf{Id}(\mathtt{a}_0, \mathtt{a}_1)) \\
&\quad\quad \Rightarrow \mathsf{mkId}\big(\langle \mathtt{i} \rangle \mathsf{proj}_{\mathsf{path}}(\mathtt{p}, \mathtt{i}), \mathsf{proj}_{\mathsf{cof}}(\mathtt{p})\big) \equiv \mathtt{p} : \mathsf{Id}(\mathtt{a}_0, \mathtt{a}_1)
\end{aligned}
$$

Then one can emulate Martin-Löf's intensional identity types including the equality rule; see [41, Section 9.1] for details. Although the new identity types ensure that CTT interprets Martin-Löf type theory, path types are preferred to identity types in CTT.

Using path types, Π-types, and Σ-types, we can define a type $\mathsf{IsEquiv}(\mathtt{f})$ asserting that a function $\mathtt{f}$ is an equivalence.

$$\mathsf{IsEquiv} : ([\mathtt{A} : () \to U], [\mathtt{B} : () \to U], \mathtt{f} : () \to \mathtt{A} \texttt{ -> } \mathtt{B}) \Rightarrow U$$

See [41, Section 5] for details. We then define the type of equivalences

$$\mathsf{Equiv}(\mathtt{A}, \mathtt{B}) = \sum_{\mathtt{f:A->B}} \mathsf{IsEquiv}(\mathtt{f}).$$

We think of $\mathsf{Equiv}(\mathtt{A}, \mathtt{B})$ as a subtype of $\mathtt{A} \texttt{ -> } \mathtt{B}$, so we have $\mathtt{f}(\mathtt{a}) : \mathtt{B}$ for $\mathtt{f} : \mathsf{Equiv}(\mathtt{A}, \mathtt{B})$ and $\mathtt{a} : \mathtt{A}$.

Another new type constructor is *glueing* declared as follows.

$$
\begin{aligned}
&\mathsf{Glue} : (\mathtt{P} : () \to \mathsf{Cof}, \mathtt{A} : (_ : \mathtt{P}) \to U, \mathtt{B} : () \to U, \mathtt{f} : (_ : \mathtt{P}) \to \mathsf{Equiv}(\mathtt{A}, \mathtt{B})) \Rightarrow U \\
&\quad _ : \mathsf{Glue} : (\mathtt{P} : () \to \mathsf{Cof}, \mathtt{A} : (_ : \mathtt{P}) \to U, \mathtt{B} : () \to U, \mathtt{f} : (_ : \mathtt{P}) \to \mathsf{Equiv}(\mathtt{A}, \mathtt{B}), \\
&\qquad _ : () \to \mathtt{P}) \Rightarrow \mathsf{Glue}(\mathtt{P}, \mathtt{A}, \mathtt{B}, \mathtt{f}) \equiv \mathtt{A} : U \\
&\mathsf{glue} : ([\mathtt{P} : () \to \mathsf{Cof}], [\mathtt{A} : (_ : \mathtt{P}) \to U], [\mathtt{B} : () \to U], [\mathtt{f} : (_ : \mathtt{P}) \to \mathsf{Equiv}(\mathtt{A}, \mathtt{B})], \\
&\qquad \mathtt{a} : (_ : \mathtt{P}) \to \mathtt{A}, \mathtt{b} : () \to \mathtt{B}, _ : (_ : \mathtt{P}) \to \mathtt{b} \equiv \mathtt{f}(\mathtt{a})) \Rightarrow \mathsf{Glue}(\mathtt{P}, \mathtt{A}, \mathtt{B}, \mathtt{f}) \\
&\quad _ : (\mathtt{P} : () \to \mathsf{Cof}, \mathtt{A} : (_ : \mathtt{P}) \to U, \mathtt{B} : () \to U, \mathtt{f} : (_ : \mathtt{P}) \to \mathsf{Equiv}(\mathtt{A}, \mathtt{B}), \\
&\qquad \mathtt{a} : (_ : \mathtt{P}) \to \mathtt{A}, \mathtt{b} : () \to \mathtt{B}, _ : (_ : \mathtt{P}) \to \mathtt{b} \equiv \mathtt{f}(\mathtt{a}), _ : () \to \mathtt{P}) \\
&\qquad \Rightarrow \mathsf{glue}(\mathtt{a}, \mathtt{b}) \equiv \mathtt{a} : \mathtt{A} \\
&\mathsf{unglue} : ([\mathtt{P} : () \to \mathsf{Cof}], [\mathtt{A} : (_ : \mathtt{P}) \to U], [\mathtt{B} : () \to U], [\mathtt{f} : (_ : \mathtt{P}) \to \mathsf{Equiv}(\mathtt{A}, \mathtt{B})], \\
&\qquad \mathtt{c} : () \to \mathsf{Glue}(\mathtt{P}, \mathtt{A}, \mathtt{B}, \mathtt{f})) \Rightarrow \mathtt{B} \\
&\quad _ : (\mathtt{P} : () \to \mathsf{Cof}, \mathtt{A} : (_ : \mathtt{P}) \to U, \mathtt{B} : () \to U, \mathtt{f} : (_ : \mathtt{P}) \to \mathsf{Equiv}(\mathtt{A}, \mathtt{B}), \\
&\qquad \mathtt{c} : () \to \mathsf{Glue}(\mathtt{P}, \mathtt{A}, \mathtt{B}, \mathtt{f}), _ : () \to \mathtt{P}) \Rightarrow \mathsf{unglue}(\mathtt{c}) \equiv \mathtt{f}(\mathtt{c}) \\
&\quad _ : (\mathtt{P} : () \to \mathsf{Cof}, \mathtt{A} : (_ : \mathtt{P}) \to U, \mathtt{B} : () \to U, \mathtt{f} : (_ : \mathtt{P}) \to \mathsf{Equiv}(\mathtt{A}, \mathtt{B}), \\
&\qquad \mathtt{c} : () \to \mathsf{Glue}(\mathtt{P}, \mathtt{A}, \mathtt{B}, \mathtt{f})) \Rightarrow \mathsf{glue}(\mathtt{c}, \mathsf{unglue}(\mathtt{c})) \equiv \mathtt{c} : \mathsf{Glue}(\mathtt{P}, \mathtt{A}, \mathtt{B}, \mathtt{f}) \\
&\quad _ : (\mathtt{P} : () \to \mathsf{Cof}, \mathtt{A} : (_ : \mathtt{P}) \to U, \mathtt{B} : () \to U, \mathtt{f} : (_ : \mathtt{P}) \to \mathsf{Equiv}(\mathtt{A}, \mathtt{B}), \\
&\qquad \mathtt{a} : (_ : \mathtt{P}) \to \mathtt{A}, \mathtt{b} : () \to \mathtt{B}, _ : (_ : \mathtt{P}) \to \mathtt{b} \equiv \mathtt{f}(\mathtt{a})) \\
&\qquad \Rightarrow \mathsf{unglue}(\mathsf{glue}(\mathtt{a}, \mathtt{b})) \equiv \mathtt{a} : \mathtt{A}
\end{aligned}
$$

The glueing operation is closely related to Voevodsky's proof of univalence in the simplicial set model [102] (see also [147]), and univalence is indeed provable over CTT. We can also view the gluing operation as the composition operation for universes. For example, let P be the cofibration $(\mathtt{j} == 0_{\mathbb{I}}) \vee (\mathtt{j} == 1_{\mathbb{I}})$ over the context $(\mathtt{j} : \mathbb{I})$ and consider the gluing $\mathsf{Glue}(P, A, B, f)$ where $(\mathtt{j} : \mathbb{I}, _ : P) \to A : U$, $(\mathtt{j} : \mathbb{I}) \to B : U$, and $(\mathtt{j} : \mathbb{I}, _ : P) \to f : \mathsf{Equiv}(A, B)$. By the definition of P, the type A is determined by the two types $A \cdot (\mathtt{j} := 0_{\mathbb{I}})$ and $A \cdot (\mathtt{j} := 1_{\mathbb{I}})$, and the equivalence f is determined by $f \cdot (\mathtt{j} := 0_{\mathbb{I}})$ and $f \cdot (\mathtt{j} := 1_{\mathbb{I}})$. Then A, B, and f form the following "open box".

$$
\begin{array}{ccc}
A \cdot (\mathtt{j} := 0_{\mathbb{I}}) & \xrightarrow[\simeq]{f \cdot (\mathtt{j} := 0_{\mathbb{I}})} & B \cdot (\mathtt{j} := 0_{\mathbb{I}}) \\
& & \Big\downarrow{\scriptstyle B} \\
A \cdot (\mathtt{j} := 1_{\mathbb{I}}) & \xrightarrow[f \cdot (\mathtt{j} := 1_{\mathbb{I}})]{\simeq} & B \cdot (\mathtt{j} := 1_{\mathbb{I}})
\end{array}
$$

Strictly, this is not an open box because the horizontal arrows are equivalences rather than paths, but univalence suggests that equivalences between two types can be regarded as paths. Then the glueing $\mathsf{Glue}(P, A, B, f)$ with the condition

$\mathsf{Glue}(P, A, B, f) \equiv A$ over $(_ : P)$ gives the "missing face" in this picture.

$$\begin{array}{ccc} A \cdot (\mathtt{j} := 0_{\mathbb{I}}) & \xrightarrow[\simeq]{f \cdot (\mathtt{j} := 0_{\mathbb{I}})} & B \cdot (\mathtt{j} := 0_{\mathbb{I}}) \\ \Big\downarrow{\scriptstyle \mathsf{Glue}(P,A,B,f)} & & \Big\downarrow{\scriptstyle B} \\ A \cdot (\mathtt{j} := 1_{\mathbb{I}}) & \xrightarrow[f \cdot (\mathtt{j} := 1_{\mathbb{I}})]{\simeq} & B \cdot (\mathtt{j} := 1_{\mathbb{I}}) \end{array}$$

Higher inductive types

Higher inductive types in CTT [44] follow a common pattern and are specified by the following data:

1. a type formation rule;
2. point constructors;
3. *path constructors* which are constructors depending on the interval $\mathbb{I}$ and the values at the endpoints are specified;
4. a *homogeneous composition structure*, which is considered as a kind of constructor;
5. when the higher inductive type takes some parameters, a *transport structure* and suitable equation rules for it;
6. elimination and equation rules.

A higher inductive type is thus the type freely generated by its point constructors, path constructors, and homogeneous composition structure. We note that although the homogeneous composition structure is a constructor, we need not specify the value at the homogeneous composition in the elimination rule, because any type in CTT carries a canonical composition structure.

4.6.20. Example. The *circle* is the higher inductive type generated by a point and a loop on the point. We first add the formation rule and constructors.

$$\begin{aligned} &\mathsf{Circle} : () \Rightarrow U \\ &\mathsf{base} : () \Rightarrow \mathsf{Circle} \\ &\mathsf{loop} : (\mathtt{i} : () \to \mathbb{I}) \Rightarrow \mathsf{Circle} \end{aligned}$$

For the path constructor loop, we specify the endpoints.

$$\begin{aligned} &_ : () \Rightarrow \mathsf{loop}(0_{\mathbb{I}}) \equiv \mathsf{base} : \mathsf{Circle} \\ &_ : () \Rightarrow \mathsf{loop}(1_{\mathbb{I}}) \equiv \mathsf{base} : \mathsf{Circle} \end{aligned}$$

We adjoin a homogeneous composition structure as a constructor.

$$\begin{aligned}
\mathsf{hcomp}_{\mathsf{Circle}} : (&\mathtt{P} : () \to \mathsf{Cof}, \mathtt{a} : (_ : \mathtt{P}, \mathtt{i} : \mathbb{I}) \to \mathsf{Circle}, \mathtt{a}_0 : () \to \mathsf{Circle},\\
&_ : (_ : \mathtt{P}) \to \mathtt{a}_0 \equiv \mathtt{a}(0_{\mathbb{I}}) : \mathsf{Circle}) \Rightarrow \mathsf{Circle}\\
_ : (&\mathtt{P} : () \to \mathsf{Cof}, \mathtt{a} : (_ : \mathtt{P}, \mathtt{i} : \mathbb{I}) \to \mathsf{Circle}, \mathtt{a}_0 : () \to \mathsf{Circle},\\
&_ : (_ : \mathtt{P}) \to \mathtt{a}_0 \equiv \mathtt{a}(0_{\mathbb{I}}) : \mathsf{Circle}, _ : () \to \mathtt{P})\\
&\Rightarrow \mathsf{hcomp}_{\mathsf{Circle}}(\mathtt{P}, \mathtt{a}, \mathtt{a}_0) \equiv \mathtt{a}(1_{\mathbb{I}}) : \mathsf{Circle}
\end{aligned}$$

The elimination and equation rules assert that we can construct a function from Circle by specifying the values at base and loop (but the value at $\mathsf{hcomp}_{\mathsf{Circle}}$ is not necessary).

$$\begin{aligned}
\mathsf{elim}_{\mathsf{Circle}} : (&\mathtt{B} : (\mathtt{x} : \mathsf{Circle}) \to U, \mathtt{b} : () \to \mathtt{B}(\mathsf{base}), \mathtt{p} : (\mathtt{i} : \mathbb{I}) \to \mathtt{B}(\mathsf{loop}(\mathtt{i})),\\
&_ : () \to \mathtt{p}(0_{\mathbb{I}}) \equiv \mathtt{b} : \mathtt{B}(\mathsf{base}), _ : () \to \mathtt{p}(1_{\mathbb{I}}) \equiv \mathtt{b} : \mathtt{B}(\mathsf{base}),\\
&\mathtt{a} : () \to \mathsf{Circle}) \Rightarrow \mathtt{B}(\mathtt{a})\\
_ : (&\mathtt{B} : (\mathtt{x} : \mathsf{Circle}) \to U, \mathtt{b} : () \to \mathtt{B}(\mathsf{base}), \mathtt{p} : (\mathtt{i} : \mathbb{I}) \to \mathtt{B}(\mathsf{loop}(\mathtt{i})),\\
&_ : () \to \mathtt{p}(0_{\mathbb{I}}) \equiv \mathtt{b} : \mathtt{B}(\mathsf{base}), _ : () \to \mathtt{p}(1_{\mathbb{I}}) \equiv \mathtt{b} : \mathtt{B}(\mathsf{base}))\\
&\Rightarrow \mathsf{elim}_{\mathsf{Circle}}(\mathtt{B}, \mathtt{b}, \mathtt{p}, \mathsf{base}) \equiv \mathtt{b} : \mathtt{B}(\mathsf{base})\\
_ : (&\mathtt{B} : (\mathtt{x} : \mathsf{Circle}) \to U, \mathtt{b} : () \to \mathtt{B}(\mathsf{base}), \mathtt{p} : (\mathtt{i} : \mathbb{I}) \to \mathtt{B}(\mathsf{loop}(\mathtt{i})),\\
&_ : () \to \mathtt{p}(0_{\mathbb{I}}) \equiv \mathtt{b} : \mathtt{B}(\mathsf{base}), _ : () \to \mathtt{p}(1_{\mathbb{I}}) \equiv \mathtt{b} : \mathtt{B}(\mathsf{base}), \mathtt{i} : () \to \mathbb{I})\\
&\Rightarrow \mathsf{elim}_{\mathsf{Circle}}(\mathtt{B}, \mathtt{b}, \mathtt{p}, \mathsf{loop}(\mathtt{i})) \equiv \mathtt{p}(\mathtt{i}) : \mathtt{B}(\mathsf{loop}(\mathtt{i}))
\end{aligned}$$

The circle also has an equation rule for $\mathsf{elim}_{\mathsf{Circle}}(\mathtt{B}, \mathtt{b}, \mathtt{p}, \mathsf{hcomp}_{\mathsf{Circle}}(\dots))$ asserting that this term is computed using the composition structure on B, but we omit it. There is no transport structure as Circle does not take parameters.

4.6.21. Example. The *suspension* is the higher inductive type with the following formation rule and point and path constructors.

$$\begin{aligned}
\mathsf{Susp} &: (\mathtt{A} : () \to U) \Rightarrow U\\
\mathsf{N} &: ([\mathtt{A} : () \to U]) \Rightarrow \mathsf{Susp}(\mathtt{A})\\
\mathsf{S} &: ([\mathtt{A} : () \to U]) \Rightarrow \mathsf{Susp}(\mathtt{A})\\
\mathsf{merid} &: ([\mathtt{A} : () \to U], \mathtt{a} : () \to \mathtt{A}, \mathtt{i} : \mathbb{I}) \Rightarrow \mathsf{Susp}(\mathtt{A})\\
_ &: (\mathtt{A} : () \to U, \mathtt{a} : () \to \mathtt{A}) \Rightarrow \mathsf{merid}(\mathtt{a}, 0_{\mathbb{I}}) \equiv \mathsf{N} : \mathsf{Susp}(\mathtt{A})\\
_ &: (\mathtt{A} : () \to U, \mathtt{a} : () \to \mathtt{A}) \Rightarrow \mathsf{merid}(\mathtt{a}, 1_{\mathbb{I}}) \equiv \mathsf{S} : \mathsf{Susp}(\mathtt{A})
\end{aligned}$$

We omit the homogeneous composition structure and the elimination rule. The transport structure on suspensions is defined by case analysis as follows.

$$
\begin{aligned}
\mathsf{transp}_{\mathsf{Susp}} : {} & (\mathtt{A} : (\mathtt{i} : \mathbb{I}) \to U, \mathtt{P} : () \to \mathsf{Cof}, _ : (_ : \mathtt{P}, \mathtt{i} : \mathbb{I}) \to \mathtt{A}(\mathtt{i}) \equiv \mathtt{A}(0_{\mathbb{I}}) : U, \\
& \mathsf{a}_0 : () \to \mathsf{Susp}(\mathtt{A}(0_{\mathbb{I}}))) \Rightarrow \mathsf{Susp}(\mathtt{A}(1_{\mathbb{I}})) \\
_ : {} & (\mathtt{A} : (\mathtt{i} : \mathbb{I}) \to U, \mathtt{P} : () \to \mathsf{Cof}, _ : (_ : \mathtt{P}, \mathtt{i} : \mathbb{I}) \to \mathtt{A}(\mathtt{i}) \equiv \mathtt{A}(0_{\mathbb{I}}) : U, \\
& \mathsf{a}_0 : () \to \mathsf{Susp}(\mathtt{A}(0_{\mathbb{I}})), _ : () \to \mathtt{P}) \\
& \Rightarrow \mathsf{transp}_{\mathsf{Susp}}(\mathtt{A}, \mathtt{P}, \mathsf{a}_0) \equiv \mathsf{a}_0 : \mathsf{Susp}(\mathtt{A}(0_{\mathbb{I}})) \\
_ : {} & (\mathtt{A} : (\mathtt{i} : \mathbb{I}) \to U, \mathtt{P} : () \to \mathsf{Cof}, _ : (_ : \mathtt{P}, \mathtt{i} : \mathbb{I}) \to \mathtt{A}(\mathtt{i}) \equiv \mathtt{A}(0_{\mathbb{I}}) : U) \\
& \Rightarrow \mathsf{transp}_{\mathsf{Susp}}(\mathtt{A}, \mathtt{P}, \mathsf{N}) \equiv \mathsf{N} : \mathsf{Susp}(\mathtt{A}(1_{\mathbb{I}})) \\
_ : {} & (\mathtt{A} : (\mathtt{i} : \mathbb{I}) \to U, \mathtt{P} : () \to \mathsf{Cof}, _ : (_ : \mathtt{P}, \mathtt{i} : \mathbb{I}) \to \mathtt{A}(\mathtt{i}) \equiv \mathtt{A}(0_{\mathbb{I}}) : U) \\
& \Rightarrow \mathsf{transp}_{\mathsf{Susp}}(\mathtt{A}, \mathtt{P}, \mathsf{S}) \equiv \mathsf{S} : \mathsf{Susp}(\mathtt{A}(1_{\mathbb{I}})) \\
_ : {} & (\mathtt{A} : (\mathtt{i} : \mathbb{I}) \to U, \mathtt{P} : () \to \mathsf{Cof}, _ : (_ : \mathtt{P}, \mathtt{i} : \mathbb{I}) \to \mathtt{A}(\mathtt{i}) \equiv \mathtt{A}(0_{\mathbb{I}}) : U, \\
& \mathsf{a}_0 : () \to \mathtt{A}, \mathtt{i} : () \to \mathbb{I}) \\
& \Rightarrow \mathsf{transp}_{\mathsf{Susp}}(\mathtt{A}, \mathtt{P}, \mathsf{merid}(\mathsf{a}_0, \mathtt{i})) \equiv \mathsf{merid}(\mathsf{transp}(\mathtt{A}, \mathtt{P}, \mathsf{a}_0), \mathtt{i}) : \mathsf{Susp}(\mathtt{A}(1_{\mathbb{I}}))
\end{aligned}
$$

4.6.22. Example. The *propositional truncation* is the higher inductive type with the following formation rule and point and path constructors.

$$
\begin{aligned}
\mathsf{Trunc} : {} & (\mathtt{A} : () \to U) \Rightarrow U \\
\mathsf{trunc} : {} & ([\mathtt{A} : () \to U], \mathsf{a} : () \to \mathtt{A}) \Rightarrow \mathsf{Trunc}(\mathtt{A}) \\
\mathsf{sq} : {} & ([\mathtt{A} : () \to U], \mathsf{b}_0 : () \to \mathsf{Trunc}(\mathtt{A}), \mathsf{b}_1 : () \to \mathsf{Trunc}(\mathtt{A}), \mathtt{i} : () \to \mathbb{I}) \\
& \Rightarrow \mathsf{Trunc}(\mathtt{A}) \\
_ : {} & (\mathtt{A} : () \to U, \mathsf{b}_0 : () \to \mathsf{Trunc}(\mathtt{A}), \mathsf{b}_1 : () \to \mathsf{Trunc}(\mathtt{A})) \\
& \Rightarrow \mathsf{sq}(\mathsf{b}_0, \mathsf{b}_1, 0_{\mathbb{I}}) \equiv \mathsf{b}_0 : \mathsf{Trunc}(\mathtt{A}) \\
_ : {} & (\mathtt{A} : () \to U, \mathsf{b}_0 : () \to \mathsf{Trunc}(\mathtt{A}), \mathsf{b}_1 : () \to \mathsf{Trunc}(\mathtt{A})) \\
& \Rightarrow \mathsf{sq}(\mathsf{b}_0, \mathsf{b}_1, 1_{\mathbb{I}}) \equiv \mathsf{b}_1 : \mathsf{Trunc}(\mathtt{A})
\end{aligned}
$$

We omit the homogeneous composition structure and the elimination rule. The transport structure is defined in a similar way to suspensions.

4.6.4 Second-order algebraic theories

We see that second-order algebraic theories of Fiore and Mahmoud [57] are naturally transformed into SOGATs.

4.6.23. Definition. A *second-order signature* $\Sigma = (O, |-|)$ is specified by a set of operators O and an arity function $|-| : O \to \mathbb{N}^*$ where $\mathbb{N}^*$ denotes the set of lists of natural numbers. For an operator $\mathsf{o} \in O$, we write $\mathsf{o} : (n_1, \ldots, n_k)$ to mean that $|\mathsf{o}| = (n_1, \ldots, n_k)$.

$$\frac{}{\Theta \triangleright \Gamma \vdash t_1 \equiv t_2}\ ((t_1, t_2) \in E) \qquad \frac{}{\Theta \triangleright \Gamma \vdash t \equiv t} \qquad \frac{\Theta \triangleright \Gamma \vdash t_1 \equiv t_2}{\Theta \triangleright \Gamma \vdash t_2 \equiv t_1}$$

$$\frac{\Theta \triangleright \Gamma \vdash t_1 \equiv t_2 \qquad \Theta \triangleright \Gamma \vdash t_2 \equiv t_3}{\Theta \triangleright \Gamma \vdash t_1 \equiv t_3}$$

$$\frac{\mathtt{X}_1 : (m_1), \ldots, \mathtt{X}_k : (m_k) \triangleright \Delta \vdash t \equiv s \qquad \{\Theta \triangleright \Gamma, \vec{\mathtt{x}_i} \vdash t_i \equiv s_i\}_{1 \leq i \leq k}}{\Theta \triangleright \Gamma, \Delta \vdash t \cdot (\mathtt{X}_i := \langle \vec{\mathtt{x}_i} \rangle t_i)_{1 \leq i \leq k} \equiv s \cdot (\mathtt{X}_i := \langle \vec{\mathtt{x}_i} \rangle s_i)_{1 \leq i \leq k}}$$

Figure 4.4: Second-order equational logic

A *context* $\Theta \triangleright \Gamma$ is specified by a sequence Θ of the form

$$\mathtt{X}_1 : (m_1), \ldots, \mathtt{X}_k : (m_k),$$

where $\mathtt{X}_1, \ldots, \mathtt{X}_k$ are distinct *metavariables* and m_i's are natural numbers, and a sequence $\Gamma = (\mathtt{x}_1, \ldots, \mathtt{x}_n)$ of distinct *variables*. Given a second-order signature Σ, the *terms over* Σ *in a context* $\Theta \triangleright \Gamma$ are inductively defined as follows, where we write $\Theta \triangleright \Gamma \vdash t$ when t is a term in $\Theta \triangleright \Gamma$:

- for any $\mathtt{x} \in \Gamma$, we have $\Theta \triangleright \Gamma \vdash \mathtt{x}$;
- for any $(\mathtt{X} : (m)) \in \Theta$ and any $\Theta \triangleright \Gamma \vdash t_i$ for $1 \leq i \leq m$, we have $\Theta \triangleright \Gamma \vdash \mathtt{X}(t_1, \ldots, t_m)$;
- for any operator $\mathtt{o} : (n_1, \ldots, n_k)$ and any $\Theta \triangleright \Gamma, \vec{\mathtt{x}_i} \vdash t_i$ for $1 \leq i \leq k$ where $\vec{\mathtt{x}_i} = (\mathtt{x}_{i,1}, \ldots, \mathtt{x}_{i,n_i})$, we have $\Theta \triangleright \Gamma \vdash \mathtt{o}(\langle \vec{\mathtt{x}_1} \rangle t_1, \ldots, \langle \vec{\mathtt{x}_k} \rangle t_k)$.

Variables $\vec{\mathtt{x}_i}$ in $\langle \vec{\mathtt{x}_i} \rangle t_i$ are bound and terms are considered up to α-equivalence.

4.6.24. Definition. An *equation* is a pair of terms in the same context. An *equational presentation* (Σ, E) is specified by a second-order signature Σ together with a set E of equations over it.

Given an equational presentation (Σ, E), the set of derivable equations, written $\Theta \triangleright \Gamma \vdash t_1 \equiv t_2$, is defined by the inference rules listed in Fig. 4.4.

4.6.25. Remark. As noted in [56, Section 3.5], any equation $\Theta \triangleright \Gamma \vdash t_1 \equiv t_2$ can be transformed into an equivalent equation $\Theta, \Gamma^\dagger \triangleright () \vdash t_1' \equiv t_2'$ in the empty variable context, which is understood as the untyped version of contextual completeness. We thus assume that all the equations in an equational presentation are in the empty variable context.

4.6.26. Construction. Let (Σ, E) be an equational presentation. We define a SOGAT $\mathrm{T}(\Sigma, E)$ as follows. We first define $\mathrm{T}(\Sigma)$ to be the SOGAT consisting of the following data:

- a representable type symbol $* : () \Rightarrow \mathsf{type}$;
- a term symbol $\mathtt{o} : (\mathtt{X}_1 : \vec{*_1} \to *, \ldots, \mathtt{X}_k : \vec{*_k} \to *) \Rightarrow *$ for any operator $\mathtt{o} : (n_1, \ldots, n_k)$, where $\vec{*_i}$ denotes the context $(\mathtt{x}_{i,1} : *, \ldots, \mathtt{x}_{i,n_i} : *)$.

Then terms over (Σ, E) are naturally regarded as term expressions over $\mathrm{T}(\Sigma)$. Any context $\Theta \triangleright \Gamma$ yields an environment Θ^* and a context Γ^* defined by $\Theta^* = (\mathtt{X}_1 : \vec{*_1} \to *, \ldots, \mathtt{X}_k : \vec{*_k} \to *)$ when $\Theta = (\mathtt{X}_1 : (m_1), \ldots, \mathtt{X}_k : (m_k))$ and $\Gamma^* = (\mathtt{x}_1 : *, \ldots, \mathtt{x}_n : *)$ when $\Gamma = (\mathtt{x}_1, \ldots, \mathtt{x}_n)$. We then define $\mathrm{T}(\Sigma, E)$ by extending $\mathrm{T}(\Sigma)$ with an axiom $\Theta^* \Rightarrow t_1 \equiv t_2 : *$ for any equation $\Theta \triangleright () \vdash t_1 \equiv t_2$ in E.

4.6.27. Proposition. *Let (Σ, E) be an equational presentation.*

1. *For any term $\Theta \triangleright \Gamma \vdash t$ over Σ, we have $\mathrm{T}(\Sigma, E), \Theta^* \vdash \Gamma^* \to t : *$.*
2. *An equation $\Theta \triangleright \Gamma \vdash t_1 \equiv t_2$ is derivable over E if and only if $\mathrm{T}(\Sigma, E), \Theta^* \vdash \Gamma^* \to t_1 \equiv t_2 : *$.*

Proof:
Straightforward by induction. □

4.6.28. Remark. In a similar way to Construction 4.6.26, we can transform many-sorted equational presentations of Fiore and Hur [55] into SOGATs. A change is that we add a representable type symbol $\sigma : () \to \mathsf{type}$ for any type of a given signature rather than add a single representable type symbol.

4.6.5 Generalized algebraic theories

A *generalized algebraic theory* of Cartmell [35] is considered as a SOGAT with no variable binding. Roughly, a generalized algebraic theory is a SOGAT with no representable type, proposition, or representable proposition symbols. In the absence of representable type or representable proposition symbols, the only possible context is the empty one, and then environments play the same role as contexts in the theory of generalized algebraic theories.

4.6.29. Example. The generalized algebraic theory of categories is represented as a SOGAT as follows.

$$
\begin{aligned}
\mathsf{Obj} &: () \Rightarrow \mathsf{Type} \\
\mathsf{Hom} &: (\mathtt{X} : () \to \mathsf{Obj}, \mathtt{Y} : () \to \mathsf{Obj}) \Rightarrow \mathsf{Type} \\
\mathsf{id} &: (\mathtt{X} : () \to \mathsf{Obj}) \Rightarrow \mathsf{Hom}(\mathtt{X}, \mathtt{X}) \\
\mathsf{comp} &: ([\mathtt{X}_1 : () \to \mathsf{Obj}], [\mathtt{X}_2 : () \to \mathsf{Obj}], [\mathtt{X}_3 : () \to \mathsf{Obj}], \mathtt{g} : () \to \mathsf{Hom}(\mathtt{X}_2, \mathtt{X}_3), \\
&\qquad \mathtt{f} : () \to \mathsf{Hom}(\mathtt{X}_1, \mathtt{X}_2)) \Rightarrow \mathsf{Hom}(\mathtt{X}_1, \mathtt{X}_3) \\
_ &: (\mathtt{X} : () \to \mathsf{Obj}, \mathtt{Y} : () \to \mathsf{Obj}, \mathtt{f} : () \to \mathsf{Hom}(\mathtt{X}, \mathtt{Y})) \\
&\qquad \Rightarrow \mathsf{comp}(\mathsf{id}(\mathtt{Y}), \mathtt{f}) \equiv \mathtt{f} : \mathsf{Hom}(\mathtt{X}, \mathtt{Y}) \\
_ &: (\mathtt{X} : () \to \mathsf{Obj}, \mathtt{Y} : () \to \mathsf{Obj}, \mathtt{f} : () \to \mathsf{Hom}(\mathtt{X}, \mathtt{Y})) \\
&\qquad \Rightarrow \mathsf{comp}(\mathtt{f}, \mathsf{id}(\mathtt{X})) \equiv \mathtt{f} : \mathsf{Hom}(\mathtt{X}, \mathtt{Y}) \\
_ &: (\mathtt{X}_1 : () \to \mathsf{Obj}, \mathtt{X}_2 : () \to \mathsf{Obj}, \mathtt{X}_3 : () \to \mathsf{Obj}, \mathtt{X}_4 : () \to \mathsf{Obj}, \\
&\qquad \mathtt{f}_3 : () \to \mathsf{Hom}(\mathtt{X}_3, \mathtt{X}_4), \mathtt{f}_2 : () \to \mathsf{Hom}(\mathtt{X}_2, \mathtt{X}_3), \mathtt{f}_1 : () \to \mathsf{Hom}(\mathtt{X}_1, \mathtt{X}_3)) \\
&\qquad \Rightarrow \mathsf{comp}(\mathtt{f}_3, \mathsf{comp}(\mathtt{f}_2, \mathtt{f}_1)) \equiv \mathsf{comp}(\mathsf{comp}(\mathtt{f}_3, \mathtt{f}_2), \mathtt{f}_1) : \mathsf{Hom}(\mathtt{X}_1, \mathtt{X}_4)
\end{aligned}
$$

There are some differences between generalized algebraic theories and SOGATs of this sort. First, type equalities are allowed in generalized algebraic theories but not in SOGATs. Therefore, only generalized algebraic theories without type equalities are regarded as SOGATs. Second, environments over a SOGAT can contain term equalities but contexts over a generalized algebraic theory cannot. This is not a big difference because, as Cartmell [35] noted, we can always add to a generalized algebraic theory symbols and axioms that axiomatize the equality predicate on a given type.

4.6.6 Bauer et al.'s general type theories

The syntax of SOGATs is inspired by the general definition of dependent type theories given by Bauer, Haselwarter, and Lumsdaine [20], but there are some changes.

SOGATs can have judgment forms other than "being a type" and "being an element of a type". With this flexibility, we can define complex type theories such as cubical type theory [41] as SOGATs. Bauer et al.'s type theories are regarded as SOGATs such that:

- they have two symbols $U : () \Rightarrow \mathsf{Type}$ and $E : (\mathtt{A} : () \to U) \Rightarrow \mathsf{type}$;
- all the other symbols are term symbols;
- all the axioms are equalities.

We have kept distinguishing metavariables from symbols and introduced environments, while Bauer et al. introduced metavariable symbols and metavariable

extensions. This change is natural from the aspect that SOGATs are a generalization of second-order algebraic theories. More importantly, environments will play a central role in the semantics of SOGATs given in Section 4.8.

In Bauer et al.'s definition, *presuppositivity* is one of the properties that a type theory should satisfy, while we include all the presuppositions in the premises of each inference rule. Because of this change, well-formed pretheories in our sense and acceptable type theories in their sense do not match. For example, consider the following pretheory which appears in [20, Example 5.14] and is acceptable in the sense of Bauer et al.

$$
\begin{aligned}
&U : () \Rightarrow \mathsf{Type} \\
&E : (\mathtt{A} : () \to U) \Rightarrow \mathsf{type} \\
&\mathsf{u} : () \Rightarrow E(\mathsf{el}(\mathsf{u})) \\
&\mathsf{el} : (\mathsf{a} : () \to E(\mathsf{el}(\mathsf{u}))) \Rightarrow U
\end{aligned}
$$

For the rule associated with the symbol u to be well-formed/acceptable, we have to derive the presupposition $() \to \mathsf{el}(\mathsf{u}) : U$. The only way of deriving this is to use the rule associated with the symbol el, and thus we have to derive $() \to \mathsf{u} : E(\mathsf{el}(\mathsf{u}))$. If we do not include the presuppositions in the premises of the rule associated with the symbol u, then we immediately derive $() \to \mathsf{u} : E(\mathsf{el}(\mathsf{u}))$. If we include the presuppositions in the premises of the rule associated with u, then we have to derive $() \to \mathsf{el}(\mathsf{u}) : U$ to derive $() \to \mathsf{u} : E(\mathsf{el}(\mathsf{u}))$, but $() \to \mathsf{el}(\mathsf{u}) : U$ was what we are trying to derive. Because of this circularity, this pretheory is not well-formed in our sense.

Nevertheless, this change on presuppositivity does not affect the notion of derivability for well-presented/well-ordered theories. For a well-presented type theory in the sense of Bauer et al., the presuppositions of the rule associated with each symbol are required to be derivable over the previously introduced rules. Therefore, we can always provide derivations of the presuppositions on each use of the rule.

Bauer et al. deal with more general well-founded relations than well-orderings on symbols and axioms. Assuming classical axioms, this is not a big difference because one can extend any well-founded relation to a well-ordering.

4.7 Theories over a SOGAT

A type theory is specified by a set of inference rules and gives a way of building a set of derivable judgments, but one can adjoin some judgments as assumptions. By a *theory* over a type theory, we mean a set of judgments adjoined to the type theory as assumptions. In the language of SOGATs, an environment plays the role of assumptions, since the set of derivable judgments is produced for each environment. We further require a theory to be well-ordered and finitary.

4.7.1. Definition. Let T be a SOGAT. A *T-theory* or *theory over T* is a well-ordered finitary environment over T. A *morphism $\Phi \to \Psi$ of T-theories* is an instantiation $T|_{\mathsf{expr}}, \Psi|_{\mathsf{term}} \vdash I : () \Rightarrow \Phi|_{\mathsf{term}}$ such that $T, \Psi \vdash () \to I : \Phi$. Two morphisms $I_1, I_2 : \Phi \to \Psi$ are *equivalent* when $T, \Psi \vdash () \to I_1 \equiv I_2 : \Phi$. The T-theories and the equivalence classes of their morphisms form a category $\mathbf{Th}(T)$ since derivable judgments are stable under instantiations (Propositions 4.4.8 and 4.4.9).

4.7.2. Example. The theories over DTT are precisely the generalized algebraic theories of Cartmell [35]. For example, the generalized algebraic theory of categories is defined as the following theory over DTT.

$$
\begin{aligned}
\mathsf{Obj} &: () \to U \\
\mathsf{Hom} &: (\mathtt{x} : \mathsf{Obj}, \mathtt{y} : \mathsf{Obj}) \to U \\
\mathsf{id} &: (\mathtt{x} : \mathsf{Obj}) \to \mathsf{Hom}(\mathtt{x}, \mathtt{x}) \\
\mathsf{comp} &: ([\mathtt{x}_1 : \mathsf{Obj}], [\mathtt{x}_2 : \mathsf{Obj}], [\mathtt{x}_3 : \mathsf{Obj}], \mathtt{g} : \mathsf{Hom}(\mathtt{x}_2, \mathtt{x}_3), \mathtt{f} : \mathsf{Hom}(\mathtt{x}_1, \mathtt{x}_2)) \\
&\qquad \to \mathsf{Hom}(\mathtt{x}_1, \mathtt{x}_3) \\
_ &: (\mathtt{x} : \mathsf{Obj}, \mathtt{y} : \mathsf{Obj}, \mathtt{f} : \mathsf{Hom}(\mathtt{x}, \mathtt{y})) \to \mathsf{comp}(\mathsf{id}(\mathtt{y}), \mathtt{f}) \equiv \mathtt{f} : \mathsf{Hom}(\mathtt{x}, \mathtt{y}) \\
_ &: (\mathtt{x} : \mathsf{Obj}, \mathtt{y} : \mathsf{Obj}, \mathtt{f} : \mathsf{Hom}(\mathtt{x}, \mathtt{y})) \to \mathsf{comp}(\mathtt{f}, \mathsf{id}(\mathtt{x})) \equiv \mathtt{f} : \mathsf{Hom}(\mathtt{x}, \mathtt{y}) \\
_ &: (\mathtt{x}_1 : \mathsf{Obj}, \mathtt{x}_2 : \mathsf{Obj}, \mathtt{x}_3 : \mathsf{Obj}, \mathtt{x}_4 : \mathsf{Obj}, \mathtt{f}_3 : \mathsf{Hom}(\mathtt{x}_3, \mathtt{x}_4), \mathtt{f}_2 : \mathsf{Hom}(\mathtt{x}_2, \mathtt{x}_3), \\
&\quad \mathtt{f}_1 : \mathsf{Hom}(\mathtt{x}_1, \mathtt{x}_3)) \\
&\qquad \to \mathsf{comp}(\mathtt{f}_3, \mathsf{comp}(\mathtt{f}_2, \mathtt{f}_1)) \equiv \mathsf{comp}(\mathsf{comp}(\mathtt{f}_3, \mathtt{f}_2), \mathtt{f}_1) : \mathsf{Hom}(\mathtt{x}_1, \mathtt{x}_4)
\end{aligned}
$$

4.7.3. Example. Type theories with Π-types are widely used as *logical frameworks* [75, 133, 132], that is, type theories for defining type theories. For example, a fragment of Martin-Löf type theory is defined as the following theory over the dependent type theory with Π-types.

$$
\begin{aligned}
\mathtt{U} &: () \to U \\
\mathtt{E} &: (\mathtt{A} : \mathtt{U}) \to U \\
\Pi &: (\mathtt{A} : \mathtt{U}, \mathtt{B} : \mathtt{E}(\mathtt{A}) \texttt{ -> } \mathtt{U}) \to \mathtt{U} \\
\mathtt{abs} &: ([\mathtt{A} : \mathtt{U}], [\mathtt{B} : \mathtt{E}(\mathtt{A}) \texttt{ -> } \mathtt{U}], \mathtt{b} : (\mathtt{x} : \mathtt{E}(\mathtt{A})) \texttt{ -> } \mathtt{E}(\mathtt{B}(\mathtt{x}))) \to \mathtt{E}(\Pi(\mathtt{A}, \mathtt{B})) \\
\mathtt{app} &: ([\mathtt{A} : \mathtt{U}], [\mathtt{B} : \mathtt{E}(\mathtt{A}) \texttt{ -> } \mathtt{U}], \mathtt{f} : \mathtt{E}(\Pi(\mathtt{A}, \mathtt{B})), \mathtt{a} : \mathtt{E}(\mathtt{A})) \to \mathtt{E}(\mathtt{B}(\mathtt{a})) \\
_ &: (\mathtt{A} : \mathtt{U}, \mathtt{B} : \mathtt{E}(\mathtt{A}) \texttt{ -> } \mathtt{U}, \mathtt{b} : (\mathtt{x} : \mathtt{E}(\mathtt{A})) \texttt{ -> } \mathtt{E}(\mathtt{B}(\mathtt{x})), \mathtt{a} : \mathtt{E}(\mathtt{A})) \\
&\qquad \to \mathtt{app}(\mathtt{abs}(\mathtt{b}), \mathtt{a}) \equiv \mathtt{b}(\mathtt{a}) : \mathtt{E}(\mathtt{B}(\mathtt{a})) \\
_ &: (\mathtt{A} : \mathtt{U}, \mathtt{B} : \mathtt{E}(\mathtt{A}) \texttt{ -> } \mathtt{U}, \mathtt{f} : \mathtt{E}(\Pi(\mathtt{A}, \mathtt{B}))) \to \mathtt{abs}(\lambda \mathtt{x}.\mathtt{app}(\mathtt{f}, \mathtt{x})) \equiv \mathtt{f} : \mathtt{E}(\Pi(\mathtt{A}, \mathtt{B})) \\
&\vdots
\end{aligned}
$$

In this section, we show that the category $\mathbf{Th}(T)$ of T-theories is compactly generated (Theorem 4.7.10). We begin by constructing colimits of T-theories.

4.7.4. Proposition. *For any SOGAT T, the category* $\mathbf{Th}(T)$ *is cocomplete. More precisely:*

1. *the coproduct of a family of T-theories $\{\Phi_\xi\}_{\xi\in\Xi}$ is given by the disjoint union of Φ_ξ's;*
2. *the coequalizer of a pair of morphisms $I_1, I_2 : \Phi \to \Psi$ is given by Ψ extended with assumptions $H_{\mathtt{X}} : \Gamma \cdot I_1 \to I_1(\mathtt{X}) \equiv I_2(\mathtt{X}) : K \cdot I_1$ for all metavariables $(\mathtt{X} : \Gamma \to K) \in \Phi$.*

Proof:
Let $\{\Phi_\xi\}_{\xi\in\Xi}$ be a family of T-theories. Choosing well-orderings on Ξ and Φ_ξ's, one can make the disjoint coproduct $\coprod_{\xi\in\Xi}\Phi_\xi$ a well-ordered finitary environment. The universal property is immediate from the definition of morphisms.

Let $I_1, I_2 : \Phi \to \Psi$ be morphisms of T-theories. Let $(I_1 \equiv I_2)$ be the relative environment over Ψ consisting of assumptions $H_{\mathtt{X}} : \Gamma \cdot I_1 \to I_1(\mathtt{X}) \equiv I_2(\mathtt{X}) : K \cdot I_1$ for all metavariable $(\mathtt{X} : \Gamma \to K) \in \Phi$ and well-ordered by $H_{\mathtt{X}} < H_{\mathtt{Y}}$ if $\mathtt{X} < \mathtt{Y}$. The well-formedness of $(I_1 \equiv I_2)$ is proved by induction on $\mathtt{X} \in \Phi$. If $(I_1 \equiv I_2)_{<H_{\mathtt{X}}}$ is well-formed, then we have $T, \Psi \,.\, (I_1 \equiv I_2)_{<H_{\mathtt{X}}} \vdash 0 \to (I_1)_{<\mathtt{X}} \equiv (I_2)_{<\mathtt{X}} : \Phi_{<\mathtt{X}}$. By equality instantiation, we have $T, \Psi \,.\, (I_1 \equiv I_2)_{<H_{\mathtt{X}}} \vdash \Gamma \cdot I_1 \to I_2(\mathtt{X}) : K \cdot I_1$, and thus $H_{\mathtt{X}}$ is well-formed over $(I_1 \equiv I_2)_{<H_{\mathtt{X}}}$. The universal property is immediate from the definition of morphisms. □

4.7.5. Corollary. *Any morphism $I : \Phi \to \Psi$ in* $\mathbf{Th}(T)$ *factors as an inclusion of environments followed by an isomorphism.*

Proof:
Ψ is isomorphic to the coequalizer Ψ' of $\mathrm{inl}, \mathrm{inr} \circ I : \Phi \to \Phi + \Psi$. Then I factors as

$$\Phi \xhookrightarrow{\mathrm{inl}} \Phi + \Psi \hookrightarrow \Psi' \xrightarrow{\cong} \Psi.$$

□

For some special colimits, there are better constructions.

4.7.6. Proposition. *Let Φ be a T-theory. Then Φ is the colimit of the chain $\{\Phi_{\leq x}\}_{x\in\Phi}$.*

Proof:
Let Ψ be a T-theory and $\{I_x : \Phi_{\leq x} \to \Psi\}_{x\in T}$ a cone in $\mathbf{Th}(T)$. For $x \leq y$, the restriction of the instantiation I_y to $\Phi_{\leq x}$ is equivalent to I_x but not necessarily equal, so we cannot simply take the union of $\{I_x\}_{x\in T}$. By transfinite induction, we replace $\{I_x\}_{x\in T}$ by some $\{I'_x\}_{x\in T}$ such that I_x is equivalent to I'_x and the restriction of I'_y to $\Phi_{\leq x}$ is equal to I'_x, and then the union of $\{I'_x\}_{x\in T}$ determines

a morphism $I : \Phi \to \Psi$ whose restriction to $\Phi_{\leq x}$ is I_x. Let $(x : \Gamma \to e) \in T$ be a metavariable or assumption and suppose that I'_y is defined for all $y < x$. Then the union of $\{I'_y\}_{y<x}$ determines an instantiation $I'_{<x}$ of $\Phi_{<x}$ in Ψ, and the restriction of I_x to $\Phi_{<x}$ is equivalent to $I'_{<x}$. By equality instantiation, we derive $\Gamma \cdot I'_{<x} \to I_x(x) : e \cdot I'_{<x}$. Thus, the extension of $I'_{<x}$ by $x \mapsto I_x(x)$ determines an instantiation I'_x of $\Phi_{\leq x}$ in Ψ equivalent to I_x.

Let $I_1, I_2 : \Phi \to \Psi$ be two morphisms that agree on every $\Phi_{\leq x}$. That is, $T, \Phi_{\leq x} \vdash \Gamma \cdot I_1 \to I_1(\mathtt{X}) \equiv I_2(\mathtt{X}) : K \cdot I_1$ for any $x \in T$ and any metavariable $(\mathtt{X} : \Gamma \to K) \leq x$. Then we have $T, \Phi \vdash \Gamma \cdot I_1 \to I_1(\mathtt{X}) \equiv I_2(\mathtt{X}) : K \cdot I_1$ for any metavariable $(\Phi : \Gamma \to K) \in \Phi$, and thus I_1 and I_2 are equal morphisms. □

4.7.7. Proposition. *Let $I : \Phi_1 \to \Phi_2$ be a morphism of T-theories. If Φ_1 is extended by a metavariable or assumption $(x : \Gamma \to e)$, then Φ_2 is extended by $(x : \Gamma \cdot I \to e \cdot I)$ and the square*

$$\begin{array}{ccc} \Phi_1 & \xrightarrow{\qquad I \qquad} & \Phi_2 \\ \big\downarrow & & \big\downarrow \\ \Phi_1 \,.\, (x : \Gamma \to e) & \xrightarrow[I+\mathrm{id}]{} & \Phi_2 \,.\, (x : \Gamma \cdot I \to e \cdot I) \end{array}$$

is a pushout in $\mathbf{Th}(T)$.

Proof:
Let $J_1 : \Phi_1 \,.\, (x : \Gamma \to e) \to \Psi$ and $J_2 : \Phi_2 \to \Psi$ be morphisms such that $J_1|_{\Phi_1} = J_2 \circ I$, that is, the instantiations $J_1|_{\Phi_1}$ and $J_2 \circ I$ are equivalent. By equality instantiation, we derive $T, \Psi \vdash \Gamma \cdot (J_2 \circ I) \to J_1(x) : e \cdot (J_2 \circ I)$ which is equivalent to $T, \Psi \vdash (\Gamma \cdot J_2) \cdot I \to J_1(x) : (e \cdot J_2) \cdot I$. Thus, the extension of J_2 by $x \mapsto J_1(x)$ determines a morphism $J : \Phi_2 \,.\, (x : \Gamma \cdot I \to e \cdot I) \to \Psi$ such that $J \circ (I + \mathrm{id}) = J_1$ and $J|_{\Phi_2} = J_2$.

For two morphisms $J_1, J_2 : \Phi_2 \,.\, (x : \Gamma \cdot I \to e \cdot I) \to \Psi$, if they agree on $\Phi_1 \,.\, (x : \Gamma \to e)$ and Φ_2, then clearly $J_1 = J_2$. □

To see that $\mathbf{Th}(T)$ is compactly generated, we observe that every T-theory is built out of finite theories under colimits. By Proposition 4.7.6, any T-theory Φ is the colimit of the chain $\{\Phi_{\leq x}\}_{x \in \Phi}$. We write each step $\Phi_{<x} \to \Phi_{\leq x}$ as a pushout of a morphism between finite T-theories as follows. First, fix a derivation D_x over $\Phi_{<x}$ of the well-formedness condition of each $x \in \Phi$. We write $y \lhd x$ when y appears in D_x, and let $\lhd^+$ and $\lhd^*$ be the transitive closure and the reflexive and transitive closure, respectively, of $\lhd$. Since D_x is finite and Φ is well-ordered, the

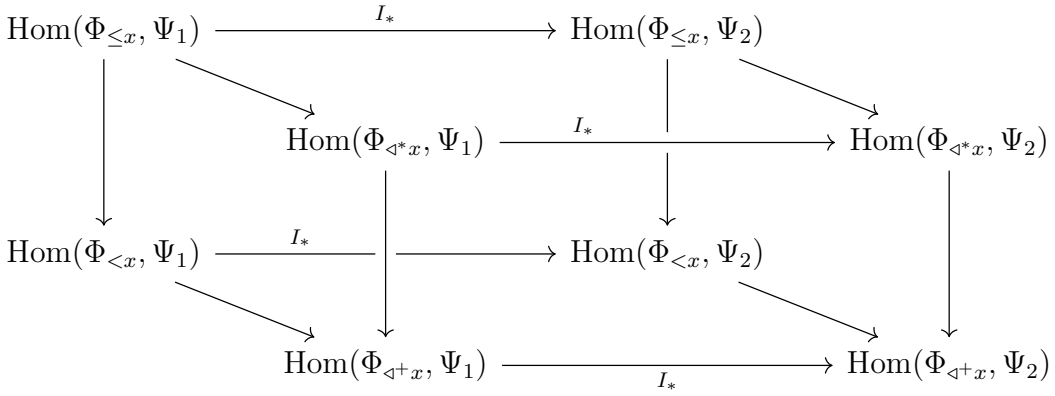
subtheory $\Phi_{\lhd^* x} \subset \Phi_{\leq x}$ is finite. By Proposition 4.7.7, the square of inclusions

$$\begin{array}{ccc} \Phi_{\lhd^+ x} & \hookrightarrow & \Phi_{<x} \\ \downarrow & & \downarrow \\ \Phi_{\lhd^* x} & \hookrightarrow & \Phi_{\leq x} \end{array}$$

is a pushout in $\mathbf{Th}(T)$.

4.7.8. Lemma. *The finite T-theories form a strong generator of $\mathbf{Th}(T)$, that is, the family of functors $\mathrm{Hom}(\Phi, -) : \mathbf{Th}(T) \to \mathbf{Set}$ indexed over the finite T-theories is jointly faithful and jointly conservative.*

Proof:
Let $I : \Psi_1 \to \Psi_2$ be a morphism of T-theories. For a T-theory Φ, the map $I_* : \mathrm{Hom}(\Phi, \Psi_1) \to \mathrm{Hom}(\Phi, \Psi_2)$ is the limit of $I_* : \mathrm{Hom}(\Phi_{\leq x}, \Psi_1) \to \mathrm{Hom}(\Phi_{\leq x}, \Psi_2)$ for all $x \in \Phi$. Moreover, the map I_* fits into the following diagram



in which the side squares are pullbacks. Then, by transfinite induction, we see: if $I_* = I'_* : \mathrm{Hom}(\Phi, \Psi_1) \to \mathrm{Hom}(\Phi, \Psi_2)$ for any finite Φ, then $I = I'$; and if $I_* : \mathrm{Hom}(\Phi, \Psi_1) \to \mathrm{Hom}(\Phi, \Psi_2)$ is invertible for any finite Φ, then I is invertible. □

4.7.9. Lemma. *Any finite T-theory is compact in $\mathbf{Th}(T)$.*

Proof:
Let $\{\Phi_\xi\}_{\xi \in \Xi}$ be a directed diagram of T-theories and let Φ denote its colimit. Let $\Phi'_\xi = \mathrm{colim}_{\xi' \leq \xi} \Phi_{\xi'}$. The diagram $\{\Phi'_\xi\}_{\xi \in \Xi}$ is isomorphic to $\{\Phi_\xi\}_{\xi \in \Xi}$, but it has a better property. From the construction of colimits described in Proposition 4.7.4, the morphism $\Phi'_{\xi_1} \to \Phi'_{\xi_2}$ for $\xi_1 \leq \xi_2$ is an inclusion of environments and Φ is the directed union of $\{\Phi'_\xi\}_{\xi \in \Xi}$. For any derivation of $T, \Phi \vdash \Gamma \to \mathcal{H}$ with Γ finite, there exists a $\xi \in \Xi$ such that the context Γ and the judgment head $\mathcal{H}$ are over Φ'_ξ

and $T, \Phi'_\xi \vdash \Gamma \to \mathcal{H}$, because the number of metavariables and assumptions used in the derivation is finite. Then it follows that any morphism $I : \Psi \to \Phi$ with Ψ finite factors through some Φ'_ξ and that, if $I_1 : \Psi \to \Phi'_{\xi_1}$ and $I_2 : \Psi \to \Phi'_{\xi_2}$ become equal in Φ, then they become equal at some $\xi \geq \xi_1, \xi_2$. Hence, for any finite T-theory Ψ, the functor $\mathbf{Th}(T)(\Psi, -) : \mathbf{Th}(T) \to \mathbf{Set}$ preserves directed colimits. □

4.7.10. Theorem. *For any SOGAT T, the category $\mathbf{Th}(T)$ is compactly generated and the compact objects are those isomorphic to a finite T-theory.*

Proof:
By Proposition 4.7.4, $\mathbf{Th}(T)$ is cocomplete. By Lemmas 4.7.8 and 4.7.9, $\mathbf{Th}(T)$ has a strong generator consisting of compact objects. Hence, $\mathbf{Th}(T)$ is compactly generated. From the construction of colimits in Proposition 4.7.4, finite T-theories are closed under finite colimits, and thus any compact object is isomorphic to a finite T-theory. □

4.8 Semantics of SOGATs

We give a connection between SOGATs and CwRs. In Section 4.8.1 we construct a CwR $\mathbf{Cl}(T)$ called the *syntactic CwR* of a SOGAT T. The goal of this section is to spell out the universal property of $\mathbf{Cl}(T)$. We introduce *interpretations* of a SOGAT T in a CwR $\mathcal{C}$ in Section 4.8.2 and show in Section 4.8.3 that the interpretations of T in $\mathcal{C}$ are equivalent to the morphisms of CwRs from $\mathbf{Cl}(T)$ to $\mathcal{C}$. An interpretation of T is defined to be a function on symbols of T, from which we can derive the universal property of $\mathbf{Cl}(T)$. We also show in Section 4.8.4 that any CwR is equivalent to the syntactic category of some SOGAT.

4.8.1 Syntactic categories

4.8.1. Definition. Let T be a SOGAT. We define a category $\mathbf{Cl}(T)$ as follows.

- The objects are the well-ordered finite environments over T.
- The morphisms $\Phi \to \Psi$ are the instantiations $T, \Phi \vdash () \to I : \Psi$. We identify two morphisms $I_1, I_2 : \Phi \to \Psi$ when $T, \Phi \vdash () \to I_1 \equiv I_2 : \Psi$.

Alternatively, $\mathbf{Cl}(T)$ is the full subcategory of $\mathbf{Th}(T)^{\mathrm{op}}$ spanned by the finite T-theories. We call $\mathbf{Cl}(T)$ the *syntactic CwR* because of Proposition 4.8.4 below.

4.8.2. Proposition. *For any SOGAT T, the category $\mathbf{Cl}(T)$ has finite limits, and we have an equivalence*

$$\mathbf{Th}(T) \simeq \mathbf{Lex}(\mathbf{Cl}(T), \mathbf{Set}).$$

Proof:
By Theorem 4.7.10. □

Recall from Section 4.4.3 that, for a context $T, \Phi \vdash \Gamma\ \mathsf{ok}$, we have the relative environment $\Gamma^\dagger$ over Φ, the substitution $T, \Phi \,.\, \Gamma^\dagger \vdash () \to \omega_\Gamma : \Gamma$, and the instantiation $T, \Phi \vdash \Gamma \to \Omega_\Gamma : \Gamma^\dagger$. Moreover, the action of the substitution ω_Γ and the action of the instantiation Ω_Γ are mutual inverses and induce a bijective correspondence between derivable judgments over $\Phi \,.\, \Gamma^\dagger$ and derivable judgments over Γ.

4.8.3. Definition. Let T be a SOGAT. We say a morphism in $\mathbf{Th}(T)^{\mathrm{op}}$ is a *representable map* if it is isomorphic to the projection

$$\Phi \,.\, \Gamma^\dagger \to \Phi$$

for some well-ordered finite context $T, \Phi \vdash \Gamma\ \mathsf{ok}$.

4.8.4. Proposition. *Let T be a SOGAT. Then $\mathbf{Th}(T)^{\mathrm{op}}$ is a CwR and $\mathbf{Cl}(T) \subset \mathbf{Th}(T)^{\mathrm{op}}$ is a sub-CwR.*

Proof:
The pullback of $\Phi \,.\, \Gamma^\dagger$ along a morphism $I : \Phi' \to \Phi$ is given by $\Phi' \,.\, (\Gamma \cdot I)^\dagger$, and thus representable maps of $\mathbf{Th}(T)^{\mathrm{op}}$ are closed under pullbacks. The identity on Φ is isomorphic to the representable map $\Phi \,.\, ()^\dagger \to \Phi$. The composite of two representable maps $\Phi \,.\, \Gamma^\dagger \to \Phi$ and $\Phi \,.\, \Gamma^\dagger \,.\, \Delta^\dagger \to \Phi \,.\, \Gamma^\dagger$ is isomorphic to the representable map $\Phi \,.\, (\Gamma \,.\, (\Delta \cdot \Omega_\Gamma))^\dagger \to \Phi$. It remains to show the exponentiability of a representable map $\Phi \,.\, \Gamma^\dagger \to \Phi$. By Corollary 4.7.5, any object over $\Phi \,.\, \Gamma^\dagger$ is isomorphic to a projection $\Phi \,.\, \Gamma^\dagger \,.\, \Psi \to \Phi \,.\, \Gamma^\dagger$. Then the pushforward of $\Phi \,.\, \Gamma^\dagger \,.\, \Psi$ along the representable map $\Phi \,.\, \Gamma^\dagger \to \Phi$ is given by $\Phi \,.\, (\Psi \cdot \Omega_\Gamma)$. Note that the action of the instantiation Ω_Γ changes the arity of metavariables and assumptions from Ψ, but $\Psi \cdot \Omega_\Gamma$ remains finitary because Γ is finite. The universal property of $\Phi \,.\, (\Psi \cdot \Omega_\Gamma)$ follows from the contextual completeness: the sections of $\Phi \,.\, (\Psi \cdot \Omega_\Gamma) \to \Phi$ bijectively correspond to the sections of $\Phi \,.\, \Gamma^\dagger \,.\, \Psi \to \Phi \,.\, \Gamma^\dagger$. We have proved that $\mathbf{Th}(T)^{\mathrm{op}}$ is a category with representable maps. Since the construction of the pushforward along a representable map preserves finiteness, $\mathbf{Cl}(T) \subset \mathbf{Th}(T)^{\mathrm{op}}$ is a subcategory with representable maps. □

In the next few sections, we show that the syntactic CwR $\mathbf{Cl}(T)$ has an appropriate universal property as follows.

1. When T is the union of a chain $(T_\alpha)_{\alpha<\lambda}$ indexed over a limit ordinal λ, the CwR $\mathbf{Cl}(T)$ is the colimit of $\mathbf{Cl}(T_\alpha)$'s. As a special case, the syntactic CwR of the empty SOGAT is the initial CwR.

2. When $T = T' \,.\, (S : \Phi \Rightarrow \mathsf{Type})$, the CwR $\mathbf{Cl}(T)$ is obtained from $\mathbf{Cl}(T')$ by freely adjoining the morphism $\Phi \,.\, (\mathtt{X} : () \to S(\mathrm{id}_\Phi)) \to \Phi$.

3. When $T = T' \,.\, (S : \Phi \Rightarrow \mathsf{type})$, the CwR $\mathbf{Cl}(T)$ is obtained from $\mathbf{Cl}(T')$ by freely adjoining the representable map $\Phi \,.\, (\mathtt{x} : S(\mathrm{id}_\Phi))^\dagger \to \Phi$.

4. When $T = T' \,.\, (S : \Phi \Rightarrow \mathsf{Prop})$, the CwR $\mathbf{Cl}(T)$ is obtained from $\mathbf{Cl}(T')$ by freely adjoining the monomorphism $\Phi \,.\, (_ : () \to S(\mathrm{id}_\Phi)) \to \Phi$.

5. When $T = T' \,.\, (S : \Phi \Rightarrow \mathsf{prop})$, the CwR $\mathbf{Cl}(T)$ is obtained from $\mathbf{Cl}(T')$ by freely adjoining the representable monomorphism $\Phi \,.\, (_ : S(\mathrm{id}_\Phi))^\dagger \to \Phi$.

6. When $T = T' \,.\, (S : \Phi \Rightarrow K)$, the CwR $\mathbf{Cl}(T)$ is obtained from $\mathbf{Cl}(T')$ by freely adjoining the section $S(\mathrm{id}_\Phi)$ of the morphism $\Phi \,.\, (\mathtt{X} : () \to K) \to \Phi$.

7. When $T = T' \,.\, (_ : \Phi \Rightarrow P)$, the CwR $\mathbf{Cl}(T)$ is obtained from $\mathbf{Cl}(T')$ by freely inverting the monomorphism $\Phi \,.\, (_ : () \to P) \to \Phi$.

4.8.5. Example. Let T_0 be DTT (Example 4.6.1). Then the syntactic CwR $\mathbf{Cl}(T_0)$ is obtained from the initial one as follows:

1. adjoin an object U;

2. adjoin a representable map $E \to U$.

In other words, $\mathbf{Cl}(T_0)$ is the free CwR generated by a representable map $\partial : E \to U$. Hence, models of $\mathbf{Cl}(T_0)$ in the sense of Definition 3.2.4 are equivalent to natural models. We also note that by Example 4.7.2 and Proposition 4.8.2, we have an alternative proof of the result of [171]: the essentially algebraic theory of generalized algebraic theories is identified with the category with finite limits free generated by an exponentiable arrow.

4.8.6. Example. Let T_1 be the extension of DTT with Π-types (Example 4.6.3). Then the syntactic CwR $\mathbf{Cl}(T_1)$ is obtained from $\mathbf{Cl}(T_0)$ by freely adjoining a pullback of the form

$$\begin{array}{ccc} \mathrm{P}_\partial(E) & \overset{\lambda}{\dashrightarrow} & E \\ {\scriptstyle \mathrm{P}_\partial(\partial)}\downarrow & \lrcorner & \downarrow{\scriptstyle \partial} \\ \mathrm{P}_\partial(U) & \underset{\Pi}{\dashrightarrow} & U. \end{array}$$

The formation rule Π corresponds to the arrow $\Pi : \mathrm{P}_\partial(U) \to U$ and the constructor λ corresponds to the arrow $\lambda : \mathrm{P}_\partial(E) \to E$. The destructor @ and the equational axioms correspond to the inverse of the induced arrow $\mathrm{P}_\partial(E) \to \Pi^* E$.

4.8.2 Interpretations

Let T be a SOGAT and $\mathcal{C}$ a CwR. We introduce a notion of an *interpretation of T in $\mathcal{C}$*. Since SOGATs are dependently-typed, the definition of an interpretation is not simple. Intuitively, an interpretation φ of T in $\mathcal{C}$ assigns a component of $\mathcal{C}$ to

each symbol of T and is extended to an interpretation $[\![-]\!](\varphi)$ of all well-formed environments, contexts, and expressions over T. Moreover, it should satisfy the following conditions.

- Each sort symbol $S : \Phi \Rightarrow c$ is interpreted as an object $\varphi_S \in \mathcal{C}/[\![\Phi]\!]_\varphi$ where Φ is interpreted as an object of $\mathcal{C}$. If c is type (Prop, prop), then φ_S is a representable map (monomorphism, representable monomorphism, respectively).
- Each term symbol $S : \Phi \Rightarrow K$ is interpreted as a section φ_S of $[\![K]\!](\varphi)$ where K is interpreted as an object of $\mathcal{C}/[\![\Phi]\!](\varphi)$.
- For any axiom $H : \Phi \Rightarrow P$, the arrow $[\![P]\!](\varphi) \to [\![\Phi]\!](\varphi)$ is invertible where P is interpreted as a subobject of $[\![\Phi]\!](\varphi)$.

There are two problems. First, the naive interpretation of types as objects causes a *coherence problem*. To solve the coherence problem, we use the splitting technique of Hofmann [78]. Second, the definition of an interpretation and the extension $[\![-]\!]$ of an interpretation are mutually dependent. To resolve this dependency, we first define an interpretation φ of the underlying symbol signature $T|_{\mathsf{expr}}$, extend it to an interpretation $[\![-]\!](\varphi)$ of all expressions over $T|_{\mathsf{expr}}$, and then define an interpretation of T to be an interpretation of $T|_{\mathsf{expr}}$ satisfying certain well-formedness conditions. Since there can be meaningless expressions over $T|_{\mathsf{expr}}$, we have to define the extended interpretation $[\![e]\!](\varphi)$ as a *partial* interpretation, following Pitts [138], Streicher [160], and *Initiality Project* [86]. We will show a form of soundness: $[\![e]\!](\varphi)$ is defined whenever e is well-formed.

We deal with partiality inside the topos $\mathcal{X} = \mathbf{Fun}(\mathcal{C}^{\mathrm{op}}, \mathbf{Set})$ of presheaves over $\mathcal{C}$. Hofmann's splitting of $\mathcal{C}$ gives rise to a representable map $\mathsf{typeof}_{\mathcal{X}} : \mathsf{term}_{\mathcal{X}} \to \mathsf{Type}_{\mathcal{X}}$ of presheaves over $\mathcal{C}$. We think of $\mathsf{Type}_{\mathcal{X}}$ and $\mathsf{term}_{\mathcal{X}}$ as presheaves of "semantic types" and "semantic terms", respectively. The fundamental idea is to interpret a type expression (term expression) over a variable signature γ as a partial map from $\mathsf{term}^{\gamma}_{\mathcal{X}}$ to $\mathsf{Type}_{\mathcal{X}}$ (to $\mathsf{term}_{\mathcal{X}}$, respectively) of presheaves. Working in the topos of presheaves also allows us to reuse the construction in another topos with sufficient structure. For example, we will use the topos $\mathcal{X}^{\swarrow\searrow}$ of spans in $\mathcal{X}$ to define a notion of an isomorphism between interpretations.

In this section, we assume that the reader is familiar with basic topos theory and the internal language of a topos [106, 119, 92, 93].

4.8.7. Remark. One way of understanding the definition of $\underline{\mathrm{Itpr}}_{\mathcal{X}}(T)$ below is first to translate complex type dependency into higher-order logic [cf. 91, 53] and then to interpret the resulting higher-order theory in the topos $\mathcal{X}$.

The internal language of a topos

We extensively use the internal language of a topos [106, 93]. The *internal language of a topos* $\mathcal{X}$ is the simply-typed lambda calculus in which the types are

the objects in $\mathcal{X}$ and the terms are the maps in $\mathcal{X}$. We write $\mathrm{t} : 1 \to \Omega$ for the subobject classifier and $\mathcal{P}\,A$ for the power object of A, and use the notations

$$\{\mathtt{x} : A \mid P\} = \lambda \mathtt{x}.P$$
$$a \in \alpha = \alpha(a)$$

for terms $\Gamma, \mathtt{x} : A \vdash P : \Omega$, $\Gamma \vdash a : A$, and $\Gamma \vdash \alpha : \mathcal{P}\,A$. The subobject classifier Ω supports logical connectives $\top, \bot, \wedge, \vee, \mathtt{->}$ and quantifiers $\forall, \exists$.

We further extend the language by the *partiality monad*. For an object $A \in \mathcal{X}$, the *partial map classifier* $A_\bot$ is defined to be $\mathrm{P_t}(A)$, where $\mathrm{P_t}$ is the polynomial functor associated with t. In the internal language, $A_\bot$ can be defined to be the subobject of $\Omega \times \mathcal{P}\,A$ named by

$$\{(P, \alpha) : \Omega \times \mathcal{P}\,A \mid (\forall \mathtt{x}_1 \mathtt{x}_2.\mathtt{x}_1 \in \alpha \wedge \mathtt{x}_2 \in \alpha \mathtt{\ ->\ } \mathtt{x}_1 \mathtt{\ ==\ } \mathtt{x}_2) \wedge ((\exists \mathtt{x}.\mathtt{x} \in \alpha) \leftrightarrow P)\}.$$

In other words, an element of $A_\bot$ is a pair (P, α) consisting of a proposition P and a function α from P to A. The assignment $A \mapsto A_\bot$ is part of a strong monad and Ω is an algebra for the monad. We thus introduce the "let" notation [127]. Concretely, given terms $\Gamma \vdash a : A_\bot$ and $\Gamma, \mathtt{x} : A \vdash P : \Omega$, we define a term

$$\Gamma \vdash \mathsf{let}\ \mathtt{x} \Leftarrow a\ \mathsf{in}\ P : \Omega$$

to be

$$\mathsf{proj}_1(a) \wedge \forall \mathtt{x} \in \mathsf{proj}_2(a).P.$$

When $a = (Q, \alpha)$, the proposition $\mathsf{let}\ \mathtt{x} \Leftarrow a\ \mathsf{in}\ P$ is true if Q is true and P is true for the unique element $\mathtt{x} \in \alpha$. We also introduce the notation

$$a\downarrow := \mathsf{let}\ _ \Leftarrow a\ \mathsf{in}\ \top.$$

We write $A \rightharpoonup B$ for the type $A \mathtt{\ ->\ } B_\bot$ of partial maps from A to B. Then, for a partial map $f : A \rightharpoonup B$ and an element $a : A$, the notation $f(a)\downarrow$ means that f is defined at a. We write $\mathrm{dom}\, f$ for the domain of a partial map $f : A \rightharpoonup B$, that is, $\mathrm{dom}\, f = \{\mathtt{x} : A \mid f(\mathtt{x})\downarrow\} : \mathcal{P}\,A$.

Interpretations

Let $\mathcal{C}$ be a CwR and $\mathcal{X}$ denote the topos of presheaves over $\mathcal{C}$.

4.8.8. Construction. The topos $\mathcal{X}$ contains the following objects and maps.

- The *weak representable map classifier* $\mathsf{typeof}_\mathcal{X} : \mathsf{term}_\mathcal{X} \to \mathsf{Type}_\mathcal{X}$ in the sense that any representable map of presheaves over $\mathcal{C}$ is (not uniquely) a pullback of $\mathsf{typeof}_\mathcal{X}$. Concretely, one can define $\mathsf{Type}_\mathcal{X}$ to be the right adjoint splitting [158] of the codomain fibration $\mathcal{C}^{\to} \to \mathcal{C}$.

- The subobject $\mathsf{type}_{\mathcal{X}} \subset \mathsf{Type}_{\mathcal{X}}$ spanned by the representable maps in $\mathcal{C}$, that is, a section $\mathrm{Y}\, x \to \mathsf{Type}_{\mathcal{X}}$ factors through $\mathsf{type}_{\mathcal{X}}$ if and only if the corresponding arrow $y \to x$ in $\mathcal{C}$ is a representable map;

- The subobject $\mathsf{Prop}_{\mathcal{X}} \subset \mathsf{Type}_{\mathcal{X}}$ spanned by the monomorphisms. The pullback of $\mathsf{typeof}_{\mathcal{X}}$ along the inclusion $\mathsf{Prop}_{\mathcal{X}} \to \mathsf{Type}_{\mathcal{X}}$ is a monomorphism, and let $\mathsf{true}_{\mathcal{X}} : \mathsf{Prop}_{\mathcal{X}} \to \Omega$ denote the characteristic map.

- The subobject $\mathsf{prop}_{\mathcal{X}} = \mathsf{type}_{\mathcal{X}} \cap \mathsf{Prop}_{\mathcal{X}} \subset \mathsf{Type}_{\mathcal{X}}$.

- A pullback of the form

$$\begin{array}{ccc} \mathsf{term}_{\mathcal{X}} & \xrightarrow{\mathsf{refl}_{\mathcal{X}}} & \mathsf{term}_{\mathcal{X}} \\ {\scriptstyle\Delta}\downarrow & \lrcorner & \downarrow{\scriptstyle\mathsf{typeof}_{\mathcal{X}}} \\ \mathsf{term}_{\mathcal{X}} \times_{\mathsf{Type}_{\mathcal{X}}} \mathsf{term}_{\mathcal{X}} & \xrightarrow[\mathsf{Id}_{\mathcal{X}}]{} & \mathsf{Type}_{\mathcal{X}} \end{array}$$

  defined by (chosen) finite limits in $\mathcal{C}$. Since the diagonal map is a monomorphism, $\mathsf{Id}_{\mathcal{X}}$ factors through $\mathsf{Prop}_{\mathcal{X}}$.

We view $\mathsf{Type}_{\mathcal{X}}$ ($\mathsf{type}_{\mathcal{X}}$, $\mathsf{Prop}_{\mathcal{X}}$, $\mathsf{prop}_{\mathcal{X}}$, $\mathsf{term}_{\mathcal{X}}$) as a presheaf of "semantic" types (representable types, propositions, representable propositions, terms, respectively). We will interpret expressions as partial functions on semantic terms:

- define $\underline{\mathrm{Itpr}}_{\mathcal{X}}(\gamma) = \mathsf{term}_{\mathcal{X}}^{\gamma}$ for any variable signature γ;

- define $\underline{\mathrm{Expr}}_{\mathcal{X}}(\gamma, c) = \underline{\mathrm{Itpr}}_{\mathcal{X}}(\gamma) \rightharpoonup c_{\mathcal{X}}$ for any variable signature γ and any syntactic class c.

We define a presheaf of interpretations of a symbol (metavariable) signature as follows:

- $\underline{\mathrm{Itpr}}_{\mathcal{X}}(\mu) = \prod_{(\mathtt{X}:\gamma)\in\mu} \underline{\mathrm{Expr}}_{\mathcal{X}}(\gamma, \mathsf{term})$ for any metavariable signature μ;

- $\underline{\mathrm{Itpr}}_{\mathcal{X}}(\Sigma) = \prod_{(S:\mu\Rightarrow c)\in\Sigma} \underline{\mathrm{Itpr}}_{\mathcal{X}}(\mu) \rightharpoonup c_{\mathcal{X}}$ for any symbol signature Σ;

- $\underline{\mathrm{Itpr}}_{\mathcal{X}}(\Sigma, \mu) = \underline{\mathrm{Itpr}}_{\mathcal{X}}(\Sigma) \times \underline{\mathrm{Itpr}}_{\mathcal{X}}(\mu)$.

We also define "semantic" substitutions and instantiations:

- $\underline{\mathrm{Subst}}_{\mathcal{X}}(\gamma, \delta) = \underline{\mathrm{Expr}}_{\mathcal{X}}(\gamma, \mathsf{term})^{\delta}$ for any variable signatures γ and δ;

- $\underline{\mathrm{Inst}}_{\mathcal{X}}(\gamma, \nu) = \prod_{(\mathtt{Y}:\delta)\in\nu} \underline{\mathrm{Expr}}_{\mathcal{X}}(\gamma + \delta, \mathsf{term})$ for any variable signature γ and any metavariable signature ν.

The action of semantic substitutions

$$\underline{\mathrm{Expr}}_{\mathcal{X}}(\delta, c) \times \underline{\mathrm{Subst}}_{\mathcal{X}}(\gamma, \delta) \to \underline{\mathrm{Expr}}_{\mathcal{X}}(\gamma, c)$$

is defined by the composition of partial maps. For the action of semantic instantiations

$$\Big(\underline{\mathrm{Itpr}}_{\mathcal{X}}(\nu) \texttt{ -> } \underline{\mathrm{Expr}}_{\mathcal{X}}(\gamma, c)\Big) \times \underline{\mathrm{Inst}}_{\mathcal{X}}(\gamma', \nu) \to \underline{\mathrm{Expr}}_{\mathcal{X}}(\gamma' + \gamma, c),$$

observe that $\underline{\mathrm{Expr}}_{\mathcal{X}}(\gamma' + \gamma, c) \cong \underline{\mathrm{Itpr}}_{\mathcal{X}}(\gamma')$ `->` $\underline{\mathrm{Expr}}_{\mathcal{X}}(\gamma, c)$. We then have a map

$$\begin{aligned} &\underline{\mathrm{Itpr}}_{\mathcal{X}}(\nu) \texttt{ -> } \underline{\mathrm{Expr}}_{\mathcal{X}}(\gamma, c) \\ \to &\Big(\underline{\mathrm{Itpr}}_{\mathcal{X}}(\gamma') \texttt{ -> } \underline{\mathrm{Itpr}}_{\mathcal{X}}(\nu)\Big) \texttt{ -> } \Big(\underline{\mathrm{Itpr}}_{\mathcal{X}}(\gamma') \texttt{ -> } \underline{\mathrm{Expr}}_{\mathcal{X}}(\gamma, c)\Big) \\ \cong &\underline{\mathrm{Inst}}_{\mathcal{X}}(\gamma', \nu) \texttt{ -> } \underline{\mathrm{Expr}}_{\mathcal{X}}(\gamma' + \gamma, c), \end{aligned}$$

and the function application defines the action of semantic instantiations.

Let Σ be a symbol signature and μ a metavariable signature. Then the family of sets $\mathrm{Hom}(\underline{\mathrm{Itpr}}_{\mathcal{X}}(\Sigma, \mu), \underline{\mathrm{Expr}}_{\mathcal{X}}(\gamma, c))$ indexed over pairs of a variable signature γ and a syntactic class c has the same structure as $\mathrm{Expr}_{\Sigma,\mu}(\gamma, c)$ defined by the following operations:

- $\mathsf{var}_{\mathcal{X}}(\mathtt{x}) : 1 \to \underline{\mathrm{Expr}}_{\mathcal{X}}(\gamma, \mathsf{term})$ for any variable $\mathtt{x} \in \gamma$ defined by the $\mathtt{x}$-th projection $\mathsf{term}_{\mathcal{X}}^{\gamma} \to \mathsf{term}_{\mathcal{X}}$;
- $\mathsf{mvar}_{\mathcal{X}}(\mathtt{X}) : \underline{\mathrm{Itpr}}_{\mathcal{X}}(\mu) \times \underline{\mathrm{Subst}}_{\mathcal{X}}(\gamma, \delta) \to \underline{\mathrm{Expr}}_{\mathcal{X}}(\gamma, \mathsf{term})$ for any metavariable $(\mathtt{X} : \delta) \in \mu$ defined by the $\mathtt{X}$-th projection followed by the action of substitutions;
- $\mathsf{sym}_{\mathcal{X}}(S) : \underline{\mathrm{Itpr}}_{\mathcal{X}}(\Sigma) \times \underline{\mathrm{Inst}}_{\mathcal{X}}(\gamma, \nu) \to \underline{\mathrm{Expr}}_{\mathcal{X}}(\gamma, c)$ for any symbol $(S : \nu \Rightarrow c) \in \Sigma$ defined by the S-th projection followed by the action of instantiations;
- $\underline{\mathrm{Expr}}_{\mathcal{X}}(\gamma, \mathsf{type}) \to \underline{\mathrm{Expr}}_{\mathcal{X}}(\gamma, \mathsf{Type})$ induced by the inclusion $\mathsf{type}_{\mathcal{X}} \to \mathsf{Type}_{\mathcal{X}}$;
- $\underline{\mathrm{Expr}}_{\mathcal{X}}(\gamma, \mathsf{prop}) \to \underline{\mathrm{Expr}}_{\mathcal{X}}(\gamma, \mathsf{Prop})$ induced by the inclusion $\mathsf{prop}_{\mathcal{X}} \to \mathsf{Prop}_{\mathcal{X}}$;
- $\mathsf{eq}_{\mathcal{X}} : \underline{\mathrm{Expr}}_{\mathcal{X}}(\gamma, \mathsf{Type}) \times \underline{\mathrm{Expr}}_{\mathcal{X}}(\gamma, \mathsf{term}) \times \underline{\mathrm{Expr}}_{\mathcal{X}}(\gamma, \mathsf{term}) \to \underline{\mathrm{Expr}}_{\mathcal{X}}(\gamma, \mathsf{Prop})$ induced by the map $\mathsf{Id}_{\mathcal{X}}$.

We then have a unique structure-preserving map

$$[\![-]\!] : \mathrm{Expr}_{\Sigma,\mu}(\gamma, c) \to \mathrm{Hom}\Big(\underline{\mathrm{Itpr}}_{\mathcal{X}}(\Sigma, \mu), \underline{\mathrm{Expr}}_{\mathcal{X}}(\gamma, c)\Big).$$

It is immediate from the definition that $[\![-]\!]$ preserves the action of substitutions and instantiations: a substitution $\Sigma, \mu \vdash f : \gamma' \to \gamma$ is interpreted as a map

$[\![f]\!] : \underline{\mathrm{Itpr}}_{\mathcal{X}}(\Sigma, \mu) \to \underline{\mathrm{Subst}}_{\mathcal{X}}(\gamma', \gamma)$ and $[\![e \cdot f]\!]$ is equal to the action of $[\![f]\!]$ on $[\![e]\!]$; an instantiation $\Sigma, \mu' \vdash I : \gamma' \Rightarrow \mu$ is interpreted as a map $[\![I]\!] : \underline{\mathrm{Itpr}}_{\mathcal{X}}(\Sigma, \mu') \to \underline{\mathrm{Inst}}_{\mathcal{X}}(\gamma', \mu)$ and $[\![e \cdot I]\!]$ is equal to the action of $[\![I]\!]$ on $[\![e]\!]'$ where $[\![e]\!]' : \underline{\mathrm{Itpr}}_{\mathcal{X}}(\Sigma) \to (\underline{\mathrm{Itpr}}_{\mathcal{X}}(\mu)$ `->` $\underline{\mathrm{Expr}}_{\mathcal{X}}(\gamma, c))$ is the transpose of $[\![e]\!]$. As a special case, $[\![-]\!]$ preserves weakening on both variable signatures and metavariable signatures.

We will define $\underline{\mathrm{Itpr}}_{\mathcal{X}}(T)$ as a subobject of $\underline{\mathrm{Itpr}}_{\mathcal{X}}(T|_{\mathsf{expr}})$. For a judgment head $\mathcal{H}$ over Σ, μ, and γ, we define a map

$$(- \models \mathcal{H}) : \underline{\mathrm{Itpr}}_{\mathcal{X}}(\Sigma, \mu) \times \underline{\mathrm{Itpr}}_{\mathcal{X}}(\gamma) \to \Omega$$

by

$$\begin{aligned} ((\varphi, \psi) \models a : K) &= \mathsf{let}\ \mathtt{u} \Leftarrow [\![a]\!](\varphi, \psi), \mathtt{v} \Leftarrow [\![K]\!](\varphi, \psi)\ \mathsf{in}\ \mathsf{typeof}_{\mathcal{X}}(\mathtt{u})\ \texttt{==}\ \mathtt{v} \\ ((\varphi, \psi) \models P) &= \mathsf{let}\ \mathtt{v} \Leftarrow [\![P]\!](\varphi, \psi)\ \mathsf{in}\ \mathsf{true}_{\mathcal{X}}(\mathtt{v}). \end{aligned}$$

For a context Γ over Σ and μ, we define a map

$$[\![\Gamma]\!] : \underline{\mathrm{Itpr}}_{\mathcal{X}}(\Sigma, \mu) \to \mathcal{P}\Big(\underline{\mathrm{Itpr}}_{\mathcal{X}}(\Gamma|_{\mathsf{term}})\Big)$$

by

$$[\![\Gamma]\!](\varphi) = \{\psi \in \underline{\mathrm{Itpr}}_{\mathcal{X}}(\Gamma|_{\mathsf{term}}) \mid (\bigwedge_{(\mathtt{x}:A) \in \Gamma} (\varphi, \psi) \models \mathtt{x} : A) \wedge (\bigwedge_{(H:p) \in \Gamma} (\varphi, \psi) \models p)\}.$$

For a judgment $\Gamma \to \mathcal{H}$, we define a map

$$(- \models \Gamma \to \mathcal{H}) : \underline{\mathrm{Itpr}}_{\mathcal{X}}(\Sigma, \mu) \to \Omega$$

by

$$(\varphi \models \Gamma \to \mathcal{H}) = \forall \psi \in [\![\Gamma]\!](\varphi).(\varphi, \psi) \models \mathcal{H}.$$

We extend the notation $\models$ for a family of judgments by conjunction. For an environment Φ over Σ, we define a map

$$[\![\Phi]\!] : \underline{\mathrm{Itpr}}_{\mathcal{X}}(\Sigma) \to \mathcal{P}\Big(\underline{\mathrm{Itpr}}_{\mathcal{X}}(\Phi|_{\mathsf{term}})\Big)$$

by

$$\begin{aligned} [\![\Phi]\!](\varphi) = \{\psi \in \underline{\mathrm{Itpr}}_{\mathcal{X}}(\Phi|_{\mathsf{term}}) \mid (&\bigwedge_{(\mathtt{X}:\Gamma \to K) \in \Phi} \mathrm{dom}\, \psi_{\mathtt{X}}\ \texttt{==}\ [\![\Gamma]\!](\varphi, \psi) \\ &\wedge (\varphi, \psi) \models \Gamma \to \mathtt{X}(\mathrm{id}_\Gamma) : K) \wedge (\bigwedge_{(H:\Gamma \to P)} (\varphi, \psi) \models \Gamma \to P)\}. \end{aligned}$$

For an environment Φ and a judgment $\Gamma \to \mathcal{H}$ over Φ, we define a map

$$(- \models \Phi, \Gamma \to \mathcal{H}) : \underline{\mathrm{Itpr}}_{\mathcal{X}}(\Sigma) \to \Omega$$

by

$$(\varphi \models \Phi, \Gamma \to \mathcal{H}) = \forall \psi \in [\![\Phi]\!](\varphi).(\varphi, \psi) \models \Gamma \to \mathcal{H}.$$

For a pretheory T, we define a subobject $\underline{\mathrm{Itpr}}_{\mathcal{X}}(T) \subset \underline{\mathrm{Itpr}}_{\mathcal{X}}(T|_{\mathsf{expr}})$ to be named by

$$\{\varphi \in \underline{\mathrm{Itpr}}_{\mathcal{X}}(T|_{\mathsf{expr}}) \mid (\bigwedge_{(S:\Phi \Rightarrow c) \in T} \mathrm{dom}\, \varphi_S == [\![\Phi]\!](\varphi)) \wedge (\bigwedge_{(S:\Phi \Rightarrow K) \in T} \mathrm{dom}\, \varphi_S == [\![\Phi]\!](\varphi) \wedge \varphi \models \Phi, () \to S(\mathrm{id}_\Phi) : K) \wedge (\bigwedge_{(H:\Phi \Rightarrow P) \in T} \varphi \models \Phi, () \to P)\}.$$

By definition, a map $\varphi : A \to \underline{\mathrm{Itpr}}_{\mathcal{X}}(T)$ assigns to each symbol $S : \Phi \Rightarrow c$ of T a partial map φ_S in $\mathcal{X}/A$ represented by a span

$$A^* \underline{\mathrm{Itpr}}_{\mathcal{X}}(\Phi|_{\mathsf{term}}) \longleftarrow \mathrm{dom}\, \varphi_S \xrightarrow{\varphi_S} A^* c_{\mathcal{X}}$$

whose left leg is a monomorphism. The domain $\mathrm{dom}\, \varphi_S$ of this partial map is required to be named by the map $[\![\Phi]\!](\varphi) : A \to \mathcal{P}\Big(\underline{\mathrm{Itpr}}_{\mathcal{X}}(\Phi|_{\mathsf{term}})\Big)$. When S is a term symbol $\Phi \Rightarrow K$, the partial map $[\![K]\!](\varphi) : A \rightharpoonup \mathsf{Type}_{\mathcal{X}}$ is defined on $\mathrm{dom}\, \varphi_S$ by soundness below, and the following diagram commutes.

$$\begin{array}{ccc} & & \mathsf{term}_{\mathcal{X}} \\ & \overset{\varphi_S}{\nearrow} & \big\downarrow {\scriptstyle \mathsf{typeof}_{\mathcal{X}}} \\ \mathrm{dom}\, \varphi_S & \xrightarrow[[\![K]\!](\varphi)]{} & \mathsf{Type}_{\mathcal{X}} \end{array}$$

For any axiom $_ : \Phi \Rightarrow P$ of T, the partial map $[\![P]\!](\varphi) : A \rightharpoonup \mathsf{Prop}_{\mathcal{X}}$ is defined on $[\![\Phi]\!](\varphi)$ and factors through $\mathsf{term}_{\mathcal{X}}$.

$$\begin{array}{ccc} & & \mathsf{term}_{\mathcal{X}} \\ & \cdots\cdots\nearrow & \big\downarrow {\scriptstyle \mathsf{typeof}_{\mathcal{X}}} \\ [\![\Phi]\!](\varphi) & \xrightarrow[[\![P]\!](\varphi)]{} & \mathsf{Prop}_{\mathcal{X}} \end{array}$$

4.8.9. Proposition (Soundness). *If $T, \Phi \vdash \Gamma \to \mathcal{H}$, then the topos $\mathcal{X}$ satisfies*

$$\forall \varphi \in \underline{\mathrm{Itpr}}_{\mathcal{X}}(T).\varphi \models \Phi, \Gamma \to \mathcal{H}.$$

Proof:
By induction on derivation. Since equality $\equiv$ is interpreted as the equality in $\mathcal{X}$, the congruence rules and the conversion rules are trivial. By construction, the following hold in the topos $\mathcal{X}$,

$$\begin{aligned}
&\varphi \models \Phi, \Gamma \to \mathtt{x} : A && ((\mathtt{x} : A) \in \Gamma)\\
&\varphi \models \Phi, \Gamma \to p && ((H : p) \in \Gamma)\\
&\varphi \models \Phi, \Gamma \to \mathtt{X}(\mathrm{id}_\Gamma) : K && ((\mathtt{X} : \Gamma \to K) \in \Phi)\\
&\varphi \models \Phi, \Gamma \to P && ((H : \Gamma \to P) \in \Phi)\\
&\varphi \models \Phi, () \to S(\mathrm{id}_\Phi) : K && ((S : \Phi \Rightarrow K) \in T)\\
&\varphi \models \Phi, () \to P && ((H : \Phi \Rightarrow P) \in T)
\end{aligned}$$

where $\varphi \in \underline{\mathrm{Itpr}}_{\mathcal{X}}(T)$. Therefore, the substitution lemma and the instantiation lemma below complete the proof. □

4.8.10. Lemma (Substitution Lemma). *The topos $\mathcal{X}$ satisfies*

$$\begin{aligned}
\forall \varphi \in \underline{\mathrm{Itpr}}_{\mathcal{X}}(T).\forall \psi \in [\![\Phi]\!](\varphi).((\varphi, \psi) \models \Delta \to \mathcal{H}) \texttt{->} ((\varphi, \psi) \models \Gamma \to f : \Delta)\\
\texttt{->} ((\varphi, \psi) \models \Gamma \to \mathcal{H} \cdot f)
\end{aligned}$$

for any judgment $\Delta \to \mathcal{H}$ over $T|_{\mathsf{expr}}$ and $\Phi|_{\mathsf{term}}$ and any substitution $T|_{\mathsf{expr}}, \Phi|_{\mathsf{term}} \vdash f : \Gamma|_{\mathsf{term}} \to \Delta|_{\mathsf{term}}$.

Proof:
Straightforward. □

4.8.11. Lemma (Instantiation Lemma). *The topos $\mathcal{X}$ satisfies*

$$\begin{aligned}
\forall \varphi \in \underline{\mathrm{Itpr}}_{\mathcal{X}}(T).(\varphi \models \Psi, \Delta \to \mathcal{H}) \texttt{->} (\varphi \models \Phi, \Gamma \to I : \Psi)\\
\texttt{->} (\varphi \models \Phi, \Gamma \,.\, (\Delta \cdot I) \to \mathcal{H} \cdot I)
\end{aligned}$$

for any judgment $\Delta \to \mathcal{H}$ over $T|_{\mathsf{expr}}$ and $\Psi|_{\mathsf{term}}$ and any instantiation $T|_{\mathsf{expr}}, \Phi|_{\mathsf{term}} \vdash I : \Gamma|_{\mathsf{term}} \Rightarrow \Psi|_{\mathsf{term}}$.

Proof:
Straightforward. □

Well-ordered interpretations

When T is a SOGAT, in which symbols and axioms are well-ordered, the presheaf $\underline{\mathrm{Itpr}}_{\mathcal{X}}(T)$ can be described inductively. We begin with interpretations of well-ordered contexts. For a context Γ over T and Φ, let $\underline{\mathrm{Itpr}}_{\mathcal{X}}(T, \Phi)$ denote the subobject of $\underline{\mathrm{Itpr}}_{\mathcal{X}}(T) \times \underline{\mathrm{Itpr}}_{\mathcal{X}}(\Phi|_{\mathsf{term}})$ named by $[\![\Phi]\!]$ and $\underline{\mathrm{Itpr}}_{\mathcal{X}}(T, \Phi, \Gamma)$ the subobject of $\underline{\mathrm{Itpr}}_{\mathcal{X}}(T, \Phi) \times \underline{\mathrm{Itpr}}_{\mathcal{X}}(\Gamma|_{\mathsf{term}})$ named by $[\![\Gamma]\!]$.

4.8.12. Proposition. *For a well-ordered finite context Γ over T and Φ, we have the following description of $\underline{\mathrm{Itpr}}_{\mathcal{X}}(T, \Phi, \Gamma)$.*

1. *When $\Gamma = ()$, we have $\underline{\mathrm{Itpr}}_{\mathcal{X}}(T, \Phi, \Gamma) \cong \underline{\mathrm{Itpr}}_{\mathcal{X}}(T, \Phi)$.*
2. *When $\Gamma = \Gamma' \,.\, (\mathtt{x} : A)$ for a small type $T, \Phi \vdash \Gamma' \to A\ \mathsf{type}$, the partial map $[\![A]\!]$ is defined on $[\![\Gamma']\!]$ by soundness, and we have a pullback*

$$\begin{array}{ccc} \underline{\mathrm{Itpr}}_{\mathcal{X}}(T, \Phi, \Gamma) & \longrightarrow & \mathsf{term}_{\mathcal{X}} \\ \downarrow & \lrcorner & \downarrow{\scriptstyle \mathsf{typeof}_{\mathcal{X}}} \\ \underline{\mathrm{Itpr}}_{\mathcal{X}}(T, \Phi, \Gamma') & \xrightarrow[[\![A]\!]]{} & \mathsf{type}_{\mathcal{X}}. \end{array}$$

3. *When $\Gamma = \Gamma' \,.\, (H : p)$ for a small proposition $T, \Phi \vdash \Gamma' \to p\ \mathsf{prop}$, the partial map $[\![p]\!]$ is defined on $[\![\Gamma']\!]$ by soundness, and we have a pullback*

$$\begin{array}{ccc} \underline{\mathrm{Itpr}}_{\mathcal{X}}(T, \Phi, \Gamma) & \longrightarrow & \mathsf{term}_{\mathcal{X}} \\ \downarrow & \lrcorner & \downarrow{\scriptstyle \mathsf{typeof}_{\mathcal{X}}} \\ \underline{\mathrm{Itpr}}_{\mathcal{X}}(T, \Phi, \Gamma') & \xrightarrow[[\![p]\!]]{} & \mathsf{prop}_{\mathcal{X}}. \end{array}$$

Proof:
Item 1 is trivial. Item 2 is equivalent to that $\mathcal{X}$ satisfies

$$\begin{aligned} \forall \varphi \in \underline{\mathrm{Itpr}}_{\mathcal{X}}(T, \Phi). \forall \psi \in \underline{\mathrm{Itpr}}_{\mathcal{X}}(\Gamma|_{\mathsf{term}}). \psi \in [\![\Gamma]\!](\varphi) \leftrightarrow (\psi_{\Gamma'} \in [\![\Gamma']\!](\varphi) \\ \wedge\ \mathsf{let}\ \mathtt{u} \Leftarrow [\![A]\!](\psi_{\Gamma'})\ \mathsf{in}\ \mathsf{typeof}_{\mathcal{X}}(\psi_{\mathtt{x}}) == \mathtt{u}). \end{aligned} \tag{4.2}$$

By definition, $\psi \in [\![\Gamma]\!](\varphi)$ if and only if $(\varphi, \psi) \models \mathtt{y} : B$ for any variable $(\mathtt{y} : B) \in \Gamma$ and if $(\varphi, \psi) \models p$ for any hypothesis $(H : p) \in \Gamma$. For any variable or hypothesis $(y : e)$ of Γ, the expression e is over Γ' by the well-orderedness of Γ. Since $[\![-]\!]$ preserves weakening, we see that $(\varphi, \psi) \models y : e$ is equivalent to $(\varphi, \psi_{\Gamma'}) \models y : e$ for $(y : e) \in \Gamma'$ and that $(\varphi, \psi) \models \mathtt{x} : A$ is equivalent to $\mathsf{let}\ \mathtt{u} \Leftarrow [\![A]\!](\psi_{\Gamma'})\ \mathsf{in}\ \mathsf{typeof}_{\mathcal{X}}(\psi_{\mathtt{x}})$ == u, from which Eq. (4.2) follows. Item 3 is similarly proved. □

4.8.13. Corollary. *For any well-ordered finite context Γ over $T|_{\mathsf{expr}}$ and $\Phi|_{\mathsf{term}}$, the projection map $\underline{\mathrm{Itpr}}_{\mathcal{X}}(T, \Phi, \Gamma) \to \underline{\mathrm{Itpr}}_{\mathcal{X}}(T, \Phi)$ is representable by representable maps in $\mathcal{C}$ in the sense that, for any section $\varphi : \mathrm{Y}_{\mathcal{C}}\, x \to \underline{\mathrm{Itpr}}_{\mathcal{X}}(T, \Phi)$, the pullback $\varphi^* \underline{\mathrm{Itpr}}_{\mathcal{X}}(T, \Phi, \Gamma)$ is representable by some object $y \in \mathcal{C}$ and the arrow $y \to x$ is a representable map in $\mathcal{C}$.*

Proof:
This is because the map $\mathsf{typeof}_{\mathcal{X}} : \mathsf{term}_{\mathcal{X}} \to \mathsf{type}_{\mathcal{X}}$ has this property. □

4.8.14. Proposition. *For a well-ordered finite environment Φ over T, we have the following description of $\underline{\mathrm{Itpr}}_{\mathcal{X}}(T,\Phi)$.*

1. *When $\Phi = ()$, we have $\underline{\mathrm{Itpr}}_{\mathcal{X}}(T,\Phi) \cong \underline{\mathrm{Itpr}}_{\mathcal{X}}(T)$.*
2. *When $\Phi = \Phi' \,.\, (\mathtt{X} : \Gamma \to K)$ for a type $T, \Phi' \vdash \Gamma \to K$ Type, the partial map $[\![K]\!]$ is defined on $[\![\Gamma]\!]$ by soundness. Let $\widetilde{[\![K]\!]}$ be the pullback*

$$\begin{array}{ccc} \widetilde{[\![K]\!]} & \longrightarrow & \mathsf{term}_{\mathcal{X}} \\ \downarrow & \lrcorner & \downarrow{\scriptstyle \mathsf{typeof}_{\mathcal{X}}} \\ \underline{\mathrm{Itpr}}_{\mathcal{X}}(T,\Phi',\Gamma) & \xrightarrow[[\![K]\!]]{} & \mathsf{Type}_{\mathcal{X}}, \end{array}$$

   and we have an isomorphism

$$\underline{\mathrm{Itpr}}_{\mathcal{X}}(T,\Phi) \cong \textstyle\prod_{\underline{\mathrm{Itpr}}_{\mathcal{X}}(T,\Phi',\Gamma)} \widetilde{[\![K]\!]}$$

   over $\underline{\mathrm{Itpr}}_{\mathcal{X}}(T,\Phi')$.

3. *When $\Phi = \Phi' \,.\, (H : \Gamma \to P)$ for a proposition $T, \Phi' \vdash \Gamma \to P$ Prop, the partial map $[\![P]\!]$ is defined on $[\![\Gamma]\!]$ by soundness. Let $\widetilde{[\![P]\!]}$ be the pullback*

$$\begin{array}{ccc} \widetilde{[\![P]\!]} & \longrightarrow & \mathsf{term}_{\mathcal{X}} \\ \downarrow & \lrcorner & \downarrow{\scriptstyle \mathsf{typeof}_{\mathcal{X}}} \\ \underline{\mathrm{Itpr}}_{\mathcal{X}}(T,\Phi',\Gamma) & \xrightarrow[[\![P]\!]]{} & \mathsf{Prop}_{\mathcal{X}}, \end{array}$$

   and we have an isomorphism

$$\underline{\mathrm{Itpr}}_{\mathcal{X}}(T,\Phi) \cong \textstyle\prod_{\underline{\mathrm{Itpr}}_{\mathcal{X}}(T,\Phi',\Gamma)} \widetilde{[\![P]\!]}$$

   over $\underline{\mathrm{Itpr}}_{\mathcal{X}}(T,\Phi')$.

Proof:
Item 1 is trivial. Item 2 is equivalent to that $\mathcal{X}$ satisfies

$$\begin{aligned} \forall \varphi \in \underline{\mathrm{Itpr}}_{\mathcal{X}}(T). \forall \psi \in \underline{\mathrm{Itpr}}_{\mathcal{X}}(\Phi|_{\mathsf{term}}). \psi \in [\![\Phi]\!](\varphi) \leftrightarrow (\psi_{\Phi'} \in [\![\Phi']\!](\varphi) \\ \wedge \operatorname{dom} \psi_{\mathtt{X}} == [\![\Gamma]\!](\varphi, \psi_{\Phi'}) \wedge (\forall \chi \in [\![\Gamma]\!](\varphi,\psi_{\Phi'}).\ \mathsf{let}\ \mathtt{u} \Leftarrow \psi_{\mathtt{X}}(\chi), \\ \mathtt{v} \Leftarrow [\![K]\!](\varphi,\psi_{\Phi'},\chi)\ \mathsf{in}\ \mathsf{typeof}_{\mathcal{X}}(\mathtt{u}) == \mathtt{v})). \end{aligned}$$

This is easily verified by the definition of $[\![\Phi]\!]$ since $[\![-]\!]$ preserves weakening. Item 3 is similarly proved. □

4.8.15. Corollary. *For any well-ordered finite environment Φ over T, the projection map $\underline{\mathrm{Itpr}}_{\mathcal{X}}(T,\Phi) \to \underline{\mathrm{Itpr}}_{\mathcal{X}}(T)$ is a representable map of presheaves over $\mathcal{C}$.*

Proof:
The map $\mathsf{typeof}_{\mathcal{X}} : \mathsf{term}_{\mathcal{X}} \to \mathsf{Type}_{\mathcal{X}}$ is a representable map of presheaves over $\mathcal{C}$ by definition. Since representable maps in $\mathcal{C}$ is exponentiable, representable maps of presheaves over $\mathcal{C}$ are closed under pushforwards along representable maps in $\mathcal{C}$. Thus, by Corollary 4.8.13, representable maps of presheaves over $\mathcal{C}$ are closed under pushforwards along the projection $\underline{\mathrm{Itpr}}_{\mathcal{X}}(T,\Phi',\Gamma) \to \underline{\mathrm{Itpr}}_{\mathcal{X}}(T,\Phi')$. □

4.8.16. Proposition. *For a SOGAT T, we have the following description of $\underline{\mathrm{Itpr}}_{\mathcal{X}}(T)$.*

1. *When T is the union of a chain $(T_\alpha)_{\alpha<\lambda}$ indexed over a limit ordinal λ, we have an isomorphism*

$$\underline{\mathrm{Itpr}}_{\mathcal{X}}(T) \cong \lim_{\alpha<\lambda} \underline{\mathrm{Itpr}}_{\mathcal{X}}(T_\alpha).$$

   As a special case, we have $\underline{\mathrm{Itpr}}_{\mathcal{X}}(T) \cong 1$ when $T = ()$.

2. *When $T = T'.(S : \Phi \Rightarrow c)$ for a well-formed finite environment Φ over T' and $c \in \{\mathsf{Type}, \mathsf{type}, \mathsf{Prop}, \mathsf{prop}\}$, we have an isomorphism*

$$\underline{\mathrm{Itpr}}_{\mathcal{X}}(T) \cong \prod_{\underline{\mathrm{Itpr}}_{\mathcal{X}}(T',\Phi)} c_{\mathcal{X}}$$

   over $\underline{\mathrm{Itpr}}_{\mathcal{X}}(T')$.

3. *When $T = T' \,.\, (S : \Phi \Rightarrow K)$ for a type $T', \Phi \vdash () \to K\ \mathsf{Type}$, the partial map $[\![K]\!]$ is defined on $[\![()]\!]$ by soundness. Let $\widetilde{[\![K]\!]}$ be the pullback*

$$\begin{array}{ccc} \widetilde{[\![K]\!]} & \longrightarrow & \mathsf{term}_{\mathcal{X}} \\ \downarrow & \lrcorner & \downarrow{\scriptstyle \mathsf{typeof}_{\mathcal{X}}} \\ \underline{\mathrm{Itpr}}_{\mathcal{X}}(T',\Phi) & \xrightarrow[[\![K]\!]]{} & \mathsf{Type}_{\mathcal{X}}, \end{array}$$

   and we have an isomorphism

$$\underline{\mathrm{Itpr}}_{\mathcal{X}}(T) \cong \prod_{\underline{\mathrm{Itpr}}_{\mathcal{X}}(T',\Phi)} \widetilde{[\![K]\!]}$$

   over $\underline{\mathrm{Itpr}}_{\mathcal{X}}(T')$.

4. *When* $T = T' \,.\, (H : \Phi \Rightarrow P)$ *for a proposition* $T', \Phi \vdash () \to P\ \mathsf{Prop}$*, the partial map* $[\![P]\!]$ *is defined on* $[\![()]\!]$ *by soundness. Let* $\widetilde{[\![P]\!]}$ *be the pullback*

$$\begin{array}{ccc} \widetilde{[\![P]\!]} & \longrightarrow & \mathsf{term}_{\mathcal{X}} \\ \downarrow & \lrcorner & \downarrow{\scriptstyle \mathsf{typeof}_{\mathcal{X}}} \\ \underline{\mathrm{Itpr}}_{\mathcal{X}}(T', \Phi) & \xrightarrow[[\![P]\!]]{} & \mathsf{Prop}_{\mathcal{X}}, \end{array}$$

and we have an isomorphism

$$\underline{\mathrm{Itpr}}_{\mathcal{X}}(T) \cong \prod_{\underline{\mathrm{Itpr}}_{\mathcal{X}}(T', \Phi)} \widetilde{[\![P]\!]}$$

over $\underline{\mathrm{Itpr}}_{\mathcal{X}}(T')$.

Proof:
Similar to Proposition 4.8.14. □

Isomorphisms of interpretations

When defining the object $\underline{\mathrm{Itpr}}_{\mathcal{X}}(T)$, we have only used the objects and the maps listed in Construction 4.8.8. Furthermore, we do not need the fact that $\mathsf{typeof}_{\mathcal{X}}$ is the weak representable map classifier for constructing $\underline{\mathrm{Itpr}}_{\mathcal{X}}(T)$. Therefore, the same construction works in an arbitrary topos with sufficient structure. We define a presheaf of isomorphisms of interpretations by applying the construction to the topos of spans $\mathcal{X}^{\swarrow\searrow}$, that is, the category of functors from the category $\{0 \leftarrow 01 \rightarrow 1\}$ to $\mathcal{X}$.

We define a span $(\pi_1, \pi_2) : \mathsf{Type}_{\mathcal{X}^{\swarrow\searrow}} \to \mathsf{Type}_{\mathcal{X}} \times \mathsf{Type}_{\mathcal{X}}$ to be the morphism part of the internal groupoid associated to the map $\mathsf{typeof}_{\mathcal{X}} : \mathsf{term}_{\mathcal{X}} \to \mathsf{Type}_{\mathcal{X}}$, that is, for a pair of maps $(a_1, a_2) : A \to \mathsf{Type}_{\mathcal{X}} \times \mathsf{Type}_{\mathcal{X}}$, a section of $\mathsf{Type}_{\mathcal{X}^{\swarrow\searrow}}$ over (a_1, a_2) corresponds to an isomorphism $a_1^*\mathsf{term}_{\mathcal{X}} \cong a_2^*\mathsf{term}_{\mathcal{X}}$ over A. It thus has the generic isomorphism $\pi_1^*\mathsf{term}_{\mathcal{X}} \cong \pi_2^*\mathsf{term}_{\mathcal{X}}$, and we have a map of spans

$$\begin{array}{ccccc} \mathsf{term}_{\mathcal{X}} & \longleftarrow & \mathsf{term}_{\mathcal{X}^{\swarrow\searrow}} & \longrightarrow & \mathsf{term}_{\mathcal{X}} \\ \downarrow{\scriptstyle \mathsf{typeof}_{\mathcal{X}}} & & \downarrow{\scriptstyle \mathsf{typeof}_{\mathcal{X}^{\swarrow\searrow}}} & & \downarrow{\scriptstyle \mathsf{typeof}_{\mathcal{X}}} \\ \mathsf{Type}_{\mathcal{X}} & \longleftarrow & \mathsf{Type}_{\mathcal{X}^{\swarrow\searrow}} & \longrightarrow & \mathsf{Type}_{\mathcal{X}} \end{array} \tag{4.3}$$

such that both squares are pullbacks. Let $\mathsf{type}_{\mathcal{X}^{\swarrow\searrow}} \subset \mathsf{Type}_{\mathcal{X}^{\swarrow\searrow}}$ be the pullback of $\mathsf{Type}_{\mathcal{X}^{\swarrow\searrow}}$ along the inclusion $\mathsf{type}_{\mathcal{X}} \times \mathsf{type}_{\mathcal{X}} \to \mathsf{Type}_{\mathcal{X}} \times \mathsf{Type}_{\mathcal{X}}$. The objects $\mathsf{Prop}_{\mathcal{X}^{\swarrow\searrow}}$ and $\mathsf{prop}_{\mathcal{X}^{\swarrow\searrow}}$ are constructed in the same way as those in $\mathcal{X}$. One can lift $\mathsf{Id}_{\mathcal{X}}$ and $\mathsf{refl}_{\mathcal{X}}$ to maps of spans.

We define $\underline{\mathrm{Itpr}}_{\mathcal{X}\swarrow\searrow}(-)$ in the same way as $\underline{\mathrm{Itpr}}_{\mathcal{X}}(-)$. Since the projection $\mathcal{X}^{\swarrow\searrow} \to \mathcal{X} \times \mathcal{X}$ preserves all the structures involved with the construction, $\underline{\mathrm{Itpr}}_{\mathcal{X}\swarrow\searrow}(-)$ is a span of the form

$$\underline{\mathrm{Itpr}}_{\mathcal{X}\swarrow\searrow}(-) \to \underline{\mathrm{Itpr}}_{\mathcal{X}}(-) \times \underline{\mathrm{Itpr}}_{\mathcal{X}}(-).$$

For example, we have a span $\underline{\mathrm{Itpr}}_{\mathcal{X}\swarrow\searrow}(T) \to \underline{\mathrm{Itpr}}_{\mathcal{X}}(T) \times \underline{\mathrm{Itpr}}_{\mathcal{X}}(T)$ for any SOGAT T. Sections of $\underline{\mathrm{Itpr}}_{\mathcal{X}\swarrow\searrow}(T)$ are *isomorphisms* of interpretations.

We would like to make the span $\underline{\mathrm{Itpr}}_{\mathcal{X}\swarrow\searrow}(-)$ part of an internal groupoid in $\mathcal{X}$. Since $\underline{\mathrm{Itpr}}_{\mathcal{X}\swarrow\searrow}(\gamma)$ is the product of copies of $\mathsf{term}_{\mathcal{X}\swarrow\searrow}$ for a variable signature γ, it carries an internal groupoid structure. However, we cannot make $\underline{\mathrm{Itpr}}_{\mathcal{X}\swarrow\searrow}(\mu)$ an internal groupoid for a metavariable signature μ. To see this, let $\varphi_1, \varphi_2, \varphi_3 : A \to \underline{\mathrm{Itpr}}_{\mathcal{X}}(\mu)$ be maps, $f_1 : A \to \underline{\mathrm{Itpr}}_{\mathcal{X}\swarrow\searrow}(\mu)$ a section over (φ_1, φ_2), and $f_2 : A \to \underline{\mathrm{Itpr}}_{\mathcal{X}\swarrow\searrow}(\mu)$ a section over (φ_2, φ_3). These data induce partial maps in $\mathcal{X}/A$ fitting into the diagram in Fig. 4.5 for each metavariable $(\mathtt{X} : \gamma) \in \mu$. One would define the composition of f_1 and f_2 by taking the pullback of $(f_1)_{\mathtt{X}}$ and $(f_2)_{\mathtt{X}}$ over $(\varphi_2)_{\mathtt{X}}$ and composing it with the composition operators on $\mathsf{term}_{\mathcal{X}\swarrow\searrow}$ and $\underline{\mathrm{Itpr}}_{\mathcal{X}\swarrow\searrow}(\gamma)$. However, this would not work because the induced map $\mathrm{dom}(f_1)_{\mathtt{X}} \times_{\mathrm{dom}(\varphi_2)_{\mathtt{X}}} \mathrm{dom}(f_2)_{\mathtt{X}} \to A^* \underline{\mathrm{Itpr}}_{\mathcal{X}\swarrow\searrow}(\gamma)$ need not be a monomorphism. A sufficient condition for this map to be a monomorphism is that the vertical maps in the middle column in Fig. 4.5 are all isomorphisms. We define a subobject $\underline{\mathrm{Itpr}}^{\cong}_{\mathcal{X}\swarrow\searrow}(\mu) \subset \underline{\mathrm{Itpr}}_{\mathcal{X}\swarrow\searrow}(\mu)$ in such a way that a map $f : A \to \underline{\mathrm{Itpr}}_{\mathcal{X}\swarrow\searrow}(\mu)$ over (φ, ψ) factors through $\underline{\mathrm{Itpr}}^{\cong}_{\mathcal{X}\swarrow\searrow}(\mu)$ if and only if the maps $\mathrm{dom}\, f_{\mathtt{X}} \to \mathrm{dom}\, \varphi_{\mathtt{X}}$ and $\mathrm{dom}\, f_{\mathtt{X}} \to \mathrm{dom}\, \psi_{\mathtt{X}}$ are isomorphisms for any metavariable $\mathtt{X}$ in μ. Then, if f_1 and f_2 factors through $\underline{\mathrm{Itpr}}^{\cong}_{\mathcal{X}\swarrow\searrow}(\mu)$, then they can be composed and the composition again factors through $\underline{\mathrm{Itpr}}^{\cong}_{\mathcal{X}\swarrow\searrow}(\mu)$. In this way, $\underline{\mathrm{Itpr}}^{\cong}_{\mathcal{X}\swarrow\searrow}(\mu)$ is equipped with an internal groupoid structure.

For the same reason, we cannot define an internal groupoid structure on $\underline{\mathrm{Itpr}}_{\mathcal{X}\swarrow\searrow}(\Sigma)$ for a symbol signature but can define on a subobject $\underline{\mathrm{Itpr}}^{\cong}_{\mathcal{X}\swarrow\searrow}(\Sigma) \subset \underline{\mathrm{Itpr}}^{\mathrm{i}\cong}_{\mathcal{X}\swarrow\searrow}(\Sigma)$ where a map $f : A \to \underline{\mathrm{Itpr}}_{\mathcal{X}\swarrow\searrow}(\Sigma)$ over (φ, ψ) factors through $\underline{\mathrm{Itpr}}^{\mathrm{i}\cong}_{\mathcal{X}\swarrow\searrow}(\Sigma)$ if and only if for any symbol $(S : \mu \Rightarrow c) \in \Sigma$, the domain $\mathrm{dom}\, f_S \hookrightarrow A^* \underline{\mathrm{Itpr}}_{\mathcal{X}\swarrow\searrow}(\mu)$ factors through $A^* \underline{\mathrm{Itpr}}^{\cong}_{\mathcal{X}\swarrow\searrow}(\mu)$ and the maps $\mathrm{dom}\, f_S \to \mathrm{dom}\, \varphi_S$ and $\mathrm{dom}\, f_S \to \mathrm{dom}\, \psi_S$ are isomorphisms.

For the composition operator on $\underline{\mathrm{Itpr}}_{\mathcal{X}\swarrow\searrow}(T)$ for a SOGAT T, observe the following.

4.8.17. Proposition. *Let Φ be a well-ordered finite environment Φ over T and Γ a well-ordered finite context over T and Φ. Then all the squares in the following*

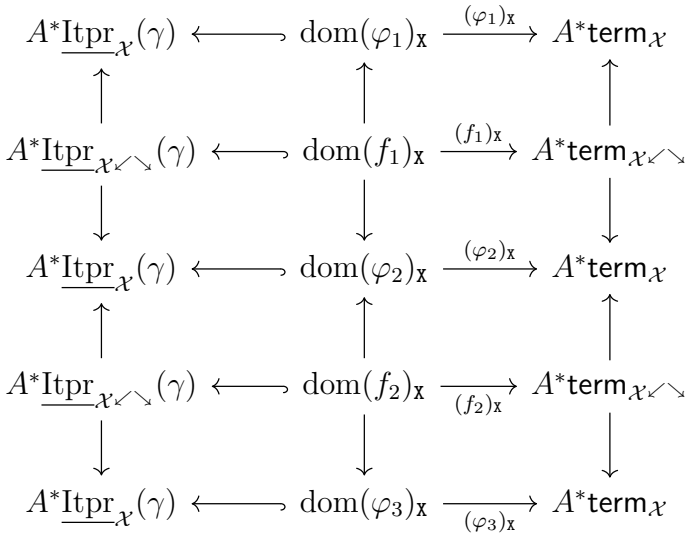


Figure 4.5: Composition of isomorphisms of interpretations

diagram are pullbacks.

$$\begin{array}{ccccc}
\underline{\mathrm{Itpr}}_{\mathcal{X}}(T,\Phi,\Gamma) & \longleftarrow & \underline{\mathrm{Itpr}}_{\mathcal{X}\swarrow\searrow}(T,\Phi,\Gamma) & \longrightarrow & \underline{\mathrm{Itpr}}_{\mathcal{X}}(T,\Phi,\Gamma) \\
\downarrow & & \downarrow & & \downarrow \\
\underline{\mathrm{Itpr}}_{\mathcal{X}}(T,\Phi) & \longleftarrow & \underline{\mathrm{Itpr}}_{\mathcal{X}\swarrow\searrow}(T,\Phi) & \longrightarrow & \underline{\mathrm{Itpr}}_{\mathcal{X}}(T,\Phi) \\
\downarrow & & \downarrow & & \downarrow \\
\underline{\mathrm{Itpr}}_{\mathcal{X}}(T) & \longleftarrow & \underline{\mathrm{Itpr}}_{\mathcal{X}\swarrow\searrow}(T) & \longrightarrow & \underline{\mathrm{Itpr}}_{\mathcal{X}}(T)
\end{array}$$

Proof:
This follows from Propositions 4.8.12 and 4.8.14, since both squares in Eq. (4.3) are pullbacks. □

Proposition 4.8.17 implies that a section $f : A \to \underline{\mathrm{Itpr}}_{\mathcal{X}\swarrow\searrow}(T,\Phi)$ over a map $(\varphi,\psi) : A \to \underline{\mathrm{Itpr}}_{\mathcal{X}}(T,\Phi) \times \underline{\mathrm{Itpr}}_{\mathcal{X}}(T,\Phi)$ induces isomorphisms

$$\varphi^*\underline{\mathrm{Itpr}}_{\mathcal{X}}(T,\Phi,\Gamma) \xleftarrow{\cong} f^*\underline{\mathrm{Itpr}}_{\mathcal{X}\swarrow\searrow}(T,\Phi,\Gamma) \xrightarrow{\cong} \psi^*\underline{\mathrm{Itpr}}_{\mathcal{X}}(T,\Phi,\Gamma).$$

For a metavariable $(\mathtt{X} : \Gamma \to K) \in \Phi$, the presheaf $\varphi^*\underline{\mathrm{Itpr}}_{\mathcal{X}}(T,\Phi,\Gamma)$ is precisely the domain of the partial map $\varphi_{\mathtt{X}}$, and thus we have $\underline{\mathrm{Itpr}}_{\mathcal{X}\swarrow\searrow}(T,\Phi) \subset \underline{\mathrm{Itpr}}_{\mathcal{X}\swarrow\searrow}(T) \times \underline{\mathrm{Itpr}}^{\cong}_{\mathcal{X}\swarrow\searrow}(\Phi|_{\mathsf{term}})$. Similarly, $\underline{\mathrm{Itpr}}_{\mathcal{X}\swarrow\searrow}(T) \subset \underline{\mathrm{Itpr}}^{\cong}_{\mathcal{X}\swarrow\searrow}(T|_{\mathsf{expr}})$. One can see that $\underline{\mathrm{Itpr}}_{\mathcal{X}}(T)$ is closed under the internal groupoid structure.

4.8.3 Functorial semantics

Let T be a SOGAT and $\mathcal{C}$ a CwR. We define a groupoid $\mathrm{Itpr}(T, \mathcal{C})$ of *interpretations of T in $\mathcal{C}$* to be $\mathrm{Hom}\Big(1, \underline{\mathrm{Itpr}}_{\mathcal{X}}(T)\Big)$ where $\mathcal{X}$ is the topos of presheaves over $\mathcal{C}$ and we view $\underline{\mathrm{Itpr}}_{\mathcal{X}}(T)$ as an internal groupoid with the span $\underline{\mathrm{Itpr}}_{\mathcal{X}\swarrow\searrow}(T)$.

The assignment $\Phi \mapsto \underline{\mathrm{Itpr}}_{\mathcal{X}}(T, \Phi)$ is naturally extended to a functor

$$\underline{\mathrm{Itpr}}_{\mathcal{X}}(T, -) : \mathbf{Cl}(T) \to \mathcal{X}/\underline{\mathrm{Itpr}}_{\mathcal{X}}(T).$$

By Corollary 4.8.15, its image is in the class of representable maps over $\underline{\mathrm{Itpr}}_{\mathcal{X}}(T)$. Thus, for any interpretation $\varphi : 1 \to \underline{\mathrm{Itpr}}_{\mathcal{X}}(T)$, the composite of $\underline{\mathrm{Itpr}}_{\mathcal{X}}(T, -)$ and the pullback functor along φ factors through the Yoneda embedding. We refer to the induced functor $\mathbf{Cl}(T) \to \mathcal{C}$ as $\varphi^\star$.

$$\begin{array}{ccc} \mathbf{Cl}(T) & \overset{\varphi^\star}{\dashrightarrow} & \mathcal{C} \\ {\scriptstyle \underline{\mathrm{Itpr}}_{\mathcal{X}}(T,-)}\downarrow & & \downarrow{\scriptstyle \mathrm{Y}_{\mathcal{C}}} \\ \mathcal{X}/\underline{\mathrm{Itpr}}_{\mathcal{X}}(T) & \xrightarrow[\varphi^*]{} & \mathcal{X} \end{array}$$

By Proposition 4.8.17, any isomorphism $\varphi \cong \psi$ of interpretations induces a natural isomorphism between $\varphi^\star$ and $\psi^\star$, making $(-)^\star$ a functor

$$(-)^\star : \mathrm{Itpr}(T, \mathcal{C}) \to \mathrm{k}(\mathbf{Fun}(\mathbf{Cl}(T), \mathcal{C})).$$

By Corollary 4.8.13 and Proposition 4.8.14, we see that $\varphi^\star$ preserves representable maps and pushforwards along representable maps, and thus the functor $(-)^\star$ factors through $\mathbf{CwR}(\mathbf{Cl}(T), \mathcal{C})$.

4.8.18. Theorem. *For any SOGAT T and any CwR $\mathcal{C}$, the functor $(-)^\star$ is an equivalence of groupoids*

$$\mathrm{Itpr}(T, \mathcal{C}) \simeq \mathbf{CwR}(\mathbf{Cl}(T), \mathcal{C}).$$

To give an inverse, let $F : \mathbf{Cl}(T) \to \mathcal{C}$ be a morphism of CwRs. We would like to construct an interpretation $\varphi \in \mathrm{Itpr}(T, \mathcal{C})$ such that $\varphi^\star \cong F$. For a type symbol $(S : \Phi \Rightarrow \mathsf{Type}) \in T$, we have a morphism $F(\Phi \,.\, (\mathtt{X} : () \to S)) \to F(\Phi)$ in $\mathcal{C}$ which corresponds to some map $F(S) : \mathrm{Y}\, F(\Phi) \to \mathsf{Type}_{\mathcal{X}}$ of presheaves. Intuitively, we would define φ_S to be the map $F(S)$, but the formal definition of an interpretation requires that φ_S is a partial map from $\underline{\mathrm{Itpr}}_{\mathcal{X}}(\Phi|_{\mathsf{term}})$ to $\mathsf{Type}_{\mathcal{X}}$. We thus have to make a monomorphism $\mathrm{Y}\, F(\Phi) \hookrightarrow \underline{\mathrm{Itpr}}_{\mathcal{X}}(\Phi|_{\mathsf{term}})$.

We first choose a pullback square

$$\begin{array}{ccc} \mathrm{Y}\, F(\Phi \,.\, (x : () \to S)) & \dashrightarrow & \mathsf{term}_{\mathcal{X}} \\ \downarrow & \lrcorner & \downarrow{\scriptstyle \mathsf{typeof}_{\mathcal{X}}} \\ \mathrm{Y}\, F(\Phi) & \underset{F(S)}{\dashrightarrow} & c_{\mathcal{X}} \end{array}$$

for any sort symbol $(S : \Phi \Rightarrow c) \in T$. Then, for any sort $T, \Phi \vdash \Gamma \to e\ \mathsf{ok}$ where $e = S(I)$ for a sort symbol $S : \Psi \Rightarrow c$ and an instantiation I, we have a map $F(e \cdot \omega) : \mathrm{Y}\, F\big(\Phi \,.\, \Gamma^\dagger\big) \to c_{\mathcal{X}}$ defined by the composite

$$\mathrm{Y}\, F\big(\Phi \,.\, \Gamma^\dagger\big) \xrightarrow{F(I)} \mathrm{Y}\, F(\Psi) \xrightarrow{F(S)} c_{\mathcal{X}} \tag{4.4}$$

and have a pullback

$$\begin{array}{ccc} \mathrm{Y}\, F\big(\Phi \,.\, \Gamma^\dagger \,.\, (x : () \to e \cdot \omega)\big) & \xrightarrow{\pi_x} & \mathsf{term}_{\mathcal{X}} \\ \downarrow & \lrcorner & \downarrow{\scriptstyle \mathsf{typeof}_{\mathcal{X}}} \\ \mathrm{Y}\, F\big(\Phi \,.\, \Gamma^\dagger\big) & \xrightarrow[F(e\cdot\omega)]{} & c_{\mathcal{X}}. \end{array}$$

Let Φ be a well-ordered finite environment over T and Γ a well-ordered finite context over T and Φ. We view $\mathrm{Y}\, F\big(\Phi \,.\, \Gamma^\dagger\big)$ as a subobject of $\mathrm{Y}\, F(\Phi) \times \underline{\mathrm{Itpr}}_{\mathcal{X}}(\Gamma|_{\mathsf{term}})$ by induction on the length of Γ. When $\Gamma = ()$, we have $F\big(\Phi \,.\, \Gamma^\dagger\big) \cong \overline{F(\Phi)}$. When $\Gamma = \Gamma_0 \,.\, (\mathtt{x} : A)$ for a small type $T, \Phi \vdash \Gamma_0 \to A\ \mathsf{type}$, we have the pullback

$$\begin{array}{ccc} \mathrm{Y}\, F\big(\Phi \,.\, \Gamma^\dagger\big) & \xrightarrow{\pi_{\mathtt{x}}} & \mathsf{term}_{\mathcal{X}} \\ \downarrow & \lrcorner & \downarrow{\scriptstyle \mathsf{typeof}_{\mathcal{X}}} \\ \mathrm{Y}\, F\Big(\Phi \,.\, \Gamma_0^\dagger\Big) & \xrightarrow[F(A\cdot\omega)]{} & \mathsf{type}_{\mathcal{X}}. \end{array}$$

Then the projections $\mathrm{Y}\, F(\Phi \,.\, \Gamma^\dagger) \to \mathrm{Y}\, F(\Phi \,.\, \Gamma_0^\dagger) \hookrightarrow F(\Phi) \times \underline{\mathrm{Itpr}}_{\mathcal{X}}(\Gamma_0|_{\mathsf{term}})$ and $\mathrm{Y}\, F(\Phi.\Gamma^\dagger) \to \mathsf{term}_{\mathcal{X}}$ are jointly monic and thus determine a subobject of $\mathrm{Y}\, F(\Phi) \times \underline{\mathrm{Itpr}}_{\mathcal{X}}(\Gamma|_{\mathsf{term}})$. When $\Gamma = \Gamma_0 \,.\, (H : p)$ for a small proposition $T, \Phi \vdash \Gamma_0 \to p\ \mathsf{type}$, we have the composite of monomorphisms $\mathrm{Y}\, F(\Phi \,.\, \Gamma^\dagger) \hookrightarrow \mathrm{Y}\, F(\Phi \,.\, \Gamma_0^\dagger) \hookrightarrow F(\Phi) \times \underline{\mathrm{Itpr}}_{\mathcal{X}}(\Gamma|_{\mathsf{term}})$.

Let Φ be a well-ordered finite environment over T. We view $\mathrm{Y}\, F(\Phi)$ as a subobject of $\underline{\mathrm{Itpr}}_{\mathcal{X}}(\Phi|_{\mathsf{term}})$ by induction the length of Φ. When $\Phi = ()$, we have $F(\Phi) \cong 1$. When $\Phi = \Phi_0 \,.\, (\mathtt{X} : \Gamma \to K)$ for a type $T, \Phi_0 \vdash \Gamma \to K\ \mathsf{Type}$, the object $F(\Phi)$ is the pushforward of $F\big(\Phi_0 \,.\, \Gamma^\dagger \,.\, (\mathtt{x} : K \cdot \omega)\big)$ along the representable map $F\big(\Phi_0 \,.\, \Gamma^\dagger\big) \to F(\Phi_0)$. We have the pullback

$$\begin{array}{ccc} \mathrm{Y}\, F\big(\Phi_0 \,.\, \Gamma^\dagger \,.\, (\mathtt{X} : () \to K \cdot \omega)\big) & \xrightarrow{\pi_{\mathtt{X}}} & \mathsf{term}_{\mathcal{X}} \\ \downarrow & & \downarrow{\scriptstyle \mathsf{typeof}_{\mathcal{X}}} \\ \mathrm{Y}\, F\big(\Phi_0 \,.\, \Gamma^\dagger\big) & \xrightarrow[F(K\cdot\omega)]{} & \mathsf{Type}_{\mathcal{X}} \end{array}$$

and then $F(K \cdot \omega)$ determines a partial map from $\mathrm{Y}\, F(\Phi_0) \times \underline{\mathrm{Itpr}}_{\mathcal{X}}(\Gamma|_{\mathsf{term}})$ to $\mathsf{Type}_{\mathcal{X}}$. Let $F(K) : \mathrm{Y}\, F(\Phi_0) \to (\underline{\mathrm{Itpr}}_{\mathcal{X}}(\Gamma|_{\mathsf{term}}) \rightharpoonup \mathsf{Type}_{\mathcal{X}})$ denote its transpose.

Writing dependent maps as partial maps, we see that $F(\Phi)$ fits into the pullback

$$\begin{array}{ccc} \mathrm{Y}\, F(\Phi) & \longrightarrow & \underline{\mathrm{Itpr}}_{\mathcal{X}}(\Gamma|_{\mathsf{term}}) \rightharpoonup \mathsf{term}_{\mathcal{X}} \\ \downarrow & \lrcorner & \downarrow{\scriptstyle \underline{\mathrm{Itpr}}_{\mathcal{X}}(\Gamma|_{\mathsf{term}}) \rightharpoonup \mathsf{typeof}_{\mathcal{X}}} \\ \mathrm{Y}\, F(\Phi_0) & \xrightarrow[F(K)]{} & \underline{\mathrm{Itpr}}_{\mathcal{X}}(\Gamma|_{\mathsf{term}}) \rightharpoonup \mathsf{Type}_{\mathcal{X}}. \end{array}$$

Then the projections $\mathrm{Y}\, F(\Phi) \to \mathrm{Y}\, F(\Phi_0) \hookrightarrow \underline{\mathrm{Itpr}}_{\mathcal{X}}(\Phi_0|_{\mathsf{term}})$ and $\mathrm{Y}\, F(\Phi) \to (\underline{\mathrm{Itpr}}_{\mathcal{X}}(\Gamma|_{\mathsf{term}}) \rightharpoonup \mathsf{term}_{\mathcal{X}})$ are jointly monic and thus determine a subobject of $\underline{\mathrm{Itpr}}_{\mathcal{X}}(\Phi|_{\mathsf{term}})$. When $\Phi = \Phi_0 \,.\, (H : \Gamma \to P)$ for a proposition $T, \Phi_0 \vdash \Gamma \to P\ \mathsf{Prop}$, we have the composite of monomorphisms $\mathrm{Y}\, F(\Phi) \hookrightarrow \mathrm{Y}\, F(\Phi_0) \hookrightarrow \underline{\mathrm{Itpr}}_{\mathcal{X}}(\Phi|_{\mathsf{term}})$.

We are now ready to define an interpretation $\varphi : 1 \to \underline{\mathrm{Itpr}}_{\mathcal{X}}(T)$. We first define a global section $\varphi : 1 \to \underline{\mathrm{Itpr}}_{\mathcal{X}}(T|_{\mathsf{expr}})$ as follows.

- For a sort symbol $(S : \Phi \Rightarrow c) \in T$, we define φ_S to be the partial map

$$\underline{\mathrm{Itpr}}_{\mathcal{X}}(\Phi|_{\mathsf{term}}) \hookleftarrow \mathrm{Y}\, F(\Phi) \xrightarrow{F(S)} c_{\mathcal{X}}.$$

- For a term symbol $(S : \Phi \Rightarrow K) \in T$, the term $S(\mathrm{id}_\Phi)$ determines a section of the projection $\Phi \,.\, (\mathtt{X} : () \to K) \to \Phi$. We define φ_S to be the partial map

$$\underline{\mathrm{Itpr}}_{\mathcal{X}}(\Phi|_{\mathsf{term}}) \hookleftarrow \mathrm{Y}\, F(\Phi) \xrightarrow{F(S(\mathrm{id}_\Phi))} \mathrm{Y}\, F(\Phi \,.\, (\mathtt{X} : () \to K)) \xrightarrow{\pi_{\mathtt{X}}} \mathsf{term}_{\mathcal{X}}.$$

For a well-formed sort expression $T, \Phi \vdash \Gamma \to e\ c$ or term expression $T, \Phi \vdash \Gamma \to e : K$, we have a partial map $F(e \cdot \omega)$ from $\underline{\mathrm{Itpr}}_{\mathcal{X}}(\Phi|_{\mathsf{term}}, \Gamma|_{\mathsf{term}})$ to $c_{\mathcal{X}}$: the case when e is a sort expression is given by Eq. (4.4); when e is a term expression, we have the partial map

$$\underline{\mathrm{Itpr}}_{\mathcal{X}}(\Phi|_{\mathsf{term}}, \Gamma|) \hookleftarrow \mathrm{Y}\, F\big(\Phi \,.\, \Gamma^\dagger\big) \xrightarrow{F(e\cdot\omega)} \mathrm{Y}\, F\big(\Phi \,.\, \Gamma^\dagger \,.\, (\mathtt{X} : () \to K)\big) \xrightarrow{\pi_{\mathtt{X}}} \mathsf{term}_{\mathcal{X}}.$$

4.8.19. Lemma. *1. For any well-ordered finite environment $T \vdash \Phi\ \mathsf{ok}$, the subobject $\mathrm{Y}\, F(\Phi) \subset \underline{\mathrm{Itpr}}_{\mathcal{X}}(\Phi|_{\mathsf{term}})$ is named by $[\![\Phi]\!](\varphi) : 1 \to \mathcal{P}\Big(\underline{\mathrm{Itpr}}_{\mathcal{X}}(\Phi|_{\mathsf{term}})\Big)$.*

2. For any well-ordered finite context $T, \Phi \vdash \Gamma\ \mathsf{ok}$, the subobject $\mathrm{Y}\, F\big(\Phi \,.\, \Gamma^\dagger\big) \subset \underline{\mathrm{Itpr}}_{\mathcal{X}}(\Phi|_{\mathsf{term}}, \Gamma|_{\mathsf{term}})$ is named by $[\![\Gamma]\!](\varphi, -) : \underline{\mathrm{Itpr}}_{\mathcal{X}}(\Phi|_{\mathsf{term}}) \to \mathcal{P}\Big(\underline{\mathrm{Itpr}}_{\mathcal{X}}(\Gamma|_{\mathsf{term}})\Big)$.

3. For any sort expression $T, \Phi \vdash \Gamma \to e\ c$ or term expression $T, \Phi \vdash \Gamma \to e : K$, the partial map $F(e \cdot \omega)$ from $\underline{\mathrm{Itpr}}_{\mathcal{X}}(\Phi|_{\mathsf{term}}, \Gamma|_{\mathsf{term}})$ to $c_{\mathcal{X}}$ is equal to $[\![e]\!](\varphi, -)$.

Proof:
By induction on the well-orderings on Φ and Γ and the derivation of $\Gamma \to e\ c$ or $\Gamma \to e : K$. For Item 1 use Proposition 4.8.14. For Item 2 use Proposition 4.8.12. □

4.8.20. Lemma. *The global section $\varphi : 1 \to \underline{\mathrm{Itpr}}_{\mathcal{X}}(T|_{\mathsf{expr}})$ factors through $\underline{\mathrm{Itpr}}_{\mathcal{X}}(T)$ and we have $\varphi^\star \cong F$.*

Proof:
Immediate from Lemma 4.8.19. □

Proof of Theorem 4.8.18:
It remains to show that the functor $(-)^\star$ is fully faithful. Let $\varphi, \psi : 1 \to \underline{\mathrm{Itpr}}_{\mathcal{X}}(T)$ be two interpretations and $\sigma : \varphi^\star \cong \psi^\star$ a natural isomorphism. We construct a unique isomorphism $f : \varphi \cong \psi$ such that $f^\star = \sigma$.

Let $(S : \Phi \Rightarrow \mathsf{Type}) \in T$ be a type symbol. φ_S is a partial map from $\underline{\mathrm{Itpr}}_{\mathcal{X}}(\Phi|_{\mathsf{term}})$ to $\mathsf{Type}_{\mathcal{X}}$ and its domain is $\varphi^* \underline{\mathrm{Itpr}}_{\mathcal{X}}(T, \Phi)$. By the definition of $\varphi^\star$, the presheaf $\varphi^* \underline{\mathrm{Itpr}}_{\mathcal{X}}(T, \Phi)$ is representable by $\varphi^\star(\Phi)$, and thus we may regard φ_S as a map $\mathrm{Y}\, \varphi^\star(\Phi) \to \mathsf{Type}_{\mathcal{X}}$. Since $\varphi_S^* \mathsf{term}_{\mathcal{X}} \cong \mathrm{Y}\, \varphi^\star(\Phi\ .\ (\mathtt{X} : () \to S(\mathrm{id}_\Phi)))$ by Proposition 4.8.14, we have isomorphisms

$$\begin{array}{ccc} \varphi_S^* \mathsf{term}_{\mathcal{X}} & \xrightarrow[\cong]{\sigma_S} & \psi_S^* \mathsf{term}_{\mathcal{X}} \\ \downarrow & & \downarrow \\ \mathrm{Y}\, \varphi^\star(\Phi) & \xrightarrow[\sigma_\Phi]{\cong} & \mathrm{Y}\, \psi^\star(\Phi). \end{array}$$

By the definition of $\mathsf{Type}_{\mathcal{X}\swarrow\searrow}$, the isomorphism σ_S corresponds to a map $f_S : \mathrm{Y}\, \varphi^\star(\Phi) \to \mathsf{Type}_{\mathcal{X}\swarrow\searrow}$ such that the following diagram commutes.

$$\begin{array}{ccc} \mathrm{Y}\, \varphi^\star(\Phi) & \xrightarrow{\varphi_S} & \mathsf{Type}_{\mathcal{X}} \\ \| & & \uparrow \\ \mathrm{Y}\, \varphi^\star(\Phi) & \xrightarrow{f_S} & \mathsf{Type}_{\mathcal{X}\swarrow\searrow} \\ {\scriptstyle \sigma_\Phi}\downarrow{\scriptstyle \cong} & & \downarrow \\ \mathrm{Y}\, \psi^\star(\Phi) & \xrightarrow[\psi_S]{} & \mathsf{Type}_{\mathcal{X}} \end{array}$$

This essentially defines the S-component of f, but formally it should be a partial map from $\underline{\mathrm{Itpr}}_{\mathcal{X}\swarrow\searrow}(\Phi|_{\mathsf{term}})$ to $\mathsf{Type}_{\mathcal{X}\swarrow\searrow}$. We thus make a monomorphism $\mathrm{Y}\, \varphi^\star(\Phi) \hookrightarrow \underline{\mathrm{Itpr}}_{\mathcal{X}\swarrow\searrow}(\Phi|_{\mathsf{term}})$ in the same way as the proof of the essential surjectivity. For other sort symbols and term symbols, we can build the components of f in the same way, and it satisfies $f^\star = \sigma$ by construction. The uniqueness of f

is immediate since components of f are determined by components of σ. □

Propositions 4.8.12, 4.8.14 and 4.8.16 and Theorem 4.8.18 give the syntactic CwR $\mathbf{Cl}(T)$ an appropriate universal property.

1. When T is the union of a chain $(T_\alpha)_{\alpha<\lambda}$ indexed over a limit ordinal λ, we have an equivalence

$$\mathbf{CwR}(\mathbf{Cl}(T), \mathcal{C}) \simeq \lim_{\alpha<\lambda} \mathbf{CwR}(\mathbf{Cl}(T_\alpha), \mathcal{C}).$$

   Hence, $\mathbf{Cl}(T)$ is the colimit of $\mathbf{Cl}(T_\alpha)$'s. In the case when $T = ()$, we have $\mathbf{CwR}(\mathbf{Cl}(T), \mathcal{C}) \simeq 1$ and $\mathbf{Cl}(T)$ is the initial CwR.

2. When $T = T'.(S : \Phi \Rightarrow \mathsf{Type})$, we had $\underline{\mathrm{Itpr}}_{\mathcal{X}}(T) \cong \prod_{\underline{\mathrm{Itpr}}_{\mathcal{X}}(T',\Phi)} \mathsf{Type}_{\mathcal{X}}$. Then a global section of $\underline{\mathrm{Itpr}}_{\mathcal{X}}(T)$ corresponds to a pair consisting of a global section $\varphi : 1 \to \underline{\mathrm{Itpr}}_{\mathcal{X}}(T')$ and a section $\varphi_S : \varphi^\star(\Phi) \to \mathsf{Type}_{\mathcal{X}}$. Since $\mathsf{Type}_{\mathcal{X}}$ is a weak representable map classifier, the section φ_S corresponds to an object of $\mathcal{C}/\varphi^\star(\Phi)$. Therefore, we have a pullback

$$\begin{array}{ccc} \mathbf{CwR}(\mathbf{Cl}(T), \mathcal{C}) & \longrightarrow & \mathrm{k}(\mathcal{C}^{\to}) \\ \downarrow & \lrcorner & \downarrow \\ \mathbf{CwR}(\mathbf{Cl}(T'), \mathcal{C}) & \xrightarrow[\mathrm{ev}_\Phi]{} & \mathrm{k}(\mathcal{C}). \end{array}$$

   Hence, $\mathbf{Cl}(T)$ is obtained from $\mathbf{Cl}(T')$ by freely adjoining an object over Φ.

3. Similarly, when $T = T' \,.\, (S : \Phi \Rightarrow \mathsf{type})$, we have a pullback

$$\begin{array}{ccc} \mathbf{CwR}(\mathbf{Cl}(T), \mathcal{C}) & \longrightarrow & \mathrm{k}(\mathrm{R}_{\mathcal{C}}) \\ \downarrow & \lrcorner & \downarrow \\ \mathbf{CwR}(\mathbf{Cl}(T'), \mathcal{C}) & \xrightarrow[\mathrm{ev}_\Phi]{} & \mathrm{k}(\mathcal{C}) \end{array}$$

   where $\mathrm{R}_{\mathcal{C}} \subset \mathcal{C}^{\to}$ is the full subcategory spanned by the representable maps. Hence $\mathbf{Cl}(T)$ is obtained from $\mathbf{Cl}(T')$ by freely adjoining a representable map over Φ.

4. Similarly, when $T = T' \,.\, (S : \Phi \Rightarrow \mathsf{Prop})$, we have a pullback

$$\begin{array}{ccc} \mathbf{CwR}(\mathbf{Cl}(T), \mathcal{C}) & \longrightarrow & \mathrm{k}(\mathbf{Sub}_{\mathcal{C}}) \\ \downarrow & \lrcorner & \downarrow \\ \mathbf{CwR}(\mathbf{Cl}(T'), \mathcal{C}) & \xrightarrow[\mathrm{ev}_\Phi]{} & \mathrm{k}(\mathcal{C}) \end{array}$$

   where $\mathbf{Sub}_{\mathcal{C}} \subset \mathcal{C}^{\to}$ is the full subcategory spanned by the monomorphisms. Hence, $\mathbf{Cl}(T)$ is obtained from $\mathbf{Cl}(T')$ by freely adjoining a subobject of Φ.

5. Similarly, when $T = T' \,.\, (S : \Phi \Rightarrow \mathsf{prop})$, we have a pullback

$$\begin{array}{ccc} \mathbf{CwR}(\mathbf{Cl}(T), \mathcal{C}) & \longrightarrow & \mathrm{k}\big(\mathbf{Sub}_{\mathcal{C}}^{\mathrm{R}}\big) \\ \downarrow & \lrcorner & \downarrow \\ \mathbf{CwR}(\mathbf{Cl}(T'), \mathcal{C}) & \xrightarrow[\mathrm{ev}_{\Phi}]{} & \mathrm{k}(\mathcal{C}) \end{array}$$

where $\mathbf{Sub}_{\mathcal{C}}^{\mathrm{R}} \subset \mathbf{Sub}_{\mathcal{C}}$ is the full subcategory spanned by the representable monomorphisms. Hence, $\mathbf{Cl}(T)$ is obtained from $\mathbf{Cl}(T')$ by freely adjoining a representable monomorphism over Φ.

6. When $T = T' \,.\, (S : \Phi \Rightarrow K)$, we had $\underline{\mathrm{Itpr}}_{\mathcal{X}}(T) \cong \prod_{\underline{\mathrm{Itpr}}_{\mathcal{X}}(T',\Phi)} \widetilde{[\![K]\!]}$ where $\widetilde{[\![K]\!]}$ is the pullback of $\mathsf{term}_{\mathcal{X}}$ along $[\![K]\!] : \underline{\mathrm{Itpr}}_{\mathcal{X}}(T', \Phi) \to \mathsf{Type}_{\mathcal{X}}$. Then a global section $1 \to \underline{\mathrm{Itpr}}_{\mathcal{X}}(T)$ corresponds to a pair consisting of a global section $\varphi : 1 \to \underline{\mathrm{Itpr}}_{\mathcal{X}}(T')$ and a section φ_S of $\varphi^\star(\Phi \,.\, (\mathtt{X} : () \to K)) \to \varphi^\star(\Phi)$. Therefore, we have a pullback

$$\begin{array}{ccc} \mathbf{CwR}(\mathbf{Cl}(T), \mathcal{C}) & \longrightarrow & \mathrm{k}\big(\mathcal{C}^{\underline{\Delta}}\big) \\ \downarrow & \lrcorner & \downarrow \\ \mathbf{CwR}(\mathbf{Cl}(T'), \mathcal{C}) & \xrightarrow[\mathrm{ev}_{\Phi.(\mathtt{X}:() \to K) \to \Phi}]{} & \mathrm{k}(\mathcal{C}^{\to}) \end{array}$$

where $\underline{\Delta}$ denotes the category $\begin{array}{ccc} & & 1 \\ & \nearrow & \downarrow \\ 0 & = & 0 \end{array}$ and thus $\mathcal{C}^{\underline{\Delta}}$ is the category of sections. Hence, $\mathbf{Cl}(T)$ is obtained from $\mathbf{Cl}(T')$ by freely adjoining a section of $\Phi \,.\, (\mathtt{X} : K) \to \Phi$.

7. Similarly, when $T = T' \,.\, (H : \Phi \Rightarrow P)$, we have a pullback

$$\begin{array}{ccc} \mathbf{CwR}(\mathbf{Cl}(T), \mathcal{C}) & \longrightarrow & \mathrm{k}(\mathcal{C}^{\cong}) \\ \downarrow & \lrcorner & \downarrow \\ \mathbf{CwR}(\mathbf{Cl}(T'), \mathcal{C}) & \xrightarrow[\mathrm{ev}_{\Phi.(H:() \to P) \to \Phi}]{} & \mathrm{k}(\mathbf{Sub}_{\mathcal{C}}). \end{array}$$

Hence, $\mathbf{Cl}(T)$ is obtained from $\mathbf{Cl}(T')$ by freely inverting $\Phi \,.\, (H : () \to P) \to \Phi$.

4.8.4 The internal SOGAT of a CwR

Given a CwR $\mathcal{C}$, we construct a SOGAT $\mathrm{L}(\mathcal{C})$ such that $\mathbf{Cl}(\mathrm{L}(\mathcal{C})) \simeq \mathcal{C}$. We call $\mathrm{L}(\mathcal{C})$ the *internal SOGAT of* $\mathcal{C}$.

We prepare some constructions of SOGATs.

4.8.21. Construction. Let T be a SOGAT.

1. For an object $\Phi \in \mathbf{Cl}(T)$, a well-ordered finite context over T and Φ and a type $T, \Phi \vdash \Gamma \to K$ ok, the SOGAT $T\,.\,\big(S : \Phi\,.\,\Gamma^\dagger \Rightarrow K \cdot \omega_\Gamma\big)$ is the one obtained from T by freely adjoining a section of the projection $\Phi\,.\,(\mathtt{X} : \Gamma \to K) \to \Phi$.

2. For a proposition $T, \Phi \vdash \Gamma \to P$ ok, the *SOGAT* $T\,.\big(H : \Phi\,.\,\Gamma^\dagger \Rightarrow P \cdot \omega_\Gamma\big)$ is the one obtained from T by freely inverting the projection $\Phi.(H : \Gamma \to P) \to \Phi$.

3. Repeating Items 1 and 2, we have the SOGAT obtained from T by freely adjoining a section of the projection $\Phi\,.\,\Psi \to \Phi$ for an arbitrary well-ordered finite relative environment Ψ over Φ. As a special case, we have the one obtained from T by freely adjoining a morphism $\Phi \to \Psi$ between arbitrary objects of $\mathbf{Cl}(T)$.

4. For two morphisms $I_1, I_2 : \Phi \to \Psi$ in $\mathbf{Cl}(T)$, we have a SOGAT $T\,.\,(I_1 \equiv I_2)$ by adjoining an axiom $\Phi\,.\,(\Gamma \cdot I)^\dagger \Rightarrow I_1(\mathtt{X})\cdot\omega_{\Gamma\cdot I_1} \equiv I_2(\mathtt{X})\cdot\omega_{\Gamma\cdot I_1} : (K \cdot I_1)\cdot\omega_{\Gamma\cdot I_1}$ for each metavariable $(\mathtt{X} : \Gamma \to K) \in \Psi$. The SOGAT $T\,.\,(I_1 \equiv I_2)$ is thus the one obtained from T by freely equalizing I_1 and I_2.

5. For a morphism $I : \Phi \to \Psi$ in $\mathbf{Cl}(T)$, we have the SOGAT obtained from T by freely inverting I, that is, by freely adjoining a morphism $I^{-1} : \Psi \to \Phi$ and equalizing $I^{-1} \circ I \equiv \mathrm{id}$ and $I \circ I^{-1} \equiv \mathrm{id}$.

6. For a morphism $I : \Phi \to \Psi$ in $\mathbf{Cl}(T)$, we have the SOGAT obtained from T by freely making I a representable map, that is, by adding a small type symbol $S_1 : \Psi \Rightarrow \mathsf{type}$ and a term symbol $S_2 : \Phi \Rightarrow S_1(I)$ and then freely inverting the morphism $I\,.\,(\mathtt{X} := S_2(\mathrm{id}_\Phi)) : \Phi \to \Psi\,.\,(\mathtt{X} : () \to S_1(\mathrm{id}_\Psi))$.

We define $\mathrm{L}(\mathcal{C})$ in the following steps.

1. Begin with the empty SOGAT.

2. Add a type symbol $x : () \Rightarrow \mathsf{Type}$ for any object $x \in \mathcal{C}$.

3. Add a morphism $u : (\mathtt{X} : () \to x) \to (\mathtt{Y} : () \to y)$ for any arrow $u : x \to y$ in $\mathcal{C}$.

4. Equalize two morphisms $(\mathtt{X} := \mathrm{id}_x(\mathtt{X})), (\mathtt{X} := \mathtt{X}) : (\mathtt{X} : () \to x) \to (\mathtt{X} : () \to x)$ for any object $x \in \mathcal{C}$.

5. Equalize two morphisms $(\mathtt{X}_3 := (v \circ u)(\mathtt{X}_1)), (\mathtt{X}_3 := v(u(\mathtt{X}_1))) : (\mathtt{X}_1 : () \to x_1) \to (\mathtt{X}_3 : () \to x_3)$ for any arrows $u : x_1 \to x_2$ and $v : x_2 \to x_3$ in $\mathcal{C}$.

6. Invert the canonical morphism $(\mathtt{X} : () \to \lim_{\xi\in\Xi} x_\xi) \to \lim_{\xi\in\Xi}(\mathtt{X}_\xi : () \to x_\xi)$ for any finite diagram $x : \Xi \to \mathcal{C}$.

7. Make the morphism $u : (\mathtt{Y} : () \to y) \to (\mathtt{X} : () \to x)$ representable for any representable map $u : y \to x$ in $\mathcal{C}$.

8. Invert the canonical morphism $(\mathtt{Z} : () \to u_* z) \to u_*(\mathtt{Z} : () \to z)$ for any arrow $v : z \to y$ and any representable map $u : y \to x$ in $\mathcal{C}$.

By construction, an interpretation of $\mathrm{L}(\mathcal{C})$ in a CwR $\mathcal{D}$ is nothing but a morphism of CwRs $\mathcal{C} \to \mathcal{D}$. Thus, we have a natural equivalence

$$\mathbf{CwR}(\mathbf{Cl}(\mathrm{L}(\mathcal{C})), \mathcal{D}) \simeq \mathbf{CwR}(\mathcal{C}, \mathcal{D})$$

and then

$$\mathbf{Cl}(\mathrm{L}(\mathcal{C})) \simeq \mathcal{C}.$$

We have proved that every SOGAT generates the syntactic CwR with an appropriate universal property and that every CwR is the syntactic CwR of some SOGAT. We expect that the construction of syntactic CwRs is part of an equivalence of $(2,1)$-categories. To make it precise, we would have to define morphisms and 2-morphisms of SOGATs and prove the fully faithfulness. Since we mainly work with CwRs in this thesis and use SOGATs only for presenting some concrete CwRs, further study of SOGATs is left as future work.

Chapter 5

The theory of type theories

In this chapter, we establish basic results in the semantics of type theory based on our definition of a type theory (Definition 3.2.3). The *initiality theorem/conjecture* asserts that a type theory has an initial model constructed out of the syntax of the type theory. We give an explicit construction of an initial model of a type theory and see that the construction coincides with the traditional syntactic construction when the type theory is presented by a SOGAT. The initial model construction is extended to the construction of *syntactic models* generated by certain data. To make it precise, we introduce a notion of a *theory over a type theory*, which intuitively consists of constants and axioms, and construct the syntactic model generated by a theory. The syntactic model construction has a right adjoint which assigns to each model a theory called the *internal language* of the model. One might expect that the adjunction between theories and models is an equivalence, but this is not the case. This is because objects in the base category of a syntactic model look like *contexts* while in general, a model of a type theory can contain a lot of junk objects in the base category. We say a model of a type theory is *democratic* when all the objects in the base category look like contexts and show that the syntactic model construction induces an equivalence between theories and democratic models.

In Section 5.1, we formulate the main result, the equivalence of theories and democratic models. This result is proved in Section 5.4 after developing technical lemmas on the $(2,1)$-categories of type theories (Section 5.3) and of models of a type theory (Section 5.2).

5.0.1. Remark. The proof of the main theorem in this thesis is different from the proof in the earlier paper [169]. In the earlier paper, the author explicitly constructed the left adjoint, but in this thesis, we use the *adjoint functor theorem* for presentable categories to ensure the existence of the left adjoint. The concrete description of the initial model (Section 5.4.1) is still relevant in the proof in this thesis to analyze the adjunction, but we do not need the concrete description of a general syntactic model. The current version of the proof of the equivalence

of theories and democratic models was obtained through the development of ∞-type theories (Chapter 6), which is joint work with Hoang Kim Nguyen, and is designed to work also in the $(\infty, 1)$-categorical context.

5.1 Theories and models

A *theory over a type theory* is roughly a set of constants and axioms to be adjoined to the type theory. Proposition 4.8.2 justifies the following definition.

5.1.1. Definition. Let $\mathcal{T}$ be a type theory. A *theory over* $\mathcal{T}$ or *$\mathcal{T}$-theory* is a left exact functor $\mathcal{T} \to \mathbf{Set}$. We write $\mathbf{Th}(\mathcal{T})$ for the category of $\mathcal{T}$-theories, that is, the full subcategory of $\mathbf{Fun}(\mathcal{T}, \mathbf{Set})$ spanned by the left exact functors.

Models of a type theory and morphisms of models are defined in Definitions 3.2.4 and 3.2.5. Models of a type theory form a $(2,1)$-category.

5.1.2. Definition. Let $F, G : \mathcal{M} \to \mathcal{N}$ be morphisms of models of a type theory $\mathcal{T}$. A 2-*morphism* $\sigma : F \Rightarrow G$ consists of the following data:

- a natural isomorphism $\sigma_\diamond : F_\diamond \cong G_\diamond : \mathcal{M}(\diamond) \to \mathcal{N}(\diamond)$;
- for any object $x \in \mathcal{T}$, a natural isomorphism $\sigma_x : F_x \cong G_x : \mathcal{M}(x) \to \mathcal{N}(x)$ over $\sigma_\diamond$.

We write $\mathbf{Mod}(\mathcal{T})$ for the $(2,1)$-category of models of $\mathcal{T}$, morphisms of models, and 2-morphisms.

Let $\mathcal{T}$ be a type theory and $\mathcal{M}$ a model of $\mathcal{T}$. Taking the fiber over the final object of the base category $\mathcal{M}(\diamond)$, we have a $\mathcal{T}$-theory

$$\mathcal{T} \xrightarrow{\mathcal{M}} \mathbf{DFib}_{\mathcal{M}(\diamond)} \xrightarrow{A \mapsto A_1} \mathbf{Set}$$

which we call the *internal language of* $\mathcal{M}$ and is denoted by $\mathrm{L}(\mathcal{M})$. The assignment $\mathcal{M} \mapsto \mathrm{L}(\mathcal{M})$ is extended to a functor $\mathrm{L} : \mathbf{Mod}(\mathcal{T}) \to \mathbf{Th}(\mathcal{T})$.

The goal of this chapter is to show that the internal language functor induces an equivalence between theories and *democratic* models. In Section 5.1.1, we concretely describe $\mathrm{L}(\mathcal{M})$ when $\mathcal{T}$ is presented by a certain SOGAT to justify calling $\mathrm{L}(\mathcal{M})$ the internal language of $\mathcal{M}$. We introduce democratic models in Section 5.1.2. In Section 5.1.3, we state the main theorem which will be proved in Section 5.4.

5.1.1 The internal language at work

We concretely describe the internal languages of models of some type theories presented by SOGATs. Let T_0 denote DTT which was defined to be the SOGAT consisting of the following two symbols (Example 4.6.1).

$$U : () \Rightarrow \mathsf{Type}$$
$$E : (\mathtt{A} : () \to U) \Rightarrow \mathsf{type}$$

We also recall that the application of E is omitted. Let $\mathcal{T}_0 = \mathbf{Cl}(T_0)$ be the syntactic CwR. The universal property of $\mathcal{T}_0$ is that it is the free CwR generated by a representable map $\partial : E \to U$. Concretely, the object U is represented by the environment $(\mathtt{A} : () \to U)$, the object E is represented by the environment $(\mathtt{A} : () \to U, \mathtt{a} : () \to \mathtt{A})$, and the representable map ∂ is represented by the instantiation $(\mathtt{A} := \mathtt{A})$.

We first describe components of a T_0-theory. Let $\Phi \in \mathbf{Th}(T_0) \simeq \mathbf{Lex}(\mathcal{T}_0, \mathbf{Set})$ be a T_0-theory. Let $\mathrm{Y} : \mathcal{T}_0^{\mathrm{op}} \to \mathbf{Lex}(\mathcal{T}_0, \mathbf{Set})$ denote the Yoneda embedding, so the component $\Phi(x)$ at an object $x \in \mathcal{T}_0$ is isomorphic to the set of morphisms of T_0-theories $\mathrm{Y}\, x \to \Phi$. By the definitions of U and E, a morphism of T_0-theories $A : \mathrm{Y}\, U \to \Phi$ corresponds to a closed type $T_0, \Phi \vdash () \to A : U$ and an extension $a : \mathrm{Y}\, E \to \Phi$ of A along $\mathrm{Y}\, \partial : \mathrm{Y}\, U \to \mathrm{Y}\, E$ corresponds to a closed term $T_0, \Phi \vdash () \to a : A$. For open types and terms, observe that the object $\mathrm{P}^n_\partial(1) \in \mathcal{T}_0$ is represented by the environment

$$\begin{aligned}
&\mathtt{A}_1 : () \to U \\
&\mathtt{A}_2 : (\mathtt{x}_1 : \mathtt{A}_1) \to U \\
&\quad\vdots \\
&\mathtt{A}_n : (\mathtt{x}_1 : \mathtt{A}_1, \mathtt{x}_2 : \mathtt{A}_2(\mathtt{x}_1), \ldots, \mathtt{x}_{n-1} : \mathtt{A}_{n-1}(\mathtt{x}_1, \ldots, \mathtt{x}_{n-2})) \to U
\end{aligned}$$

by the construction of the pushforward along ∂ in $\mathcal{T}_0 = \mathbf{Cl}(T_0)$. Then a morphism $\Gamma : \mathrm{Y}(\mathrm{P}^n_\partial(1)) \to \Phi$ corresponds to a context $T_0, \Phi \vdash \Gamma\ \mathsf{ok}$ of length n, an extension $A : \mathrm{Y}(\mathrm{P}^n_\partial(U)) \to \Phi$ of Γ along the morphism $\mathrm{Y}(\mathrm{P}^n_\partial(1)) \to \mathrm{Y}(\mathrm{P}^n_\partial(U))$ corresponds to a type $T_0, \Phi \vdash \Gamma \to A : U$, and an extension $a : \mathrm{Y}(\mathrm{P}^n_\partial(E)) \to \Phi$ of A along $\mathrm{Y}(\mathrm{P}^n_\partial(\partial))$ corresponds to a term $T_0, \Phi \vdash \Gamma \to a : A$. We also note that for two morphisms $A_1, A_2 : \mathrm{Y}(\mathrm{P}^n_\partial(U)) \to \Phi$ over Γ (two morphisms $a_1, a_2 : \mathrm{Y}(\mathrm{P}^n_\partial(E)) \to \Phi$ over A), the following are equivalent:

1. A_1 and A_2 (a_1 and a_2, respectively) are equal;
2. the induced morphism $(A_1, A_2) : \mathrm{Y}(\mathrm{P}^n_\partial(U \times U)) \to \Phi$ (morphism $(a_1, a_2) : \mathrm{Y}(\mathrm{P}^n_\partial(E \times_U E)) \to \Phi$, respectively) factors through the codiagonal morphism $\mathrm{Y}(\mathrm{P}^n_\partial(U \times U)) \to \mathrm{Y}(\mathrm{P}^n_\partial(U))$ (morphism $\mathrm{Y}(\mathrm{P}^n_\partial(E \times_U E)) \to \mathrm{Y}(\mathrm{P}^n_\partial(E))$, respectively);

3. we derive $T_0, \Phi \vdash \Gamma \to A_1 \equiv A_2 : U$ (derive $T_0, \Phi \vdash \Gamma \to a_1 \equiv a_2 : A$, respectively).

Now let $\mathcal{M}$ be a model of $\mathcal{T}_0$ and consider the internal language $\mathrm{L}(\mathcal{M})$. By definition, a morphism $\mathrm{Y}\, x \to \mathrm{L}(\mathcal{M})$ corresponds to an element of the fiber $\mathcal{M}(x)_1$ over the final object which corresponds to a global section $\mathcal{M}(\diamond) \simeq \mathcal{M}(\diamond)/1 \to \mathcal{M}(x)$ of discrete fibrations over $\mathcal{M}(\diamond)$ by Yoneda. Then a context $T_0, \mathrm{L}(\mathcal{M}) \vdash \Gamma\ \mathsf{ok}$ of length n corresponds to a global section $\Gamma : \mathcal{M}(\diamond) \to \mathcal{M}(\mathrm{P}_\partial^n(1))$. Since $\mathcal{M}$ preserves the polynomial functor P_∂, such a global section corresponds to a finite sequence $(A_1, \ldots, A_n)$ where

$$\begin{aligned} A_1 &: \mathcal{M}(\diamond) \to \mathcal{M}(U) \\ A_2 &: \mathcal{M}(\diamond)/\{A_1\} \to \mathcal{M}(U) \\ &\vdots \\ A_n &: \mathcal{M}(\diamond)/\{A_{n-1}\} \to \mathcal{M}(U) \end{aligned}$$

are maps of discrete fibrations over $\mathcal{M}(\diamond)$ and $\{-\}$ denotes the context comprehension with respect to $\mathcal{M}(\partial)$. In particular, we have an object $\{A_n\}$. Therefore, ignoring the length, a context over $\mathrm{L}(\mathcal{M})$ corresponds to an object of the base category $\mathcal{M}(\diamond)$ obtained by context comprehension. Under this identification, a type $T_0, \mathrm{L}(\mathcal{M}) \vdash \Gamma \to A : U$ corresponds to a section $A : \mathcal{M}(\diamond)/\Gamma \to \mathcal{M}(U)$ and a term $T_0, \mathrm{L}(\mathcal{M}) \vdash \Gamma \to a : A$ corresponds to a section $a : \mathcal{M}(\diamond)/\Gamma \to \mathcal{M}(E)$ over A. Moreover, we derive $T_0, \mathrm{L}(\mathcal{M}) \vdash \Gamma \to A_1 \equiv A_2 : U$ (derive $T_0, \mathrm{L}(\mathcal{M}) \vdash \Gamma \to a_1 \equiv a_2 : A$) if and only if the corresponding sections $A_1, A_2 : \mathcal{M}(\diamond)/\Gamma \to \mathcal{M}(U)$ (sections $a_1, a_2 : \mathcal{M}(\diamond)/\Gamma \to \mathcal{M}(E)$, respectively) are equal. Hence, the internal language $\mathrm{L}(\mathcal{M})$ provides $\mathcal{M}$ with a syntactic way of constructing sections of $\mathcal{M}(U)$ and $\mathcal{M}(E)$ and proving equalities between them.

The internal language becomes more powerful when we extend T_0 by various type constructors. For example, let T_1 be the extension of T_0 by Π-types and intensional identity types, and take the syntactic CwR $\mathcal{T}_1 = \mathbf{Cl}(T_1)$. Suppose that we have a model $\mathcal{M}$ of $\mathcal{T}_1$, sections $A : \mathcal{M}(\diamond) \to \mathcal{M}(U)$ and $B : \mathcal{M}(\diamond)/\{A\} \to \mathcal{M}(U)$, and a section $a : \mathcal{M}(\diamond) \to \mathcal{M}(E)$ over A. We can easily construct a compound type like

$$T_1, \mathrm{L}(\mathcal{M}) \vdash () \to \prod_{\mathtt{x}:A} a\ \texttt{==}\ \mathtt{x}\ \texttt{->}\ B(a)\ \texttt{->}\ B(\mathtt{x}) : U,$$

and then we have the corresponding section $\mathcal{M}(\diamond) \to \mathcal{M}(U)$. It is hardly advisable to construct such a section in the language of category theory because we would have to explicitly write morphisms representing weakening and substitution which are silently performed syntactically.

5.1.2 Democratic models

Syntactic models of a type theory usually have the property that any object in the base category is a context and thus represented by a finite sequence of types. We will call such a model *democratic*, generalizing the notion of a democratic category with families [39, 40].

5.1.3. Definition. Let $\mathcal{M}$ be a model of $\mathcal{T}$, $u : y \to x$ a representable map in $\mathcal{T}$, $\Gamma \in \mathcal{M}(\diamond)$ an object and $a : \mathcal{M}(\diamond)/\Gamma \to \mathcal{M}(x)$ a section. Let $\mathrm{q}_u : \mathcal{M}(x) \to \mathcal{M}(y)$ be the right adjoint of $\mathcal{M}(u)$. The element $\mathrm{q}_u(a) \in \mathcal{M}(y)$ corresponds to a pair of an object $\{a\}_u \in \mathcal{M}(\diamond)$ and a section $\mathrm{q}_u(a) : \mathcal{M}(\diamond)/\{a\}_u \to \mathcal{M}(y)$. The counit $\mathrm{p}_u(a) : u \cdot \mathrm{q}_u(a) \to a$ then corresponds to a pullback square of the form

$$\begin{array}{ccc} \mathcal{M}(\diamond)/\{a\}_u & \xrightarrow{\mathrm{q}_u(a)} & \mathcal{M}(y) \\ {\scriptstyle \mathrm{p}_u(a)}\downarrow & \lrcorner & \downarrow{\scriptstyle \mathcal{M}(u)} \\ \mathcal{M}(\diamond)/\Gamma & \xrightarrow[a]{} & \mathcal{M}(x). \end{array}$$

We refer to the object $\{a\}_u \in \mathcal{M}(\diamond)$ as the *context comprehension of a with respect to u*.

5.1.4. Definition. Let $\mathcal{M}$ be a model of $\mathcal{T}$. The class of *contextual objects of $\mathcal{M}$* is the smallest replete class of objects of $\mathcal{M}(\diamond)$ containing the terminal object and closed under context comprehension. We say $\mathcal{M}$ is *democratic* if all the objects of $\mathcal{M}(\diamond)$ are contextual. We write $\mathbf{Mod}^{\mathrm{dem}}(\mathcal{T})$ the full subcategory of $\mathbf{Mod}(\mathcal{T})$ spanned by the democratic models.

5.1.5. Proposition. *Let $\mathcal{M}$ be a model of $\mathcal{T}$ and $\mathcal{M}'(\diamond) \subset \mathcal{M}(\diamond)$ a full subcategory closed under terminal objects. The following are equivalent:*

1. *the pullback of $\mathcal{M} : \mathcal{T} \to \mathbf{DFib}_{\mathcal{M}(\diamond)}$ along the inclusion $\mathcal{M}'(\diamond) \to \mathcal{M}(\diamond)$ determines a model $\mathcal{M}'$ of $\mathcal{T}$ and the natural transformation $\mathcal{M}' \Rightarrow \mathcal{M}$ is a morphism of models of $\mathcal{T}$;*

2. *$\mathcal{M}'(\diamond) \subset \mathcal{M}(\diamond)$ is closed under context comprehension.*

Proof:
Let $\mathcal{M}'(x)$ be the pullback

$$\begin{array}{ccc} \mathcal{M}'(x) & \hookrightarrow & \mathcal{M}(x) \\ \downarrow & \lrcorner & \downarrow \\ \mathcal{M}'(\diamond) & \hookrightarrow & \mathcal{M}(\diamond). \end{array}$$

The functor $\mathcal{M}' : \mathcal{T} \to \mathbf{DFib}_{\mathcal{M}'(\diamond)}$ preserves finite limits as $\mathcal{M}$ does. For a representable map $u : y \to x$ in $\mathcal{T}$, consider the square

$$\begin{array}{ccc} \mathcal{M}'(y) & \hookrightarrow & \mathcal{M}(y) \\ {\scriptstyle \mathcal{M}'(u)}\downarrow & & \downarrow{\scriptstyle \mathcal{M}(u)} \\ \mathcal{M}'(x) & \hookrightarrow & \mathcal{M}(x). \end{array} \tag{5.1}$$

The following are equivalent: the functor $\mathcal{M}'(u)$ has a right adjoint and Eq. (5.1) satisfies the Beck-Chevalley condition; the composite $\mathcal{M}'(x) \hookrightarrow \mathcal{M}(x) \xrightarrow{\mathrm{q}_u} \mathcal{M}(y)$ factors through $\mathcal{M}'(y)$. The former is equivalent to that $\mathcal{M}'$ is a model of $\mathcal{T}$ and the natural transformation $\mathcal{M}' \Rightarrow \mathcal{M}$ is a morphism of models. The latter is equivalent to that $\mathcal{M}'(\diamond)$ is closed under context comprehension. □

5.1.6. Definition. For a model $\mathcal{M}$ of $\mathcal{T}$, we define a model $\mathcal{M}^\heartsuit$ called the *heart of* $\mathcal{M}$ as follows: the base category $\mathcal{M}^\heartsuit(\diamond)$ is the full subcategory of $\mathcal{M}(\diamond)$ spanned by the contextual objects; $\mathcal{M}^\heartsuit(x)$ is the pullback of $\mathcal{M}(x)$ along the inclusion $\mathcal{M}^\heartsuit(\diamond) \to \mathcal{M}(\diamond)$. By Proposition 5.1.5, $\mathcal{M}^\heartsuit(\diamond)$ is indeed a model of $\mathcal{T}$ and the inclusion $\mathcal{M}^\heartsuit \to \mathcal{M}$ is a morphism of models.

By construction, $\mathcal{M}^\heartsuit$ is the largest democratic model contained in $\mathcal{M}$ in the following sense.

5.1.7. Proposition. *For any democratic model $\mathcal{M}$ of $\mathcal{T}$ and any model $\mathcal{N}$ of $\mathcal{T}$, the inclusion $\mathcal{N}^\heartsuit \hookrightarrow \mathcal{N}$ induces an equivalence of groupoids*

$$\mathbf{Mod}^{\mathrm{dem}}(\mathcal{T})\big(\mathcal{M}, \mathcal{N}^\heartsuit\big) \simeq \mathbf{Mod}(\mathcal{T})(\mathcal{M}, \mathcal{N}).$$

□

5.1.3 The theory-model correspondence

The goal of this chapter is to show the following.

5.1.8. Theorem. *For any type theory $\mathcal{T}$, the functor* $\mathrm{L} : \mathbf{Mod}(\mathcal{T}) \to \mathbf{Th}(\mathcal{T})$ *has a left adjoint and induces an equivalence*

$$\mathbf{Mod}^{\mathrm{dem}}(\mathcal{T}) \simeq \mathbf{Th}(\mathcal{T}).$$

The left adjoint $\mathrm{S} : \mathbf{Th}(\mathcal{T}) \to \mathbf{Mod}(\mathcal{T})$ of L assigns the *syntactic model* to each $\mathcal{T}$-theory.

The outline of the proof of Theorem 5.1.8 is as follows. To obtain the left adjoint S of L, we apply the *adjoint functor theorem* for presentable categories. By definition, the category $\mathbf{Th}(\mathcal{T})$ is compactly generated, and thus it remains

to show that the $(2,1)$-category $\mathbf{Mod}(\mathcal{T})$ is compactly generated (Section 5.2.1) and that L preserves limits and filtered colimits. The equivalence $\mathbf{Mod}^{\mathrm{dem}}(\mathcal{T}) \simeq \mathbf{Th}(\mathcal{T})$ is proved by concretely describing the left adjoint S. Since $\mathbf{Th}(\mathcal{T}) \simeq \mathbf{Lex}(\mathcal{T}, \mathbf{Set})$ is generated by representable functors $\mathrm{Y}_{\mathcal{T}^{\mathrm{op}}} x : \mathcal{T} \to \mathbf{Set}$ under filtered colimits and since both L and S preserve filtered colimits, the behavior of the adjunction $\mathrm{S} \dashv \mathrm{L}$ is completely determined by the values at representable functors $\mathrm{Y}_{\mathcal{T}^{\mathrm{op}}} x$. We begin by describing the *initial model* of $\mathcal{T}$ (Section 5.4.1) which is the syntactic model $\mathrm{S}(\mathrm{Y}_{\mathcal{T}^{\mathrm{op}}} 1)$ since $\mathrm{Y}_{\mathcal{T}^{\mathrm{op}}} 1$ is the initial object in $\mathbf{Th}(\mathcal{T})$. We then observe that the syntactic model $\mathrm{S}(\mathrm{Y}_{\mathcal{T}^{\mathrm{op}}} x)$ for an object $x \in \mathcal{T}$ is obtained from the initial model of the *slice type theory* $\mathcal{T}/x$ (Section 5.3.1).

5.2 The category of models of a type theory

In this section, we study the $(2,1)$-category of models of a type theory $\mathcal{T}$. We view a model of $\mathcal{T}$ as a category-valued functor as follows. A model of $\mathcal{T}$ is a pair $(\mathcal{M}(\diamond), \mathcal{M})$ consisting of a category $\mathcal{M}(\diamond)$ and a functor $\mathcal{M} : \mathcal{T} \to \mathbf{DFib}_{\mathcal{M}(\diamond)} \subset \mathbf{Cat}/\mathcal{M}(\diamond)$. Such a pair $(\mathcal{M}(\diamond), \mathcal{M})$ can be regarded as a functor $\mathcal{M} : \mathcal{T}^{\triangleright} \to \mathbf{Cat}$ where $\mathcal{T}^{\triangleright}$ is the category obtained from $\mathcal{T}$ by adjoining a new final object $\diamond$. A morphism $F : \mathcal{M} \to \mathcal{N}$ of models of $\mathcal{T}$ is then regarded as a natural transformation $F : \mathcal{M} \Rightarrow \mathcal{N} : \mathcal{T}^{\triangleright} \to \mathbf{Cat}$. One can see that the $(2,1)$-category $\mathbf{Mod}(\mathcal{T})$ is the subcategory of $\mathbf{Fun}(\mathcal{T}^{\triangleright}, \mathbf{Cat})$ spanned by the models of $\mathcal{T}$ and the morphisms between them.

5.2.1 Presentability of the category of models

The following result is in collaboration with John Bourke.

5.2.1. Proposition. *For any type theory $\mathcal{T}$, the $(2,1)$-category $\mathbf{Mod}(\mathcal{T})$ is compactly generated. Moreover, the inclusion functor $\mathbf{Mod}(\mathcal{T}) \to \mathbf{Fun}(\mathcal{T}^{\triangleright}, \mathbf{Cat})$ is conservative and preserves limits and filtered colimits.*

5.2.2. Remark. While the author was visiting Masaryk University, John and the author confirmed that $\mathbf{Mod}(\mathcal{T})$ seen as a strict $(2,2)$-category belongs to $\mathbf{LP}$ of Bourke [28] in the same way as the proof of Proposition 5.2.1 below. Consequently, $\mathbf{Mod}(\mathcal{T})$ has not only $(2,1)$-categorical (co)limits but also bi(co)limits and enjoys a form of biadjoint functor theorem [29]. For the results in this thesis the $(2,1)$-categorical structure of $\mathbf{Mod}(\mathcal{T})$ is enough, and a further $(2,2)$-categorical study of $\mathbf{Mod}(\mathcal{T})$ is left as future work.

5.2.3. Lemma. *Let* $\mathrm{cod} : \mathbf{DFib} \to \mathbf{Cat}$ *denote the functor that maps a discrete fibration to its codomain, and we view it as a functor between $(2,2)$-categories. For any $(2,2)$-category $\mathcal{C}$, the pullback operator*

$$\mathbf{Fun}(\mathcal{C}, \mathbf{Cat}) \ni F \mapsto F^{*}\mathbf{DFib} \in \widehat{(2,2)\text{-}\mathbf{Cat}}/\mathcal{C}$$

is part of a functor $\mathbf{Fun}(\mathcal{C}, \mathbf{Cat})^{\mathrm{coop}} \to \widehat{(2,2)\text{-}\mathbf{Cat}}/\mathcal{C}$*, where* $\widehat{(2,2)\text{-}\mathbf{Cat}}$ *is the* $(3,2)$*-category of large* $(2,2)$*-categories.*

Proof:
This follows from the fact that cod : **DFib** $\to$ **Cat** is a 2-fibration in the sense of Buckley [32]; see also [107]. For concreteness, we describe the action of natural transformations and modifications between functors $\mathcal{C} \to \mathbf{Cat}$.

Let $\sigma : F \Rightarrow G : \mathcal{C} \to \mathbf{Cat}$ be a natural transformation. The functor $\sigma^* : G^*\mathbf{DFib} \to F^*\mathbf{DFib}$ over $\mathcal{C}$ is defined as follows. An object of $G^*\mathbf{DFib}$ is a pair (x, B) consisting of an object $x \in \mathcal{C}$ and a discrete fibration B over $G(x)$. We define $\sigma^*(x, B) \in F^*\mathbf{DFib}$ to be $(x, \sigma_x^* B)$ where $\sigma_x^* B$ is the pullback of discrete fibrations

$$\begin{array}{ccc} \sigma_x^* B & \longrightarrow & B \\ \downarrow & \lrcorner & \downarrow \\ F(x) & \xrightarrow[\sigma_x]{} & G(x). \end{array}$$

Let $\vartheta : \sigma \Rrightarrow \tau : F \Rightarrow G : \mathcal{C} \to \mathbf{Cat}$ be a modification. The natural transformation $\vartheta^* : \tau^* \Rightarrow \sigma^* : G^*\mathbf{DFib} \to F^*\mathbf{DFib}$ is defined as follows. Let (x, B) be an object of $G^*\mathbf{DFib}$. We construct a map $\vartheta^* : \tau_x^* B \to \sigma_x^* B$ of discrete fibrations over $F(x)$. An object of $\tau_x^* B$ is a pair (y, b) consisting of an object $y \in F(x)$ and a section b of B over $\tau_x(y)$. We define $\vartheta^*(y, b) \in \sigma_x^* B$ to be $\big(y, \vartheta_{x,y}^* b\big)$ where $\vartheta_{x,y}^* b$ is the unique lift

$$\begin{array}{ccc} \vartheta_{x,y}^* b & \dashrightarrow & b \\ & & \\ \sigma_x(y) & \xrightarrow[\vartheta_{x,y}]{} & \tau_x(y). \end{array}$$

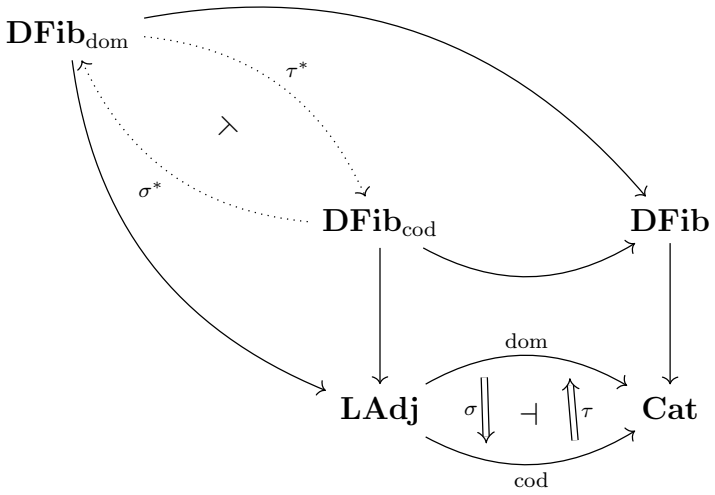
□

Proof of Proposition 5.2.1:
From the definition of a model of $\mathcal{T}$, a functor $\mathcal{M} : \mathcal{T}^{\triangleright} \to \mathbf{Cat}$ belongs to $\mathbf{Mod}(\mathcal{T})$ if and only if the following conditions are satisfied:

1. $\mathcal{M}(\diamond)$ has a terminal object;
2. for any object $x \in \mathcal{T}$, the functor $\mathcal{M}(x) \to \mathcal{M}(\diamond)$ is a discrete fibration;
3. for any finite diagram $(x_\xi)_{\xi \in \Xi}$ in $\mathcal{T}$, the canonical functor $\mathcal{M}(\lim_{\xi \in \Xi} x_\xi) \to \lim_{\xi \in \Xi} \mathcal{M}(x_\xi)$ is invertible, where the latter limit is taken in $\mathbf{Cat}/\mathcal{M}(\diamond)$;
4. for any representable map $u : y \to x$ in $\mathcal{T}$, the functor $\mathcal{M}(u) : \mathcal{M}(y) \to \mathcal{M}(x)$ has a right adjoint;
5. for any representable map $u : y \to x$ in $\mathcal{T}$ and any object $z \in \mathcal{T}/y$, the canonical map $\mathcal{M}(u_* z) \to \mathcal{M}(u)_* \mathcal{M}(z)$ in $\mathbf{DFib}_{\mathcal{M}(\diamond)}$ is invertible.

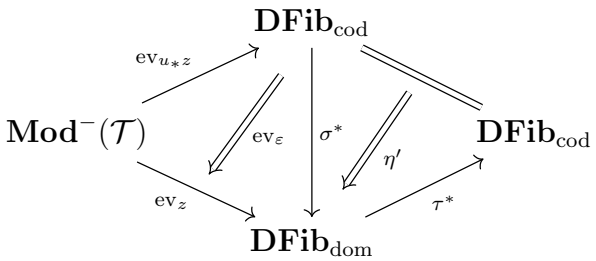
Let $\mathbf{Mod}^{-}(\mathcal{T}) \subset \mathbf{Fun}(\mathcal{T}^{\triangleright}, \mathbf{Cat})$ be the subcategory spanned by the functors satisfying Items 1 to 4 and the natural transformations compatible with the terminal object of $\mathcal{M}(\diamond)$ and the right adjoint of $\mathcal{M}(u)$ for any representable map u in $\mathcal{T}$. Items 1 to 4 are structures and properties defined by finite limits and adjoints, and thus it is straightforward to check that $\mathbf{Mod}^{-}(\mathcal{T})$ is compactly generated and the inclusion $\mathbf{Mod}^{-}(\mathcal{T}) \to \mathbf{Fun}(\mathcal{T}^{\triangleright}, \mathbf{Cat})$ is conservative and preserves limits and filtered colimits.

The tricky part is Item 5 because functors preserving pushforwards need not form a compactly generated category. The key idea is that the pushforward $\mathcal{M}(u)_*$ here is defined by the pullback along the right adjoint of $\mathcal{M}(u)$ (Proposition 3.1.17), and thus Item 5 is essentially a condition for the preservation of finite limits. To make it precise, we construct $\mathbf{Mod}(\mathcal{T})$ inside $\mathbf{Pr}^{\mathrm{R}}_{\omega}$. Let $u : y \to x$ be a representable map in $\mathcal{T}$ and $z \in \mathcal{T}/y$ an arbitrary object. Let $\mathbf{Mod}^{-,(u,z)}(\mathcal{T}) \subset \mathbf{Mod}^{-}(\mathcal{T})$ denote the full subcategory spanned by the functors satisfying Item 5 for u and z. Since $\mathbf{Mod}(\mathcal{T})$ is the wide pullback of all $\mathbf{Mod}^{-,(u,z)}(\mathcal{T})$'s over $\mathbf{Mod}^{-}(\mathcal{T})$, it suffices to construct $\mathbf{Mod}^{-,(u,z)}(\mathcal{T})$ inside $\mathbf{Pr}^{\mathrm{R}}_{\omega}$. By definition, the evaluation at u defines a functor $\mathrm{ev}_u : \mathbf{Mod}^{-}(\mathcal{T}) \to \mathbf{LAdj}$. Let $\mathbf{DFib}_{\mathrm{dom}}$ and $\mathbf{DFib}_{\mathrm{cod}}$ denote the pullbacks of $\mathrm{cod} : \mathbf{DFib} \to \mathbf{Cat}$ along $\mathrm{dom} : \mathbf{LAdj} \to \mathbf{Cat}$ and $\mathrm{cod} : \mathbf{LAdj} \to \mathbf{Cat}$, respectively. The canonical natural transformation $\sigma : \mathrm{dom} \Rightarrow \mathrm{cod}$ induces the pullback functor $\sigma^* : \mathbf{DFib}_{\mathrm{cod}} \to \mathbf{DFib}_{\mathrm{dom}}$ over $\mathbf{LAdj}$ by Lemma 5.2.3. Since σ has a right adjoint $\tau : \mathrm{cod} \Rightarrow \mathrm{dom}$ (in the $(2,2)$-category $\mathbf{Fun}(\mathbf{LAdj}, \mathbf{Cat})$), we also have the pullback functor $\tau^* : \mathbf{DFib}_{\mathrm{dom}} \to \mathbf{DFib}_{\mathrm{cod}}$ which is right adjoint to σ^* by Lemma 5.2.3.



The evaluations at z and u_*z define functors $\mathrm{ev}_z : \mathbf{Mod}^{-}(\mathcal{T}) \to \mathbf{DFib}_{\mathrm{dom}}$ and $\mathrm{ev}_{u_*z} : \mathbf{Mod}^{-}(\mathcal{T}) \to \mathbf{DFib}_{\mathrm{cod}}$, respectively, over ev_u, and the counit $\varepsilon : u^*u_*z \to z$ induces a natural transformation $\mathrm{ev}_\varepsilon : \sigma^* \circ \mathrm{ev}_{u_*z} \Rightarrow \mathrm{ev}_z$ over ev_u. Let ρ be the

composite of natural transformations



where η' is the unit of the adjunction $\sigma^* \dashv \tau^*$. Then, $\mathbf{Mod}^{-,(u,z)}(\mathcal{T})$ is the inverter of ρ, that is, the pullback

$$\begin{array}{ccc} \mathbf{Mod}^{-,(u,z)}(\mathcal{T}) & \longrightarrow & \mathbf{DFib}^{\cong}_{\mathrm{cod}} \\ \downarrow & \lrcorner & \downarrow \\ \mathbf{Mod}^{-}(\mathcal{T}) & \xrightarrow[\rho]{} & \mathbf{DFib}^{\to}_{\mathrm{cod}} \end{array}$$

where $\mathcal{C}^{\cong} \subset \mathcal{C}^{\to}$ denotes the full subcategory spanned by the invertible arrows. Indeed, the component of ρ at an object $\mathcal{M} \in \mathbf{Mod}^{-}(\mathcal{T})$ is

$$\mathcal{M}(u_* z) \xrightarrow{\eta'} \tau^* \sigma^* \mathcal{M}(u_* z) \cong \tau^* \mathcal{M}(u^* u_* z) \xrightarrow{\tau^* \mathcal{M}(\varepsilon)} \tau^* \mathcal{M}(z) \cong \mathcal{M}(u)_* \mathcal{M}(z),$$

and $\mathbf{Mod}^{-,(u,z)}(\mathcal{T})$ is precisely the full subcategory of $\mathbf{Mod}^{-}(\mathcal{T})$ spanned by those object $\mathcal{M}$ such that this canonical map is invertible. Since all the functors involved with this construction preserve limits and filtered colimits, $\mathbf{Mod}^{-,(u,z)}(\mathcal{T})$ is constructed inside $\mathbf{Pr}^{\mathrm{R}}_{\omega}$. □

5.2.2 The universal property of the category of models

We view $\mathbf{Mod}(\mathcal{T})$ as a $(2,1)$-category over $\mathbf{Lex}^{(\emptyset)}$ with the functor $\mathbf{Mod}(\mathcal{T}) \to \mathbf{Lex}^{(\emptyset)}$ sending a model of $\mathcal{T}$ to its base category and give a universal property, where $\mathbf{Lex}^{(\emptyset)}$ is the $(2,1)$-category of categories with limits of shape $\emptyset$, that is, categories with final objects. In this section, we describe the universal property of $\mathbf{Mod}(\mathcal{T})$ in $\widehat{(2,1)\text{-}\mathbf{Cat}}/\mathbf{Lex}^{(\emptyset)}$. A corollary is that $\mathcal{T} \mapsto \mathbf{Mod}(\mathcal{T})$ is part of a limit-preserving functor $\mathbf{TT}^{\mathrm{op}} \to \widehat{(2,1)\text{-}\mathbf{Cat}}/\mathbf{Lex}^{(\emptyset)}$, and thus a universal property of a type theory $\mathcal{T}$ is transferred to $\mathbf{Mod}(\mathcal{T})$.

We first redefine discrete fibrations in a manner invariant under equivalence.

5.2.4. Definition. Let $\mathcal{X}$ be a finitely complete $(2,2)$-category. We say a map $p : A \to \mathcal{C}$ in $\mathcal{X}$ is a *right fibration* if the square

$$\begin{array}{ccc} A^{\to} & \xrightarrow{\mathrm{cod}} & A \\ {\scriptstyle p^{\to}}\downarrow & & \downarrow{\scriptstyle p} \\ \mathcal{C}^{\to} & \xrightarrow[\mathrm{cod}]{} & \mathcal{C} \end{array}$$

is a pullback. By a *right fibration over* $\mathcal{C}$ we mean a right fibration of the form $A \to \mathcal{C}$. We write $\mathbf{RFib}(\mathcal{X})_{\mathcal{C}} \subset \mathcal{X}/\mathcal{C}$ for the full $(2,1)$-subcategory spanned by the right fibrations over $\mathcal{C}$. A *discrete fibration over* $\mathcal{C}$ is a 0-truncated object in $\mathbf{RFib}(\mathcal{X})_{\mathcal{C}}$. We write $\mathbf{DFib}(\mathcal{X})_{\mathcal{C}} \subset \mathbf{RFib}(\mathcal{X})_{\mathcal{C}}$ for the full subcategory spanned by the discrete fibrations over $\mathcal{C}$.

5.2.5. Proposition. *When* $\mathcal{X} = \mathbf{Cat}$*, the category* $\mathbf{DFib}(\mathbf{Cat})_{\mathcal{C}}$ *is equivalent to* $\mathbf{DFib}_{\mathcal{C}}$*.*

Proof:
Recall that any right fibration A over $\mathcal{C}$ is equivalent to the category of elements of the groupoid-valued presheaf $x \mapsto \mathbf{RFib}(\mathbf{Cat})_{\mathcal{C}}(\mathcal{C}/x, A)$, which is the right adjoint splitting of A [158]. When A is 0-truncated, the groupoid $\mathbf{RFib}(\mathbf{Cat})_{\mathcal{C}}(\mathcal{C}/x, A)$ is equivalent to a discrete one, and thus A is equivalent to an object from $\mathbf{DFib}_{\mathcal{C}}$. □

5.2.6. Definition. We say a map $f : B \to A$ in $\mathbf{DFib}(\mathcal{X})_{\mathcal{C}}$ is a *representable map* if it has a right adjoint in $\mathcal{X}$.

We have the following basic properties in the same way as $\mathbf{DFib}_{\mathcal{C}}$.

5.2.7. Lemma. *For any discrete fibration* A *over* $\mathcal{C}$*, the equivalence* $(\mathcal{X}/\mathcal{C})/A \simeq \mathcal{X}/A$ *is restricted to an equivalence* $\mathbf{DFib}(\mathcal{X})_{\mathcal{C}}/A \simeq \mathbf{DFib}(\mathcal{X})_{A}$*.* □

5.2.8. Lemma. *Any representable map* $f : B \to A$ *in* $\mathbf{DFib}(\mathcal{X})_{\mathcal{C}}$ *is exponentiable, and the pushforward along* f *is given by the pullback along the right adjoint* $A \to B$ *of* f*.* □

5.2.9. Proposition. *For any finitely complete* $(2,2)$*-category* $\mathcal{X}$*, the category* $\mathbf{DFib}(\mathcal{X})_{\mathcal{C}}$ *is a CwR. Furthermore, any left exact functor* $F : \mathcal{X} \to \mathcal{Y}$ *induces a morphism of CwRs* $\mathbf{DFib}(\mathcal{X})_{\mathcal{C}} \to \mathbf{DFib}(\mathcal{Y})_{F(\mathcal{C})}$*.*

Proof:
By construction. □

Let Ξ be a $(2,1)$-category and $\mathcal{C} : \Xi \to \mathbf{Cat}$ a functor. We have the CwR $\mathbf{DFib}(\mathbf{Fun}(\Xi, \mathbf{Cat}))_{\mathcal{C}}$. Let $\mathcal{T}$ be a type theory. We have equivalences of 2-groupoids

$$\begin{aligned} & \widehat{(2,1)\text{-}\mathbf{Cat}}/\mathbf{Cat}((\Xi, \mathcal{C}), (\mathbf{Fun}(\mathcal{T}^{\triangleright}, \mathbf{Cat}), \mathrm{ev}_{\diamond})) \\ \simeq & \qquad \text{(transposition)} \\ & \{\diamond\}/\widehat{(2,1)\text{-}\mathbf{Cat}}((\mathcal{T}^{\triangleright}, \diamond), (\mathbf{Fun}(\Xi, \mathbf{Cat}), \mathcal{C})) \\ \simeq & \qquad \text{(definition of } \mathcal{T}^{\triangleright}) \\ & \widehat{(2,1)\text{-}\mathbf{Cat}}(\mathcal{T}, \mathbf{Fun}(\Xi, \mathbf{Cat})/\mathcal{C}). \end{aligned}$$

5.2.10. Proposition. *Let Ξ be a $(2,1)$-category and $\mathcal{C} : \Xi \to \mathbf{Lex}^{(\emptyset)} \subset \mathbf{Cat}$ a functor. For a type theory $\mathcal{T}$ and a functor $F : \mathcal{T} \to \mathbf{Fun}(\Xi, \mathbf{Cat})/\mathcal{C}$, the following are equivalent:*

1. *F factors through $\mathbf{DFib}(\mathbf{Fun}(\Xi, \mathbf{Cat}))_{\mathcal{C}}$ and is a morphism of CwRs;*
2. *the transpose $F' : \Xi \to \mathbf{Fun}(\mathcal{T}^{\triangleright}, \mathbf{Cat})$ factors through $\mathbf{Mod}(\mathcal{T})$.*

Consequently, we have an equivalence of groupoids

$$\widehat{(2,1)\text{-}\mathbf{Cat}}/\mathbf{Lex}^{(\emptyset)}((\Xi, \mathcal{C}), (\mathbf{Mod}(\mathcal{T}), \mathrm{ev}_{\diamond})) \simeq \widehat{\mathbf{CwR}}(\mathcal{T}, \mathbf{DFib}(\mathbf{Fun}(\Xi, \mathbf{Cat}))_{\mathcal{C}}).$$

Proof:
This is immediate from the definitions of $\mathbf{DFib}(\mathbf{Fun}(\Xi, \mathbf{Cat}))_{\mathcal{C}}$ and $\mathbf{Mod}(\mathcal{T})$. □

5.2.11. Remark. We can also obtain a stronger universal property viewing $\mathbf{Mod}(\mathcal{T})$ as a $(2,2)$-category over $\mathbf{Lex}^{(\emptyset)}$.

5.2.12. Corollary. *The assignment $\mathcal{T} \mapsto \mathbf{Mod}(\mathcal{T})$ is part of a limit-preserving functor $\mathbf{TT}^{\mathrm{op}} \to \widehat{(2,1)\text{-}\mathbf{Cat}}/\mathbf{Lex}^{(\emptyset)}$.* □

5.3 The category of type theories

In this section, we study the $(2,1)$-category of type theories. *Slice type theories* will play an important role in the proof of the theory-model correspondence. We also show that the $(2,1)$-category of type theories is compactly generated, although we do not use it in this chapter. The presentability will be a major source of examples of higher dimensional type theories (Chapter 6) where syntactic presentations have not yet been available.

5.3.1 Slice type theories

Let $\mathcal{C}$ be a CwR and $x \in \mathcal{C}$ an object. The slice category $\mathcal{C}/x$ is a CwR in which an arrow u is a representable map if it is a representable map in $\mathcal{C}$. We show that $\mathcal{C}/x$ is the CwR obtained from $\mathcal{C}$ by freely adjoining a global section of x.

5.3.1. PROPOSITION. *For any arrow $u : x \to y$ in a CwR $\mathcal{C}$, the pullback functor $u^* : \mathcal{C}/y \to \mathcal{C}/x$ is a morphism of CwRs.*

Proof:
The pullback functor commutes with any limits and pushforwards. It also preserves representable maps since representable maps are closed under pullbacks. □

In particular, we have the morphism of CwRs $x^* : \mathcal{C} \to \mathcal{C}/x$ defined by the pullback along the final projection $x \to 1$. We regard the diagonal arrow $\Delta_x : x \to x \times x$ as a global section $1_x \to x^* x$ in $\mathcal{C}/x$.

5.3.2. PROPOSITION. *Let $\mathcal{C}$ be a CwR and $x \in \mathcal{C}$ an object. For any CwR $\mathcal{D}$, the square*

$$\begin{array}{ccc} \mathbf{CwR}(\mathcal{C}/x, \mathcal{D}) & \xrightarrow{F \mapsto F(\Delta_x)} & \mathrm{k}(1/\mathcal{D}) \\ {\scriptstyle (-\circ x^*)}\downarrow & & \downarrow{\scriptstyle \mathrm{cod}} \\ \mathbf{CwR}(\mathcal{C}, \mathcal{D}) & \xrightarrow[F \mapsto F(x)]{} & \mathrm{k}(\mathcal{D}) \end{array}$$

is a pullback in the $(2,1)$-category of groupoids.

Proof:
For a morphism $F : \mathcal{C} \to \mathcal{D}$ and a global section $u : 1 \to F(x)$, we have a morphism

$$\mathcal{C}/x \xrightarrow{F/x} \mathcal{D}/F(x) \xrightarrow{u^*} \mathcal{D},$$

which defines an inverse of the functor $\mathbf{CwR}(\mathcal{C}/x, \mathcal{D}) \to \mathbf{CwR}(\mathcal{C}, \mathcal{D}) \times_{\mathrm{k}(\mathcal{D})} \mathrm{k}(1/\mathcal{D})$. □

5.3.2 Presentability of the category of type theories

Let $\mathbf{Cat}^+$ denote the $(2,1)$-category in which the objects are the small categories equipped with a class of arrows and the morphisms are the functors preserving the specified arrows. $\mathbf{Cat}^+$ fits into the pullback

$$\begin{array}{ccc} \mathbf{Cat}^+ & \longrightarrow & \mathbf{Sub}(\mathbf{Set}) \\ \downarrow & \lrcorner & \downarrow \\ \mathbf{Cat} & \xrightarrow[\mathrm{Hom}]{} & \mathbf{Set} \end{array}$$

and thus is compactly generated. We view the $(2,1)$-category $\mathbf{TT}$ of type theories as a subcategory of $\mathbf{Cat}^+$.

5.3.3. Proposition. *The $(2,1)$-category $\mathbf{TT}$ is compactly generated. Moreover, the forgetful functor $\mathbf{TT} \to \mathbf{Cat}^+$ preserves limits and filtered colimits and is conservative.*

Proof:
Let $\mathbf{Lex}^+$ denote the $(2,1)$-category of finitely complete categories equipped with a pullback-stable class of arrows. It is defined as a full subcategory of $\mathbf{Lex} \times_{\mathbf{Cat}} \mathbf{Cat}^+$. The inclusion $\mathbf{Lex}^+ \to \mathbf{Lex} \times_{\mathbf{Cat}} \mathbf{Cat}^+$ has a left adjoint by taking the pullback-stable closure of a class of arrows and is closed under filtered colimits. Then $\mathbf{TT}$ fits into the pullback

$$\begin{array}{ccc} \mathbf{TT} & \longrightarrow & \mathbf{LAdj} \\ \downarrow & \lrcorner & \downarrow \\ \mathbf{Lex}^+ & \xrightarrow[F]{} & \mathbf{Cat}^{\to}, \end{array}$$

where $\mathbf{LAdj}$ is the category of left adjoints (so a functor that belongs to $\mathbf{LAdj}$ has a right adjoint), the bottom functor F sends an object $\mathcal{C} \in \mathbf{Lex}^+$ to the pullback functor $\mathcal{C}^{\not\Downarrow} \to \mathcal{C}^{\downarrow\not\Rightarrow}$, $\mathcal{C}^{\not\Downarrow}$ is the subcategory of the category of diagrams $\begin{smallmatrix} & & x_1 \\ & & \downarrow \\ x_2 & \xrightarrow[u]{} & x_3 \end{smallmatrix}$ in $\mathcal{C}$ spanned by those diagrams such that u is a representable map and those morphisms whose components at x_2 and x_3 are invertible, and $\mathcal{C}^{\downarrow\not\Rightarrow}$ is similarly defined. By construction, $\mathbf{TT}$ is compactly generated and the forgetful functor $\mathbf{TT} \to \mathbf{Cat}^+$ preserves limits and filtered colimits and is conservative. □

5.4 The theory-model correspondence

In this section, we prove Theorem 5.1.8: the internal language functor $\mathrm{L} : \mathbf{Mod}(\mathcal{T}) \to \mathbf{Th}(\mathcal{T})$ has a left adjoint and induces an equivalence $\mathbf{Mod}^{\mathrm{dem}}(\mathcal{T}) \simeq \mathbf{Th}(\mathcal{T})$.

5.4.1. Proposition. *For any type theory $\mathcal{T}$, the functor $\mathrm{L} : \mathbf{Mod}(\mathcal{T}) \to \mathbf{Th}(\mathcal{T})$ preserves limits and filtered colimits.*

Proof:
This is because limits and filtered colimits in $\mathbf{Mod}(\mathcal{T})$ and $\mathbf{Th}(\mathcal{T})$ are computed in $\mathbf{Fun}(\mathcal{T}^{\triangleright}, \mathbf{Cat})$ and $\mathbf{Fun}(\mathcal{T}, \mathbf{Set})$, respectively. □

Then, by the adjoint functor theorem for presentable $(2,1)$-categories, the functor L has a left adjoint $\mathrm{S} : \mathbf{Th}(\mathcal{T}) \to \mathbf{Mod}(\mathcal{T})$. For a $\mathcal{T}$-theory Φ, we call $\mathrm{S}(\Phi)$ the *syntactic model generated by* Φ.

The goal of this section is to show that the adjunction $\mathrm{S} \dashv \mathrm{L}$ induces an equivalence between theories and democratic models. Since $\mathbf{Th}(\mathcal{T})$ is the completion of $\mathcal{T}^{\mathrm{op}}$ under filtered colimits and any left adjoint preserves colimits, S is completely determined by the values at the representable functors $\mathrm{Y}_{\mathcal{T}^{\mathrm{op}}}(x)$. We thus concretely describe syntactic models of the form $\mathrm{S}(\mathrm{Y}_{\mathcal{T}^{\mathrm{op}}}(x))$ to understand S. The initial model is the special case when x is the final object of $\mathcal{T}$ and studied in Section 5.4.1. Other syntactic models are described using the initial model of $\mathcal{T}/x$ in Section 5.4.2. We prove the main result in Section 5.4.3.

5.4.1 The initial model

5.4.2. Definition. Recall that the Yoneda embedding $\mathrm{Y}_{\mathcal{T}} : \mathcal{T} \to \mathbf{DFib}_{\mathcal{T}}$ preserves all existing limits and pushforwards. Therefore, the pair $(\mathcal{T}, \mathrm{Y}_{\mathcal{T}})$ is a model of $\mathcal{T}$. We define the *initial model* $\mathrm{I}(\mathcal{T})$ to be the heart of $(\mathcal{T}, \mathrm{Y}_{\mathcal{T}})$.

We will show that $\mathrm{I}(\mathcal{T})$ is indeed an initial object of $\mathbf{Mod}(\mathcal{T})$. Since it is constructed from the Yoneda embedding, the initiality of $\mathrm{I}(\mathcal{T})$ essentially follows from the Yoneda Lemma.

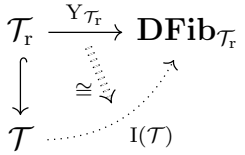
By definition, the model $\mathrm{I}(\mathcal{T})$ is described as follows:

- the base category is $\mathcal{T}_{\mathrm{r}}$, the full subcategory of $\mathcal{T}$ spanned by the objects x such that the final projection $x \to 1$ is a representable map;
- $\mathrm{I}(\mathcal{T})(y) = \mathcal{T}_{\mathrm{r}}/y$ defined by the pullback

$$\begin{array}{ccc} \mathcal{T}_{\mathrm{r}}/y & \hookrightarrow & \mathcal{T}/y \\ \downarrow & & \downarrow \\ \mathcal{T}_{\mathrm{r}} & \hookrightarrow & \mathcal{T} \end{array}$$

  for $y \in \mathcal{T}$.

Alternatively, the functor $\mathrm{I}(\mathcal{T}) : \mathcal{T} \to \mathbf{DFib}_{\mathcal{T}_{\mathrm{r}}}$ is defined as the left Kan extension of the Yoneda embedding $\mathrm{Y}_{\mathcal{T}_{\mathrm{r}}} : \mathcal{T}_{\mathrm{r}} \to \mathbf{DFib}_{\mathcal{T}_{\mathrm{r}}}$ along the inclusion $\mathcal{T}_{\mathrm{r}} \hookrightarrow \mathcal{T}$.



5.4.3. Example. The construction of the initial model coincides with the traditional syntactic construction. Let T be a SOGAT. Then the base category of the initial model $\mathrm{I}(\mathbf{Cl}(T))$ is described as follows:

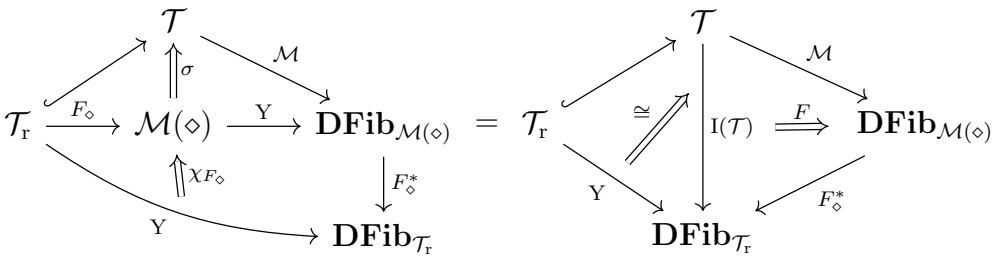
- the objects are the well-formed finite environments over T of the form $\Gamma^\dagger$ for a context Γ;
- the morphisms are the equivalence classes of instantiations.

By contextual completeness, an instantiation of $\Delta^\dagger$ in $\Gamma^\dagger$ is equivalent to a substitution of Δ in Γ. Therefore, the base category is the category of contexts and substitutions. For an object $\Phi \in \mathbf{Cl}(T)$, a section $\mathrm{I}(\mathbf{Cl}(T))(\diamond)/\Gamma^\dagger \to \mathrm{I}(\mathbf{Cl}(T))(\Phi)$ is represented by an instantiation of Φ in $\Gamma^\dagger$ which is equivalent to an instantiation of Φ over Γ by contextual completeness.

5.4.4. Theorem. *For any type theory $\mathcal{T}$, the model $\mathrm{I}(\mathcal{T})$ is an initial object in the $(2,1)$-category $\mathbf{Mod}(\mathcal{T})$.*

Proof:
Let $\mathcal{M}$ be a model of $\mathcal{T}$. Note that a morphism $F : \mathrm{I}(\mathcal{T}) \to \mathcal{M}$ is regarded as a pair $(F_\diamond, F)$ consisting of a functor $F_\diamond : \mathcal{T}_\mathrm{r} \to \mathcal{M}(\diamond)$ and a natural transformation $F : \mathrm{I}(\mathcal{T}) \Rightarrow F_\diamond^* \mathcal{M} : \mathcal{T} \to \mathbf{DFib}_{\mathcal{T}_\mathrm{r}}$. We first show that there is at most one morphism $\mathrm{I}(\mathcal{T}) \to \mathcal{M}$ up to contractible choice.

Let $F : \mathrm{I}(\mathcal{T}) \to \mathcal{M}$ be a morphism. We have the natural transformation $\sigma : \mathrm{Y}_{\mathcal{M}(\diamond)} \circ F_\diamond \Rightarrow \mathcal{M}|_{\mathcal{T}_\mathrm{r}} : \mathcal{T}_\mathrm{r} \to \mathbf{DFib}_{\mathcal{M}(\diamond)}$ whose component at $x \in \mathcal{T}_\mathrm{r}$ is $F_x(\mathrm{id}_x) : \mathcal{M}(\diamond)/F_\diamond(x) \to \mathcal{M}(x)$. The natural transformation σ is characterized as the one such that



(5.2)

where $\chi_{F_\diamond}$ is the natural transformation defined by the morphism part of the functor $F_\diamond$. The Beck-Chevalley condition for a representable map $u : y \to x$ implies that for any object $(v : z \to x) \in \mathcal{T}_\mathrm{r}/x$, the square

$$\begin{array}{ccc} \mathcal{M}(\diamond)/F_\diamond(v^*y) & \xrightarrow{F_y(u^*v)} & \mathcal{M}(y) \\ {\scriptstyle v^*u}\downarrow & & \downarrow{\scriptstyle \mathcal{M}(u)} \\ \mathcal{M}(\diamond)/F_\diamond(z) & \xrightarrow[F_x(v)]{} & \mathcal{M}(x) \end{array}$$

is a pullback. From the special case when x is the final object, we see that the canonical map $F_y(\mathrm{id}_y) : \mathcal{M}(\diamond)/F_\diamond(y) \to \mathcal{M}(y)$ is invertible for any object $y \in \mathcal{T}_\mathrm{r}$, that is, the natural transformation σ is invertible. Then, since the Yoneda

embedding is fully faithful, the functor $F_\diamond$ is unique up to contractible choice. The natural transformation $F : \mathrm{I}(\mathcal{T}) \Rightarrow F_\diamond^* \circ \mathcal{M}$ is also unique up to contractible choice because $\mathrm{I}(\mathcal{T})$ is the left Kan extension of the Yoneda embedding along the inclusion $\mathcal{T}_\mathrm{r} \to \mathcal{T}$.

It remains to construct a morphism $F : \mathrm{I}(\mathcal{T}) \to \mathcal{M}$. Since the base category $\mathcal{M}(\diamond)$ has a final object, if a discrete fibration A over $\mathcal{M}(\diamond)$ satisfies that the final projection $A \to 1$ is a representable map, then A is representable. Then, by the definition of $\mathcal{T}_\mathrm{r}$, the restriction of $\mathcal{M} : \mathcal{T} \to \mathbf{DFib}_{\mathcal{M}(\diamond)}$ to $\mathcal{T}_\mathrm{r}$ factors through the Yoneda embedding. Let $F_\diamond : \mathcal{T}_\mathrm{r} \to \mathcal{M}(\diamond)$ denote the induced functor.

$$\begin{array}{ccc} \mathcal{T}_\mathrm{r} & \overset{F_\diamond}{\cdots\cdots\to} & \mathcal{M}(\diamond) \\ \downarrow & \overset{\sigma}{\swarrow\!\cong} & \downarrow{\scriptstyle \mathrm{Y}} \\ \mathcal{T} & \xrightarrow[\mathcal{M}]{} & \mathbf{DFib}_{\mathcal{M}(\diamond)} \end{array}$$

Since $\mathrm{I}(\mathcal{T})$ is the left Kan extension of the Yoneda embedding along the inclusion $\mathcal{T}_\mathrm{r} \to \mathcal{T}$, we have a unique natural transformation $F : \mathrm{I}(\mathcal{T}) \Rightarrow F_\diamond^* \circ \mathcal{M} : \mathcal{T} \to \mathbf{DFib}_{\mathcal{T}_\mathrm{r}}$ satisfying Eq. (5.2). It remains to check that F is indeed a morphism of models. By construction, the functor $F_\diamond$ preserves final objects. Let $u : y \to x$ be a representable map in $\mathcal{T}$. We have to show that the square

$$\begin{array}{ccc} \mathcal{T}_\mathrm{r}/y & \xrightarrow{F_y} & \mathcal{M}(y) \\ {\scriptstyle u}\downarrow & & \downarrow{\scriptstyle \mathcal{M}(u)} \\ \mathcal{T}_\mathrm{r}/x & \xrightarrow[F_x]{} & \mathcal{M}(x) \end{array}$$

satisfies the Beck-Chevalley condition. Since the Beck-Chevalley condition is verified at each object of $\mathcal{T}_\mathrm{r}/x$ and since any object $(v : z \to x) \in \mathcal{T}_\mathrm{r}/x$ is the image of id_z by the map $v : \mathcal{T}_\mathrm{r}/z \to \mathcal{T}_\mathrm{r}/x$, it suffices to show that the composite of squares

$$\begin{array}{ccccc} \mathcal{T}_\mathrm{r}/v^*y & \xrightarrow{u^*v} & \mathcal{T}_\mathrm{r}/y & \xrightarrow{F_y} & \mathcal{M}(y) \\ {\scriptstyle v^*u}\downarrow & & \downarrow{\scriptstyle u} & & \downarrow{\scriptstyle \mathcal{M}(u)} \\ \mathcal{T}_\mathrm{r}/z & \xrightarrow[v]{} & \mathcal{T}_\mathrm{r}/x & \xrightarrow[F_x]{} & \mathcal{M}(x) \end{array} \tag{5.3}$$

satisfies the Beck-Chevalley condition for any arrow $v : z \to x$ with $z \in \mathcal{T}_\mathrm{r}$. By the definition of F, Eq. (5.3) is isomorphic to

$$\begin{array}{ccccccc} \mathcal{T}_\mathrm{r}/v^*y & \xrightarrow{F_\diamond} & \mathcal{M}(\diamond)/F_\diamond(v^*y) & \xrightarrow{\cong} & \mathcal{M}(v^*y) & \xrightarrow{\mathcal{M}(u^*v)} & \mathcal{M}(y) \\ {\scriptstyle v^*u}\downarrow & & {\scriptstyle F_\diamond(v^*u)}\downarrow & & {\scriptstyle \mathcal{M}(v^*u)}\downarrow & & \downarrow{\scriptstyle \mathcal{M}(u)} \\ \mathcal{T}_\mathrm{r}/z & \xrightarrow[F_\diamond]{} & \mathcal{M}(\diamond)/F_\diamond(z) & \xrightarrow[\cong]{} & \mathcal{M}(z) & \xrightarrow[\mathcal{M}(v)]{} & \mathcal{M}(x). \end{array} \tag{5.4}$$

The right square of Eq. (5.4) is a pullback in $\mathbf{DFib}_{\mathcal{M}(\diamond)}$ and thus satisfies the Beck-Chevalley condition by Proposition 3.1.15. The Beck-Chevalley condition for the left square of Eq. (5.4) asserts that $F_\diamond$ preserves pullbacks of representable maps in $\mathcal{T}_{\mathrm{r}}$, which is true by the definition of $F_\diamond$. □

5.4.2 Syntactic models generated by compact theories

Recall that the compact objects in $\mathbf{Lex}(\mathcal{C}, \mathbf{Set})$ for a finitely complete category $\mathcal{C}$ are precisely the representable functors $\mathrm{Y}_{\mathcal{C}^{\mathrm{op}}} x = \mathrm{Hom}(x, -) : \mathcal{C} \to \mathbf{Set}$. We may thus call a $\mathcal{T}$-theory of the form $\mathrm{Y}_{\mathcal{T}^{\mathrm{op}}} x$ a *compact $\mathcal{T}$-theory*. In this section, we describe the syntactic model $\mathrm{S}(\mathrm{Y}_{\mathcal{T}^{\mathrm{op}}}(x))$.

5.4.5. Proposition. *For any object $x \in \mathcal{T}$, we have a pullback*

$$\begin{array}{ccc} \mathbf{Mod}(\mathcal{T}/x) & \longrightarrow & \mathrm{Y}_{\mathcal{T}^{\mathrm{op}}}(x)/\mathbf{Th}(\mathcal{T}) \\ \downarrow & \lrcorner & \downarrow \\ \mathbf{Mod}(\mathcal{T}) & \xrightarrow[\mathrm{L}]{} & \mathbf{Th}(\mathcal{T}). \end{array}$$

Proof:
By Propositions 5.2.10 and 5.3.2, a model of $\mathcal{T}/x$ corresponds to a model $\mathcal{M}$ of $\mathcal{T}$ equipped with a global section $a : \mathcal{M}(\diamond) \to \mathcal{M}(x)$. Since the base category $\mathcal{M}(\diamond)$ has a final object 1, the global section a corresponds to an element in the fiber $\mathcal{M}(x)_1$ over the final object. We thus have a pullback

$$\begin{array}{ccc} \mathbf{Mod}(\mathcal{T}/x) & \longrightarrow & 1/\mathbf{Set} \\ \downarrow & \lrcorner & \downarrow \\ \mathbf{Mod}(\mathcal{T}) & \xrightarrow[\mathcal{M} \mapsto \mathcal{M}(x)_1]{} & \mathbf{Set}. \end{array}$$

By the definition of L, the bottom functor is isomorphic to the composite

$$\mathbf{Mod}(\mathcal{T}) \xrightarrow{\mathrm{L}} \mathbf{Th}(\mathcal{T}) \xrightarrow{\mathrm{ev}_x} \mathbf{Set}.$$

By Yoneda, we have a pullback

$$\begin{array}{ccc} \mathrm{Y}_{\mathcal{T}^{\mathrm{op}}}(x)/\mathbf{Th}(\mathcal{T}) & \longrightarrow & 1/\mathbf{Set} \\ \downarrow & \lrcorner & \downarrow \\ \mathbf{Th}(\mathcal{T}) & \xrightarrow[\mathrm{ev}_x]{} & \mathbf{Set}, \end{array}$$

and then we obtain the desired pullback by the two-pullbacks lemma. □

By Proposition 5.4.5, we get an equivalence

$$\mathbf{Mod}(\mathcal{T}/x) \simeq (\mathrm{Y}_{\mathcal{T}^{\mathrm{op}}}(x) \downarrow \mathrm{L}).$$

Since the syntactic model $\mathrm{S}(\mathrm{Y}_{\mathcal{T}^{\mathrm{op}}}(x))$ is the initial object of $(\mathrm{Y}_{\mathcal{T}^{\mathrm{op}}}(x) \downarrow \mathrm{L})$, it is obtained from the initial model $\mathrm{I}(\mathcal{T}/x)$ of $\mathcal{T}/x$ by restricting $\mathrm{I}(\mathcal{T}/x) : \mathcal{T}/x \to \mathbf{DFib}_{\mathrm{I}(\mathcal{T}/x)(\diamond)}$ along $x^* : \mathcal{T} \to \mathcal{T}/x$. We thus have a concrete description of $\mathrm{S}(\mathrm{Y}_{\mathcal{T}^{\mathrm{op}}}(x))$ as follows:

- the base category $\mathrm{S}(\mathrm{Y}_{\mathcal{T}^{\mathrm{op}}}(x))(\diamond)$ is the full subcategory of $\mathcal{T}/x$ spanned by the representable maps $y \to x$;
- for objects $y \in \mathcal{T}$ and $(u : x' \to x) \in \mathrm{S}(\mathrm{Y}_{\mathcal{T}^{\mathrm{op}}}(x))(\diamond)$, the fiber of $\mathrm{S}(\mathrm{Y}_{\mathcal{T}^{\mathrm{op}}}(x))(y)$ over u is $\mathcal{T}/x(u, x^*y) \cong \mathcal{T}(x', y)$.

General syntactic models

Although we do not need concrete descriptions of general syntactic models to prove the main theorem of this chapter, we can construct general syntactic models using initial models. Let Φ be a $\mathcal{T}$-theory, that is, a left exact functor $\Phi : \mathcal{T} \to \mathbf{Set}$. We define a type theory $\mathcal{T}[\Phi]$ to be the filtered colimit

$$\mathcal{T}[\Phi] = \mathrm{colim}_{(x,a) \in \int_{\mathcal{T}} \Phi} \mathcal{T}/x$$

in $\mathbf{TT}$. Since $\mathcal{T}/x$ is the type theory obtained from $\mathcal{T}$ by freely adjoining a global section of x, the type theory $\mathcal{T}[\Phi]$ is the one obtained from $\mathcal{T}$ by adjoining a global section a of x for any object $x \in \mathcal{T}$ and any element $a \in \Phi(x)$.

5.4.6. Remark. In terms of SOGATs, $\mathcal{T}[\Phi]$ is the SOGAT obtained by replacing each metavariable (assumption) of the environment Φ with a term symbol (axiom) in a similar way to the metavariable replacement.

Since the functor $\mathbf{Mod}(-)$ sends colimits of type theories to limits over $\mathbf{Lex}^{(\emptyset)}$ by Corollary 5.2.12, we have a pullback

$$\begin{array}{ccc} \mathbf{Mod}(\mathcal{T}[\Phi]) & \longrightarrow & \Phi/\mathbf{Th}(\mathcal{T}) \\ \downarrow & \lrcorner & \downarrow \\ \mathbf{Mod}(\mathcal{T}) & \xrightarrow[\mathrm{L}]{} & \mathbf{Th}(\mathcal{T}) \end{array}$$

by Proposition 5.4.5. Therefore, we have an equivalence

$$\mathbf{Mod}(\mathcal{T}[\Phi]) \simeq (\Phi \downarrow \mathrm{L}),$$

and thus the syntactic model $\mathrm{S}(\Phi)$ is obtained from the initial model $\mathrm{I}(\mathcal{T}[\Phi])$ of $\mathcal{T}[\Phi]$ by restricting $\mathrm{I}(\mathcal{T}[\Phi]) : \mathcal{T}[\Phi] \to \mathbf{DFib}_{\mathrm{I}(\mathcal{T}[\Phi])(\diamond)}$ along the canonical morphism $\mathcal{T} \to \mathcal{T}[\Phi]$.

5.4.3 The equivalence of theories and democratic models

We are now ready to prove the main result: for any type theory $\mathcal{T}$, the restriction of the functor $\mathrm{L} : \mathbf{Mod}(\mathcal{T}) \to \mathbf{Th}(\mathcal{T})$ to $\mathbf{Mod}^{\mathrm{dem}}(\mathcal{T}) \subset \mathbf{Mod}(\mathcal{T})$ is an equivalence

$$\mathbf{Mod}^{\mathrm{dem}}(\mathcal{T}) \simeq \mathbf{Th}(\mathcal{T}).$$

This is proved as follows:

1. the unit of the adjunction is invertible (Lemma 5.4.7);
2. the left adjoint $\mathrm{S} : \mathbf{Th}(\mathcal{T}) \to \mathbf{Mod}(\mathcal{T})$ factors through $\mathbf{Mod}^{\mathrm{dem}}(\mathcal{T}) \subset \mathbf{Mod}(\mathcal{T})$ (Lemma 5.4.8);
3. the restriction of the right adjoint L to $\mathbf{Mod}^{\mathrm{dem}}(\mathcal{T}) \subset \mathbf{Mod}(\mathcal{T})$ is conservative (Lemma 5.4.9), and thus the counit at a democratic model is invertible by Item 1 and one of the triangle identities.

5.4.7. Lemma. *The unit of the adjunction* $\mathrm{S} \dashv \mathrm{L} : \mathbf{Th}(\mathcal{T}) \to \mathbf{Mod}(\mathcal{T})$ *is invertible.*

Proof:
Since both functors S and L preserve filtered colimits, it suffices to show that the unit is invertible at $\mathrm{Y}_{\mathcal{T}^{\mathrm{op}}}(x)$ for every object $x \in \mathcal{T}$. From the description of $\mathrm{S}(\mathrm{Y}_{\mathcal{T}^{\mathrm{op}}}(x))$ in Section 5.4.2, the unit is given by the isomorphism $\mathrm{Y}_{\mathcal{T}^{\mathrm{op}}}(x)(y) = \mathcal{T}(x, y) \cong \mathrm{L}(\mathrm{S}(\mathrm{Y}_{\mathcal{T}^{\mathrm{op}}}(x)))(y)$. □

5.4.8. Lemma. *The functor* $\mathrm{S} : \mathbf{Th}(\mathcal{T}) \to \mathbf{Mod}(\mathcal{T})$ *factors through* $\mathbf{Mod}^{\mathrm{dem}}(\mathcal{T}) \subset \mathbf{Mod}(\mathcal{T})$.

Proof:
Since $\mathbf{Mod}^{\mathrm{dem}}(\mathcal{T}) \subset \mathbf{Mod}(\mathcal{T})$ is a coreflective subcategory by Proposition 5.1.7, it is closed under colimits. Thus, it suffices to show that $\mathrm{S}(\mathrm{Y}_{\mathcal{T}^{\mathrm{op}}}(x))$ is democratic for any object $x \in \mathcal{T}$, but this is obvious from the description of $\mathrm{S}(\mathrm{Y}_{\mathcal{T}^{\mathrm{op}}}(x))$ in Section 5.4.2. □

5.4.9. Lemma. *The restriction of* $\mathrm{L} : \mathbf{Mod}(\mathcal{T}) \to \mathbf{Th}(\mathcal{T})$ *to* $\mathbf{Mod}^{\mathrm{dem}}(\mathcal{T}) \subset \mathbf{Mod}(\mathcal{T})$ *is conservative.*

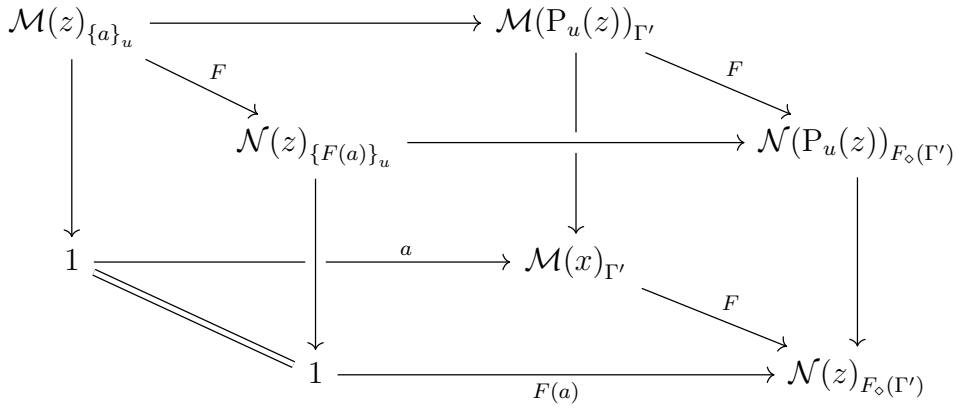
For Lemma 5.4.9, we prepare a sublemma.

5.4.10. Lemma. *Let* $F : \mathcal{M} \to \mathcal{N}$ *be a morphism of models of* $\mathcal{T}$ *such that* $\mathrm{L}(F) : \mathrm{L}(\mathcal{M}) \to \mathrm{L}(\mathcal{N})$ *is invertible. Then* F *induces an isomorphism between fibers*

$$\mathcal{M}(x)_{\Gamma} \to \mathcal{N}(x)_{F_{\diamond}(\Gamma)}$$

for any object $x \in \mathcal{T}$ *and any contextual object* $\Gamma \to \mathcal{M}(\diamond)$.

Proof:
By induction on the contextual object $\Gamma \in \mathcal{M}(\diamond)$. The case when $\Gamma = 1$ is immediate from the assumption that $\mathrm{L}(F)$ is invertible. Suppose that Γ is $\{a\}_u$ for some representable map $u : y \to x$ in $\mathcal{T}$, contextual object $\Gamma' \in \mathcal{M}(\diamond)$ and section $a : \mathcal{M}(\diamond)/\Gamma' \to \mathcal{M}(x)$. Since $\mathcal{M} : \mathcal{T} \to \mathbf{DFib}_{\mathcal{M}(\diamond)}$ commutes with the polynomial functor P_u, the sections $\mathcal{M}(\diamond)/\{a\}_u \to \mathcal{M}(z)$ correspond to the sections of $\mathcal{M}(\mathrm{P}_u(z)) \to \mathcal{M}(x)$ over a. The same argument applies to $\mathcal{N}$ and $F(a)$, and thus we have a commutative diagram



in which the front and back squares are pullbacks. Since F is invertible at Γ' by the induction hypothesis, we conclude that the map $F : \mathcal{M}(z)_{\{a\}_u} \to \mathcal{N}(z)_{\{F(a)\}_u}$ is invertible. □

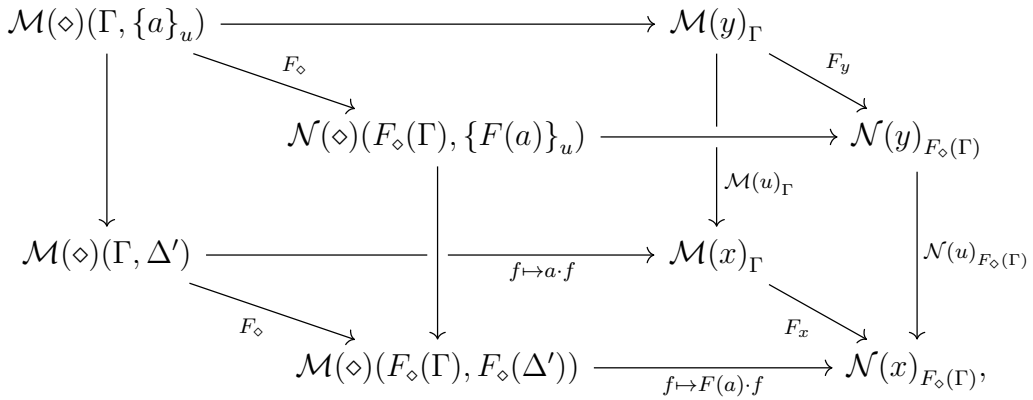
Proof of Lemma 5.4.9:
Let $F : \mathcal{M} \to \mathcal{N}$ be a morphism between democratic models of $\mathcal{T}$ and suppose that $\mathrm{L}(F) : \mathrm{L}(\mathcal{M}) \to \mathrm{L}(\mathcal{N})$ is invertible. We show that F is invertible, that is, $F_x : \mathcal{M}(x) \to \mathcal{N}(x)$ is invertible for any $x \in \mathcal{T}^{\triangleright}$. Lemma 5.4.10 implies that the square

$$\begin{array}{ccc} \mathcal{M}(x) & \xrightarrow{F_x} & \mathcal{N}(x) \\ \downarrow & & \downarrow \\ \mathcal{M}(\diamond) & \xrightarrow[F_\diamond]{} & \mathcal{N}(\diamond) \end{array}$$

is a pullback for any $x \in \mathcal{T}$. Thus, it suffices to show that the functor $F_\diamond : \mathcal{M}(\diamond) \to \mathcal{N}(\diamond)$ is an equivalence.

For the fully-faithfulness of $F_\diamond$, we show by induction on Δ that the map $F_\diamond : \mathcal{M}(\diamond)(\Gamma, \Delta) \to \mathcal{N}(\diamond)(F_\diamond(\Gamma), F_\diamond(\Delta))$ is invertible for any objects $\Gamma, \Delta \in \mathcal{M}(\diamond)$. The case when $\Delta = 1$ is trivial. Suppose that $\Delta = \{a\}_u$ for some representable map $u : y \to x$ in $\mathcal{T}$, object $\Delta' \in \mathcal{M}(\diamond)$ and section $a : \mathcal{M}(\diamond)/\Delta' \to \mathcal{M}(x)$. In

the following commutative diagram



the front and back squares are pullbacks by the definition of $\{a\}_u$, the maps F_y and F_x are invertible by Lemma 5.4.10, and the map $F_\diamond$ is invertible at Δ' by the induction hypothesis. Thus, $F_\diamond$ is invertible at $\{a\}_u$.

For the essential surjectivity of $F_\diamond$, we show by induction on Δ that, for any object $\Delta \in \mathcal{N}(\diamond)$, there exists an object $\Gamma \in \mathcal{M}(\diamond)$ such that $F_\diamond(\Gamma) \cong \Delta$. The case when $\Delta = 1$ is trivial. Suppose that $\Delta = \{b\}_u$ for some representable map $u : y \to x$ in $\mathcal{T}$, object $\Gamma' \in \mathcal{N}(\diamond)$ and section $b : \mathcal{N}(\diamond)/\Delta' \to \mathcal{N}(x)$. By the induction hypothesis, we have an object $\Gamma' \in \mathcal{M}(\diamond)$ and an isomorphism $f : F_\diamond(\Gamma') \cong \Delta'$. By Lemma 5.4.10, we have a section $a : \mathcal{M}(\diamond)/\Gamma' \to \mathcal{M}(x)$ such that $F(a) = b \cdot f$. Then $F_\diamond(\{a\}_u) \cong \{b\}_u$. □

Chapter 6

∞-type theories

In this chapter, we introduce a notion of an ∞-*type theory* to tackle the conjecture that the homotopy theory of type theories with intensional identity types is equivalent to the homotopy theory of $(\infty, 1)$-categories with finite limits [100].

Once we identify a type theory with a CwR, its higher categorical generalization makes sense.

6.0.1. Definition. An $(\infty, 1)$-*category with representable maps* ($(\infty, 1)$-*CwR*) is an $(\infty, 1)$-category $\mathcal{C}$ with finite limits equipped with a pullback-stable class $\mathrm{R}_{\mathcal{C}}$ of exponentiable arrows. Arrows in $\mathrm{R}_{\mathcal{C}}$ are called *representable maps*. A *morphism of* $(\infty, 1)$-*CwRs* $\mathcal{C} \to \mathcal{D}$ is a functor $F : \mathcal{C} \to \mathcal{D}$ preserving finite limits, representable maps and pushforwards along representable maps.

6.0.2. Definition. An ∞-*type theory* is a small $(\infty, 1)$-CwR. For $1 \leq n < \infty$, by an n-*type theory*, we mean an ∞-type theory whose underlying $(\infty, 1)$-category is an $(n, 1)$-category.

A 1-type theory in this sense is of course a type theory in the sense of Definition 3.2.3. It turns out that a model of an ∞-type theory may be regarded as a *non-split* model of a 1-type theory that naturally arises in the categorical semantics of type theory. For example, let $\mathbb{D}$ be the 1-type theory freely generated by a representable map $\partial : E \to U$ (Example 3.2.6) and $\mathbb{D}_1^-$ the 2-type theory freely generated by a representable map $\partial : E \to U$. Models of $\mathbb{D}$ are ordinary natural models. Models of $\mathbb{D}_1^-$ are then $(2, 1)$-categorical analogue of natural models, that is, a model of $\mathbb{D}_1^-$ consists of the following data:

- a $(2, 1)$-category $\mathcal{M}(\diamond)$ with a final object;
- a representable map $\mathcal{M}(\partial) : \mathcal{M}(E) \to \mathcal{M}(U)$ of groupoid-valued presheaves over $\mathcal{M}(\diamond)$.

A groupoid-valued presheaf is a functor $\mathcal{M}(\diamond)^{\mathrm{op}} \to \mathbf{Gpd}$ in the $(2,1)$-categorical sense and preserves composition only up to isomorphism. We could try to interpret the action $A \cdot f$ of a substitution in a type theory as the right action of the groupoid-valued presheaf, but this interpretation satisfies the substitution law $A \cdot (f \circ g) = (A \cdot f) \cdot g$ only up to isomorphism. In this sense, models of $\mathbb{D}_1^-$ are not models of $\mathbb{D}$ but seem to interpret components of $\mathbb{D}$ with a weaker notion of equality.

Constructing models of an ∞-type theory is often easier than constructing models of a 1-type theory. For example, any category $\mathcal{C}$ with finite limits induces a groupoid-valued presheaf $\mathcal{C}^{\mathrm{op}} \ni x \mapsto \mathrm{k}(\mathcal{C}/x) \in \mathbf{Gpd}$, where the right action is given by pullbacks, and this is part of a model of $\mathbb{D}_1^-$. Since pullbacks are determined up to isomorphism, this presheaf hardly preserves composition strictly. Therefore, to get a model of $\mathbb{D}$ from a category with finite limits, we need a splitting technique [78, 116], that is, we have to replace the groupoid-valued presheaf by a set-valued presheaf.

We are faced with a dilemma: we can easily get models of the 2-type theory $\mathbb{D}_1^-$ from structured categories; in the real world, however, we work with the 1-type theory $\mathbb{D}$, not $\mathbb{D}_1^-$. We thus want to justify interpreting $\mathbb{D}$ in models of $\mathbb{D}_1^-$ to use plenty of models of $\mathbb{D}_1^-$ for the study of $\mathbb{D}$. Such a situation is called a *coherence problem* which is traditionally formulated as the problem of interpreting a 1-type theory in non-split models, but here non-split models are replaced by models of an ∞-type theory.

Coherence problems become much more serious when we try to interpret a type theory in structured $(\infty,1)$-categories. The idea of homotopy type theory is to interpret Martin-Löf's intensional type theory in structured $(\infty,1)$-categories, but the interpretation is far from obvious because most equations hold only up to homotopy in $(\infty,1)$-categories while Martin-Löf type theory features a more strict notion of equality, judgmental equality. It is not difficult to show that certain structured $(\infty,1)$-categories are models of an ∞-type theory, and then a coherence problem is again the problem of interpreting a 1-type theory in models of an ∞-type theory.

Once a coherence problem is solved, we can often establish the correspondence between theories and non-split models. For example, Clairambault and Dybjer [39, 40] showed that theories over Martin-Löf' extensional type theory are biequivalent to locally cartesian closed categories. Kapulkin and Lumsdaine [100] conjectured that theories over Martin-Löf's intensional type theory are equivalent to locally cartesian closed $(\infty,1)$-categories in a suitable sense. Such a strong correspondence between theories and non-split models justifies using the *internal language* of a non-split model.

The notion of ∞-type theories provides a precise and unified formulation of general coherence problems in both 1-categorical and $(\infty,1)$-categorical semantics of type theories. We view a coherence problem as the problem of interpreting a 1-type theory in a model of an ∞-type theory. Since 1-type theories are special

∞-type theories, a coherence problem is now formulated in the language of ∞-type theories and related concepts so that we can deal with both 1-categorical and $(\infty,1)$-categorical coherence problems in the same language. Of course, this is just a rephrasing of a coherence problem and does not provide any technique of solving the problem. However, it does provide a technique of strengthening a coherence theorem to the correspondence between theories and non-split models. We demonstrate that if the coherence problem between a 1-type theory $\mathcal{T}$ and an ∞-type theory $\mathcal{T}$ is solved, then we can systematically prove that the $(\infty,1)$-category $\mathbf{Mod}^{\mathrm{dem}}(\mathcal{T}_\infty)$ of democratic models of $\mathcal{T}_\infty$ is a *localization* of the category $\mathbf{Th}(\mathcal{T})$ of theories, that is, $\mathbf{Mod}^{\mathrm{dem}}(\mathcal{T}_\infty)$ is obtained from $\mathbf{Th}(\mathcal{T})$ by formally inverting some morphisms. As an application, we explain the idea of solving the conjecture by Kapulkin and Lumsdaine [100] that the $(\infty,1)$-category of finitely complete $(\infty,1)$-categories is a localization of the category of theories over the dependent type theory with intensional identity types.

We introduce basic concepts around ∞-type theories in Section 6.1. There are two important concepts not found in the theory of 1-type theories: *univalent* representable maps; the *representable map classifier* of right fibrations over an $(\infty,1)$-category. ∞-type theories with univalence are considered much better than ones without univalence. For example, various type-theoretic structures become unique up to contractible choice under univalence and thus are treated as properties rather than structures. The representable map classifier is the one and only source of representable maps of right fibrations: for any $(\infty,1)$-category $\mathcal{C}$, we obtain a representable map of right fibrations over $\mathcal{C}$ called the *generic representable map* for free; any representable map of right fibrations over $\mathcal{C}$ is the pullback of the generic representable map along a unique map.

We study some concrete examples of ∞-type theories in Section 6.2. Unlike the 1-categorical case, we have not yet found syntactic counterparts of ∞-type theories, so *free constructions* are a major source of examples of ∞-type theories. The most fundamental example of an ∞-type theory is the ∞-analogue of the dependent type theory with Σ-types, unit type, and extensional identity types which we refer to as $\mathbb{E}_\infty$. We show that the $(\infty,1)$-category $\mathbf{Th}(\mathbb{E}_\infty)$ of theories over $\mathbb{E}_\infty$ is equivalent to the $(\infty,1)$-category $(\infty,1)$-$\mathbf{Lex}$ of small $(\infty,1)$-categories with finite limits. This is understood as an ∞-analogue of the equivalence of theories with extensional identity types and categories with finite limits [39, 40]. Although we have not found a syntactic counterpart of $\mathbb{E}_\infty$, the equivalence $\mathbf{Th}(\mathbb{E}_\infty) \simeq (\infty,1)$-$\mathbf{Lex}$ has an interesting consequence: from a universal property of $\mathbb{E}_\infty$, we can derive a universal property of $(\infty,1)$-$\mathbf{Lex}$.

In Section 6.3, we develop a technique of establishing the correspondence between theories and non-split models that applies to both 1-categorical and $(\infty,1)$-categorical coherence problems. As a special case, we sketch the idea of solving the internal language conjecture of Kapulkin and Lumsdaine [100]: the $(\infty,1)$-category of finitely complete $(\infty,1)$-categories is a localization of the category of theories over the type theory with intensional identity types.

In this chapter, we assume that the reader is familiar with basic $(\infty, 1)$-category theory [117, 38, 145].

The contents of this chapter are joint work with Hoang Kim Nguyen [130].

6.1 The theory of ∞-type theories

6.1.1. Definition. A functor $p : A \to \mathcal{C}$ between $(\infty, 1)$-category is a *right fibration* if the square

$$\begin{array}{ccc} A^{\to} & \xrightarrow{\mathrm{cod}} & A \\ {\scriptstyle p^{\to}}\downarrow & & \downarrow{\scriptstyle p} \\ \mathcal{C}^{\to} & \xrightarrow[\mathrm{cod}]{} & \mathcal{C} \end{array}$$

is a pullback of $(\infty, 1)$-categories. For an $(\infty, 1)$-category $\mathcal{C}$, by a *right fibration over* $\mathcal{C}$, we mean a right fibration of the form $A \to \mathcal{C}$. We write $\mathbf{RFib}_{\mathcal{C}} \subset (\infty, 1)\text{-}\mathbf{Cat}\,/\mathcal{C}$ for the full subcategory spanned by the right fibrations over $\mathcal{C}$.

6.1.2. Definition. An $(\infty, 1)$-*category with representable maps* ($(\infty, 1)$-*CwR*) is an $(\infty, 1)$-category $\mathcal{C}$ with finite limits equipped with a pullback-stable class $\mathrm{R}_{\mathcal{C}}$ of exponentiable arrows. Arrows in $\mathrm{R}_{\mathcal{C}}$ are called *representable maps*. A *morphism of* $(\infty, 1)$-*CwRs* $\mathcal{C} \to \mathcal{D}$ is a functor $F : \mathcal{C} \to \mathcal{D}$ preserving finite limits, representable maps and pushforwards along representable maps. We write $(\infty, 1)$-$\mathbf{CwR}$ for the $(\infty, 1)$-category of $(\infty, 1)$-CwRs and morphisms of $(\infty, 1)$-CwR.

6.1.3. Example. For any $(\infty, 1)$-category $\mathcal{C}$, the $(\infty, 1)$-category $\mathbf{RFib}_{\mathcal{C}}$ of right fibrations over $\mathcal{C}$ is an $(\infty, 1)$-CwR in which a map is representable if it has a right adjoint seen as a functor.

6.1.4. Definition. An ∞-*type theory* is a small $(\infty, 1)$-CwR. A *morphism of* ∞-*type theories* is a morphism of $(\infty, 1)$-CwRs. We write ∞-$\mathbf{TT}$ for the $(\infty, 1)$-category of ∞-type theories which is nothing but $(\infty, 1)$-$\mathbf{CwR}$. For $1 \leq n < \infty$, by an n-*type theory*, we mean an ∞-type theory whose underlying $(\infty, 1)$-category is an $(n, 1)$-category.

6.1.5. Definition. Let $\mathcal{T}$ be an ∞-type theory. A *model* $\mathcal{M}$ *of* $\mathcal{T}$ consists of the following data:

- an $(\infty, 1)$-category $\mathcal{M}(\diamond)$ with a terminal object;
- a morphism of $(\infty, 1)$-CwRs $\mathcal{M} : \mathcal{T} \to \mathbf{RFib}_{\mathcal{M}(\diamond)}$.

We view a model of $\mathcal{T}$ as a functor $\mathcal{T}^{\triangleright} \to (\infty, 1)\text{-}\mathbf{Cat}$ where $\mathcal{T}^{\triangleright}$ is the $(\infty, 1)$-category obtained from $\mathcal{T}$ by adjoining a new final object $\diamond$.

6.1.6. Definition. Let $\mathcal{T}$ be an ∞-type theory and $\mathcal{M}$ and $\mathcal{N}$ models of $\mathcal{T}$. A *morphism $F : \mathcal{M} \to \mathcal{N}$ of models of $\mathcal{T}$* is a natural transformation $F : \mathcal{M} \Rightarrow \mathcal{N} : \mathcal{T}^{\triangleright} \to (\infty, 1)\text{-}\mathbf{Cat}$ satisfying the following conditions:

1. the component $F_{\diamond} : \mathcal{M}(\diamond) \to \mathcal{N}(\diamond)$ preserves final objects;

2. for any representable map $u : x \to y$ in $\mathcal{T}$, the square

$$\begin{array}{ccc} \mathcal{M}(x) & \xrightarrow{F_x} & \mathcal{N}(x) \\ {\scriptstyle \mathcal{M}(u)}\downarrow & & \downarrow{\scriptstyle \mathcal{N}(u)} \\ \mathcal{M}(y) & \xrightarrow[F_y]{} & \mathcal{N}(y) \end{array} \tag{6.1}$$

   satisfies the Beck-Chevalley condition.

We write $\mathbf{Mod}(\mathcal{T})$ for the subcategory of $\mathbf{Fun}(\mathcal{T}^{\triangleright}, (\infty, 1)\text{-}\mathbf{Cat})$ spanned by the models of $\mathcal{T}$ and the morphisms between them.

We have presented the proofs in Chapter 5 in such a way that they also work in the $(\infty, 1)$-categorical context. Therefore, we have analogous results including the following.

6.1.7. Proposition. *The $(\infty, 1)$-category ∞-$\mathbf{TT}$ is compactly generated and limits and filtered colimits of ∞-type theories are computed component-wise.*

6.1.8. Proposition. *For any ∞-type theory, the $(\infty, 1)$-category $\mathbf{Mod}(\mathcal{T})$ is compactly generated and the forgetful functor $\mathbf{Mod}(\mathcal{T}) \to \mathbf{Fun}(\mathcal{T}^{\triangleright}, (\infty, 1)\text{-}\mathbf{Cat})$ preserves limits and filtered colimits and is conservative.*

6.1.9. Definition. Let $\mathcal{T}$ be an ∞-type theory. A *theory over $\mathcal{T}$* or *$\mathcal{T}$-theory* is a left exact functor $\mathcal{T} \to \mathbf{Space}$ where $\mathbf{Space}$ is the $(\infty, 1)$-category of spaces. We write $\mathbf{Th}(\mathcal{T})$ for the $(\infty, 1)$-category of $\mathcal{T}$-theories, that is, the full subcategory of $\mathbf{Fun}(\mathcal{T}, \mathbf{Space})$ spanned by the left exact functors.

The internal language functor $\mathrm{L} : \mathbf{Mod}(\mathcal{T}) \to \mathbf{Th}(\mathcal{T})$ is defined in the same way as the 1-categorical case. We also have the full subcategory $\mathbf{Mod}^{\mathrm{dem}}(\mathcal{T}) \subset \mathbf{Mod}(\mathcal{T})$ spanned by the democratic models.

6.1.10. Theorem. *For any ∞-type theory $\mathcal{T}$, the internal language functor $\mathrm{L} : \mathbf{Mod}(\mathcal{T}) \to \mathbf{Th}(\mathcal{T})$ has a left adjoint and induces an equivalence*

$$\mathbf{Mod}^{\mathrm{dem}}(\mathcal{T}) \simeq \mathbf{Th}(\mathcal{T}).$$

6.1.1 Univalent representable maps

We review the theory of *univalent maps* [69, 141, 142] in the context of $(\infty,1)$-CwR.

6.1.11. Definition. For objects x and y of an $(\infty,1)$-category $\mathcal{C}$ with finite products, let $\mathrm{Map}(x,y) \to \mathcal{C}$ denote the right fibration whose fiber over z is $\mathcal{C}/z(x\times z, y\times z) \simeq \mathcal{C}(x\times z, y)$. It is defined by the pullback

$$\begin{array}{ccc} \mathrm{Map}(x,y) & \longrightarrow & \mathcal{C}/y \\ \downarrow & \lrcorner & \downarrow \\ \mathcal{C} & \xrightarrow[(x\times -)]{} & \mathcal{C}. \end{array}$$

If $\mathrm{Map}(x,y)$ is representable, we write $\underline{\mathrm{Map}}(x,y)$ for the representing object. Let $\mathrm{Eqv}(x,y)$ denote the subfibration of $\mathrm{Map}(x,y)$ spanned by the invertible arrows $x\times z \simeq y\times z$. If $\mathrm{Eqv}(x,y)$ is representable, we write $\underline{\mathrm{Eqv}}(x,y)$ for the representing object.

6.1.12. Definition. Let $u : y \to x$ be an arrow in an $(\infty,1)$-category $\mathcal{C}$ with finite limits. We regard $u\times x : y\times x \to x\times x$ and $x\times u : x\times y \to x\times x$ as objects of $\mathcal{C}/x\times x$ and define $\mathrm{Eqv}(u)$ to be the right fibration $\mathrm{Eqv}(u\times x, x\times u) \to \mathcal{C}/x\times x$. If $\mathrm{Eqv}(u)$ is representable, we write $\underline{\mathrm{Eqv}}(u)$ for the representing object.

By definition, an arrow $z \to \underline{\mathrm{Eqv}}(u)$ corresponds to a triple (v_1, v_2, w) consisting of arrows $v_1, v_2 : z \to x$ and an invertible arrow $w : v_1^* y \simeq v_2^* y$ over z.

6.1.13. Definition. Let $u : y \to x$ be an arrow in an $(\infty,1)$-category $\mathcal{C}$ with finite limits such that $\mathrm{Eqv}(u)$ is representable. We have a section $\lceil \mathrm{id} \rceil : x \to \underline{\mathrm{Eqv}}(u)$ over the diagonal $\Delta : x \to x\times x$ corresponding to the identity on y. We say u is *univalent* if the arrow $\lceil \mathrm{id} \rceil : x \to \underline{\mathrm{Eqv}}(u)$ is invertible.

For an $(\infty,1)$-category $\mathcal{C}$ with pullbacks, let $\mathcal{O}_{\mathcal{C}}$ denote the cartesian fibration over $\mathcal{C}$ defined by the codomain functor $\mathcal{C}^{\to} \to \mathcal{C}$ and $\mathcal{O}_{\mathcal{C}}^{(\mathrm{all})}$ the largest right fibration over $\mathcal{C}$ contained in $\mathcal{O}_{\mathcal{C}}$.

6.1.14. Proposition. *Let $u : y \to x$ be an arrow in an $(\infty,1)$-category $\mathcal{C}$ with finite limits such that* $\mathrm{Eqv}(u)$ *is representable. We identify u with the corresponding map of right fibrations $\mathcal{C}/x \to \mathcal{O}_{\mathcal{C}}^{(\mathrm{all})}$ via Yoneda. The following are equivalent:*

1. *u is univalent;*

2. *the square*

$$\begin{array}{ccc} \mathcal{C}/x & \xrightarrow{\quad u \quad} & \mathcal{O}_{\mathcal{C}}^{(\mathrm{all})} \\ {\scriptstyle\Delta}\downarrow & & \downarrow{\scriptstyle\Delta} \\ \mathcal{C}/x \times x & \xrightarrow[u\times u]{} & \mathcal{O}_{\mathcal{C}}^{(\mathrm{all})} \times_{\mathcal{C}} \mathcal{O}_{\mathcal{C}}^{(\mathrm{all})} \end{array}$$

   is a pullback of right fibrations over $\mathcal{C}$;

3. *$u : \mathcal{C}/x \to \mathcal{O}_{\mathcal{C}}^{(\mathrm{all})}$ is a monomorphism of right fibrations over $\mathcal{C}$.*

Proof:
The same proof as [69, Proposition 3.8 (1)–(3)] works only assuming the representability of $\mathrm{Eqv}(u)$. □

6.1.15. Proposition. *Let x and y be exponentiable objects in an $(\infty,1)$-category $\mathcal{C}$ with finite limits.*

1. *The right fibration $\mathrm{Eqv}(x,y) \to \mathcal{C}$ is representable.*

2. *Let $\mathcal{D}$ be an $(\infty,1)$-category with finite limits and $F : \mathcal{C} \to \mathcal{D}$ a left exact functor. If F sends x and y to exponentiable objects over $F(z)$ and commutes with exponentiation by x and y, then the canonical arrow $F\big(\underline{\mathrm{Eqv}}(x,y)\big) \to \underline{\mathrm{Eqv}}(F(x),F(y))$ is invertible.*

Proof:
The right fibration $\mathrm{Eqv}(x,y)$ is equivalent to the right fibration $\mathrm{BiInv}(x,y)$ of bi-invertible arrows whose fiber over $z \in \mathcal{C}$ is the space of tuples (u,v,η,w,ε) consisting of arrows $u : x \times z \to y \times z$ and $v,w : y \times z \to x \times z$ over z and homotopies $\eta : v \circ u \simeq \mathrm{id}$ and $\varepsilon : u \circ w \simeq \mathrm{id}$ over z. The right fibration $\mathrm{BiInv}(x,y)$ is representable by the exponentiability of x and y. The second assertion is clear from the construction of the representable object for $\mathrm{BiInv}(x,y)$. □

6.1.16. Corollary. *Let $u : y \to x$ be a representable map in an $(\infty,1)$-CwR.*

1. *The right fibration $\mathrm{Eqv}(u) \to \mathcal{C}/x \times x$ is representable.*

2. *If u is univalent, so is $F(u)$ for any morphism of $(\infty,1)$-CwRs $F : \mathcal{C} \to \mathcal{D}$.*

6.1.17. Definition. Let $u : y \to x$ be a univalent representable map in an $(\infty,1)$-CwR. We say an arrow $u' : y' \to x'$ is *u-small* if there exists a (necessarily unique) pullback square of the form

$$\begin{array}{ccc} y' & \dashrightarrow & y \\ {\scriptstyle u'}\downarrow & \lrcorner & \downarrow{\scriptstyle u} \\ x' & \dashrightarrow & x. \end{array}$$

6.1.2 The representable map classifier

Let $\mathcal{C}$ be a small $(\infty,1)$-category. Since the representability of a map of right fibrations over $\mathcal{C}$ is a local property, we see that the class of representable maps of right fibrations over $\mathcal{C}$ is local in the sense of Lurie [117]. Moreover, for any right fibration A over $\mathcal{C}$, the space of representable maps over A is essentially small: for any representable map $f : B \to A$, the cardinality of the fiber of B over a section $a : \mathcal{C}/x \to A$ is bounded by the cardinality of $\mathcal{C}$ since a^*B is representable. Hence, the class of representable maps of right fibrations over $\mathcal{C}$ has a classifying object by [117, Proposition 6.1.6.3] which we refer to as the *representable map classifier*.

The representable map classifier is a right fibration $\mathrm{O}_{\mathcal{C}}$ over $\mathcal{C}$ equipped with a *generic representable map* $\rho_{\mathcal{C}} : \widetilde{\mathrm{O}}_{\mathcal{C}} \to \mathrm{O}_{\mathcal{C}}$ in the sense that $\rho_{\mathcal{C}}$ is a representable map of right fibrations over $\mathcal{C}$ and, for any representable map $f : B \to A$ of right fibrations over $\mathcal{C}$, there exists a unique pullback of the form

$$\begin{array}{ccc} B & \dashrightarrow & \widetilde{\mathrm{O}}_{\mathcal{C}} \\ {\scriptstyle f}\downarrow & \lrcorner & \downarrow{\scriptstyle \rho_{\mathcal{C}}} \\ A & \dashrightarrow & \mathrm{O}_{\mathcal{C}}. \end{array}$$

By [69, Theorem 3.9], the generic representable map $\rho_{\mathcal{C}}$ is a univalent representable map in $\mathbf{RFib}_{\mathcal{C}}$.

6.1.18. Proposition. *For any small $(\infty,1)$-category $\mathcal{C}$ with pullbacks, the right fibration $\mathcal{O}_{\mathcal{C}}^{(\mathrm{all})}$ is the representable map classifier.*

This is immediate from the following observation.

6.1.19. Lemma. *Let $\mathcal{C}$ be an $(\infty,1)$-category with pullbacks and $x \in \mathcal{C}$ an object. A map $f : A \to \mathcal{C}/x$ of right fibrations over $\mathcal{C}$ is representable if and only if the right fibration A is representable.*

Proof:
By definition. □

Proof of Proposition 6.1.18:
By Lemma 6.1.19, the space of representable maps over $\mathcal{C}/x$ is equivalent to $\mathrm{k}(\mathcal{C}/x)$ which is the fiber of $\mathcal{O}_{\mathcal{C}}^{(\mathrm{all})}$ over x. □

6.2 Type-theoretic structures

In this section, we explore connections between ∞-type-theoretic structures and $(\infty,1)$-categorical structures. In Section 6.2.1 we define an ∞-type theory $\mathbb{E}_\infty$

such that

$$\mathbf{Th}(\mathbb{E}_\infty) \simeq (\infty,1)\text{-}\mathbf{Lex} \tag{6.2}$$

where the right side is the $(\infty,1)$-category of small $(\infty,1)$-categories with finite limits. Moreover, $\mathbb{E}_\infty$ is an ∞-analogue of Martin-Löf type theory with Σ-types, unit type, and extensional identity types. Therefore, Eq. (6.2) is understood as an ∞-analogue of the equivalence of theories with Σ-types, unit type, and extensional identity types and categories with finite limits [39, 40]. Another way of viewing Eq. (6.2) is that $\mathbb{E}_\infty$ is a *presentation* of the $(\infty,1)$-category $(\infty,1)$-**Lex**. This presentation has the advantage that the ∞-type theory has a simple universal property from which we can derive a universal property of $(\infty,1)$-**Lex**.

In Section 6.2.2 we define an ∞-type theory $\mathbb{E}_\infty^\Pi$ such that theories over $\mathbb{E}_\infty^\Pi$ are equivalent to locally cartesian closed $(\infty,1)$-categories. Since ∞-type theories themselves are also $(\infty,1)$-categories with structure, we can even find an ∞-type theory $\mathbb{R}_\infty$ such that theories over $\mathbb{R}_\infty$ are equivalent to ∞-type theories (Section 6.2.3).

6.2.1 Finitely complete $(\infty,1)$-categories

We define an ∞-type theory $\mathbb{E}_\infty$ such that theories over $\mathbb{E}_\infty$ are equivalent to $(\infty,1)$-categories with finite limits.

6.2.1. Definition. Let $\mathcal{C}$ be an $(\infty,1)$-CwR and $\partial : E \to U$ a representable map in $\mathcal{C}$.

- A *unit type structure on* ∂ is a pullback square of the form

$$\begin{array}{ccc} 1 & \overset{*}{\dashrightarrow} & E \\ \| & \lrcorner & \downarrow{\scriptstyle \partial} \\ 1 & \underset{1}{\dashrightarrow} & U. \end{array}$$

- A Σ-*type structure on* ∂ is a pullback square of the form

$$\begin{array}{ccc} \mathrm{dom}(\partial \otimes \partial) & \overset{\mathsf{pair}}{\dashrightarrow} & E \\ {\scriptstyle \partial\otimes\partial}\downarrow & \lrcorner & \downarrow{\scriptstyle \partial} \\ \mathrm{cod}(\partial \otimes \partial) & \underset{\Sigma}{\dashrightarrow} & U. \end{array}$$

- An Id-*type structure on* ∂ is a pullback square of the form

$$\begin{array}{ccc} E & \overset{\mathsf{refl}}{\dashrightarrow} & E \\ {\scriptstyle \Delta}\downarrow & \lrcorner & \downarrow{\scriptstyle \partial} \\ E \times_U E & \underset{\mathsf{Id}}{\dashrightarrow} & U. \end{array}$$

When ∂ is univalent, these are properties of ∂ rather than structures as follows.

6.2.2. Proposition. *Let $\partial : E \to U$ be a univalent representable map in an $(\infty, 1)$-CwR. Then unit type structures, Σ-type structures, and Id-type structures are unique up to contractible choice. Moreover, we have the following:*

1. *∂ has a unit type structure if and only if all the identity arrows are ∂-small;*
2. *∂ has a Σ-type structure if and only if ∂-small maps are closed under composition;*
3. *∂ has an Id-type structure if and only if ∂-small maps are closed under equalizers: for any ∂-small map $u : y \to x$, any object $x' \in \mathcal{C}/x$ and any arrows $v_1, v_2 : x' \to y$ in $\mathcal{C}/x$, the equalizer $x'' \to x'$ of v_1 and v_2 in $\mathcal{C}/x$ is ∂-small.*

Proof:
Since those structures are defined by pullbacks of ∂, they are unique up to contractible choice by univalence. The rest is straightforward. □

6.2.3. Definition. By a *finitely complete universe* in an $(\infty, 1)$-CwR, we mean a univalent representable map equipped with a unit type structure, a Σ-type structure, and an Id-type structure. We define $\mathbb{E}_\infty$ to be the free ∞-type theory generated by a left exact universe $\partial : E \to U$.

6.2.4. Theorem. *The functor* $\mathrm{ev}_\diamond : \mathbf{Mod}^{\mathrm{dem}}(\mathbb{E}_\infty) \to (\infty, 1)\text{-}\mathbf{Cat}$ *factors through* $(\infty, 1)\text{-}\mathbf{Lex}$ *and induces an equivalence*

$$\mathbf{Mod}^{\mathrm{dem}}(\mathbb{E}_\infty) \simeq (\infty, 1)\text{-}\mathbf{Lex}\,.$$

6.2.5. Corollary. *We have an equivalence* $\mathbf{Th}(\mathbb{E}_\infty) \simeq (\infty, 1)\text{-}\mathbf{Lex}$.

Proof:
By Theorem 6.1.10. □

To prove Theorem 6.2.4, we prepare a few lemmas.

6.2.6. Lemma. *An arrow in $\mathbb{E}_\infty$ is a representable map if and only if it is ∂-small.*

Proof:
Let $\mathbb{E}'_\infty$ denote the $(\infty, 1)$-CwR whose underlying $(\infty, 1)$-category is the same as $\mathbb{E}_\infty$ and representable maps are the ∂-small maps. $\mathbb{E}'_\infty$ is indeed an $(\infty, 1)$-CwR because ∂-small maps are closed under identities and composition by Proposition 6.2.2. By the initiality of $\mathbb{E}_\infty$, the inclusion $\mathbb{E}'_\infty \to \mathbb{E}_\infty$ has a section, and thus $\mathbb{E}'_\infty \simeq \mathbb{E}_\infty$. □

6.2.7. Definition. Let $\mathcal{M}$ be a model of an ∞-type theory $\mathcal{T}$. By a *display map*, we mean a morphism $f : \Delta \to \Gamma$ in $\mathcal{M}(\diamond)$ that is equivalent over Γ to $\mathrm{p}_u : \{a\}_u \to \Gamma$ for some representable map $u : y \to x$ in $\mathcal{T}$ and some section $a : \mathcal{M}(\diamond)/\Gamma \to \mathcal{M}(x)$. By definition, display maps are stable under pullbacks and any morphism of models of $\mathcal{T}$ preserves pullbacks of display maps.

6.2.8. Lemma. *Let $\mathcal{M}$ be a model of $\mathbb{E}_\infty$. An arrow $f : \Delta \to \Gamma$ in $\mathcal{M}(\diamond)$ is a display map if and only if the map $f : \mathcal{M}(\diamond)/\Delta \to \mathcal{M}(\diamond)/\Gamma$ of right fibrations over $\mathcal{M}(\diamond)$ is $\mathcal{M}(\partial)$-small.*

Proof:
By Lemma 6.2.6. □

Proof of Theorem 6.2.4:
Let $\mathcal{M}$ be a democratic model of $\mathbb{E}_\infty$. To show that $\mathcal{M}(\diamond)$ has finite limits, it remains to show that it has pullbacks. It suffices to show that all the morphisms of $\mathcal{M}(\diamond)$ are display maps, which also proves that any morphism between democratic models of $\mathbb{E}_\infty$ preserves finite limits in the base categories. By Proposition 6.2.2 and Lemma 6.2.8, the class of display maps in $\mathcal{M}(\diamond)$ is closed under identities, composition, and equalizers. Since $\mathcal{M}$ is democratic, the final projection $\Gamma \to 1$ is a composite of display maps for any object $\Gamma \in \mathcal{M}(\diamond)$. Then, for any morphism $f : \Gamma \to \Delta$ in $\mathcal{M}(\diamond)$, we can form the graph $\langle f \rangle$ of f by the equalizer of $f \circ \pi_1, \pi_2 : \Gamma \times \Delta \to \Delta$. The projection $\langle f \rangle \to \Gamma \times \Delta \to \Delta$ is a display map and Γ is equivalent over Δ to $\langle f \rangle$, and thus f is a display map.

We have proved that the functor $\mathrm{ev}_\diamond : \mathbf{Mod}^{\mathrm{dem}}(\mathbb{E}_\infty) \to (\infty,1)\text{-}\mathbf{Cat}$ factors through $(\infty,1)\text{-}\mathbf{Lex}$. We construct an inverse of $\mathrm{ev}_\diamond : \mathbf{Mod}^{\mathrm{dem}}(\mathbb{E}_\infty) \to (\infty,1)\text{-}\mathbf{Lex}$. Let $\mathcal{C}$ be an $(\infty,1)$-category with finite limits. The representable map classifier $\rho_{\mathcal{C}} : \widetilde{\mathrm{O}}_{\mathcal{C}} \to \mathrm{O}_{\mathcal{C}}$ is a univalent representable map and, since $\mathcal{C}$ has finite limits, representable maps of right fibrations over $\mathcal{C}$ are closed under equalizers. Therefore, $\rho_{\mathcal{C}}$ is a finitely complete universe in $\mathbf{RFib}_{\mathcal{C}}$ and thus determines a model $\mathrm{O}_{\mathcal{C}}$ of $\mathbb{E}_\infty$. Since the map $\mathcal{C}/x \to \mathcal{C}/1$ is representable for any object $x \in \mathcal{C}$, the model $\mathrm{O}_{\mathcal{C}}$ is democratic. Proposition 6.1.18 ensures that the construction $\mathcal{C} \mapsto \mathrm{O}_{\mathcal{C}}$ is functorial.

We show that the construction $\mathcal{C} \mapsto \mathrm{O}_{\mathcal{C}}$ is an inverse of $\mathrm{ev}_\diamond$. By definition, the base category of $\mathrm{O}_{\mathcal{C}}$ is $\mathcal{C}$. For the other equivalence, let $\mathcal{M}$ be a democratic model of $\mathbb{E}_\infty$. We show that $\mathcal{M}$ is naturally equivalent to $\mathrm{O}_{\mathcal{M}(\diamond)}$. By the definition of the representable map classifier, we have a unique pullback

$$\begin{array}{ccc} \mathcal{M}(E) & \dashrightarrow & \widetilde{\mathrm{O}}_{\mathcal{M}(\diamond)} \\ {\scriptstyle \mathcal{M}(\partial)}\downarrow & \lrcorner & \downarrow{\scriptstyle \rho_{\mathcal{M}(\diamond)}} \\ \mathcal{M}(U) & \underset{f}{\dashrightarrow} & \mathrm{O}_{\mathcal{M}(\diamond)}. \end{array}$$

We show that the map f is a monomorphism and surjective. Since $\mathcal{M}(\partial)$ is univalent, the map f is a monomorphism by [69, Corollary 3.10]. By Proposition 6.1.18, the objects of $\mathrm{O}_{\mathcal{M}(\diamond)}$ are the morphisms of $\mathcal{M}(\diamond)$. Then, Lemma 6.2.8 implies that the image of f is the class of display maps in $\mathcal{M}(\diamond)$. Since all the morphisms in $\mathcal{M}(\diamond)$ are display maps, we conclude that f is surjective. □

Consider the image of the representable map $\partial : E \to U$ by the inclusion

$$\mathbb{E}_\infty \to \mathbf{Th}(\mathbb{E}_\infty)^{\mathrm{op}} \simeq \mathbf{Mod}^{\mathrm{dem}}(\mathbb{E}_\infty)^{\mathrm{op}} \simeq (\infty,1)\text{-}\mathbf{Lex}^{\mathrm{op}}.$$

For an $(\infty,1)$-category $\mathcal{C}$ with finite limits, we have

$$\begin{aligned}\mathbf{Th}(\mathbb{E}_\infty)(\mathrm{Y}_{\mathbb{E}_\infty^{\mathrm{op}}}(U), \mathrm{L}(\mathrm{O}_\mathcal{C})) &\simeq \mathbf{RFib}_\mathcal{C}(\mathcal{C}, \mathrm{O}_\mathcal{C}) \\ &\simeq \mathrm{k}(\mathcal{C}) \\ \mathbf{Th}(\mathbb{E}_\infty)(\mathrm{Y}_{\mathbb{E}_\infty^{\mathrm{op}}}(E), \mathrm{L}(\mathrm{O}_\mathcal{C})) &\simeq \mathbf{RFib}_\mathcal{C}\left(\mathcal{C}, \widetilde{\mathrm{O}}_\mathcal{C}\right) \\ &\simeq \mathrm{k}(1/\mathcal{C}).\end{aligned}$$

Hence, the object U corresponds to the free finitely complete $(\infty,1)$-category $\langle\alpha\rangle$ generated by an object α, the object E corresponds to the free finitely complete $(\infty,1)$-category $\langle\gamma : 1 \to \alpha\rangle$ generated by an object α and a global section $\gamma : 1 \to \alpha$, and the arrow $\partial : E \to U$ corresponds to the inclusion $\iota : \langle\alpha\rangle \to \langle\gamma : 1 \to \alpha\rangle$. Since $\mathbf{Th}(\mathbb{E}_\infty)^{\mathrm{op}} \simeq (\infty,1)\text{-}\mathbf{Lex}^{\mathrm{op}}$ is the ω-free completion of $\mathbb{E}_\infty$, we see that ∂ remains exponentiable in $(\infty,1)\text{-}\mathbf{Lex}^{\mathrm{op}}$. We thus regard $(\infty,1)\text{-}\mathbf{Lex}^{\mathrm{op}}$ as an $(\infty,1)$-CwR in which the representable maps are the pullbacks of ι, and then ι is a finitely complete universe in $(\infty,1)\text{-}\mathbf{Lex}^{\mathrm{op}}$. From the universal property of $\mathbb{E}_\infty$, we have the following universal property of $(\infty,1)\text{-}\mathbf{Lex}^{\mathrm{op}}$.

6.2.9. Corollary. *Let $\mathcal{X}$ be an $(\infty,1)$-CwR that has small limits and $f : B \to A$ a finitely complete universe in $\mathcal{X}$. Then there exists a unique morphism of $(\infty,1)$-CwRs $F : (\infty,1)\text{-}\mathbf{Lex}^{\mathrm{op}} \to \mathcal{X}$ that sends ι to f and preserves small limits.*

We also have truncated versions of Theorem 6.2.4.

6.2.10. Definition. For $1 \leq n < \infty$, we define $\mathbb{E}_n\mathbb{E}_n$ to be the ∞-type theory obtained from $\mathbb{E}_\infty$ by forcing $\partial : E \to U$ to be $(n-1)$-truncated. Note that, by univalence, U is forced to be n-truncated, and thus $\mathbb{E}_n$ is an $(n+1)$-type theory.

6.2.11. Theorem. *For any $1 \leq n < \infty$, the functor $\mathrm{ev}_\diamond : \mathbf{Mod}^{\mathrm{dem}}(\mathbb{E}_n) \to (\infty,1)\text{-}\mathbf{Cat}$ factors through the $(n+1,1)$-category $(n,1)\text{-}\mathbf{Lex}$ of finitely complete $(n,1)$-categories and induces an equivalence*

$$\mathbf{Mod}^{\mathrm{dem}}(\mathbb{E}_n) \simeq (n,1)\text{-}\mathbf{Lex}.$$

Proof:
A democratic model of $\mathbb{E}_n$ is nothing but a democratic model $\mathcal{M}$ of $\mathbb{E}_\infty$ such that the map $\mathcal{M}(\partial) : \mathcal{M}(E) \to \mathcal{M}(U)$ is $(n-1)$-truncated. It then follows that the base category $\mathcal{M}(\diamond)$ is an $(n,1)$-category with finite limits. Conversely, if $\mathcal{C}$ is an $(n,1)$-category with finite limits, then the generic representable map $\rho_{\mathcal{C}} : \widetilde{\mathrm{O}}_{\mathcal{C}} \to \mathrm{O}_{\mathcal{C}}$ is $(n-1)$-truncated. □

6.2.2 Locally cartesian closed $(\infty,1)$-categories

We show a result similar to Theorem 6.2.4 for locally cartesian closed $(\infty,1)$-categories.

6.2.12. Definition. Let $\mathcal{C}$ be an $(\infty,1)$-CwR and $\partial : E \to U$ a representable map in $\mathcal{C}$. A Π-*type structure on* ∂ is a pullback square of the form

$$\begin{array}{ccc} \mathrm{P}_\partial(E) & \overset{\lambda}{\dashrightarrow} & E \\ {\scriptstyle \mathrm{P}_\partial(\partial)}\downarrow & & \downarrow{\scriptstyle \partial} \\ \mathrm{P}_\partial(U) & \underset{\Pi}{\dashrightarrow} & U \end{array}$$

The following is immediate from the definition.

6.2.13. Proposition. *Let $\partial : E \to U$ be a univalent representable map in an $(\infty,1)$-CwR. Then Π-type structures on ∂ are unique up to contractible choice. Moreover, ∂ has a Π-type structure if and only if ∂-small maps are closed under pushforwards along ∂-maps.*

6.2.14. Definition. We define $\mathbb{E}^{\Pi}_{\infty}$ to be the free ∞-type theory generated by a finitely complete universe $\partial : E \to U$ with a Π-type structure.

6.2.15. Theorem. *The functor* $\mathrm{ev}_\diamond : \mathbf{Mod}^{\mathrm{dem}}\big(\mathbb{E}^{\Pi}_{\infty}\big) \to (\infty,1)\text{-}\mathbf{Cat}$ *factors through* $(\infty,1)\text{-}\mathbf{LCCC}$ *and induces an equivalence*

$$\mathbf{Mod}^{\mathrm{dem}}\big(\mathbb{E}^{\Pi}_{\infty}\big) \simeq (\infty,1)\text{-}\mathbf{LCCC},$$

where $(\infty,1)\text{-}\mathbf{LCCC}$ *is the* $(\infty,1)$*-category of small locally cartesian closed* $(\infty,1)$*-categories and functors preserving finite limits and pushforwards.*

Proof:
The proof is analogous to that of Theorem 6.2.4. □

6.2.3 ∞-type theories

Since the structure of an ∞-type theory looks type-theoretic, we could find an ∞-type theory such that theories over it are equivalent to ∞-type theories.

6.2.16. Definition. Let $\partial_1 : E_1 \to U_1$, $\partial_2 : E_2 \to U_2$ and $\partial_3 : E_3 \to U_3$ be representable maps in an $(\infty, 1)$-CwR. A $(\partial_1, \partial_2, \partial_3)$-$\Pi$-*type structure* is a pullback square of the form

$$\begin{array}{ccc} \mathrm{P}_{\partial_1}(E_2) & \overset{\lambda}{\dashrightarrow} & E_3 \\ {\scriptstyle \mathrm{P}_{\partial_1}(\partial_2)}\downarrow & \lrcorner & \downarrow{\scriptstyle \partial_3} \\ \mathrm{P}_{\partial_1}(U_2) & \underset{\Pi}{\dashrightarrow} & U_3. \end{array}$$

The following is immediate from the definition.

6.2.17. Proposition. *If ∂_3 is univalent, then $(\partial_1, \partial_2, \partial_3)$-$\Pi$-type structures are unique up to contractible choice. Moreover, there exists a $(\partial_1, \partial_2, \partial_3)$-$\Pi$-type structure if and only if the pushforward of any ∂_2-small map along any ∂_1-small map is ∂_3-small.*

6.2.18. Definition. We define $\mathbb{R}_\infty$ to be the free ∞-type theory generated by the following data:

- a finitely complete universe $\partial : E \to U$;
- a subobject $R \subset U$. We write ∂_R for the pullback of ∂ along the inclusion $R \to U$;
- a unit type structure and a Σ-type structure on ∂_R;
- a $(\partial_R, \partial, \partial)$-$\Pi$-type structure.

6.2.19. Definition. Let $\mathcal{M}$ be a model of $\mathbb{R}_\infty$. We say a morphism in $\mathcal{M}(\diamond)$ is a *representable map* if it is a context comprehension with respect to ∂_R. Using the $(\partial_R, \partial, \partial)$-$\Pi$-type structure, we see that display maps are closed under pushforwards along representable maps. In particular, if $\mathcal{M}$ is democratic, then $\mathcal{M}(\diamond)$ is an ∞-type theory and any morphism between democratic models induces a morphism of ∞-type theories. Hence, we have a functor

$$\mathrm{ev}_\diamond : \mathbf{Mod}^{\mathrm{dem}}(\mathbb{R}_\infty) \to \infty\text{-}\mathbf{TT}\,.$$

6.2.20. Theorem. *The functor $\mathrm{ev}_\diamond : \mathbf{Mod}^{\mathrm{dem}}(\mathbb{R}_\infty) \to \infty$-$\mathbf{TT}$ is an equivalence.*

Proof:
Similar to Theorem 6.2.4. For an ∞-type theory $\mathcal{C}$, we regard the class $\mathrm{R}_\mathcal{C}$ of representable maps in $\mathcal{C}$ as a subfibration of $\mathrm{O}_\mathcal{C}$. Then the representable map classifier $\rho_\mathcal{C} : \widetilde{\mathrm{O}}_\mathcal{C} \to \mathrm{O}_\mathcal{C}$ and the subobject $\mathrm{R}_\mathcal{C} \subset \mathrm{O}_\mathcal{C}$ determine a democratic model of $\mathbb{R}_\infty$. □

6.3 Coherence problems

One of motivations for ∞-type theories is to tackle *coherence problems* in categorical semantics of type theory. Traditionally, a category $\mathcal{C}$ with finite limits is considered as a "model" of dependent type theory: contexts are interpreted as objects of $\mathcal{C}$; types over a context Γ are interpreted as objects of $\mathcal{C}/\Gamma$; terms are interpreted as sections. For a type A over a context Γ and a substitution $f : \Gamma' \to \Gamma$, the action $A \cdot f$ is interpreted as the pullback $f^*A \in \mathcal{C}/\Gamma'$. However, there is a mismatch between levels of equality: the substitution law $A \cdot (f \circ g) = (A \cdot f) \cdot g$ holds on the nose in the type theory; but we only have a coherent isomorphism $(f \circ g)^* A \cong g^* f^* A$ in the category. Therefore, a category with finite limits is only a *non-split* model of dependent type theory.

Theorem 6.2.11 suggests that it is more natural to consider categories with finite limits as models of a 2-type theory rather than of a 1-type theory, and we have not found any coherence problems in proving Theorem 6.2.11. Therefore, switching from 1-type theory to ∞-type theory, coherence problems seem to disappear. Of course, this does not solve anything because we just rephrase non-split models of a type theory by models of an ∞-type theory, and the problem is now how the ∞-type theory considered there is related to the original 1-type theory. We can deal with the situation in a broader context. Let $\mathcal{T}$ be a 1-type theory and suppose that we have an ∞-type theory $\mathcal{T}_\infty$ that looks like an ∞-analogue of $\mathcal{T}$. In many cases, we cannot find any non-trivial morphism between $\mathcal{T}$ and $\mathcal{T}_\infty$ but can find a span of ∞-type theories

$$\mathcal{T} \xleftarrow{\tau} \mathcal{T}_\infty^- \xrightarrow{\gamma} \mathcal{T}_\infty, \tag{6.3}$$

where τ is a truncation-like morphism and γ inverts some arrows in $\mathcal{T}_\infty^-$.

6.3.1. Example. Let $\mathbb{E}$ denote the free 1-type theory generated by a representable map $\partial : E \to U$ equipped with Σ-types, unit type, and extensional identity types. We cannot obtain a morphism between $\mathbb{E}$ and $\mathbb{E}_1$ preserving the generating representable map ∂: the object U is 0-truncated in $\mathbb{E}$ while not in $\mathbb{E}_1$; the representable map ∂ is univalent in $\mathbb{E}_1$ while not in $\mathbb{E}$. We thus introduce an intermediate 2-type theory $\mathbb{E}_1^-$ defined in the same way as $\mathbb{E}_1$ but without univalence of $\partial : E \to U$. Then $\mathbb{E}$ is obtained from $\mathbb{E}_1^-$ by forcing U to be 0-truncated, and $\mathbb{E}_1$ is obtained from $\mathbb{E}_1^-$ by forcing ∂ to be univalent, yielding a span

$$\mathbb{E} \xleftarrow{\tau} \mathbb{E}_1^- \xrightarrow{\gamma} \mathbb{E}_1. \tag{6.4}$$

6.3.2. Example. Let $\mathbb{I}$ denote the free 1-type theory generated by a representable map $\partial : E \to U$ equipped with Σ-types, unit type, and intensional identity types and $\mathbb{I}_\infty$ the free ∞-type theory with the same structure as $\mathbb{I}$. Then $\mathbb{I}$ is obtained from $\mathbb{I}_\infty$ by forcing U and E to be 0-truncated. Intensional identity

types in the ∞-type theory $\mathbb{I}_\infty$ is defined as a commutative square

$$\begin{array}{ccc} E & \overset{\mathsf{refl}}{\cdots\cdots\cdots\cdots\rightarrow} & E \\ {\scriptstyle\Delta}\downarrow & & \downarrow{\scriptstyle\partial} \\ E \times_U E & \underset{\mathsf{Id}}{\cdots\cdots\cdots\rightarrow} & U \end{array}$$

equipped with an arrow representing the elimination rule. If the induced arrow $\mathsf{refl}' : E \to \mathsf{Id}^* E$ is inverted, then the intensional identity types become extensional ones. Therefore, $\mathbb{E}_\infty$ is obtained from $\mathbb{I}_\infty$ by inverting refl' and by forcing ∂ to be univalent, and thus we have a span

$$\mathbb{I} \xleftarrow{\tau} \mathbb{I}_\infty \xrightarrow{\gamma} \mathbb{E}_\infty. \tag{6.5}$$

The span (6.3) induces a functor from $\mathbf{Th}(\mathcal{T})$ to $\mathbf{Th}(\mathcal{T}_\infty)$ by the composite

$$\mathbf{Th}(\mathcal{T}) \xrightarrow{\tau^*} \mathbf{Th}(\mathcal{T}_\infty^-) \xrightarrow{\gamma_!} \mathbf{Th}(\mathcal{T}_\infty),$$

where for a morphism of ∞-type theories $F : \mathcal{T}_1 \to \mathcal{T}_2$, we write $F^* : \mathbf{Th}(\mathcal{T}_2) \to \mathbf{Th}(\mathcal{T}_1)$ for the precomposition functor and $F_!$ for its left adjoint. We say the span (6.3) is a *presentation of* $\mathcal{T}_\infty$ *by* $\mathcal{T}$ if the functor

$$\gamma_! \circ \tau^* : \mathbf{Th}(\mathcal{T}) \to \mathbf{Th}(\mathcal{T}_\infty)$$

is a *localization functor* in the sense of Cisinski [38], that is, the induced functor $\mathrm{L}(\mathbf{Th}(\mathcal{T})) \to \mathbf{Th}(\mathcal{T}_\infty)$ is an equivalence where $\mathrm{L}(\mathbf{Th}(\mathcal{T}))$ is obtained from $\mathbf{Th}(\mathcal{T})$ by freely inverting those morphisms inverted by $\gamma_! \circ \tau^*$. This condition implies:

1. any $\mathcal{T}_\infty$-theory is represented by a $\mathcal{T}$-theory;
2. any morphism $I : \Phi \to \Psi$ in $\mathbf{Th}(\mathcal{T}_\infty)$ between $\mathcal{T}$-theories is represented by a zigzag of morphisms in $\mathbf{Th}(\mathcal{T})$

$$\Phi \xrightarrow{I_1^+} \Phi_{01} \xleftarrow{I_1^-} \Phi_1 \xrightarrow{I_2^+} \cdots \xleftarrow{I_n^-} \Psi$$

where I_i^-'s become invertible in $\mathbf{Th}(\mathcal{T}_\infty)$.

Roughly, all we can do with the ∞-type theory $\mathcal{T}_\infty$ are emulated by the 1-type theory $\mathcal{T}$. Since $\mathbf{Th}(\mathcal{T}_\infty) \simeq \mathbf{Mod}^{\mathrm{dem}}(\mathcal{T}_\infty)$, the equivalence $\mathrm{L}(\mathbf{Th}(\mathcal{T})) \simeq \mathbf{Th}(\mathcal{T}_\infty)$ justifies using the 1-type theory $\mathcal{T}$ as an *internal language* for democratic models of the ∞-type theory $\mathcal{T}_\infty$.

In this section, we develop a technique of proving that the span (6.3) is a presentation of $\mathcal{T}_\infty$ by $\mathcal{T}$. In general, it is difficult to prove that a functor is a localization functor. It is known that when a functor preserves certain colimits, one can verify that the functor is a localization functor by checking a couple of

conditions called the *left approximation property* [38]. We thus first show that the functor $\gamma_! \circ \tau^*$ preserves certain colimits. Since $\gamma_!$ preserves all colimits, the real problem is whether τ^* preserves certain colimits. In fact, this is what people refer to as a *coherence problem*.

To see why this is a coherence problem, we observe that the classical solutions to coherence problems by Hofmann [78] and Curien [47] essentially proves that the functor τ^* preserves certain colimits. Hofmann's approach is to replace a non-split model by a split model equivalent to the original non-split model in some sense. This shows that the functor $\tau^* : \mathbf{Mod}^{\mathrm{dem}}(\mathcal{T}) \simeq \mathbf{Th}(\mathcal{T}) \to \mathbf{Th}(\mathcal{T}^-_\infty) \simeq \mathbf{Mod}^{\mathrm{dem}}(\mathcal{T}^-_\infty)$ is surjective in some sense. The preservation of certain colimits is a consequence of this surjectivity. Curien introduced a type theory with explicit coercion in which equality between types is replaced by isomorphisms. His key result is that in the initial theory, any two parallel isomorphisms are equal. An ∞-type theory is considered as a higher dimensional extension of the type theory with explicit coercion in the sense that equality is replaced by homotopies. Then, Curien's result can be rephrased as follows: any two parallel homotopies in the initial $\mathcal{T}^-_\infty$-theory are equal. This means that the initial $\mathcal{T}^-_\infty$-theory does not contain any non-trivial homotopies, from which it follows that the functor $\tau^* : \mathbf{Th}(\mathcal{T}) \to \mathbf{Th}(\mathcal{T}^-_\infty)$ preserves initial objects.

Once we prove that the functor τ^* preserves certain colimits, it is not difficult to check the left approximation property. We explain this by examples. In Section 6.3.1 we consider the simplest case when $\mathcal{T}$ is the syntactic CwR of DTT whose models are natural models and non-split models are comprehension categories. In Section 6.3.2, we extend the result to a general coherence theorem for dependent type theories with various type constructors such as Σ-types, unit type, extensional identity types, and Π-types. Finally, in Section 6.3.3, we discuss the coherence problem for intensional identity types and finitely complete $(\infty, 1)$-categories. We give a sketch of the proof that the span (6.5) is a presentation of $\mathbb{E}_\infty$ by $\mathbb{I}$, which is a positive solution to the conjecture by Kapulkin and Lumsdaine [100] that the homotopy theory of theories over the type theory with intensional identity types is equivalent to the homotopy theory of finitely complete $(\infty, 1)$-categories.

6.3.1 Coherence for comprehension categories

A *comprehension category* [90, 89] is a cartesian fibration $\mathcal{D} \to \mathcal{C}$ equipped with a functor $\mathrm{p} : \mathcal{D} \to \mathcal{C}^{\to}$ over $\mathcal{C}$ sending cartesian arrows to pullback squares. In this thesis, we only consider the case when $\mathcal{D}$ is a right fibration over $\mathcal{C}$. For any

section $A : \mathcal{C}/\Gamma \to \mathcal{D}$, we have a pullback

$$\begin{array}{ccccc} \mathcal{C}/\operatorname{dom}(\mathrm{p}(A)) & \longrightarrow & & & \mathcal{C}^{\Delta} \\ \downarrow & \lrcorner & & & \downarrow \\ \mathcal{C}/\Gamma & \xrightarrow[A]{} & \mathcal{D} & \xrightarrow[\mathrm{p}]{} & \mathcal{C}^{\to} \end{array}$$

where $\mathcal{C}^{\Delta}$ is the category of sections in $\mathcal{C}$. Hence, the pullback $\mathrm{p}^*\mathcal{C}^{\Delta} \to \mathcal{D}$ is a representable map of right fibrations over $\mathcal{C}$. Conversely, any representable map of right fibrations induces a comprehension category. We thus identify a comprehension category with the corresponding representable map of right fibrations.

6.3.3. Definition. Let $\mathbb{D}$ denote the syntactic CwR of DTT. It is characterized as the free 1-type theory generated by a representable map $\partial : E \to U$. We define $\mathbb{D}_1$ to be the free 2-type theory generated by a 0-truncated univalent representable map $\partial : E \to U$. Let $\mathbb{D}_1^-$ be the free 2-type theory generated by a 0-truncated representable map $\partial : E \to U$. By definition, $\mathbb{D}$ is obtained from $\mathbb{D}_1^-$ by forcing U to be 0-truncated, and $\mathbb{D}_1$ is obtained from $\mathbb{D}_1^-$ by forcing ∂ to be univalent. Therefore, we have a span

$$\mathbb{D} \xleftarrow{\tau} \mathbb{D}_1^- \xrightarrow{\gamma} \mathbb{D}_1. \tag{6.6}$$

We note that a model of $\mathbb{D}_1^-$ is nothing but a comprehension category. We could say that a model of $\mathbb{D}_1$ is a *univalent* comprehension category in the sense that the corresponding representable map of right fibrations is univalent. A comprehension category $\mathrm{p} : \mathcal{D} \to \mathcal{C}^{\to}$ is univalent in this sense if and only if it is fiberwise fully faithful on isomorphisms, that is, the functor $\mathrm{p}_\Gamma : \mathcal{D}_\Gamma \to \mathrm{k}(\mathcal{C}/\Gamma)$ is fully faithful for any object $\Gamma \in \mathcal{C}$. Therefore, a univalent comprehension category is identified with a category $\mathcal{C}$ equipped with a class of arrows $\mathcal{D} \subset \mathcal{C}^{\to}$ that is stable under pullbacks. It is almost a display map category [166] or a clan [94], but the specified arrows are not required to be closed under composition. In this section, we sketch how the coherence problem for univalent comprehension categories is solved.

6.3.4. Theorem. *The span* (6.6) *is a presentation of* $\mathbb{D}_1$ *by* $\mathbb{D}$.

Although this time we only deal with $(2,1)$-categories, we use the general theory of localizations of $(\infty,1)$-categories [38, Chapter 7] in order to leave open the possibility of higher dimensional generalization. The outline is as follows.

1. We make $\mathbf{Th}(\mathbb{D})$ a *category with weak equivalences and cofibrations* in which the weak equivalences are those morphisms inverted by $\gamma_! \circ \tau^*$ such that the functor $\gamma_! \circ \tau^* : \mathbf{Th}(\mathbb{D}) \to \mathbf{Th}(\mathbb{D}_1)$ is right exact. Let $\mathrm{L}(\mathbf{Th}(\mathbb{D}))$ denote the localization.

2. We show that the functor $\gamma_! \circ \tau^*$ satisfies the *left approximation property*: for any cofibrant object $\Phi \in \mathbf{Th}(\mathbb{D})$ and any morphism $\gamma_! \tau^* \Phi \to \Psi$ in $\mathbf{Th}(\mathbb{D}_1)$, there exist a morphism $\Phi \to \Psi'$ in $\mathbf{Th}(\mathbb{D})$ and an equivalence $\gamma_! \tau^* \Psi' \simeq \Psi$ under $\gamma_! \tau^* \Phi$.

Then, the left derived functor $\mathrm{L}(\mathbf{Th}(\mathbb{D})) \to \mathbf{Th}(\mathbb{D}_1)$ is an equivalence by [38, Proposition 7.6.15].

We first recall the notion of a category with weak equivalences and cofibrations. There are some variations of this notion [30, 180, 139, 164], and here we take Cisinski's definition [38].

6.3.5. Definition. A *category with weak equivalences and cofibrations* is a category $\mathcal{C}$ equipped with two classes of arrows Weq and Cof satisfying the conditions below. Arrows in Weq are called *weak equivalences* and arrows in Cof are called *cofibrations*. An object x of $\mathcal{C}$ is *cofibrant* if the initial injection $0 \to x$ is a cofibration. An arrow is said to be a *trivial cofibration* if it is both a weak equivalence and a cofibration.

1. $\mathcal{C}$ has an initial object.

2. All the identities are trivial cofibrations, and weak equivalences and cofibrations are closed under composition.

3. The weak equivalences satisfy the *2-out-of-3* property: if u and v are a composable pair of arrows and if two of u, v, and vu are weak equivalences, then so is the rest.

4. Cofibrations are stable under pushouts along arbitrary arrows: if $i : x \to y$ is a cofibration and $u : x \to x'$ is an arbitrary arrow, then the pushout $u_! y$ exists and the arrow $x' \to u_! y$ is a cofibration.

5. Trivial cofibrations are stable under pushouts along arrows between cofibrant objects: for any pushout square

$$\begin{array}{ccc} x & \xrightarrow{u} & x' \\ {\scriptstyle i}\downarrow & \ulcorner & \downarrow{\scriptstyle i'} \\ y & \xrightarrow[v]{} & y' \end{array}$$

   in which x and x' are cofibrant and i is a cofibration, if i is a weak equivalence, then so is i'.

6. Any arrow $u : x \to y$ with cofibrant domain factors into a cofibration $x \to y'$ followed by a weak equivalence $y' \to y$.

For a category with weak equivalences and cofibrations, the *localization* $\mathrm{L}(\mathcal{C})$ is the $(\infty, 1)$-category obtained from $\mathcal{C}$ by adjoining formal inverses of weak equivalences.

6.3.6. Definition. Let $\mathcal{C}$ be a category with weak equivalences and cofibrations and $\mathcal{D}$ an $(\infty, 1)$-category with finite colimits. A functor $F : \mathcal{C} \to \mathcal{D}$ is *right exact* if it sends trivial cofibrations between cofibrant objects to invertible arrows and preserves initial objects and pushouts of cofibrations along arrows between cofibrant objects.

Any right exact functor $F : \mathcal{C} \to \mathcal{D}$ induces the *left derived functor* $\mathrm{L}(\mathcal{C}) \to \mathcal{D}$ [38, Remark 7.5.27]. If, in addition, F sends all weak equivalences to invertible arrows, then the left derived functor coincides with the one induced by the universal property of the localization $\mathrm{L}(\mathcal{C})$ [38, Lemma 7.5.24]. In this case, one can show that the left derived functor $\mathrm{L}(\mathcal{C}) \to \mathcal{D}$ is an equivalence by verifying the following *left approximation property* by [38, Proposition 7.6.15]:

1. an arrow in $\mathcal{C}$ is a weak equivalence if it becomes invertible in $\mathcal{D}$;
2. for any cofibrant object x in $\mathcal{C}$ and any arrow $u : F(x) \to y$ in $\mathcal{D}$, there exists an arrow $u' : x \to y'$ in $\mathcal{C}$ and an equivalence $F(y') \simeq y$ under $F(x)$.

We now define weak equivalences and cofibrations in $\mathbf{Th}(\mathbb{D})$.

6.3.7. Definition. We define *weak equivalences* in $\mathbf{Th}(\mathbb{D})$ to be those morphisms inverted by $\gamma_! \circ \tau^*$.

By definition, the first half of the left approximation property is satisfied.

6.3.8. Definition. The *generating cofibrations in* $\mathbf{Th}(\mathbb{D})$ are the following morphisms:

- $\mathrm{Y}(\mathrm{P}^n_\partial(1)) \to \mathrm{Y}(\mathrm{P}^n_\partial(U))$ for $n \geq 0$;
- $\mathrm{Y}(\mathrm{P}^n_\partial(\partial)) : \mathrm{Y}(\mathrm{P}^n_\partial(U)) \to \mathrm{Y}(\mathrm{P}^n_\partial(E))$ for $n \geq 0$;
- $\mathrm{Y}(\mathrm{P}^n_\partial(\Delta)) : \mathrm{Y}(\mathrm{P}^n_\partial(E \times_U E)) \to \mathrm{Y}(\mathrm{P}^n_\partial(E))$ for $n \geq 0$.

The class of *cofibrations in* $\mathbf{Th}(\mathbb{D})$ is the closure of the generating cofibrations under retracts, pushouts along arbitrary morphisms, and transfinite composition. Cofibrations in $\mathbf{Th}\big(\mathbb{D}_1^-\big)$ and $\mathbf{Th}(\mathbb{D}_1)$ are defined in the same way. We note that the functors $\tau_!$ and $\gamma_!$ preserve generating cofibrations.

Intuitively, cofibrant objects are well-behaved objects with respect to weak equivalences. For a pushout in $\mathbf{Th}(\mathbb{D})$

$$\begin{array}{ccc} \mathrm{Y}(\mathrm{P}^n_\partial(1)) & \longrightarrow & \Phi \\ \downarrow & \ulcorner & \downarrow \\ \mathrm{Y}(\mathrm{P}^n_\partial(U)) & \longrightarrow & \Phi', \end{array}$$

Φ' is an extension of Φ by a type. Similarly, the pushouts of $\mathrm{Y}(\mathrm{P}^n_\partial(\partial))$ and $\mathrm{Y}(\mathrm{P}^n_\partial(\Delta))$ create extensions by a term and by a term equality, respectively. Therefore, a cofibrant $\mathbb{D}$-theory is obtained from the empty theory by adjoining types, terms, and equations between terms. Equations between types are not well-behaved because they are too strict when we try to interpret types as objects in a category.

By definition, $\mathbf{Th}(\mathbb{D})$ satisfies Items 1 to 4 of Definition 6.3.5. For Item 6, the *small object argument* from model category theory [82] ensures that any morphism in $\mathbf{Th}(\mathbb{D})$ factors into a cofibration followed by a *trivial fibration*.

6.3.9. Definition. We say a morphism $I : \Phi \to \Psi$ in $\mathbf{Th}(\mathbb{D})$, $\mathbf{Th}\big(\mathbb{D}_1^-\big)$ or $\mathbf{Th}(\mathbb{D}_1)$ is a *trivial fibration* if it has the right lifting property with respect to cofibrations: for any commutative square

$$\begin{array}{ccc} A & \longrightarrow & \Phi \\ {\scriptstyle i}\big\downarrow & \nearrow & \big\downarrow{\scriptstyle I} \\ B & \longrightarrow & \Psi \end{array}$$

where i is a cofibration, there exists a diagonal filler denoted by the dotted arrow. By a standard argument in model category theory [82], I is a trivial fibration if and only if it has the right lifting property with respect to the generating cofibrations. Since $\mathbf{Th}(\mathbb{D})$, $\mathbf{Th}\big(\mathbb{D}_1^-\big)$ and $\mathbf{Th}(\mathbb{D}_1)$ are compactly, the small object argument proves that any morphism factors as a transfinite composite of pushouts of generating cofibrations followed by a trivial fibration.

Thus, Item 6 of Definition 6.3.5 follows from the following lemma proved later.

6.3.10. Lemma. *Any trivial fibration in* $\mathbf{Th}(\mathbb{D})$ *is inverted by the functor* $\gamma_! \circ \tau^*$.

The hardest part is to verify Item 5 of Definition 6.3.5. We first observe that the functors $\tau^* : \mathbf{Th}(\mathbb{D}) \to \mathbf{Th}\big(\mathbb{D}_1^-\big)$ and $\gamma^* : \mathbf{Th}(\mathbb{D}_1) \to \mathbf{Th}\big(\mathbb{D}_1^-\big)$ are fully faithful. Recall that theories and democratic models are equivalent (Theorems 5.1.8 and 6.1.10). Observe that the reindexing functors $\tau^* : \mathbf{Mod}(\mathbb{D}) \to \mathbf{Mod}\big(\mathbb{D}_1^-\big)$ and $\gamma^* : \mathbf{Mod}(\mathbb{D}_1) \to \mathbf{Mod}\big(\mathbb{D}_1^-\big)$ preserve democratic models. Therefore, the functors $\tau^* : \mathbf{Th}(\mathbb{D}) \to \mathbf{Th}\big(\mathbb{D}_1^-\big)$ and $\gamma^* : \mathbf{Th}(\mathbb{D}_1) \to \mathbf{Th}\big(\mathbb{D}_1^-\big)$ are identified with $\tau^* : \mathbf{Mod}^{\mathrm{dem}}(\mathbb{D}) \to \mathbf{Mod}^{\mathrm{dem}}\big(\mathbb{D}_1^-\big)$ and $\gamma^* : \mathbf{Mod}^{\mathrm{dem}}(\mathbb{D}_1) \to \mathbf{Mod}^{\mathrm{dem}}\big(\mathbb{D}_1^-\big)$, respectively. By the definitions of $\mathbb{D}$, $\mathbb{D}_1^-$, and $\mathbb{D}_1$, we see that τ^* exhibits $\mathbf{Mod}^{\mathrm{dem}}(\mathbb{D})$ as the full subcategory of $\mathbf{Mod}^{\mathrm{dem}}\big(\mathbb{D}_1^-\big)$ spanned by those democratic models $\mathcal{M}$ of $\mathbb{D}_1^-$ such that the right fibration $\mathcal{M}(U)$ is 0-truncated, and γ^* exhibits $\mathbf{Mod}^{\mathrm{dem}}(\mathbb{D}_1)$ as the full subcategory of $\mathbf{Mod}^{\mathrm{dem}}\big(\mathbb{D}_1^-\big)$ spanned by those democratic models $\mathcal{M}$ of $\mathbb{D}_1^-$ such that the representable map $\mathcal{M}(\partial) : \mathcal{M}(E) \to \mathcal{M}(U)$ is univalent. We may thus regard $\mathbf{Th}(\mathbb{D}) \simeq \mathbf{Mod}^{\mathrm{dem}}(\mathbb{D})$ as a full subcategory of $\mathbf{Th}\big(\mathbb{D}_1^-\big) \simeq \mathbf{Mod}^{\mathrm{dem}}\big(\mathbb{D}_1^-\big)$. The key lemma is as follows.

6.3.11. Lemma. *Any cofibrant $\mathbb{D}_1^-$-theory belongs to* $\mathbf{Th}(\mathbb{D})$.

This in particular implies that the inclusion $\tau^* : \mathbf{Th}(\mathbb{D}) \to \mathbf{Th}\big(\mathbb{D}_1^-\big)$ preserves initial objects and pushouts of cofibrations along morphisms between cofibrant objects. Since $\gamma_! : \mathbf{Th}\big(\mathbb{D}_1^-\big) \to \mathbf{Th}(\mathbb{D}_1)$ preserves arbitrary colimits as it is a left adjoint, the composite $\gamma_! \circ \tau^*$ also preserves initial objects and pushouts of cofibrations along morphisms between cofibrant objects. Then Item 5 of Definition 6.3.5 follows. This also proves that the functor $\gamma_! \circ \tau^*$ is right exact.

The left approximation property is proved by a combinatorial argument. We first observe that in $\mathbf{Th}(\mathbb{D}_1)$, all the morphisms are cofibrations. This is because the codiagonal $\mathrm{Y}(\mathrm{P}^n_\partial(\Delta)) : \mathrm{Y}(\mathrm{P}^n_\partial(U \times U)) \to \mathrm{Y}(\mathrm{P}^n_\partial(U))$ becomes a cofibration in $\mathbf{Th}(\mathbb{D}_1)$ thanks to the univalence of $\partial : E \to U$. Then, the small object argument shows that any morphism $I : \gamma_!\tau^*\Phi \to \Psi$ in $\mathbf{Th}(\mathbb{D}_1)$ is a transfinite composite of pushouts of generating cofibrations. Since the generating cofibrations in $\mathbf{Th}(\mathbb{D})$ are the same as those in $\mathbf{Th}(\mathbb{D}_1)$, we can copy the construction of I into $\mathbf{Th}(\mathbb{D})$ and have an approximation of I by a morphism in $\mathbf{Th}(\mathbb{D})$.

We take a closer look at Lemmas 6.3.10 and 6.3.11. Lemma 6.3.10 is proved by concretely describing the functor $\gamma_! \circ \tau^*$ in terms of democratic models.

6.3.12. Construction. For a democratic model $\mathcal{M} \in \mathbf{Mod}^{\mathrm{dem}}(\mathbb{D})$, the democratic model $\gamma_!\mathcal{M}$ of $\mathbb{D}_1$ is constructed as follows:

- the base category is the same as $\mathcal{M}$;
- the objects of $(\gamma_!\mathcal{M})(U)$ are the same as $\mathcal{M}(U)$, but the morphisms from A' to A are the pullback squares

$$\begin{array}{ccc} \{A'\} & \xrightarrow{g} & \{A\} \\ \downarrow & \lrcorner & \downarrow \\ \Gamma' & \xrightarrow[f]{} & \Gamma; \end{array}$$

- the sections of $(\gamma_!\mathcal{M})(E)$ over $A : \mathcal{M}(\diamond)/\Gamma \to (\gamma_!\mathcal{M})(U)$ are the sections of $\{A\} \to \Gamma$.

Then, for a trivial fibration $F : \mathcal{M} \to \mathcal{N}$ in $\mathbf{Mod}^{\mathrm{dem}}(\mathbb{D})$, the morphism $\gamma_!F : \gamma_!\mathcal{M} \to \gamma_!\mathcal{N}$ is an equivalence. The idea is that the lifting property with respect to the generating cofibrations implies that the map $(\gamma_!F)_U : (\gamma_!\mathcal{M})(U) \to (\gamma_!\mathcal{N})(U)$ is essentially surjective, full, and faithful.

Lemma 6.3.11 could be proved either semantically, following Hofmann [78], or syntactically, following Curien [47]. By "semantically", we mean that we work with democratic models instead of theories. Given a democratic model $\mathcal{M}$ of $\mathbb{D}_1^-$, we can construct a democratic model $\mathrm{Sp}\,\mathcal{M}$ of $\mathbb{D}$ as follows. The base category of $\mathrm{Sp}\,\mathcal{M}$ is the same as $\mathcal{M}$. The discrete fibration $(\mathrm{Sp}\,\mathcal{M})(U)$ is the so-called

right adjoint splitting [158] of the right fibration $\mathcal{M}(U)$. It is equipped with a fiberwise surjective map $\varepsilon_U : (\mathrm{Sp}\,\mathcal{M})(U) \to \mathcal{M}(U)$ of right fibrations over $\mathcal{M}(\diamond)$. The discrete fibration $(\mathrm{Sp}\,\mathcal{M})(E)$ is the pullback of $\mathcal{M}(E)$ along ε_U, and then ε_U is part of a morphism $\varepsilon : \mathrm{Sp}\,\mathcal{M} \to \mathcal{M}$ of democratic models of $\mathbb{D}_1^-$. The fiberwise surjectivity of ε_U implies that the morphism $\varepsilon : \mathrm{Sp}\,\mathcal{M} \to \mathcal{M}$ has the right lifting property with respect to the morphism $\mathrm{Y}(\mathrm{P}^n_\partial(1)) \to \mathrm{Y}(\mathrm{P}^n_\partial(U))$. The other lifting properties follow from the definition of $(\mathrm{Sp}\,\mathcal{M})(E)$, and thus ε is a trivial fibration. In summary, for any democratic model $\mathcal{M}$ of $\mathbb{D}_1^-$, we have a democratic model $\mathrm{Sp}\,\mathcal{M}$ of $\mathbb{D}$ and a trivial fibration $\varepsilon : \mathrm{Sp}\,\mathcal{M} \to \mathcal{M}$ in $\mathbf{Mod}^{\mathrm{dem}}\big(\mathbb{D}_1^-\big)$. In particular, if $\mathcal{M}$ is cofibrant, then ε has a section by the lifting property, and thus $\mathcal{M}$ is a retract of $\mathrm{Sp}\,\mathcal{M}$.

$$\begin{array}{ccc} 0 & \longrightarrow & \mathrm{Sp}\,\mathcal{M} \\ \downarrow & \nearrow & \downarrow{\scriptstyle \varepsilon} \\ \mathcal{M} & = & \mathcal{M} \end{array}$$

Since $\mathbf{Mod}^{\mathrm{dem}}(\mathbb{D}) \subset \mathbf{Mod}^{\mathrm{dem}}\big(\mathbb{D}_1^-\big)$ is closed under retracts, any cofibrant democratic model of $\mathbb{D}_1^-$ belongs to $\mathbf{Mod}^{\mathrm{dem}}(\mathbb{D})$.

A syntactic proof of Lemma 6.3.11 has not yet been completed, but Curien's argument [47] can be seen as a proof of Lemma 6.3.11. He introduced a type theory with *explicit conversion* where the usual conversion rule

$$\frac{\Gamma \vdash a : A_1 \qquad \Gamma \vdash A_1 \equiv A_2\ \mathsf{type}}{\Gamma \vdash a : A_2}$$

is replaced by an explicit conversion rule like

$$\frac{\Gamma \vdash a : A_1 \qquad \Gamma \vdash A_1 \equiv A_2\ \mathsf{type}}{\Gamma \vdash c_{A_1,A_2} \cdot a : A_2}$$

so that the use of conversion is recorded in the term $c_{A_1,A_2} \cdot a$. Thus, the judgment $A_1 \equiv A_2\ \mathsf{type}$ expresses that A_1 and A_2 are *homotopic* rather than equal, and the expression $c_{A_1,A_2} \cdot a$ is the action of the homotopy. In this sense, a type theory with explicit conversion is 2-*dimensional*, and we expect that it is a syntactic presentation of a 2-type theory. One of Curien's results is that in his type theory with explicit conversion, any two proofs of a type equation $A_1 \equiv A_2\ \mathsf{type}$ are equal [47, Theorem 6]. Lemma 6.3.11 essentially asserts that the same result holds when we extend the type theory by type constants, term constants, and equations between terms. Recall that a democratic model $\mathcal{M}$ of $\mathbb{D}_1^-$ belongs to $\mathbf{Mod}^{\mathrm{dem}}(\mathbb{D})$ if the right fibration $\mathcal{M}(U)$ is 0-truncated, that is, any two parallel homotopies between types in $\mathcal{M}$ are equal. Thus, Lemma 6.3.11 is equivalent to that if Φ is a $\mathbb{D}_1^-$-theory presented by type constants, term constants, and equations between terms but no equations between types, then any two parallel homotopies between types derivable over Φ are equal.

6.3.2 Coherence for finitely complete categories

We consider coherence problems for various extensions of $\mathbb{D}$ including the type theory $\mathbb{E}$ with Σ-types, unit type, and extensional identity types. Let $\mathcal{T}$ be a 1-type theory.

6.3.13. Assumption. $\mathcal{T}$ is obtained from $\mathbb{D}$ by the following operations:

1. adjoin a new arrow $x \to U$;
2. for an arrow $A : x \to U$, adjoin a new section $x \to E$ over A;
3. for a pair of parallel sections $a_1, a_2 : x \to E$ over the same arrow $A : x \to U$, equalize a_1 and a_2.

6.3.14. Example. Type theories with type constructors such as Π-types, Σ-types, identity types, unit type, and inductive types are constructed by the operations of Assumption 6.3.13.

By the same construction as $\mathcal{T}$, we also obtain a 2-type theory $\mathcal{T}_1^-$ from $\mathbb{D}_1^-$ and a 2-type theory $\mathcal{T}_1$ from $\mathbb{D}_1$. These fit into the span

$$\mathcal{T} \xleftarrow{\tau} \mathcal{T}_1^- \xrightarrow{\gamma} \mathcal{T}_1. \tag{6.7}$$

The goal is to show that the span (6.7) is a presentation of $\mathcal{T}_1$ by $\mathcal{T}$, under an extra assumption explained below. We follow the proof of Theorem 6.3.4. We choose generating cofibrations in the same way as $\mathbf{Th}(\mathbb{D})$. Again the key lemmas are Lemmas 6.3.10 and 6.3.11, and we can prove them by verifying the following.

6.3.15. Lemma. *For any democratic model $\mathcal{M}$ of $\mathcal{T}_1^-$, the democratic model $\mathrm{Sp}\,\mathcal{M}$ of $\mathbb{D}$ can be extended to a model of $\mathcal{T}$ such that $\varepsilon : \mathrm{Sp}\,\mathcal{M} \to \mathcal{M}$ is a morphism of models of $\mathcal{T}_1^-$.*

6.3.16. Assumption. For any democratic model $\mathcal{M}$ of $\mathcal{T}$, the democratic model $\gamma_!\mathcal{M}$ of $\mathbb{D}_1$ constructed in Construction 6.3.12 can be extended to a democratic model of $\mathcal{T}_1$ and satisfies the universal property in $\mathbf{Mod}^{\mathrm{dem}}(\mathcal{T}_1)$.

Lemma 6.3.15 is derivable from Assumption 6.3.13, but we have not found a nice sufficient condition for Assumption 6.3.16. Practically, Assumption 6.3.16 is verified by checking that all the arrows adjoined to $\mathcal{T}$ "respect isomorphisms". For example, the Π-type formation rule respects isomorphisms in the sense that for any isomorphisms of types $\vdash f : A_1 \cong A_2$ and $\mathtt{x} : A_1 \vdash g : B_1(\mathtt{x}) \cong A_2(f(\mathtt{x}))$, we have an isomorphism $\Pi(A_1, B_1) \cong \Pi(A_2, B_2)$. For a democratic model $\mathcal{M}$ of $\mathbb{D}$, since morphisms in fibers of $(\gamma_!\mathcal{M})(U)$ are isomorphisms of types, any Π-type structure on $\mathcal{M}$ induces a map of right fibrations $\Pi : (\gamma_!\mathcal{M})(\mathrm{P}_\partial(U)) \to (\gamma_!\mathcal{M})(U)$ which is part of a Π-type structure on $\gamma_!\mathcal{M}$. A difficulty in formulating

the notion of "respecting isomorphisms" is that whether a type formation rule respects isomorphisms depends on other components adjoined to $\mathcal{T}$. For example, the fact that the Π-type formation rule respects isomorphism does not follow from the Π-type formation rule alone, but we also need the introduction, elimination, and equality rules for Π-types to prove it. Another example is the formation rule for an inductive type which respects isomorphisms only in the presence of extensional identity types. Therefore, the validity of Assumption 6.3.16 depends on the interaction of components of $\mathcal{T}$ adjoined in Assumption 6.3.13.

6.3.17. Lemma. *For any right fibration A over a category $\mathcal{C}$ and any map $f : B \to A$ from a discrete fibration B over $\mathcal{C}$, there exists a map $f' : B \to \mathrm{Sp}\, A$ such that $\varepsilon \circ f' = f$.*

Proof:
For a section $b : \mathcal{C}/x \to B$, the composite $f \circ b : \mathcal{C}/x \to A$ determines a section $f'(b) : \mathcal{C}/x \to \mathrm{Sp}\, A$ by the definition of $\mathrm{Sp}\, A$. This construction is natural in x, and thus f' determines a map $B \to \mathrm{Sp}\, A$ of discrete fibrations over $\mathcal{C}$. By construction, $\varepsilon \circ f' = f$. □

Proof of Lemma 6.3.15:
Let $A : x \to U$ be one of the arrows adjoined to $\mathcal{T}$. By Lemma 6.3.17, we have a lift

$$\begin{array}{ccc} (\mathrm{Sp}\,\mathcal{M})(x) & \xrightarrow{\varepsilon} & \mathcal{M}(x) \\ {\scriptstyle (\mathrm{Sp}\,\mathcal{M})(A)}\big\downarrow & & \big\downarrow{\scriptstyle \mathcal{M}(A)} \\ (\mathrm{Sp}\,\mathcal{M})(U) & \xrightarrow[\varepsilon]{} & \mathcal{M}(U), \end{array}$$

and thus $\mathrm{Sp}\,\mathcal{M}$ models A and ε preserves the structure. Lifting the other components of $\mathcal{M}$ is straightforward because $(\mathrm{Sp}\,\mathcal{M})(E)$ is the pullback of $\mathcal{M}(E)$ along ε. □

6.3.18. Theorem. *The span* (6.7) *is a presentation of $\mathcal{T}_1$ by $\mathcal{T}$ for any* 1*-type theory $\mathcal{T}$ satisfying Assumptions 6.3.13 and 6.3.16.*

6.3.19. Example. It is straightforward to check that the type theory $\mathbb{E}$ satisfies Assumption 6.3.16. Therefore, the span Eq. (6.4) is a presentation of $\mathbb{E}_1$ as $\mathbb{E}$. We then derive from Theorem 6.2.11 that the $(2,1)$-category **Lex** is a localization of $\mathbf{Th}(\mathbb{E})$.

6.3.3 Coherence for finitely complete $(\infty, 1)$-categories

Kapulkin and Lumsdaine [100, Conjecture 3.7] conjectured that the homotopy theory of contextual categories with unit type, Σ-types, and intensional identity

types is equivalent to the homotopy theory of finitely complete $(\infty,1)$-categories. Contextual categories with these type constructors are equivalent to democratic models of the type theory $\mathbb{I}$ considered in Example 6.3.2, and thus their conjecture is equivalent to that the $(\infty,1)$-category $(\infty,1)$-**Lex** of finitely complete $(\infty,1)$-categories is a localization of the category $\mathbf{Mod}^{\mathrm{dem}}(\mathbb{I}) \simeq \mathbf{Th}(\mathbb{I})$. The following theorem gives a positive solution to their conjecture.

6.3.20. Theorem. *The span*

$$\mathbb{I} \xleftarrow{\tau} \mathbb{I}_\infty \xrightarrow{\gamma} \mathbb{E}_\infty.$$

in 6.3.2 is a presentation of $\mathbb{E}_\infty$ by $\mathbb{I}$.

6.3.21. Corollary. *The $(\infty,1)$-category $(\infty,1)$-**Lex** of finitely complete $(\infty,1)$-category is a localization of* $\mathbf{Th}(\mathbb{I})$.

Proof:
By Corollary 6.2.5. □

6.3.22. Remark. The precise statement of the conjecture by Kapulkin and Lumsdaine is that a specific functor between categories with weak equivalences induces an equivalence between the localizations. We will see that the functor $\gamma_! \circ \tau^* : \mathbf{Mod}^{\mathrm{dem}}(\mathbb{I}) \to \mathbf{Mod}^{\mathrm{dem}}(\mathbb{E}_\infty) \simeq (\infty,1)$-**Lex** coincides with their functor.

The idea of the proof of Theorem 6.3.20 is the same as Theorem 6.3.4: make $\mathbf{Th}(\mathbb{I})$ a category with weak equivalence and cofibrations; prove that the functor $\gamma_! \circ \tau^* : \mathbf{Th}(\mathbb{I}) \to \mathbf{Th}(\mathbb{E}_\infty)$ is right exact; prove that the functor $\gamma_! \circ \tau^*$ has the left approximation property. This time equality between terms is too strict compared to homotopies in an $(\infty,1)$-category, and thus we should exclude the codiagonals $\mathrm{Y}(\mathrm{P}^n_\partial(E \times_U E)) \to \mathrm{Y}(\mathrm{P}^n_\partial(E))$ from the generating cofibrations.

6.3.23. Definition. The *generating cofibrations in* $\mathbf{Th}(\mathbb{I})$ are the following morphisms:

- $\mathrm{Y}(\mathrm{P}^n_\partial(1)) \to \mathrm{Y}(\mathrm{P}^n_\partial(U))$ for $n \geq 0$;
- $\mathrm{Y}(\mathrm{P}^n_\partial(\partial)) : \mathrm{Y}(\mathrm{P}^n_\partial(U)) \to \mathrm{Y}(\mathrm{P}^n_\partial(E))$ for $n \geq 0$.

We note that this choice of generating cofibrations is the same as Kapulkin and Lumsdaine's [100, Definition 3.10].

Again the key lemmas are Lemmas 6.3.10 and 6.3.11. For Lemma 6.3.10, we have used a concrete description of the functor $\gamma_! \circ \tau^*$ in terms of democratic models. By a standard argument in the categorical semantics of homotopy type theory [e.g. 9, Theorem 3.2.5], the base category $\mathcal{M}(\diamond)$ of a democratic model $\mathcal{M}$

of $\mathbb{I}$ is a category with weak equivalences and fibrations. Then the localization $\mathrm{L}(\mathcal{M}(\diamond))$ is an $(\infty,1)$-category with finite limits [165, 38]. The democratic model $\gamma_!\mathcal{M}$ is obtained by Theorem 6.2.4. From this description, the functor $\gamma_! \circ \tau^* : \mathbf{Mod}^{\mathrm{dem}}(\mathbb{I}) \to \mathbf{Mod}^{\mathrm{dem}}(\mathbb{E}_\infty) \simeq (\infty,1)\text{-}\mathbf{Lex}$ is the same as the one considered by Kapulkin and Lumsdaine [100].

For Lemma 6.3.11, we develop a splitting technique. Techniques of replacing a structured $(\infty,1)$-category by a model of a 1-type theory have not been established unless the structured $(\infty,1)$-category is presentable [69, 151], but Shulman's result [151] of replacing any Grothendieck $(\infty,1)$-topos by a certain well-behaved model category is useful for splitting non-presentable $(\infty,1)$-categories, as he already mentioned [151, Remark 1.4]. Recall that a democratic model of $\mathbb{I}_\infty$ is a pair consisting of an $(\infty,1)$-category $\mathcal{M}(\diamond)$ and a representable map $\mathcal{M}(\partial) : \mathcal{M}(E) \to \mathcal{M}(U)$ of right fibrations over $\mathcal{M}(\diamond)$ equipped with certain type constructors. Shulman [151] showed that any Grothendieck $(\infty,1)$-topos is the localization of some *type-theoretic model topos*. Since the $(\infty,1)$-category $\mathbf{RFib}_{\mathcal{M}(\diamond)}$ of right fibrations over $\mathcal{M}(\diamond)$ is a Grothendieck $(\infty,1)$-topos, it is the localization $\gamma_{\mathcal{X}} : \mathcal{X} \to \mathbf{RFib}_{\mathcal{M}(\diamond)}$ of some type-theoretic model topos $\mathcal{X}$. Then the map $\mathcal{M}(\partial) : \mathcal{M}(E) \to \mathcal{M}(U)$ is the image by $\gamma_{\mathcal{X}}$ of some fibration $\partial_{\mathcal{X}} : E_{\mathcal{X}} \to U_{\mathcal{X}}$ between fibrant objects in $\mathcal{X}$.

6.3.24. Lemma. *We can choose $\partial_{\mathcal{X}} : E_{\mathcal{X}} \to U_{\mathcal{X}}$ that has unit type, Σ-types, and intensional identity types.*

By Lemma 6.3.24, we have a unique morphism $\mathbb{I} \to \mathcal{X}$ of CwRs, where we choose arbitrary maps as representable maps in $\mathcal{X}$. Then the composite with the Yoneda embedding $\mathbb{I} \to \mathcal{X} \to \widehat{\mathbf{DFib}_{\mathcal{X}}}$ determines a model of $\mathbb{I}$. We define $\mathrm{Sp}\,\mathcal{M}$ to be the heart of this model. We can construct a trivial fibration $\varepsilon : \mathrm{Sp}\,\mathcal{M} \to \mathcal{M}$ in $\mathbf{Mod}^{\mathrm{dem}}(\mathbb{I}_\infty)$, and then every cofibrant democratic model of $\mathbb{I}_\infty$ belongs to $\mathbf{Mod}^{\mathrm{dem}}(\mathbb{I})$ by the retract argument.

The proof of Lemma 6.3.24 is a little subtle. The intensional identity types on $\mathcal{M}(\partial)$ can be lifted to ones on $\partial_{\mathcal{X}}$, but unit types and Σ-types cannot be lifted. Recall that $\partial_{\mathcal{X}}$ has Σ-types if and only if pullbacks of $\partial_{\mathcal{X}}$ maps are closed under composition. Since $\mathcal{M}(\partial)$ has Σ-types, pullbacks of $\mathcal{M}(\partial)$ are closed under composition. However, the pullbacks of $\mathcal{M}(\partial)$ correspond to the *homotopy pullbacks* of $\partial_{\mathcal{X}}$ via the localization functor $\gamma_{\mathcal{X}} : \mathcal{X} \to \mathbf{RFib}_{\mathcal{M}(\diamond)}$, and thus we can say nothing about the composition of pullbacks of $\partial_{\mathcal{X}}$. To fix this, we replace $\partial_{\mathcal{X}}$ by a fibration $\partial'_{\mathcal{X}} : E'_{\mathcal{X}} \to U'_{\mathcal{X}}$ such that pullbacks of $\partial'_{\mathcal{X}}$ are homotopy pullbacks of $\partial_{\mathcal{X}}$.

By [151, Theorem 5.22], there exists a univalent fibration $\partial''_{\mathcal{X}} : E''_{\mathcal{X}} \to U''_{\mathcal{X}}$

between fibrant objects and a pullback

$$\begin{array}{ccc} E_{\mathcal{X}} & \dashrightarrow & E''_{\mathcal{X}} \\ {\scriptstyle \partial_{\mathcal{X}}}\downarrow & \lrcorner & \downarrow{\scriptstyle \partial''_{\mathcal{X}}} \\ U_{\mathcal{X}} & \xrightarrow[\iota]{} & U''_{\mathcal{X}} \end{array}$$

such that $\partial''_{\mathcal{X}}$ has unit type and Σ-types. Let $U'_{\mathcal{X}}$ be the object of $\mathcal{X}$ such that a map $A \to U'_{\mathcal{X}}$ corresponds to a triple (f_1, f_2, g) consisting of maps $f_1 : A \to U_{\mathcal{X}}$ and $f_2 : A \to U''_{\mathcal{X}}$ and a homotopy equivalence $g : f_1^* E_{\mathcal{X}} \simeq f_2^* E''_{\mathcal{X}}$ over A. By definition, we have a factorization $U_{\mathcal{X}} \xrightarrow{\iota'} U'_{\mathcal{X}} \xrightarrow{\iota''} U''_{\mathcal{X}}$ of ι, and ι' is a homotopy equivalence by the univalence of $\partial''_{\mathcal{X}}$. Let $\partial'_{\mathcal{X}}$ be the pullback of $\partial''_{\mathcal{X}}$ along ι'', and then $\partial_{\mathcal{X}}$ is the pullback of $\partial'_{\mathcal{X}}$ along ι'. Since ι' is a homotopy equivalence, we still have $\gamma_{\mathcal{X}}(\partial'_{\mathcal{X}}) \simeq \mathcal{M}(\partial)$. By construction, the pullbacks of $\partial'_{\mathcal{X}}$ are the maps that are both a pullback of $\partial''_{\mathcal{X}}$ and a homotopy pullback of $\partial_{\mathcal{X}}$. Then, $\partial'_{\mathcal{X}}$ has unit type and Σ-types. Indeed, since pullbacks of $\partial''_{\mathcal{X}}$ and homotopy pullbacks of $\partial_{\mathcal{X}}$ are closed under composition, pullbacks of $\partial'_{\mathcal{X}}$ are closed under composition.

We end this section with discussion about variants of Theorem 6.3.20. Although the splitting technique developed here works for Theorem 6.3.20, it is not straightforward to generalize it to extensions by Π-types and inductive types. The strategy used in Section 6.3.2 does not work. We could prove Lemma 6.3.15 if $\mathcal{T}$ is obtained from $\mathbb{I}$ by adjoining types and terms, but type constructors of interest have equality rules which are stricter than homotopies in $(\infty, 1)$-categories.

A minor modification of the proof of Lemma 6.3.24 works for Π-types. By extending $\mathbb{I}$, $\mathbb{I}_\infty$, and $\mathbb{E}_\infty$ with Π-types, we obtain a span of ∞-type theories

$$\mathbb{I}^{\Pi} \xleftarrow{\tau} \mathbb{I}^{\Pi}_{\infty} \xrightarrow{\gamma} \mathbb{E}^{\Pi}_{\infty}. \tag{6.8}$$

We follow the proof of Lemma 6.3.24, but we further require the fibration $\partial''_{\mathcal{X}}$ to also have Π-types. There is a size issue: we cannot guarantee that $\partial''_{\mathcal{X}}$ lives in the same Grothendieck universe as $\partial_{\mathcal{X}}$ does unless sufficiently many inaccessible cardinals exist [128]. We can thus only prove that any cofibrant democratic model of $\mathbb{I}^{\Pi}_{\infty}$ is a retract of a democratic model $\mathbb{I}^{\Pi}$ in a larger Grothendieck universe. Nevertheless, this is enough to prove that any cofibrant democratic model of $\mathbb{I}^{\Pi}_{\infty}$ belongs to $\mathbf{Mod}^{\mathrm{dem}}(\mathbb{I}^{\Pi})$, and we have the following.

6.3.25. Theorem. *Assuming that our ambient Grothendieck universe belongs to a larger Grothendieck universe, the span* (6.8) *is a presentation of* $\mathbb{E}^{\Pi}_{\infty}$ *by* $\mathbb{I}^{\Pi}$.

6.3.26. Corollary. *Assuming that our ambient Grothendieck universe belongs to a larger Grothendieck universe, the* $(\infty, 1)$*-category* $(\infty, 1)$-**LCCC** *of locally cartesian closed* $(\infty, 1)$*-categories is a localization of* $\mathbf{Th}(\mathbb{I}^{\Pi})$.

Proof:
By Theorem 6.2.15. □

The splitting technique developed here does not work for inductive types, because having inductive types is not a closure property of small maps. We would thus have to develop a better splitting technique. Alternatively, we should develop a syntactic proof of Lemma 6.3.11 which we expect works for a wide range of type constructors including ordinary and higher inductive types without any size issue. Of course, this is challenging since we have not known even what a syntactic presentation of an ∞-type theory would be.

Chapter 7

Models of cubical type theory

In this and the next chapters we consider particular examples of type theories, *univalent type theory* and *cubical type theory (CTT)*. By univalent type theory, we mean the extension of Martin-Löf type theory with Voevodsky's *univalence axiom* [172, Section 2.10] which asserts that equalities between two types are equivalent to invertible functions between the types. Combined with *higher inductive types* [172, Chapter 6], the univalence axiom provides proofs of theorems in homotopy theory that can be interpreted in an arbitrary Grothendieck $(\infty, 1)$-topos [151] and formalized in computer proof assistants [19, 31].

One disadvantage of the formulation of the univalence axiom and higher inductive types in [172] is that it lacks good computational behavior. Since the univalence axiom is added as a new symbol without any computation rule, terms containing the univalence axiom cannot be reduced to simpler ones. Some equality rules for higher inductive types hold only up to intensional equality rather than judgmental equality. CTT [41] is another extension of Martin-Löf type theory which provides a computational justification for the univalence axiom and higher inductive types: univalence is derivable in CTT; CTT satisfies canonicity [83]; CTT admits computational semantics [4]; higher inductive types in CTT have judgmental equality rules for path constructors [44]. From a semantic point of view, cubical methods are a good source of models of univalent type theory [22, 23, 10]. Cubical methods work in a constructive metalogic in contrast to Voevodsky's simplicial model [102] which is not valid constructively [24]. This suggests that one can build a cubical model over a base category other than **Set** such as the category of assemblies which we will study in Chapter 8. Indeed, Orton and Pitts [134, 135] and Licata et al. [111] gave a general way of constructing a model of CTT inside an (extension of) intuitionistic dependent type theory based on the idea of Coquand [43] of using the internal dependent type theory of a topos to formulate fibrations of cubical sets.

This chapter is mostly devoted to reviewing the method of constructing a model of CTT from a model of extensional type theory given by Orton and Pitts

[134, 135] and Licata et al. [111]. One new contribution is the construction of some higher inductive types which is joint work with Andrew Swan [163]. We consider four type theories.

- We temporarily define *the good type theory* to be the extension of DTT with Π-types, Σ-types, extensional identity types, unit type, strictly extensive finite colimits, natural numbers type, and propositional truncation.

- We introduce *quasi-cubical type theory (qCTT)* which is a type theory satisfying the axioms of Orton and Pitts [134, 135] for modeling CTT.

- We consider CTT in the style of Cohen et al. [41] with Π-types, Σ-types, dependent path types, identity types, unit type, finite coproducts, and natural numbers type.

- By *univalent type theory*, we mean the extension of DTT with Π-types, Σ-types, intensional identity types, unit type, finite coproducts, natural numbers type, and a countable chain of univalent universes.

The construction of a model of CTT is divided into two steps. First, given a model of the good type theory, we construct a model of qCTT consisting of *cubical objects* (Section 7.2). The second step is to transform a model of qCTT into a model of CTT applying the method of Orton and Pitts [134, 135] and Licata et al. [111] (Section 7.1). If we start from a model of the good type theory with a countable chain of universes, then we obtain a model of CTT with a countable chain of univalent universes as univalence is derivable over CTT, and thus it is considered as a model of univalent type theory.

We deal with universes separately for two reasons. First, the axioms for constructing univalent universes cannot be fully internalized in a type theory in our sense by the "no-go" theorem of Licata et al. [111]. Although they introduced a new type theory called crisp type theory to internalize their construction, that type theory has dual-contexts and is beyond the scope of this thesis. We thus need external assumptions on a model of qCTT to construct univalent universes. Second, there are a few options for the style of universes such as a hierarchy of predicative universes and an impredicative universe. We would therefore like a modular construction that works for any style of universes. We also deal with some selected higher inductive types separately.

The type theories considered in this chapter are all defined as SOGATs; see Section 4.6 for how type constructors and universes are defined. The reader need not read all the details of Chapter 4 to understand the results in this and the next chapters. All we need here is that a symbol

$$S : (\mathtt{X}_1 : \Gamma_1 \to e_1, \ldots, \mathtt{X}_n : \Gamma_n \to e_n) \Rightarrow e$$

in a SOGAT expresses an inference rule

$$\frac{\Gamma_1 \vdash \mathtt{X}_1 : e_1 \qquad \dots \qquad \Gamma_n \vdash \mathtt{X}_n : e_n}{\vdash S(\mathtt{X}_1, \dots, \mathtt{X}_n) : e}$$

and that a model of a SOGAT is an interpretation of the inference rules. For example, part of a model of CTT is as follows:

- a category $\mathcal{M}(\diamond)$ with a final object;
- a representable map $\mathcal{M}(E) \to \mathcal{M}(U)$ of discrete fibrations over $\mathcal{M}(\diamond)$;
- a representable monomorphism $\mathcal{M}(\mathsf{true}) \to \mathcal{M}(\mathsf{Cof})$ of discrete fibrations over $\mathcal{M}(\diamond)$;
- a discrete fibration $\mathcal{M}(\mathbb{I})$ over $\mathcal{M}(\diamond)$ such that the final projection $\mathcal{M}(\mathbb{I}) \to 1$ is a representable map;
- a bunch of maps between discrete fibrations over $\mathcal{M}(\diamond)$ interpreting the inference rules of CTT.

We freely use the internal language of a model $\mathcal{M}$ described in Section 5.1.1, so we may call a section of $\mathcal{M}(U)$ a type for example. Again, an informal understanding of the internal language is enough to read this and the next chapters: the internal language of a model of a type theory is the extension of the type theory where elements of the model are adjoined as constants.

7.0.1. Remark. Although we work with CTT in the style of Cohen et al. [41] for concreteness, the method also works for other styles of CTT. In the original work by Orton and Pitts [134, 135], the reversal on the interval is dropped. Angiuli et al. [5] gave a similar way of constructing a model of cartesian cubical type theory. The work by Cavallo, Mörtberg, and Swan [37] does not depend on any particular style of CTT, and different styles of models are obtained as special cases.

7.1 A general construction of a model of CTT

We first introduce a type theory called *quasi-cubical type theory (qCTT)* and explain how to transform a model of qCTT into a model of CTT. We explain qCTT in Section 7.1.1. Given a model $\mathcal{M}$ of qCTT, we construct in Section 7.1.2 a model $\mathcal{M}_{\mathrm{fib}}$ of CTT consisting of *fibrations* in the same way as the construction of a universe given by Licata et al. [111]. In Section 7.1.3, we describe how $\mathcal{M}_{\mathrm{fib}}$ models the type constructors of CTT. In Section 7.1.4, we see that if $\mathcal{M}$ models a universe, then so does $\mathcal{M}_{\mathrm{fib}}$ under suitable extra assumptions. We construct in Section 7.1.5 selected higher inductive types in $\mathcal{M}_{\mathrm{fib}}$ when $\mathcal{M}$ models W-types with reductions. In Section 7.1.6 we introduce *discrete types* in $\mathcal{M}$ which are automatically fibrations.

7.1.1 Axioms for modeling CTT

By *quasi-cubical type theory (qCTT)*, we mean a type theory satisfying the axioms of Orton and Pitts [134, 135] for modeling CTT. In this section, we rephrase their axioms in the language of SOGATs, that is, we define qCTT as a SOGAT extending the good type theory with certain symbols and axioms. We first add the *interval* as a type.

$$\begin{aligned} \mathbb{I} &: () \Rightarrow U \\ 0_{\mathbb{I}} &: () \Rightarrow \mathbb{I} \\ 1_{\mathbb{I}} &: () \Rightarrow \mathbb{I} \end{aligned}$$

We also add a De Morgan algebra structure on $\mathbb{I}$. We add a propositional universe of *cofibrations*, that is, a universe whose elements are regarded as propositions.

$$\begin{aligned} \mathsf{Cof} &: () \Rightarrow U \\ \mathsf{true} &: (\mathtt{P} : \mathsf{Cof}) \Rightarrow U \\ _ &: (\mathtt{P} : \mathsf{Cof}, \mathtt{a}_1 : \mathsf{true}(\mathtt{P}), \mathtt{a}_2 : \mathsf{true}(\mathtt{P})) \Rightarrow \mathtt{a}_1 \equiv \mathtt{a}_2 : \mathsf{true}(\mathtt{P}) \end{aligned}$$

We make Cof weakly closed (Example 4.6.11) under Σ-types, unit type, finite disjunctions, Π-types over $\mathbb{I}$, and equalities with the endpoints $0_{\mathbb{I}}$ and $1_{\mathbb{I}}$.

We further add some axioms. The endpoints of the interval should be distinct to obtain a non-trivial model.

$$_ : (_ : () \to 0_{\mathbb{I}} \equiv 1_{\mathbb{I}} : \mathbb{I}) \Rightarrow \bot$$

We also require $\mathbb{I}$ to be *connected*.

$$_ : (\mathtt{P} : (\mathtt{i} : \mathbb{I}) \to 2) \Rightarrow (\forall_{\mathtt{i}:\mathbb{I}} \mathtt{P}(\mathtt{i}) == 0) \vee (\forall_{\mathtt{i}:\mathbb{I}} \mathtt{P}(\mathtt{i}) == 1) \tag{7.1}$$

We will need the following *propositional extensionality* of cofibrations to construct a composition structure on identity types.

$$_ : (\mathtt{P} : () \to \mathsf{Cof}, \mathtt{Q} : () \to \mathsf{Cof}, _ : (_ : \mathtt{P}) \to \mathtt{Q}, _ : (_ : \mathtt{Q}) \to \mathtt{P}) \Rightarrow \mathtt{P} \equiv \mathtt{Q} : \mathsf{Cof}$$

Using Π-types, Σ-types, and extensional identity types, we can define the type of isomorphisms $\mathsf{Iso}(\mathtt{A}, \mathtt{B})$ between $\mathtt{A}$ and $\mathtt{B}$. The following *isomorphism extension structure* (on Cof valued in U) will be used for defining the glueing operation.

$$\begin{aligned} \mathsf{isoext}_1 &: (\mathtt{P} : () \to \mathsf{Cof}, \mathtt{A} : (_ : \mathtt{P}) \to U, \mathtt{B} : () \to U, \mathtt{f} : (_ : \mathtt{P}) \to \mathsf{Iso}(\mathtt{A}, \mathtt{B})) \Rightarrow U \\ _ &: (\mathtt{P} : () \to \mathsf{Cof}, \mathtt{A} : (_ : \mathtt{P}) \to U, \mathtt{B} : () \to U, \mathtt{f} : (_ : \mathtt{P}) \to \mathsf{Iso}(\mathtt{A}, \mathtt{B}), _ : () \to \mathtt{P}) \\ &\quad \Rightarrow \mathsf{isoext}_1(\mathtt{P}, \mathtt{A}, \mathtt{B}, \mathtt{f}) \equiv \mathtt{A} : U \\ \mathsf{isoext}_2 &: (\mathtt{P} : () \to \mathsf{Cof}, \mathtt{A} : (_ : \mathtt{P}) \to U, \mathtt{B} : () \to U, \mathtt{f} : (_ : \mathtt{P}) \to \mathsf{Iso}(\mathtt{A}, \mathtt{B})) \\ &\quad \Rightarrow \mathsf{Iso}(\mathsf{isoext}_1(\mathtt{P}, \mathtt{A}, \mathtt{B}, \mathtt{f}), \mathtt{B}) \\ _ &: (\mathtt{P} : () \to \mathsf{Cof}, \mathtt{A} : (_ : \mathtt{P}) \to U, \mathtt{B} : () \to U, \mathtt{f} : (_ : \mathtt{P}) \to \mathsf{Iso}(\mathtt{A}, \mathtt{B}), _ : () \to \mathtt{P}) \\ &\quad \Rightarrow \mathsf{isoext}_2(\mathtt{P}, \mathtt{A}, \mathtt{B}, \mathtt{f}) \equiv \mathtt{f} : \mathsf{Iso}(\mathtt{A}, \mathtt{B}) \end{aligned}$$

7.1.2 Fibrations

We define a type $\mathsf{Comp}(\mathtt{A})$ of *composition structures on* $\mathtt{A}$ inside qCTT. It is a type in the form of

$$\mathsf{Comp} : (\mathtt{A} : (\mathtt{i} : \mathbb{I}) \to U) \Rightarrow U$$

such that a term of the form $(\mathtt{A} : (\mathtt{i} : \mathbb{I}) \to U) \Rightarrow \mathsf{Comp}(\mathtt{A})$ corresponds to a composition operation. It is indeed definable in qCTT as

$$\begin{aligned}\mathsf{Comp}(\mathtt{A}) \equiv (\mathtt{P} : \mathsf{Cof}, \mathtt{a} : \mathtt{P} \texttt{ -> } (\mathtt{i} : \mathbb{I}) \texttt{ -> } \mathtt{A}(\mathtt{i}), \mathtt{a}_0 : \mathtt{A}(0_{\mathbb{I}})) \texttt{ -> } (\mathtt{P} \texttt{ -> } \mathtt{a}_0 \texttt{ == } \mathtt{a}(0_{\mathbb{I}})) \\ \texttt{ -> } ((\mathtt{a}_1 : \mathtt{A}(1_{\mathbb{I}})) \times (\mathtt{P} \texttt{ -> } \mathtt{a}_1 \texttt{ == } \mathtt{a}(1_{\mathbb{I}})).\end{aligned}$$

We also define the type of homogeneous composition structures $\mathsf{HComp}(\mathtt{A}) \equiv \mathsf{Comp}(\langle \mathtt{i} \rangle \mathtt{A})$ for $\mathtt{A} : () \to U$.

Let $\mathcal{M}$ be a model of qCTT. Since the final projection $\mathcal{M}(\mathbb{I}) \to 1$ is a representable map, the exponentiation by $\mathcal{M}(\mathbb{I})$ has a right adjoint by Corollary 3.1.18 which we refer to as $\surd$. Let $\mathcal{M}_{\mathrm{fib}}(U)$ be the pullback

$$\begin{array}{ccc} \mathcal{M}_{\mathrm{fib}}(U) & \longrightarrow & \surd(\mathcal{M}(E)) \\ \downarrow & \lrcorner & \downarrow \\ \mathcal{M}(U) & \xrightarrow[\mathcal{M}(\mathsf{Comp})']{} & \surd(\mathcal{M}(U)) \end{array} \tag{7.2}$$

where $\mathcal{M}(\mathsf{Comp})'$ is the transpose of $\mathcal{M}(\mathsf{Comp}) : \mathcal{M}(U)^{\mathcal{M}(\mathbb{I})} \to \mathcal{M}(U)$. We define $\mathcal{M}_{\mathrm{fib}}(E)$ to be the pullback of $\mathcal{M}(E)$ along the map $\mathcal{M}_{\mathrm{fib}}(U) \to \mathcal{M}(U)$.

7.1.1. Theorem (Orton and Pitts [134, 135]). *For any model $\mathcal{M}$ of qCTT, the representable maps $\mathcal{M}_{\mathrm{fib}}(E) \to \mathcal{M}_{\mathrm{fib}}(U)$, $\mathcal{M}(\mathsf{true}) \to \mathcal{M}(\mathsf{Cof})$, and $\mathcal{M}(\mathbb{I}) \to 1$ are part of a model $\mathcal{M}_{\mathrm{fib}}$ of CTT. Moreover, the forgetful map $\mathcal{M}_{\mathrm{fib}} \to \mathcal{M}$ preserves Π-types, Σ-types, unit type, finite coproducts, and natural numbers type.*

Proof:
Since $\bot$ is isomorphic to the empty type, we have the eliminator $\mathsf{elim}^E_\bot$. The disjointness of finite coproducts implies that for any cofibrations $\mathtt{P} : \mathsf{Cof}$ and $\mathtt{Q} : \mathsf{Cof}$, the square

$$\begin{array}{ccc} \mathtt{P} \wedge \mathtt{Q} & \hookrightarrow & \mathtt{Q} \\ \downarrow & & \downarrow \\ \mathtt{P} & \hookrightarrow & \mathtt{P} \vee \mathtt{Q} \end{array}$$

is a pushout, and thus we have the eliminator $\mathsf{elim}^E_\vee$. By the strict extensivity of finite colimits, we also have the eliminators $\mathsf{elim}^U_\bot$ and $\mathsf{elim}^U_\vee$.

By definition, $\mathcal{M}_{\mathrm{fib}}$ models the composition operation. Indeed, the transpose of the square (7.2) gives us a map

$$\begin{array}{ccc} \mathcal{M}_{\mathrm{fib}}(U)^{\mathcal{M}(\mathbb{I})} & \overset{\mathsf{comp}}{\dashrightarrow} & \mathcal{M}(E) \\ \downarrow & & \downarrow \\ \mathcal{M}(U)^{\mathcal{M}(\mathbb{I})} & \xrightarrow[\mathcal{M}(\mathsf{Comp})]{} & \mathcal{M}(U). \end{array}$$

It remains to construct type constructors including path types and glueing which we will explain in Section 7.1.3 below. □

7.1.3 Type constructors

Type constructors on $\mathcal{M}_{\mathrm{fib}}$ are constructed in the following steps. The formation, introduction, elimination, and equality rules are defined inside qCTT. We then have to prove that a compound type carries a composition structure whenever its parameters do, but we often omit the construction of the composition structure because it is identical to the equality rule presented in [41].

Π-types, Σ-types, unit type, finite coproducts, and natural numbers type on $\mathcal{M}_{\mathrm{fib}}$ are the same as those on $\mathcal{M}$. We note that we need the connectedness of $\mathbb{I}$ for the composition structures on finite coproducts and the natural numbers type. The connectedness of $\mathbb{I}$ implies that any path $\mathtt{p} : \mathbb{I} \to \mathtt{A} + \mathtt{B}$ uniquely factors through either $\mathsf{inl} : \mathtt{A} \to \mathtt{A} + \mathtt{B}$ or $\mathsf{inr} : \mathtt{B} \to \mathtt{A} + \mathtt{B}$. Thus, if we know that $\mathtt{p}(0_{\mathbb{I}})$ belongs to $\mathtt{A}$, then $\mathtt{p}$ factors through $\mathtt{A}$. For types $\mathtt{A} : (\mathtt{i} : \mathbb{I}) \to U$ and $\mathtt{B} : (\mathtt{i} : \mathbb{I}) \to U$ with composition structures, the composition $\mathsf{comp}(\langle \mathtt{i} \rangle \mathtt{A}(\mathtt{i}) + \mathtt{B}(\mathtt{i}), \mathtt{P}, \mathtt{c}, \mathtt{c}_0)$ is defined by case analysis on $\mathtt{c}_0 : \mathtt{A}(0_{\mathbb{I}}) + \mathtt{B}(1_{\mathbb{I}})$. When $\mathtt{c}_0 \equiv \mathsf{inl}(\mathtt{a}_0)$ for $\mathtt{a}_0 : \mathtt{A}(0_{\mathbb{I}})$, the connectivity of $\mathbb{I}$ implies that $\mathtt{c} : (\mathtt{i} : \mathbb{I}, _ : \mathtt{P}) \to \mathtt{A}(\mathtt{i}) + \mathtt{B}(\mathtt{i})$ is of the form $\langle \mathtt{i} \rangle \mathsf{inl}(\mathtt{a}(\mathtt{i}))$ for a unique $\mathtt{a} : (\mathtt{i} : \mathbb{I}, _ : \mathtt{P}) \to \mathtt{A}(\mathtt{i})$. Then the composition is defined by

$$\mathsf{comp}(\langle \mathtt{i} \rangle \mathtt{A}(\mathtt{i}) + \mathtt{B}(\mathtt{i}), \mathtt{P}, \langle \mathtt{i} \rangle \mathsf{inl}(\mathtt{a}(\mathtt{i})), \mathsf{inl}(\mathtt{a}_0)) \equiv \mathsf{inl}(\mathsf{comp}(\mathtt{A}, \mathtt{P}, \mathtt{a}, \mathtt{a}_0)).$$

The case when $\mathtt{c}_0 \equiv \mathsf{inr}(\mathtt{b}_0)$ is symmetric. For the composition $\mathsf{comp}(\langle \mathtt{i} \rangle \mathsf{N}, \mathtt{P}, \mathtt{n}, \mathtt{n}_0)$ for the natural numbers type, we proceed by induction on $\mathtt{n}_0 : \mathsf{N}$. When $\mathtt{n}_0 \equiv \mathsf{zero}$, we have $(\mathtt{i} : \mathbb{I}, _ : \mathtt{P}) \to \mathtt{n}(\mathtt{i}) \equiv \mathsf{zero} : \mathsf{N}$ by the connectedness of $\mathbb{I}$ and define

$$\mathsf{comp}(\langle \mathtt{i} \rangle \mathsf{N}, \mathtt{P}, \langle \mathtt{i} \rangle \mathsf{zero}, \mathsf{zero}) \equiv \mathsf{zero}.$$

When $\mathtt{n}_0 \equiv \mathsf{succ}(\mathtt{n}_0')$, we have $(\mathtt{i} : \mathbb{I}, _ : \mathtt{P}) \to \mathtt{n}(\mathtt{i}) \equiv \mathsf{succ}(\mathtt{n}'(\mathtt{i})) : \mathsf{N}$ for a unique $\mathtt{n}' : (\mathtt{i} : \mathbb{I}, _ : \mathtt{P}) \to \mathsf{N}$ by the connectedness of $\mathbb{I}$ and define

$$\mathsf{comp}(\langle \mathtt{i} \rangle \mathsf{N}, \mathtt{P}, \langle \mathtt{i} \rangle \mathsf{succ}(\mathtt{n}'(\mathtt{i})), \mathsf{succ}(\mathtt{n}_0')) \equiv \mathsf{succ}(\mathsf{comp}(\langle \mathtt{i} \rangle \mathsf{N}, \mathtt{P}, \mathtt{n}', \mathtt{n}_0')).$$

One may notice that $\mathsf{comp}(\langle \mathtt{i} \rangle \mathsf{N}, \mathtt{P}, \mathtt{n}, \mathtt{n}_0) \equiv \mathtt{n}_0$ for any $\mathtt{n}_0$ by induction. In fact, the type N is *discrete* (Section 7.1.6) and we can define $\mathsf{comp}(\langle \mathtt{i} \rangle \mathtt{A}, \mathtt{P}, \mathtt{a}, \mathtt{a}_0) \equiv \mathtt{a}_0$ for any discrete type $\mathtt{A}$.

It remains to construct path types, identity types, and glueing.

Path types and identity types

The *(dependent) path type*

$$\mathsf{Path} : (\mathtt{A} : (\mathtt{i} : \mathbb{I}) \to U, \mathtt{a}_0 : () \to \mathtt{A}(0_\mathbb{I}), \mathtt{a}_1 : () \to \mathtt{A}(1_\mathbb{I})) \Rightarrow U$$

is defined by

$$\mathsf{Path}(\mathtt{A}, \mathtt{a}_0, \mathtt{a}_1) \equiv \{\mathtt{p} : (\mathtt{i} : \mathbb{I}) \texttt{ -> } \mathtt{A}(\mathtt{i}) \mid \mathtt{p}(0_\mathbb{I}) \texttt{ == } \mathtt{a}_0 \wedge \mathtt{p}(1_\mathbb{I}) \texttt{ == } \mathtt{a}_1\}.$$

The introduction, elimination, and computation rules are obvious.

The *identity type*

$$\mathsf{Id} : ([\mathtt{A} : (\mathtt{i} : \mathbb{I}) \to U], \mathtt{a}_0 : () \to \mathtt{A}, \mathtt{a}_1 : () \to \mathtt{A}) \Rightarrow U$$

is defined by

$$\mathsf{Id}(\mathtt{a}_0, \mathtt{a}_1) \equiv (\mathtt{p} : \mathsf{Path}(\langle \mathtt{i} \rangle \mathtt{A}, \mathtt{a}_0, \mathtt{a}_1)) \times \{\mathtt{Q} : \mathsf{Cof} \mid \mathtt{Q} \texttt{ -> } \forall_{\mathtt{i}:\mathbb{I}} \mathtt{p}(\mathtt{i}) \texttt{ == } \mathtt{a}_0\}$$

which is a variant of Swan's construction [162]. The introduction, elimination, and computation rules are obvious. We note that we need the propositional extensionality of cofibrations to construct a composition operation on $\mathsf{Id}(\mathtt{a}_0, \mathtt{a}_1)$. Recall from [41, Section 9.1] that for $\mathtt{A} : (\mathtt{i} : \mathbb{I}) \to U$, $\mathtt{a}_0 : (\mathtt{i} : \mathbb{I}) \to \mathtt{A}(\mathtt{i})$, and $\mathtt{a}_1 : (\mathtt{i} : \mathbb{I}) \to \mathtt{A}(\mathtt{i})$, the composition operation for $\langle \mathtt{i} \rangle \mathsf{Id}(\mathtt{a}_0(\mathtt{i}), \mathtt{a}_1(\mathtt{i}))$ is defined by

$$\begin{aligned} &\mathsf{comp}(\langle \mathtt{i} \rangle \mathsf{Id}(\mathtt{a}_0(\mathtt{i}), \mathtt{a}_1(\mathtt{i})), \mathtt{P}, \langle \mathtt{i} \rangle (\mathtt{p}(\mathtt{i}), \mathtt{Q}(\mathtt{i})), (\mathtt{p}_0, \mathtt{Q}_0)) \\ &\qquad \equiv (\mathsf{comp}(\langle \mathtt{i} \rangle \mathsf{Path}(\langle \mathtt{i}' \rangle \mathtt{A}(\mathtt{i}), \mathtt{a}_0(\mathtt{i}), \mathtt{a}_1(\mathtt{i})), \mathtt{P}, \mathtt{p}, \mathtt{p}_0), \mathtt{P} \wedge \mathtt{Q}(1_\mathbb{I})). \end{aligned}$$

For this to be well-defined, we have to derive $\mathtt{P} \texttt{ -> } \mathtt{P} \wedge \mathtt{Q}(1_\mathbb{I}) \texttt{ == } \mathtt{Q}(1_\mathbb{I})$. It is immediate that $\mathtt{P} \texttt{ -> } (\mathtt{P} \wedge \mathtt{Q}(1_\mathbb{I}) \leftrightarrow \mathtt{Q}(1_\mathbb{I}))$, and then use the propositional extensionality.

Glueing

We would like to construct the *glueing*

$$\mathsf{Glue} : (\mathtt{P} : () \to \mathsf{Cof}, \mathtt{A} : (_ : \mathtt{P}) \to U, \mathtt{B} : () \to U, \mathtt{f} : (_ : \mathtt{P}) \to \mathsf{Equiv}(\mathtt{A}, \mathtt{B})) \Rightarrow U.$$

From the introduction, elimination, and computation rules for glueing, $\mathsf{Glue}(\mathtt{P}, \mathtt{A}, \mathtt{B}, \mathtt{f})$ must be isomorphic to

$$\mathsf{Glue}'(\mathtt{P}, \mathtt{A}, \mathtt{B}, \mathtt{f}) \equiv (\mathtt{x} : \mathtt{P} \texttt{ -> } \mathtt{A}) \times \{\mathtt{y} : \mathtt{B} \mid \mathtt{P} \texttt{ -> } \mathtt{f}(\mathtt{x}) \texttt{ == } \mathtt{y}\}.$$

We have a canonical isomorphism $\mathtt{g}' : (_ : \mathtt{P}) \to \mathsf{Iso}(\mathsf{Glue}'(\mathtt{P}, \mathtt{A}, \mathtt{B}, \mathtt{f}), \mathtt{A})$ defined by the first projection with inverse $\lambda\mathtt{x}.(\mathtt{x}, f(\mathtt{x}))$. However, the glueing in CTT requires the judgmental equality $(_ : \mathtt{P}) \to \mathsf{Glue}(\mathtt{P}, \mathtt{A}, \mathtt{B}, \mathtt{f}) \equiv \mathtt{A} : U$. To fix this, we use the isomorphism extension structure to obtain a type $\mathsf{Glue}(\mathtt{P}, \mathtt{A}, \mathtt{B}, \mathtt{f})$ and an isomorphism $\mathtt{g} : (_ : \mathtt{P}) \to \mathsf{Iso}(\mathsf{Glue}'(\mathtt{P}, \mathtt{A}, \mathtt{B}, \mathtt{f}), \mathsf{Glue}(\mathtt{P}, \mathtt{A}, \mathtt{B}, \mathtt{f}))$ such that $(_ : \mathtt{P}) \to \mathsf{Glue}(\mathtt{P}, \mathtt{A}, \mathtt{B}, \mathtt{f}) \equiv \mathtt{A} : U$ and $(_ : \mathtt{P}) \to \mathtt{g} \equiv \mathtt{g}' : \mathsf{Iso}(\mathsf{Glue}'(\mathtt{P}, \mathtt{A}, \mathtt{B}, \mathtt{f}), \mathtt{A})$.

7.1.4 Universes

Suppose that $\mathcal{M}$ is a model of qCTT that also models a universe u with closure properties sufficient to define Comp inside u. In the same way as $\mathcal{M}_{\mathrm{fib}}(U)$, we can form the discrete fibration $\mathcal{M}_{\mathrm{fib}}(\mathsf{u})$ of fibrations in $\mathcal{M}(\mathsf{u})$.

$$\begin{array}{ccc} \mathcal{M}_{\mathrm{fib}}(\mathsf{u}) & \longrightarrow & \surd(\mathcal{M}(\mathsf{el}_{\mathsf{u}})) \\ \downarrow & \lrcorner & \downarrow \\ \mathcal{M}(\mathsf{u}) & \xrightarrow[\mathcal{M}(\mathsf{Comp})']{} & \surd(\mathcal{M}(\mathsf{u})) \end{array}$$

To say that $\mathcal{M}_{\mathrm{fib}}(\mathsf{u})$ is a universe in $\mathcal{M}_{\mathrm{fib}}$, we have to show that $\mathcal{M}_{\mathrm{fib}}(\mathsf{u})$ is $\mathcal{M}(U)$-small and carries a composition structure. We need additional assumptions on $\mathcal{M}$ to derive these. Recall that the composition structure for a universe is constructed using glueing [41, Section 7.1]. Since the construction of glueing in $\mathcal{M}_{\mathrm{fib}}$ depends on the isomorphism extension structure, we need an isomorphism extension structure valued in u. For $\mathcal{M}_{\mathrm{fib}}(\mathsf{u})$ to be $\mathcal{M}(U)$-small, it is enough to assume that the functor $\surd$ preserves $\mathcal{M}(U)$-small objects, that is, for any *global* section $A : \mathcal{M}(\diamond) \to \mathcal{M}(U)$, we have a global section $\surd(A) : \mathcal{M}(\diamond) \to \mathcal{M}(U)$ such that $\surd(A)^*\mathcal{M}(E) \cong \surd(A^*\mathcal{M}(E))$.

7.1.2. Remark. We cannot realize $\surd$ as a *local* operator $\surd : \mathcal{M}(U) \to \mathcal{M}(U)$ unless $\mathcal{M}$ is trivial [111, Theorem 5.1].

If $\mathcal{M}$ models nested universes $\mathsf{u} : \widehat{\mathsf{u}}$, then we require that $\surd$ preserves $\mathcal{M}(\widehat{\mathsf{u}})$-small objects to ensure that $\mathcal{M}_{\mathrm{fib}}(\mathsf{u})$ is $\mathcal{M}_{\mathrm{fib}}(\widehat{\mathsf{u}})$-small. To summarize:

7.1.3. Theorem (Licata et al. [111]). *Let $\mathcal{M}$ be a model of qCTT. Suppose that $\mathcal{M}$ models a universe u containing Cof and $\mathbb{I}$ and weakly closed under Π-types, Σ-types, and extensional identity types and that $\mathcal{M}$ has an isomorphism extension structure on Cof valued in u. If the functor $\surd$ preserves $\mathcal{M}(U)$-small objects, then we have a universe $\mathcal{M}_{\mathrm{fib}}(\mathsf{u})$ in $\mathcal{M}_{\mathrm{fib}}$ weakly closed under Π-types, Σ-types, path types, identity types, and glueing. When u is contained in a larger universe $\mathsf{u} : \widehat{\mathsf{u}}$ and $\surd$ preserves $\mathcal{M}(\widehat{\mathsf{u}})$-small objects, $\mathcal{M}_{\mathrm{fib}}(\mathsf{u})$ is contained in $\mathcal{M}_{\mathrm{fib}}(\widehat{\mathsf{u}})$.*

7.1.5 Higher inductive types

We assume that $\mathcal{M}$ models W-types with reductions with respect to Cof. We construct some higher inductive types in the model of CTT $\mathcal{M}_{\mathrm{fib}}$. The idea is to internalize the argument of Coquand, Huber, and Mörtberg [44] using W-types with reductions. The contents of this section are joint work with Andrew Swan [163].

To begin with, we give a way of freely adjoining a homogeneous composition structure to a type in $\mathcal{M}$.

7.1.4. Definition. Given a type $\mathtt{A}$, we define the *local fibrant replacement* $\mathsf{LFR}(\mathtt{A})$ to be the W-type with reductions $\mathsf{W}(\mathtt{A}_0, \mathtt{B}_0, \mathtt{P}_0, \mathtt{f}_0)$ where

- the type $\mathtt{A}_0$ is $\mathtt{A} + \mathsf{Cof}$;
- $\mathtt{B}_0(\mathsf{inl}(\mathtt{x})) \equiv 0$ and $\mathtt{B}_0(\mathsf{inr}(\mathtt{P})) \equiv ((\mathtt{i} : \mathbb{I}) \times ((\mathtt{i} == 0_{\mathbb{I}}) \vee \mathtt{P}))$;
- $\mathtt{P}_0(\mathsf{inl}(\mathtt{x})) \equiv \bot$ and $\mathtt{P}_0(\mathsf{inr}(\mathtt{P})) \equiv \mathtt{P}$;
- $\mathtt{f}_0(\mathsf{inr}(\mathtt{P})) \equiv 1_{\mathbb{I}}$.

The constructor sup is decomposed into two constructors.

$$\mathsf{sup}_{\mathsf{inl}} : (\mathtt{a} : () \to \mathtt{A}) \Rightarrow \mathsf{LFR}(\mathtt{A})$$
$$\mathsf{sup}_{\mathsf{inr}} : (\mathtt{P} : () \to \mathsf{Cof}, \mathtt{c} : (\mathtt{i} : \mathbb{I}, _ : (\mathtt{i} == 0_{\mathbb{I}}) \vee \mathtt{P}) \to \mathsf{LFR}(\mathtt{A})) \Rightarrow \mathsf{LFR}(\mathtt{A})$$

The second one is equivalent to the following.

$$\mathsf{sup}_{\mathsf{inr}} : (\mathtt{P} : () \to \mathsf{Cof}, \mathtt{c} : (_ : \mathtt{P}, \mathtt{i} : \mathbb{I}) \to \mathsf{LFR}(\mathtt{A}), \mathtt{c}_0 : () \to \mathsf{LFR}(\mathtt{A}),$$
$$_ : (_ : \mathtt{P}) \to \mathtt{c}_0 == \mathtt{c}(0_{\mathbb{I}})) \Rightarrow \mathsf{LFR}(\mathtt{A})$$

The reduction is then given by

$$_ : (\mathtt{P} : () \to \mathsf{Cof}, \mathtt{c} : (_ : \mathtt{P}, \mathtt{i} : \mathbb{I}) \to \mathsf{LFR}(\mathtt{A}), \mathtt{c}_0 : () \to \mathsf{LFR}(\mathtt{P}),$$
$$_ : (_ : \mathtt{P}) \to \mathtt{c}_0 == \mathtt{c}(0_{\mathbb{I}}), _ : () \to \mathtt{P}) \Rightarrow \mathsf{sup}_{\mathsf{inr}}(\mathtt{P}, \mathtt{c}, \mathtt{c}_0) == \mathtt{c}(1_{\mathbb{I}}).$$

Therefore, $\mathsf{LFR}(\mathtt{A})$ is the type obtained from $\mathtt{A}$ by freely adjoining a homogeneous composition structure $\mathsf{sup}_{\mathsf{inr}}$.

7.1.5. Example. Given a type $\mathtt{A}$ in $\mathcal{M}_{\mathrm{fib}}$, the *suspension* $\mathsf{Susp}(\mathtt{A})$ is defined as follows. We first take the following pushout in $\mathcal{M}$.

$$\begin{array}{ccc} \mathtt{A} \times 2 & \longrightarrow & 2 \\ \downarrow & \ulcorner & \downarrow \\ \mathtt{A} \times \mathbb{I} & \longrightarrow & \mathsf{Susp}'(\mathtt{A}) \end{array}$$

Then $\mathsf{Susp}(\mathtt{A}) := \mathsf{LFR}(\mathsf{Susp}'(\mathtt{A}))$ is an initial $\mathsf{Susp}\,\mathtt{A}$-algebra in the sense of [44, Section 2.2] and then the same proof works.

7.1.6. Example. Given a type $\mathtt{A}$ in $\mathcal{M}_{\mathrm{fib}}$, the *propositional truncation* $\mathsf{Trunc}(\mathtt{A})$ is defined to be the W-type with reductions for the sum of the polynomial with reductions $(\mathtt{A}_0, \mathtt{B}_0, \mathtt{P}_0, \mathtt{f}_0)$ of Definition 7.1.4 and the following polynomial with reductions $(\mathtt{A}_1, \mathtt{B}_1, \mathtt{P}_1, \mathtt{f}_1)$:

- $\mathtt{A}_1 \equiv \mathbb{I}$;
- $\mathtt{B}_1(\mathtt{i}) \equiv 2$;
- $\mathtt{P}_1(\mathtt{i}) \equiv (\mathtt{i} == 0_{\mathbb{I}}) \vee (\mathtt{i} == 1_{\mathbb{I}})$;
- $\mathtt{f}_1(0_{\mathbb{I}}) \equiv 0$ and $\mathtt{f}_1(1_{\mathbb{I}}) \equiv 1$.

The constructor sup is decomposed into two constructors.

$$\mathsf{sup}_{\mathsf{inl}} : (\mathtt{a} : () \to \mathtt{A}_0, \mathtt{c} : (\mathtt{y} : \mathtt{B}_0(\mathtt{a})) \to \mathsf{Trunc}(\mathtt{A})) \Rightarrow \mathsf{Trunc}(\mathtt{A})$$
$$\mathsf{sup}_{\mathsf{inr}} : (\mathtt{i} : () \to \mathbb{I}, \mathtt{c} : 2 \to \mathsf{Trunc}(\mathtt{A})) \Rightarrow \mathsf{Trunc}(\mathtt{A})$$

The first one is decomposed as in Definition 7.1.4 and gives an inclusion $\mathtt{A}$ -> $\mathsf{Trunc}(\mathtt{A})$ and a homogeneous composition structure on $\mathsf{Trunc}(\mathtt{A})$. The second one is equivalent to

$$\mathsf{sup}_{\mathsf{inr}} : (\mathtt{c}_0 : \mathsf{Trunc}(\mathtt{A}), \mathtt{c}_1 : \mathsf{Trunc}(\mathtt{A}), \mathtt{i} : () \to \mathbb{I}) \Rightarrow \mathsf{Trunc}(\mathtt{A})$$

and the reductions are

$$_ : (\mathtt{c}_0 : \mathsf{Trunc}(\mathtt{A}), \mathtt{c}_1 : \mathsf{Trunc}(\mathtt{A})) \Rightarrow \mathsf{sup}_{\mathsf{inr}}(\mathtt{c}_0, \mathtt{c}_1, 0_{\mathbb{I}}) == \mathtt{c}_0$$
$$_ : (\mathtt{c}_0 : \mathsf{Trunc}(\mathtt{A}), \mathtt{c}_1 : \mathsf{Trunc}(\mathtt{A})) \Rightarrow \mathsf{sup}_{\mathsf{inr}}(\mathtt{c}_0, \mathtt{c}_1, 1_{\mathbb{I}}) == \mathtt{c}_1.$$

Therefore, $\mathsf{sup}_{\mathsf{inr}}$ constructs a path between arbitrary two points of $\mathsf{Trunc}(\mathtt{A})$, forcing $\mathsf{Trunc}(\mathtt{A})$ to be a proposition. We can define a composition structure for the propositional truncation in the same way as [44, Section 3.3.4].

7.1.6 Discrete types

We introduce a class of types for which composition structures are trivial.

7.1.7. Definition. We say a type $\mathtt{A} : U$ in qCTT is *discrete* if $\forall_{\mathtt{p}:\mathbb{I}\text{->}\mathtt{A}} \forall_{\mathtt{i}:\mathbb{I}} \mathtt{p}(\mathtt{i}) == \mathtt{p}(0_{\mathbb{I}})$ is inhabited. This is equivalent to that the function $\lambda \mathtt{xi.x} : \mathtt{A}$ -> $(\mathbb{I}$ -> $\mathtt{A})$ is an isomorphism with inverse $\lambda \mathtt{p.p}(0_{\mathbb{I}})$.

7.1.8. Example. The connectivity of $\mathbb{I}$ (Eq. (7.1)) is equivalent to that the type 2 is discrete. Indeed, both the connectivity of $\mathbb{I}$ and the discreteness of 2 assert that any function $\mathbb{I}$ -> 2 is constant.

7.1.9. Proposition. *Any closed discrete type $A : \mathcal{M}(\diamond) \to \mathcal{M}(U)$ carries a composition structure.*

Proof:
Given `P` : Cof, `a` : `P -> (i` : $\mathbb{I}$`) -> `A and $\mathtt{a}_0 : A$ such that `P -> a`$_0$ `==` $\mathtt{a}(0_{\mathbb{I}})$, we define $\mathtt{a}_1 \equiv \mathtt{a}_0$. By the discreteness, `a` is constantly $\mathtt{a}_0$, and thus we have `P -> a`$_1$ `==` $\mathtt{a}(1_{\mathbb{I}})$. □

7.1.10. Proposition. *Let $A : \mathcal{M}(\diamond) \to \mathcal{M}(U)$ be a discrete closed type and $B : \mathcal{M}(\diamond)/\{A\} \to \mathcal{M}(U)$ an arbitrary type. Then B carries a composition structure if and only if it carries a homogeneous composition structure.*

Proof:
A composition structure on B is given by a map of the form

$$\begin{array}{ccc} (\mathcal{M}(\diamond)/\{A\})^{\mathcal{M}(\mathbb{I})} & \dashrightarrow & \mathcal{M}(E) \\ {\scriptstyle B^{\mathcal{M}(\mathbb{I})}}\downarrow & & \downarrow \\ \mathcal{M}(U)^{\mathcal{M}(\mathbb{I})} & \xrightarrow[\mathcal{M}(\mathsf{Comp})]{} & \mathcal{M}(U). \end{array}$$

Since A is discrete, the diagonal map $\Delta : \mathcal{M}(\diamond)/\{A\} \to (\mathcal{M}(\diamond)/\{A\})^{\mathcal{M}(\mathbb{I})}$ is an isomorphism. Therefore, a composition structure on B is equivalent to a map of the form

$$\begin{array}{ccccc} \mathcal{M}(\diamond)/\{A\} & & \dashrightarrow & & \mathcal{M}(E) \\ {\scriptstyle B}\downarrow & & & & \downarrow \\ \mathcal{M}(U) & \xrightarrow[\Delta]{} & \mathcal{M}(U)^{\mathcal{M}(\mathbb{I})} & \xrightarrow[\mathcal{M}(\mathsf{Comp})]{} & \mathcal{M}(U) \end{array}$$

which is nothing but a homogeneous composition structure on B. □

7.1.11. Proposition. *Discrete types are closed under Π-types, Σ-types, unit type, path types, finite coproducts, and natural numbers type. Moreover, for any discrete type* `A`*, the path type* $\mathsf{Path}(\langle \mathtt{i} \rangle \mathtt{A}, \mathtt{a}_0, \mathtt{a}_1)$ *coincides with the extensional identity type* $\mathtt{a}_0$ `==` $\mathtt{a}_1$*.*

Proof:
Straightforward. □

7.2 Internal cubical models

Orton and Pitts [134, 135] showed that the category of cubical sets satisfies their axioms for modeling CTT. Since their proof works in a constructive metalogic, we can internalize the construction of cubical models in an arbitrary model of

extensional type theory. In this section, we review the construction of cubical models. Precisely, we construct a model of qCTT consisting of *cubical objects* in a model of the good type theory $\mathcal{M}$. The outline is as follows.

1. Choose a category $\mathcal{C}$ in $\mathcal{M}$ called a *cube category* (Section 7.2.5) and build a model $\mathbf{PSh}_{\mathcal{C}}^{\mathcal{M}}$ of the good type theory consisting of *cubical objects* in $\mathcal{M}$, that is, presheaves in $\mathcal{M}$ over $\mathcal{C}$.

2. $\mathcal{C}$ must contain a special object $\mathbb{I}$ called an *interval* (Section 7.2.3). The interval of qCTT is interpreted as the representable presheaf $\mathrm{Y}_{\mathcal{C}}\,\mathbb{I}$.

3. Cofibrations of qCTT are interpreted as *locally decidable propositions* (Section 7.2.4).

4. Any universe in $\mathcal{M}$ can be lifted to a universe in $\mathbf{PSh}_{\mathcal{C}}^{\mathcal{M}}$ (Section 7.2.2).

5. If $\mathcal{M}$ models W-types, then $\mathbf{PSh}_{\mathcal{C}}^{\mathcal{M}}$ models W-types with reductions with respect to locally decidable propositions (Section 7.2.6).

In Section 7.2.7, we introduce constant and codiscrete presheaves which automatically carry composition structures in the cubical model.

7.2.1 Internal presheaves

We begin by defining categories and presheaves in $\mathcal{M}$. Let $\mathcal{C}_0 : \mathcal{M}(\diamond) \to \mathcal{M}(U)$ and $\mathcal{C}_1 : \mathcal{M}(\diamond)/\{\mathcal{C}_0\} \times \{\mathcal{C}_0\} \to \mathcal{M}(U)$ be types, and let $\mathrm{dom}, \mathrm{cod} : \{\mathcal{C}_1\} \to \{\mathcal{C}_0\}$ denote the first and second, respectively, projections. Although $\mathcal{M}(\diamond)$ need not have general pullbacks, it has pullbacks along the projections dom and cod. Therefore, a category structure on the graph $(\mathrm{dom}, \mathrm{cod}) : \{\mathcal{C}_1\} \to \{\mathcal{C}_0\} \times \{\mathcal{C}_0\}$ makes sense. By a *category in* $\mathcal{M}$, we mean a category in $\mathcal{M}(\diamond)$ whose underlying graph is of this form. We use similar notations to ordinary categories in the internal language of $\mathcal{M}$: we refer to the type $\mathcal{C}_0$ as $\mathcal{C}$; we may write $\mathrm{Hom}(\mathtt{x}, \mathtt{y})$ for $\mathcal{C}_1(\mathtt{x}, \mathtt{y})$. A *presheaf over* $\mathcal{C}$ also makes sense and it is defined to be an object $A \in \mathcal{M}(\diamond)/\{\mathcal{C}_0\}$ equipped with a *right* $\mathcal{C}$*-action* $\mathtt{act} : \mathrm{cod}^* A \to \mathrm{dom}^* A$ which is denoted by $\cdot$ in the internal language of $\mathcal{M}$.

7.2.1. Definition. Let $\mathcal{C}$ be a category in a model $\mathcal{M}$ of the good type theory. A *presheaf over* $\mathcal{C}$ *valued in* $\mathcal{M}(U)$ is a presheaf over $\mathcal{C}$ whose underlying object over $\{\mathcal{C}_0\}$ is of the form $\{A\}$ for a section $A : \mathcal{M}(\diamond)/\{\mathcal{C}_0\} \to \mathcal{M}(U)$.

For a presheaf Γ (valued in $\mathcal{M}(U)$), let $\int_{\mathcal{C}} \Gamma$ denote its category of elements, that is, the internal category defined by the span $\Gamma \leftarrow \mathrm{cod}^*\Gamma \xrightarrow{\mathtt{act}} \Gamma$. Internalizing the standard construction of a presheaf model of type theory [e.g. 79], we have a model $\mathbf{PSh}_{\mathcal{C}}^{\mathcal{M}}$ of DTT consisting of presheaves over $\mathcal{C}$. Concretely, we define a category $\mathbf{PSh}_{\mathcal{C}}^{\mathcal{M}}(\diamond)$ and a representable map $\mathbf{PSh}_{\mathcal{C}}^{\mathcal{M}}(E) \to \mathbf{PSh}_{\mathcal{C}}^{\mathcal{M}}(U)$ of discrete fibrations over $\mathbf{PSh}_{\mathcal{C}}^{\mathcal{M}}(\diamond)$ as follows:

- the base category $\mathbf{PSh}_{\mathcal{C}}^{\mathcal{M}}(\diamond)$ is the category of presheaves over $\mathcal{C}$;
- the sections $\mathbf{PSh}_{\mathcal{C}}^{\mathcal{M}}(\diamond)/\Gamma \to \mathbf{PSh}_{\mathcal{C}}^{\mathcal{M}}(U)$ are the presheaves over $\int_{\mathcal{C}} \Gamma$ valued in $\mathcal{M}(U)$;
- the sections of $\mathbf{PSh}_{\mathcal{C}}^{\mathcal{M}}(E)$ over $A : \mathbf{PSh}_{\mathcal{C}}^{\mathcal{M}}(\diamond)/\Gamma \to \mathbf{PSh}_{\mathcal{C}}^{\mathcal{M}}(U)$ are the global sections of the presheaf $\{A\}$ over $\int_{\mathcal{C}} \Gamma$;
- the context comprehension of $A : \mathbf{PSh}_{\mathcal{C}}^{\mathcal{M}}(\diamond)/\Gamma \to \mathbf{PSh}_{\mathcal{C}}^{\mathcal{M}}(U)$ is given by the presheaf $\{A\}$.

We note that types and terms over $\Gamma \in \mathbf{PSh}_{\mathcal{C}}^{\mathcal{M}}(\diamond)$ are closed types and terms in $\mathbf{PSh}_{\int_{\mathcal{C}} \Gamma}^{\mathcal{M}}$. Therefore, when constructing something from types and terms in an arbitrary presheaf model, we may assume that the input types and terms are closed.

7.2.2. Proposition. *For any model $\mathcal{M}$ of the good type theory and any category $\mathcal{C}$ in $\mathcal{M}$, the representable map $\mathbf{PSh}_{\mathcal{C}}^{\mathcal{M}}(E) \to \mathbf{PSh}_{\mathcal{C}}^{\mathcal{M}}(U)$ is part of a model $\mathbf{PSh}_{\mathcal{C}}^{\mathcal{M}}$ of the good type theory.*

Proof:
All the type constructors except Π-types are computed pointwise. For a presheaf A over $\mathcal{C}$ and a presheaf B over $\int_{\mathcal{C}} A$, the presheaf $\Pi(A, B)$ is defined by

$$\Pi(A,B)(\mathtt{x}) = \{\mathtt{b} : \prod_{\mathtt{x}':\mathcal{C}} \prod_{\mathtt{f}:\mathrm{Hom}(\mathtt{x}',\mathtt{x})} \prod_{\mathtt{a}:A(\mathtt{x}')} B(\mathtt{x}',\mathtt{a}) \mid \forall_{\mathtt{x}',\mathtt{x}'':\mathcal{C}} \forall_{\mathtt{f}:\mathrm{Hom}(\mathtt{x}',\mathtt{x})} \forall_{\mathtt{g}:\mathrm{Hom}(\mathtt{x}'',\mathtt{x}')} \forall_{\mathtt{a}:A(\mathtt{x}')} \mathtt{b}(\mathtt{x}',\mathtt{f},\mathtt{a}) \cdot \mathtt{g} == \mathtt{b}(\mathtt{x}'',\mathtt{f}\circ\mathtt{g},\mathtt{a}\cdot\mathtt{g})\}$$

for $\mathtt{x} : \mathcal{C}$ with right action

$$\mathtt{b} \cdot \mathtt{g} = \lambda \mathtt{x}' \mathtt{f} \mathtt{a}.\mathtt{b}(\mathtt{x}', \mathtt{g} \circ \mathtt{f}, \mathtt{a}).$$

□

7.2.2 Lifting universes

It is known that any Grothendieck universe can be lifted to a universe in an arbitrary presheaf category [80]. This construction is performed inside a model of a sufficiently rich type theory. Let $\mathcal{M}$ be a model of the good type theory. Suppose that $\mathcal{M}$ also models a universe $(\mathsf{u}, \mathsf{el}_{\mathsf{u}})$. Then, for a category $\mathcal{C}$ in $\mathcal{M}$, the model $\mathbf{PSh}_{\mathcal{C}}^{\mathcal{M}}$ also models the universe as follows. Using Π-types, Σ-types, and extensional identity types, we can define a type $\underline{\mathbf{Fun}}(\mathcal{C}^{\mathrm{op}}, \mathcal{M}(\mathsf{u}))$ of presheaves over $\mathcal{C}$ valued in $\mathcal{M}(\mathsf{u})$. We then define $\mathcal{M}(\mathsf{u})_{\mathcal{C}}$ to be the presheaf over $\mathcal{C}$ defined by

$$\mathcal{M}(\mathsf{u})_{\mathcal{C}}(\mathtt{x}) = \underline{\mathbf{Fun}}((\mathcal{C}/\mathtt{x})^{\mathrm{op}}, \mathcal{M}(\mathsf{u}))$$

for $\mathtt{x} : \mathcal{C}$ with right action defined by precomposition. We define a presheaf $\mathcal{M}(\mathsf{el}_{\mathsf{u}})_{\mathcal{C}}$ over $\int_{\mathcal{C}} \mathcal{M}(\mathsf{u})_{\mathcal{C}}$ by

$$\mathcal{M}(\mathsf{el}_{\mathsf{u}})_{\mathcal{C}}(\mathtt{x}, A) = A(\mathtt{id}(\mathtt{x}))$$

for $(\mathtt{x}, A) : \int_{\mathcal{C}} \mathcal{M}(\mathsf{u})_{\mathcal{C}}$. By definition, for any presheaf $\Gamma \in \mathbf{PSh}_{\mathcal{C}}^{\mathcal{M}}(\diamond)$, the sections $\mathbf{PSh}_{\mathcal{C}}^{\mathcal{M}}(\diamond)/\Gamma \to \mathcal{M}(\mathsf{u})_{\mathcal{C}}$ bijectively correspond to the presheaves over $\int_{\mathcal{C}} \Gamma$ valued in $\mathcal{M}(\mathsf{u})$. Unless otherwise mentioned, we understand that $\mathbf{PSh}_{\mathcal{C}}^{\mathcal{M}}$ interprets the universe u as $\mathcal{M}(\mathsf{u})_{\mathcal{C}}$.

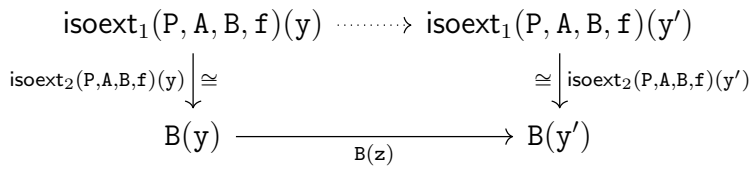
7.2.3. Proposition. *If $\mathcal{M}(\mathsf{u})$ is (weakly) closed under Σ-types (extensional identity types, unit types, disjoint finite coproducts, natural numbers type, propositional truncation), so is $\mathbf{PSh}_{\mathcal{C}}^{\mathcal{M}}(\mathsf{u})$. If $\mathcal{M}(\mathsf{u})$ is (weakly) closed under Π-types, Π-types indexed over $\mathcal{C}_0$ and $\mathcal{C}_1$, Σ-types, and extensional identity types, then $\mathbf{PSh}_{\mathcal{C}}^{\mathcal{M}}(\mathsf{u})$ is (weakly) closed under Π-types.*

Proof:
By construction. □

7.2.4. Proposition. *If $\mathcal{M}(\mathsf{u})$ is a propositional universe, then so is $\mathbf{PSh}_{\mathcal{C}}^{\mathcal{M}}(\mathsf{u})$. If, in addition, $\mathcal{M}(\mathsf{u})$ is weakly closed under finite disjunctions (satisfies the propositional extensionality axiom, has an isomorphism extension structure), so is $\mathbf{PSh}_{\mathcal{C}}^{\mathcal{M}}(\mathsf{u})$. If $\mathcal{M}(\mathsf{u})$ has an isomorphism extension structure valued in another universe $\mathcal{M}(\mathsf{u}')$, then $\mathbf{PSh}_{\mathcal{C}}^{\mathcal{M}}(\mathsf{u})$ has an isomorphism extension structure valued in $\mathbf{PSh}_{\mathcal{C}}^{\mathcal{M}}(\mathsf{u}')$.*

Proof:
By construction, it is immediate that $\mathbf{PSh}_{\mathcal{C}}^{\mathcal{M}}(\mathsf{u})$ is a propositional universe. The closure property and propositional extensionality are also obvious. The isomorphism extension structure is to be a natural transformation that takes an object $\mathtt{x} : \mathcal{C}$, presheaves $\mathtt{P} : (\mathcal{C}/\mathtt{x})^{\mathrm{op}} \to \mathcal{M}(\mathsf{u})$, $\mathtt{A} : \left(\int_{\mathcal{C}/\mathtt{x}} \mathtt{P}\right)^{\mathrm{op}} \to \mathcal{M}(U)$, and $\mathtt{B} : (\mathcal{C}/\mathtt{x})^{\mathrm{op}} \to \mathcal{M}(U)$, and a natural transformation $\mathtt{f} : \int_{\mathtt{y}:\mathcal{C}/\mathtt{x}} \mathtt{P}(\mathtt{y}) \times \mathtt{A}(\mathtt{y}) \to \mathtt{B}(\mathtt{y})$ such that $\mathtt{f}(\mathtt{y}) : \mathtt{A}(\mathtt{y}) \to \mathtt{B}(\mathtt{y})$ is an isomorphism for any $\mathtt{y} : \mathcal{C}/\mathtt{x}$ satisfying $\mathtt{P}$, and returns a presheaf $\mathsf{isoext}_1(\mathtt{P}, \mathtt{A}, \mathtt{B}, \mathtt{f}) : (\mathcal{C}/\mathtt{x})^{\mathrm{op}} \to \mathcal{M}(U)$ and an invertible natural transformation $\mathsf{isoext}_2(\mathtt{P}, \mathtt{A}, \mathtt{B}, \mathtt{f}) : \int_{\mathtt{y}:\mathcal{C}/\mathtt{x}} \mathsf{isoext}_1(\mathtt{P}, \mathtt{A}, \mathtt{B}, \mathtt{f})(\mathtt{y}) \to \mathtt{B}(\mathtt{y})$ such that $\mathsf{isoext}_1(\mathtt{P}, \mathtt{A}, \mathtt{B}, \mathtt{f})(\mathtt{y}) \equiv \mathtt{A}(\mathtt{y})$ and $\mathsf{isoext}_2(\mathtt{P}, \mathtt{A}, \mathtt{B}, \mathtt{f})(\mathtt{y}) \equiv \mathtt{f}(\mathtt{y})$ for any $\mathtt{y} : \mathcal{C}/\mathtt{x}$ satisfying $\mathtt{P}$. By the isomorphism extension structure on $\mathcal{M}(\mathsf{u})$, we have a type $\mathsf{isoext}_1(\mathtt{P}, \mathtt{A}, \mathtt{B}, \mathtt{f})(\mathtt{y}) : \mathcal{M}(U)$ and an isomorphism $\mathsf{isoext}_2(\mathtt{P}, \mathtt{A}, \mathtt{B}, \mathtt{f})(\mathtt{y}) : \mathsf{isoext}_1(\mathtt{P}, \mathtt{A}, \mathtt{B}, \mathtt{f})(\mathtt{y}) \cong \mathtt{B}(\mathtt{y})$ for any object $\mathtt{y} : \mathcal{C}/\mathtt{x}$, and these satisfy the required equations. The right action of an arrow $\mathtt{z} : \mathtt{y}' \to \mathtt{y}$ in $\mathcal{C}/\mathtt{x}$ on $\mathsf{isoext}_1(\mathtt{P}, \mathtt{A}, \mathtt{B}, \mathtt{f})$ is

defined to be the unique map that makes the following diagram commute.



Then $\mathsf{isoext}_2(\mathtt{P}, \mathtt{A}, \mathtt{B}, \mathtt{f})$ becomes a natural isomorphism by construction. The isomorphism extension structure valued in another universe is constructed in the same way. □

7.2.3 Intervals

7.2.5. Definition. Let $\mathcal{C}$ be a category with a terminal object in $\mathcal{M}$. An *interval in* $\mathcal{C}$ is an object $\mathbb{I} : \mathcal{C}$ such that the product with $\mathbb{I}$ exists and $\mathbb{I}$ is equipped with a De Morgan algebra structure.

Given an interval $\mathbb{I}$ in $\mathcal{C}$, we interpret the interval of qCTT in presheaves over $\mathcal{C}$ as $\mathbf{PSh}^{\mathcal{M}}_{\mathcal{C}}(\mathbb{I}) := \mathrm{Y}_{\mathcal{C}}\,\mathbb{I}$.

7.2.6. Proposition. *If $\mathcal{M}$ models a propositional universe* Cof *and if the equality predicate on* $\mathrm{Hom}(\mathtt{x}, \mathbb{I})$ *belongs to* $\mathcal{M}(\mathsf{Cof})$*, then the equality predicate on the representable presheaf* $\mathrm{Y}_{\mathcal{C}}\,\mathbb{I}$ *belongs to* $\mathbf{PSh}^{\mathcal{M}}_{\mathcal{C}}(\mathsf{Cof})$*.*

Proof:
By construction. □

7.2.7. Lemma. *For a presheaf A over $\mathcal{C}$, the exponential $A^{\mathrm{Y}_{\mathcal{C}}\,\mathbb{I}}$ is given by*

$$A^{\mathrm{Y}_{\mathcal{C}}\,\mathbb{I}}(\mathtt{x}) \cong A(\mathtt{x} \times \mathbb{I}).$$

Proof:
This is an internal version of Proposition 3.1.17. □

7.2.8. Proposition. *The interval in* $\mathbf{PSh}^{\mathcal{M}}_{\mathcal{C}}$ *is connected.*

Proof:
As noted in Example 7.1.8, the interval is connected if and only if $\mathsf{2}$ is discrete. The type $\mathsf{2}$ is interpreted in $\mathbf{PSh}^{\mathcal{M}}_{\mathcal{C}}$ as the constant presheaf at the two-element type in $\mathcal{M}$. Then, by Lemma 7.2.7, we have $\mathsf{2}^{\mathbb{I}} \cong \mathsf{2}$ in $\mathbf{PSh}^{\mathcal{M}}_{\mathcal{C}}$. □

7.2.9. Proposition. *If $\mathcal{M}$ models a universe* $\mathtt{u}$*, then* $\mathbf{PSh}_{\mathcal{C}}^{\mathcal{M}}(\mathtt{u})$ *is weakly closed under* Π*-types over* $\mathrm{Y}_{\mathcal{C}}\,\mathbb{I}$*.*

Proof:
By Lemma 7.2.7, $\mathbf{PSh}_{\mathcal{C}}^{\mathcal{M}}(\mathtt{u})^{\mathrm{Y}_{\mathcal{C}}\,\mathbb{I}}(\mathtt{x})$ is isomorphic to the type of presheaves over $\mathcal{C}/\mathtt{x} \times \mathbb{I}$ valued in $\mathtt{u}$. Then we obtain $\Pi_{\mathbb{I}} : \mathbf{PSh}_{\mathcal{C}}^{\mathcal{M}}(\mathtt{u})^{\mathrm{Y}_{\mathcal{C}}\,\mathbb{I}} \to \mathbf{PSh}_{\mathcal{C}}^{\mathcal{M}}(\mathtt{u})$ by sending a presheaf A over $\mathcal{C}/\mathtt{x} \times \mathbb{I}$ to the presheaf $\Pi_{\mathbb{I}}(A)$ over $\mathcal{C}/\mathtt{x}$ defined by

$$\Pi_{\mathbb{I}}(A)(\mathtt{y}) = A(\mathtt{y} \times \mathbb{I})$$

for $\mathtt{y} : \mathcal{C}/\mathtt{x}$. □

7.2.10. Proposition. *The functor* $\surd$ *preserves* $\mathbf{PSh}_{\mathcal{C}}^{\mathcal{M}}(U)$*-small objects.*

Proof:
Let $A : \mathbf{PSh}_{\mathcal{C}}^{\mathcal{M}}(\diamond) \to \mathbf{PSh}_{\mathcal{C}}^{\mathcal{M}}(U)$ be a global section, that is, a presheaf over $\mathcal{C}$ valued in $\mathcal{M}(U)$. By Lemma 7.2.7, the presheaf $\surd(A)$ over $\mathcal{C}$ is given by

$$\surd(A)(\mathtt{x}) = \int_{\mathtt{y}:\mathcal{C}} \mathrm{Hom}(\mathtt{y} \times \mathbb{I}, \mathtt{x}) \texttt{ -> } A(\mathtt{y}),$$

where the end is defined using Π-types, Σ-types, and extensional identity types, and thus $\surd(A)$ is valued in $\mathcal{M}(U)$. □

In the same way, we can show the following.

7.2.11. Corollary. *If $\mathcal{M}$ models a universe* $\mathtt{u}$ *weakly closed under* Π*-types over* $\mathcal{C}_0$ *and* $\mathcal{C}_1$*,* Σ*-types, and extensional identity types, then the functor* $\surd$ *preserves* $\mathbf{PSh}_{\mathcal{C}}^{\mathcal{M}}(\mathtt{u})$*-small objects.*

7.2.4 Locally decidable propositions

In the good type theory, the type $\mathsf{2}$ is called the *decidable subobject classifier*.

7.2.12. Definition. We say a proposition $\mathtt{P}$ is *decidable* if $\mathtt{P} + \neg\mathtt{P}$ is inhabited.

The type $\mathsf{2}$ classifies the decidable propositions in the following sense. We define a proposition

$$\mathsf{true} : (\mathtt{x} : \mathsf{2}) \Rightarrow U$$

by $\mathsf{true}(\mathtt{x}) \equiv (\mathtt{x} \texttt{ == } 0)$. For any decidable proposition $\mathtt{P} : (\mathtt{x} : \mathtt{A}) \to U$, we have a function $\chi_{\mathtt{P}} : \mathtt{A} \texttt{ -> } \mathsf{2}$ defined by $\chi_{\mathtt{P}}(\mathtt{x}) \equiv 0$ if $\mathtt{P}(\mathtt{x})$ and $\chi_{\mathtt{P}}(\mathtt{x}) \equiv 1$ if $\neg\mathtt{P}(\mathtt{x})$. This function $\chi_{\mathtt{P}}$ is uniquely characterized by the property that

$$(\mathtt{x} : \mathtt{A}) \texttt{ -> } (\mathtt{P}(\mathtt{x}) \leftrightarrow \mathsf{true}(\chi_{\mathtt{P}}(\mathtt{x}))).$$

We regard $\mathsf{2}$ as a propositional universe with the proposition true.

7.2.13. PROPOSITION. $\mathsf{2}$ *is weakly closed under* Σ*-types, unit type, and finite disjunctions, satisfies the propositional extensionality axiom, and has an isomorphism extension structure valued in* U *and an isomorphism extension structure valued in an arbitrary universe.*

Proof:
Since decidable propositions are closed under Σ-types, unit type, and finite disjunctions, $\mathsf{2}$ is weakly closed under these type constructors. The propositional extensionality is obvious. For an isomorphism extension structure, let $\mathtt{P} : () \to \mathsf{2}$ be a decidable proposition, $\mathtt{A} : (_ : \mathtt{P}) \to U$ and $\mathtt{B} : () \to U$ types and $\mathtt{f} : (_ : \mathtt{P}) \to \mathsf{Iso}(\mathtt{A}, \mathtt{B})$ an isomorphism. By the strict extensivity of the coproduct $\mathsf{2} = \mathsf{1} + \mathsf{1}$, we can proceed by case analysis on $\mathtt{P}$. When $\mathtt{P}$ is 0, we have $\mathsf{true}(\mathtt{P}) \cong \top$ and then define $\mathsf{isoext}_1(0, \mathtt{A}, \mathtt{B}, \mathtt{f}) \equiv \mathtt{A}$ and $\mathsf{isoext}_2(0, \mathtt{A}, \mathtt{B}, \mathtt{f}) \equiv \mathtt{f}$. When $\mathtt{P}$ is 1, we have $\mathsf{true}(\mathtt{P}) \cong \bot$ and then define $\mathsf{isoext}_1(1, \mathtt{A}, \mathtt{B}, \mathtt{f})$ and $\mathsf{isoext}_2(1, \mathtt{A}, \mathtt{B}, \mathtt{f})$ by the elimination rule for the empty type. An isomorphism extension structure valued in a universe is defined in the same way. □

Let $\mathcal{C}$ be a category in $\mathcal{M}$ with an interval $\mathbb{I}$. By Proposition 7.2.13, the universe $\mathbf{PSh}^{\mathcal{M}}_{\mathcal{C}}(\mathsf{2})$ satisfies most of the requirements for Cof in qCTT. However, it is never the case that the equality predicate on $\mathrm{Y}_{\mathcal{C}}\,\mathbb{I}$ is decidable while $0_{\mathbb{I}}$ and $1_{\mathbb{I}}$ are distinct. If the equality predicate on $\mathrm{Y}_{\mathcal{C}}\,\mathbb{I}$ is decidable, then the equality predicate on $\mathrm{Hom}(\mathtt{x}, \mathbb{I})$ is decidable for any $\mathtt{x} : \mathcal{C}$ and the decision is preserved by precomposition in the sense that if $\mathtt{f}_1, \mathtt{f}_2 : \mathtt{x} \to \mathbb{I}$ are distinct, then so are $\mathtt{f}_1 \circ \mathtt{g}, \mathtt{f}_2 \circ \mathtt{g} : \mathtt{x}' \to \mathbb{I}$ for any $\mathtt{g} : \mathtt{x}' \to \mathtt{x}$. In particular, the two projections $\mathbb{I} \times \mathbb{I} \to \mathbb{I}$ are equal as they are equalized by the diagonal arrow $\mathbb{I} \to \mathbb{I} \times \mathbb{I}$. It then follows that $0_{\mathbb{I}}, 1_{\mathbb{I}} : 1 \to \mathbb{I}$ are equal. We thus seek another universe for the interpretation of Cof.

Applying the construction of a universe described in Section 7.2.2 to $\mathcal{M}(\mathsf{2})$, we obtain the universe $\mathcal{M}(\mathsf{2})_{\mathcal{C}}$ in $\mathbf{PSh}^{\mathcal{M}}_{\mathcal{C}}$. By construction, $\mathcal{M}(\mathsf{2})_{\mathcal{C}}$ classifies *locally decidable propositions*, that is, presheaves P such that $P(\mathtt{x})$ is a decidable proposition for any object $\mathtt{x}$. Local decidability is a much milder condition than decidability: the equality predicate on $\mathrm{Y}_{\mathcal{C}}\,\mathbb{I}$ is locally decidable if and only if the equality predicate on $\mathrm{Hom}(\mathtt{x}, \mathbb{I})$ is decidable for any object $\mathtt{x} : \mathcal{C}$. The universe $\mathcal{M}(\mathsf{2})_{\mathcal{C}}$ also inherits good properties from $\mathcal{M}(\mathsf{2})$.

7.2.14. PROPOSITION. *Let* $\mathcal{M}$ *be a model of the good type theory and* $\mathcal{C}$ *a category in* $\mathcal{M}$*. Then the universe* $\mathcal{M}(\mathsf{2})_{\mathcal{C}}$ *in* $\mathbf{PSh}^{\mathcal{M}}_{\mathcal{C}}$ *is weakly closed under* Σ*-types, unit type, and finite disjunctions, satisfies the propositional extensionality axiom, and has an isomorphism extension structure valued in* U *and an isomorphism extension structure valued in a universe obtained by the method of Section 7.2.2.*

Proof:
By Propositions 7.2.4 and 7.2.13. □

7.2.5 Cube categories

We work in the good type theory. Let $\mathsf{Fin} : (\mathsf{x} : ()) \to \mathsf{N}) \Rightarrow U$ be the type defined by $\mathsf{Fin}(\mathsf{x}) = \{\mathsf{y} : \mathsf{N} \mid \mathsf{y} < \mathsf{x}\}$. We define a category $\mathbf{B}$ as follows. The objects are the natural numbers. The morphisms $\mathsf{x} \to \mathsf{y}$ are the functions (Fin(x) -> 2) -> (Fin(y) -> 2). Finite products in $\mathbf{B}$ are given by the addition. The hom set $\mathrm{Hom}(\mathsf{x}, \mathsf{y})$ is isomorphic to $\mathsf{Fin}\left(2^{\mathsf{x}\times\mathsf{y}}\right)$ and thus the equality predicate on $\mathrm{Hom}(\mathsf{x}, \mathsf{y})$ is decidable. We choose an interval $\mathbb{I}$ to be $1 : \mathsf{N}$. From the standard Boolean algebra structure on 2, the interval $\mathbb{I}$ is a Boolean algebra in $\mathbf{B}$. In particular, $\mathbb{I}$ is a De Morgan algebra in $\mathbf{B}$. We call a presheaf over $\mathbf{B}$ a *cubical object*.

7.2.15. Theorem (Orton and Pitts [134, 135]). *For any model $\mathcal{M}$ of the good type theory, the representable map $\mathbf{PSh}_{\mathbf{B}}^{\mathcal{M}}(E) \to \mathbf{PSh}_{\mathbf{B}}^{\mathcal{M}}(U)$, the universe $\mathcal{M}(2)_{\mathbf{B}}$, and the interval $\mathrm{Y}_{\mathbf{B}}\,\mathbb{I}$ are part of a model $\mathbf{PSh}_{\mathbf{B}}^{\mathcal{M}}$ of qCTT. Hence, we have the model $\left(\mathbf{PSh}_{\mathbf{B}}^{\mathcal{M}}\right)_{\mathrm{fib}}$ of CTT by Theorem 7.1.1.*

Proof:
We have already seen that $\mathbf{PSh}_{\mathbf{B}}^{\mathcal{M}}$ is a model of the good type theory in Proposition 7.2.2. By Proposition 7.2.14, $\mathcal{M}(2)_{\mathbf{B}}$ is a propositional universe with sufficient closure properties, propositional extensionality, and an isomorphism extension structure valued in U. By Proposition 7.2.9, $\mathcal{M}(2)_{\mathbf{B}}$ is closed under Π-types over $\mathrm{Y}_{\mathbf{B}}\,\mathbb{I}$. The equality predicate on $\mathrm{Y}_{\mathbf{B}}\,\mathbb{I}$ belongs to $\mathcal{M}(2)_{\mathbf{B}}$ by Proposition 7.2.6 and the decidability of morphisms in $\mathbf{B}$. The interval $\mathrm{Y}_{\mathbf{B}}\,\mathbb{I}$ is connected by Proposition 7.2.8. For the endpoints of $\mathrm{Y}_{\mathbf{B}}\,\mathbb{I}$ to be distinct, it suffices to show that the endpoints $0_{\mathbb{I}}, 1_{\mathbb{I}} : 0 \to \mathbb{I}$ in $\mathbf{B}$ ($0 : \mathsf{N}$ is the final object in $\mathbf{B}$) are distinct at any object $n : \mathbf{B}$, that is, the composites with the final projection $n \to 0$ are distinct, but this is obvious by the definition of $\mathbf{B}$. □

7.2.16. Proposition. *Let $\mathcal{M}$ be a model of the good type theory. Suppose that $\mathcal{M}$ models a universe u weakly closed under all the type constructors of the good type theory. Then $\mathcal{M}(\mathsf{u})_{\mathbf{B}}$ is a universe in $\mathbf{PSh}_{\mathbf{B}}^{\mathcal{M}}$ containing Cof and $\mathbb{I}$ and weakly closed under all the type constructors of qCTT. Hence, we have a universe in $\left(\mathbf{PSh}_{\mathbf{B}}^{\mathcal{M}}\right)_{\mathrm{fib}}$ by Theorem 7.1.3 and Proposition 7.2.10. Moreover, any hierarchy of universes $\mathsf{u} : \widehat{\mathsf{u}}$ is preserved by this construction by Corollary 7.2.11.*

Proof:
The closure properties of u include $2 : \mathsf{u}$ and $\mathsf{N} : \mathsf{u}$, and thus $\mathcal{M}(\mathsf{u})_{\mathbf{B}}$ contains Cof and $\mathbb{I}$. The other closure properties are proved in Proposition 7.2.3. □

The category $\mathbf{B}$ is not the only choice of the base category. What we need in the proof of Theorem 7.2.15 is:

1. the equality predicate on $\mathrm{Hom}(\mathsf{x}, \mathbb{I})$ is decidable for any object $\mathsf{x} : \mathcal{C}$;

2. the endpoints $0_{\mathbb{I}}, 1_{\mathbb{I}} : 1 \to \mathbb{I}$ in $\mathcal{C}$ are distinct at any object $\mathtt{x} : \mathcal{C}$.

We say $\mathcal{C}$ is a *cube category* if these conditions are satisfied. For example, assuming sufficient inductive types in $\mathcal{M}$, we may choose $\mathcal{C}$ to be the opposite of the category of free De Morgan algebras on finite types as in [41]. As Cohen et al. [41] remarked, the free De Morgan algebra over a finite type has decidable equality. This is because equalities between elements of a free De Morgan algebra are tested by homomorphisms to the four-element De Morgan algebra $\{0, \mathtt{x}, \mathtt{y}, 1\}$ with $1-\mathtt{x} = \mathtt{x}$, $1-\mathtt{y} = \mathtt{y}$, $\mathtt{x} \wedge \mathtt{y} = 0$, and $\mathtt{x} \vee \mathtt{y} = 1$; see [104] for an elementary proof. The interval $\mathbb{I}$ is the free De Morgan algebra on the singleton type and the endpoints $0_{\mathbb{I}}$ and $1_{\mathbb{I}}$ are the morphisms $\mathbb{I} \to \mathsf{2}$ of De Morgan algebras sending the generator to 0 and 1, respectively. Since 0 and 1 remain distinct in any free De Morgan algebras, we see that the endpoints are distinct at any object $\mathtt{x} : \mathcal{C}$.

7.2.6 W-types with reductions

This section is joint work with Andrew Swan [163]. We assume that $\mathcal{M}$ models W-types. Let $\mathcal{C}$ be a category in $\mathcal{M}$. We construct W-types with reductions in $\mathbf{PSh}_{\mathcal{C}}^{\mathcal{M}}$ with respect to the universe $\mathcal{M}(\mathsf{2})_{\mathcal{C}}$ of locally decidable propositions. The following category-theoretic result was proved by Swan.

7.2.17. Theorem (Swan [161]). *Let $\mathcal{X}$ be a locally cartesian closed category with finite colimits and disjoint coproducts and W-types, and let $\mathcal{C}$ be an internal category in $\mathcal{X}$. Then the category of internal presheaves over $\mathcal{C}$ has all locally decidable W-types with reductions.*

The W-types with reductions obtained by Theorem 7.2.17 are stable under reindexing only up to isomorphism, while in a model of a type theory everything must be strictly stable under reindexing. We thus apply a splitting technique to obtain a strictly stable one.

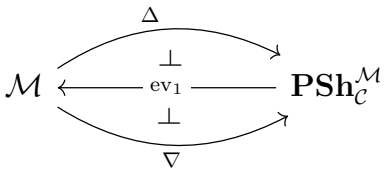
Let Γ be a context in $\mathbf{PSh}_{\mathcal{C}}^{\mathcal{M}}$, A a type over Γ, B a type over $\{A\}$, P a cofibration over $\{A\}$, and $f : P \to B$ a map over A. Unwinding the definition, Γ is a presheaf over $\mathcal{C}$, A is a presheaf over $\int_{\mathcal{C}} \Gamma$ valued in $\mathcal{M}(U)$, B is a presheaf over $\int_{\mathcal{C}} A$ valued in $\mathcal{M}(U)$, and P is a presheaf over $\int_{\mathcal{C}} A$ valued in $\mathcal{M}(\mathsf{2})$. The W-type with reductions $\mathsf{W}(A, B, P, f)$ must be a presheaf over $\int_{\mathcal{C}} \Gamma$. Let (x, c) be an object of $\int_{\mathcal{C}} \Gamma$. By Yoneda, the element c corresponds to a map $c : \mathcal{C}/x \to \int_{\mathcal{C}} \Gamma$ of discrete fibrations over $\mathcal{C}$. Pulling back along c, we have a polynomial with reductions (c^*A, c^*B, c^*P, c^*f) in the category of presheaves over $\mathcal{C}/x$. Let $\mathsf{W}'(c^*A, c^*B, c^*P, c^*f)$ be the W-type with reductions obtained by Theorem 7.2.17. We then define the value of the presheaf $\mathsf{W}(A, B, P, f)$ at (x, c) to be the fiber of $\mathsf{W}'(c^*A, c^*B, c^*P, c^*f)$ over $\mathrm{id}_x : \mathcal{C}/x$. By construction, $\mathsf{W}(A, B, P, f)$ is strictly stable under reindexing along a map $\Gamma' \to \Gamma$.

7.2.7 Constant and codiscrete presheaves

Let $\mathcal{C}$ be a cube category in a model $\mathcal{M}$ of the good type theory. We introduce two constructions of types in $\left(\mathbf{PSh}_{\mathcal{C}}^{\mathcal{M}}\right)_{\text{fib}}$ out of types in $\mathcal{M}$, *constant presheaves* and *codiscrete presheaves*.

For a closed type $A : \mathcal{M}(\diamond) \to \mathcal{M}(U)$, we define the *constant presheaf* $\Delta\, A$ by $(\Delta\, A)(\mathtt{x}) = A$ with the trivial right $\mathcal{C}$-action. For a closed type $A : \mathcal{M}(\diamond) \to \mathcal{M}(U)$, we define the *codiscrete presheaf* $\nabla\, A : \mathbf{PSh}_{\mathcal{C}}^{\mathcal{M}}(\diamond) \to \mathbf{PSh}_{\mathcal{C}}^{\mathcal{M}}(U)$ by $(\nabla\, A)(\mathtt{x}) = \mathrm{Hom}(1, \mathtt{x})$ `->` A for $\mathtt{x} : \mathcal{C}$ with composition as the right $\mathcal{C}$-action. For a type $B : \mathcal{M}(\diamond)/\{A\} \to \mathcal{M}(U)$, we define $\nabla_A\, B : \mathbf{PSh}_{\mathcal{C}}^{\mathcal{M}}(\diamond)/\{\Delta\, A\} \to \mathbf{PSh}_{\mathcal{C}}^{\mathcal{M}}(U)$ by $(\nabla_A\, B)(\mathtt{x}, \mathtt{a}) = (\nabla\, B(\mathtt{a}))(\mathtt{x})$ for $(\mathtt{x}, \mathtt{a}) : \int_{\mathcal{C}} \Delta\, A$.

Categorically, Δ and ∇ are understood as the left and right, respectively, adjoints of the evaluation $\mathrm{ev}_1 : \mathbf{PSh}_{\mathcal{C}}^{\mathcal{M}} \to \mathcal{M}$ at the final object $1 : \mathcal{C}$.



We also note that $\mathrm{ev}_1 \circ \Delta \cong \mathrm{id}$ and $\mathrm{ev}_1 \circ \nabla \cong \mathrm{id}$ by definition. Furthermore, Δ preserves the type constructors of the good type theory up to isomorphism.

7.2.18. Proposition. *For any closed type A, the constant presheaf $\Delta\, A$ is a discrete closed type of $\mathbf{PSh}_{\mathcal{C}}^{\mathcal{M}}$.*

Proof:
By Lemma 7.2.7,, we have $(\Delta\, A)^{\mathbb{I}}(\mathtt{x}) \cong (\Delta\, A)(\mathtt{x} \times \mathbb{I}) = A = (\Delta\, A)(\mathtt{x})$ for any $\mathtt{x} : \mathcal{C}$. □

7.2.19. Lemma. *For any closed type A, the codiscrete presheaf $\nabla\, A$ carries a composition structure, where we choose $\mathcal{M}(2)_{\mathcal{C}}$ as the universe of cofibrations, and is a proposition with respect to path types.*

Proof:
For a composition structure, it is enough to construct a homogeneous composition structure as $\nabla\, A$ is a closed type. Unwinding the definition, a homogeneous composition structure on $\nabla\, A$ is a natural transformation hcomp that takes an object $\mathtt{x} : \mathcal{C}$, a functor $\mathtt{P} : (\mathcal{C}/\mathtt{x})^{\mathrm{op}}$ `->` $\mathtt{2}$, a natural transformation $\mathtt{a} : \int_{\mathtt{y}:\mathcal{C}}((\mathtt{f} : \mathrm{Hom}(\mathtt{y}, \mathtt{x})) \times \mathtt{P}(\mathtt{f}) \times \mathrm{Hom}(\mathtt{y}, \mathbb{I}))$ `->` $(\nabla\, A)(\mathtt{y})$, and an element $\mathtt{a}_0 : (\nabla\, A)(\mathtt{x})$ such that $\forall_{\mathtt{y}:\mathcal{C}} \forall_{\mathtt{f}:\mathrm{Hom}(\mathtt{y},\mathtt{x})} \mathtt{P}(\mathtt{f})$ `->` $\mathtt{a}_0 \cdot \mathtt{f} == \mathtt{a}(\mathtt{f}, 0_{\mathbb{I}})$ and returns an element $\mathsf{hcomp}(\mathtt{P}, \mathtt{a}, \mathtt{a}_0) : (\nabla\, A)(\mathtt{x})$ such that $\forall_{\mathtt{y}:\mathcal{C}} \forall_{\mathtt{f}:\mathrm{Hom}(\mathtt{y},\mathtt{x})} \mathtt{P}(\mathtt{f})$ `->` $\mathsf{hcomp}(\mathtt{P}, \mathtt{a}, \mathtt{a}_0) \cdot \mathtt{f} == \mathtt{a}(\mathtt{f}, 1_{\mathbb{I}})$. We define $\mathsf{hcomp}(\mathtt{P}, \mathtt{a}, \mathtt{a}_0) : \mathrm{Hom}(1, \mathtt{x})$ `->` A by

$$\mathsf{hcomp}(\mathtt{P}, \mathtt{a}, \mathtt{a}_0)(\mathtt{f}) = \begin{cases} \mathtt{a}(\mathtt{f}, 1_{\mathbb{I}})(\mathrm{id}_1) & \text{if } \mathtt{P}(\mathtt{f}) == 0 \\ \mathtt{a}_0(\mathtt{f}) & \text{if } \mathtt{P}(\mathtt{f}) == 1 \end{cases}$$

for $\mathtt{f} : \mathrm{Hom}(1, \mathtt{x})$. Then it is natural in $\mathtt{x}$ and satisfies the required condition.

For ∇A to be a proposition, we construct a natural transformation $\mathtt{p}$ that takes an object $\mathtt{x} : \mathcal{C}$, elements $\mathtt{a}_0, \mathtt{a}_1 : (\nabla A)(\mathtt{x})$, and a morphism $\mathtt{i} : \mathrm{Hom}(\mathtt{x}, \mathbb{I})$ and returns an element $\mathtt{p}(\mathtt{a}_0, \mathtt{a}_1, \mathtt{i}) : (\nabla A)(\mathtt{x})$ such that $\mathtt{p}(\mathtt{a}_0, \mathtt{a}_1, 0_{\mathbb{I}}) == \mathtt{a}_0$ and $\mathtt{p}(\mathtt{a}_0, \mathtt{a}_1, 1_{\mathbb{I}}) == \mathtt{a}_1$. We define $\mathtt{p}(\mathtt{a}_0, \mathtt{a}_1, \mathtt{i})$ by

$$\mathtt{p}(\mathtt{a}_0, \mathtt{a}_1, \mathtt{i})(\mathtt{f}) = \begin{cases} \mathtt{a}_0(\mathtt{f}) & \text{if } \mathtt{i} \cdot \mathtt{f} == 0_{\mathbb{I}} \\ \mathtt{a}_1(\mathtt{f}) & \text{otherwise} \end{cases}$$

for $\mathtt{f} : \mathrm{Hom}(1, \mathtt{x})$. Then it is natural in $\mathtt{x}$ and satisfies the required conditions. □

7.2.20. Proposition. *For any closed type A and any type B over A, the codiscrete presheaf $\nabla_A B$ carries a composition structure, where we choose $\mathcal{M}(2)_{\mathcal{C}}$ as the universe of cofibrations, and is a proposition with respect to path types.*

Proof:
From the definition of $\nabla_A B$, this is confirmed point-wise. We thus just apply Lemma 7.2.19. □

Chapter 8

Cubical assembly models of type theory

In this chapter, we study models of univalent type theory and CTT in *cubical assemblies* obtained by applying the method described in Chapter 7 to the model of extensional type theory in assemblies.

An interesting feature of the assembly model is that it has a *proof-relevant impredicative universe*. We say a universe is impredicative if it is weakly closed under dependent products over arbitrary types. The subobject classifier of a topos is an impredicative universe in this sense but proof-irrelevant in the sense that any type classified by the universe has at most one element. It is known that any impredicative universe in a Grothendieck topos is proof-irrelevant because of the result of Freyd [58] that any small complete category is a preorder. In contrast, the impredicative universe in the assembly model is proof-relevant and admits an interpretation of polymorphic type theory such as System F [73] and the Calculus of Constructions [45]. In Section 8.2, we construct an impredicative universe in the cubical assembly model by applying the construction of universes given in Chapter 7. Since univalence is derivable over CTT, this universe is also univalent, and thus univalence is consistent with impredicativity.

When a type theory has an impredicative universe u, the inclusion from u-small types to all types has a reflection. In general, this reflection is not an equivalence, because if u itself belongs to u then one can derive a contradiction (Girard's paradox). However, if we restrict our attention to propositions, there is a possibility that the reflection of propositions to u-small propositions gives an equivalence. This is a formulation of the *propositional resizing axiom* [172, Section 3.5] in the presence of an impredicative universe. The impredicative universe in the assembly model of extensional type theory admits this form of propositional resizing. Van den Berg [174] constructed a proof-relevant, impredicative, and univalent universe satisfying propositional resizing, but this universe is 1-truncated in the sense that it classifies a class of 0-truncated types. It is unknown whether there is an untruncated, proof-relevant, impredicative, and univalent universe, and we show in Section 8.3 that the impredicative universe in the cubical assembly model

of CTT does not admit propositional resizing.

Another important aspect of the assembly model is that it is a model of *Constructive Recursive Mathematics*, a form of constructivism based on *algorithms* or *recursive functions*. In Constructive Recursive Mathematics, the following two principles are accepted.

Markov's Principle If it is not the case that an algorithm does not terminate, then the algorithm terminates.

Church's Thesis For any function on natural numbers, there exists an algorithm to compute the function.

The assembly model of extensional type theory satisfies these principles. We construct a model of univalent type theory to show the consistency of the univalence axiom and the main principles of Constructive Recursive Mathematics. We show in Section 8.4 that Markov's Principle holds in the cubical assembly model. On the other hand, Church's Thesis does not hold in the cubical assembly model and, even worse, the negation of Church's Thesis holds in any cubical model constructed by the method given in Chapter 7. Nevertheless, we can find a reflective subuniverse of cubical assemblies in which Church's Thesis and Markov's Principle hold. We prove these results on Church's Thesis in Section 8.5.

The contents of Sections 8.2 and 8.3 are based on the author's paper [170]. Sections 8.4 and 8.5 are joint work with Andrew Swan [163].

8.0.1. Remark. The results in Sections 8.4 and 8.5 also hold when we begin with any model of extensional type theory satisfying Markov's Principle and Church's Thesis instead of the assembly model. For example, we may take the *effective topos* [85] which includes the category of assemblies and is categorically more well-behaved than the category of assemblies. However, the impredicative universe in the assembly model fails to be impredicative in the effective topos if types are interpreted as arbitrary objects of the effective topos. Indeed, the internal category given by Hyland [84] is only weakly complete, and one has to interpret types as assemblies to model polymorphic type theory.

8.1 Assemblies

We review the assembly model of type theory. The standard references are [176, 114, 85, 137, 89].

Let $e \cdot n$ denote the partial operator on natural numbers that is defined if e is a code of a partial recursive function and if the value at n is defined and returns the output.

8.1.1. Definition. An *assembly* or *ω-set* is a set A equipped with a non-empty set $\mathrm{E}_A(a)$ of natural numbers for any $a \in A$. When $n \in \mathrm{E}_A(a)$, we say n is a

realizer for a or n *realizes* a. A *morphism* $f : A \to B$ *of assemblies* is a map $f : A \to B$ between the underlying sets such that there exists a natural number e such that, for any $a \in A$ and $n \in \mathrm{E}_A(a)$, the application $e \cdot a$ is defined and belongs to $\mathrm{E}_B(f(a))$. In that case, we say f is *tracked by* e or e is a *tracker of* f. We write **Asm** for the category of assemblies and their morphisms.

8.1.2. Remark. Assemblies can be more generally defined for an arbitrary partial combinatory algebra [176]. An assembly in the sense of Definition 8.1.1 is an assembly for Kleene's first model $\mathcal{K}_1$. All the results in Sections 8.2 and 8.3 holds for an arbitrary non-trivial partial combinatory algebra.

We make **Asm** part of a model of the good type theory. The base category $\mathbf{Asm}(\diamond)$ is the category of assemblies. The sections $\mathbf{Asm}(\diamond)/A \to \mathbf{Asm}(U)$ are the families of assemblies indexed over the underlying set of A. The sections of $\mathbf{Asm}(E)$ over a section $B : \mathbf{Asm}(\diamond)/A \to \mathbf{Asm}(U)$ are the dependent maps $b : \prod_{a \in A} B(a)$ such that there exists a natural number e such that for any $a \in A$ and $n \in \mathrm{E}_A(a)$, the application $e \cdot n$ is defined and belongs to $\mathrm{E}_{B(a)}(b(a))$. The context comprehension $\{B\}$ is the assembly whose underlying set is $\sum_{a \in A} B(a)$ and set of realizers is $\left\{\langle n_1, n_2\rangle \mid n_1 \in \mathrm{E}_A a, n_2 \in \mathrm{E}_{B(a)}(b)\right\}$, where $\langle n_1, n_2\rangle$ is a fixed effective encoding of tuples of natural numbers.

We refer the reader to [176, 114, 89] for how to model various type constructors. Propositional truncation might not appear explicitly in the literature. It is known [16] that propositional truncation in extensional type theory corresponds to image factorization in category theory. From the construction of coequalizers in **Asm** [176, Theorem 1.5.2], the image $\operatorname{Im} f$ of a morphism of assemblies $f : B \to A$ is given by the set-theoretic image $\{a \in A \mid \exists_{b \in B} f(b) = a\}$ with realizers $\mathrm{E}_{\operatorname{Im} f}(a) = \bigcup_{b \in f^{-1}(a)} \mathrm{E}_B(b)$. Then, for a family of assemblies B over A, the propositional truncation $\mathsf{Trunc}(B)$ is the family of assemblies over A defined by $\mathsf{Trunc}(B)(a) = \{* \mid \exists_{b \in B(a)} \top\}$ with realizers $\mathrm{E}_{\mathsf{Trunc}(B)(a)}(*) = \bigcup_{b \in B(a)} \mathrm{E}_{B(a)}(b)$.

By Theorem 7.2.15, we have the model $\left(\mathbf{PSh}_{\mathbf{B}}^{\mathbf{Asm}}\right)_{\mathrm{fib}}$ of CTT which we will refer to as **CAsm**. We call an object of **CAsm** a *cubical assembly*.

8.1.3. Remark. We can choose a different cube category. The choice of cube category does not matter for the results in this chapter.

For any Grothendieck universe $\mathcal{U}$, we have an assembly $\mathbf{Asm}_{\mathcal{U}}$ whose underlying set is the $\mathcal{U}$-small assemblies and any natural number realizes any element of $\mathbf{Asm}_{\mathcal{U}}$. The inclusion $\mathbf{Asm}_{\mathcal{U}} \subset \mathbf{Asm}$ determines a family of assemblies and thus $\mathbf{Asm}_{\mathcal{U}}$ is a universe in the model **Asm**. The universe $\mathbf{Asm}_{\mathcal{U}}$ is closed under all the type constructors of the good type theory. We thus apply Proposition 7.2.16 to obtain universes in **CAsm**.

It is known that the model **Asm** has W-types and an explicit construction is found in [173, Section 2.2]. We thus apply the constructions in Sections 7.1.5 and 7.2.6 to obtain higher inductive types in **CAsm** including suspensions and propositional truncation.

8.2 Impredicative universe

8.2.1. Definition. We say a universe $\mathtt{u} : U$ is *impredicative* if it is weakly closed under Π-types over arbitrary types. Precisely, $\mathtt{u}$ is required to be equipped with a type formation rule

$$\Pi_{\mathtt{u}} : (\mathtt{A} : () \to U, \mathtt{B} : (\mathtt{x} : \mathtt{A}) \to \mathtt{u}) \Rightarrow \mathtt{u}$$

and the introduction, elimination, and equality rules for Π-types. Notice that $\mathtt{A}$ need not belong to the universe $\mathtt{u}$.

With an impredicative universe, one can represent inductive types as certain types of polymorphic functions [73]. For example, the natural numbers type is defined to be

$$\mathsf{N} \equiv \Pi_{\mathtt{A}:\mathtt{u}} \mathtt{A} \texttt{ -> } (\mathtt{A} \texttt{ -> } \mathtt{A}) \texttt{ -> } \mathtt{A} \tag{8.1}$$

with zero $\mathsf{zero} \equiv \lambda \mathtt{Axf.x} : \mathsf{N}$ and successor $\mathsf{succ} \equiv \lambda \mathtt{n}.\lambda \mathtt{Axf.f}(\mathtt{n}(\mathtt{A}, \mathtt{x}, \mathtt{f})) : \mathsf{N} \texttt{ -> } \mathsf{N}$. By the impredicativity, the type N belongs to $\mathtt{u}$. We have the non-dependent elimination rule for N

$$\begin{aligned}
&\mathsf{rec}_{\mathsf{N}} : ([\mathtt{A} : () \to \mathtt{u}], \mathtt{a} : () \to \mathtt{A}, \mathtt{f} : (\mathtt{x} : \mathtt{A}) \to \mathtt{A}, \mathtt{n} : () \to \mathsf{N}) \Rightarrow \mathtt{A} \\
&\quad _ : (\mathtt{A} : () \to \mathtt{u}, \mathtt{a} : () \to \mathtt{A}, \mathtt{f} : (\mathtt{x} : \mathtt{A}) \to \mathtt{A}) \Rightarrow \mathsf{rec}_{\mathsf{N}}(\mathtt{a}, \mathtt{f}, \mathsf{zero}) \equiv \mathtt{a} : \mathtt{A} \\
&\quad _ : (\mathtt{A} : () \to \mathtt{u}, \mathtt{a} : () \to \mathtt{A}, \mathtt{f} : (\mathtt{x} : \mathtt{A}) \to \mathtt{A}, \mathtt{n} : () \to \mathsf{N}) \\
&\qquad \Rightarrow \mathsf{rec}_{\mathsf{N}}(\mathtt{a}, \mathtt{f}, \mathsf{succ}(\mathtt{n})) \equiv \mathtt{f}(\mathsf{rec}_{\mathsf{N}}(\mathtt{a}, \mathtt{f}, \mathtt{n})) : \mathtt{A}
\end{aligned}$$

defined by $\mathsf{rec}_{\mathsf{N}}(\mathtt{a}, \mathtt{f}, \mathtt{n}) \equiv \mathtt{n}(\mathtt{A}, \mathtt{a}, \mathtt{f})$. However, the general elimination rule for N is not derivable [159, 70]. In the presence of identity types, we can refine the definition (8.1) by equipping with certain coherence data, and then it satisfies the elimination principle under some restriction on truncation levels [156]. Identity types also allow us to represent higher inductive types as types of polymorphic functions [155, 13]. In this section, we construct a model of a universe that is both impredicative and univalent to justify the use of an impredicative universe in homotopy type theory.

The model **Asm** of the good type theory has an impredicative universe PER. The underlying set of the assembly PER is the set of partial equivalence relations on the set of natural numbers n, that is, symmetric and transitive relations. Any natural number realizes any $R \in \mathrm{PER}$. The family of assemblies $\mathsf{el}_{\mathrm{PER}}$ is defined as $\mathsf{el}_{\mathrm{PER}}(R) = \mathbb{N}/R$, the set of R-equivalence classes on $\{n \in \mathbb{N} \mid (n, n) \in R\}$, with realizers $\mathrm{E}_{\mathbb{N}/R}(A) = A$. The universe PER classifies *modest families*.

8.2.2. Definition. We say a family of assemblies B over A is *modest* if for any $a \in A$ and $b_1, b_2 \in B(A)$, if b_1 and b_2 have a common realizer then $b_1 = b_2$.

$\mathbb{N}/R$ is modest for any $R \in \mathrm{PER}$. Conversely, for a modest family of assemblies B over A, we have a morphism $R : A \to \mathrm{PER}$ defined by $(n_1, n_2) \in R(a)$ if there exists some $b \in B(a)$ realized by both n_1 and n_2.

8.2.3. PROPOSITION. *The universe* PER *in* **Asm** *is weakly closed under all the type constructors of the good type theory,* Π*-types over arbitrary types and* W*-types.*

Proof:
This is proved by checking that modest assemblies are closed under those type constructors. See [84, 114, 89] for details. An explicit construction of W-types is found in [173, Section 2.2]. □

8.2.4. REMARK. The universe PER does not satisfy equalities of type constructors such as $\mathsf{el}_{\mathrm{PER}}\big(\prod_{a\in A} R(a)\big) = \prod_{a\in A} \mathsf{el}_{\mathrm{PER}}(R(a))$, simply because elements of the left side are sets of natural numbers while elements of the right side are functions. Therefore, PER is only weakly closed under type constructors.

8.2.5. THEOREM. *The model of CTT* **CAsm** *has a univalent and impredicative universe.*

Proof:
Apply Proposition 7.2.16 to construct a universe in **CAsm** out of PER. The impredicativity is inherited from PER. Univalence is derivable in CTT. □

8.3 Failure of propositional resizing

In a type theory with a well-behaved equality type ==, a type A is a *proposition* if

$$\mathsf{IsProp}(\mathtt{A}) := \forall_{\mathtt{x}_1:\mathtt{A}} \forall_{\mathtt{x}_2:\mathtt{A}} \mathtt{x}_1 == \mathtt{x}_2$$

is inhabited. For a universe u, we define the universe of propositions by

$$\mathsf{Prop}(\mathsf{u}) := \{\mathtt{A} : \mathsf{u} \mid \mathsf{IsProp}(\mathtt{A})\}.$$

For a larger universe $\widehat{\mathsf{u}}$ the inclusion u -> $\widehat{\mathsf{u}}$ induces a function Prop(u) -> Prop($\widehat{\mathsf{u}}$). The *propositional resizing axiom* [172, Section 3.5] asserts that the function Prop(u) -> Prop($\widehat{\mathsf{u}}$) is an equivalence.

The propositional resizing axiom (for all universes) implies that the universe Prop(u) is impredicative in a slightly weaker sense. Given a type $\mathtt{A} : \widehat{\mathsf{u}}$ in a larger universe and a proposition $\mathtt{B} : \mathtt{A} \to \mathsf{Prop}(\mathsf{u})$, we have $\Pi(\mathtt{A}, \mathtt{B}) : \mathsf{Prop}(\widehat{\mathsf{u}})$ and then use the propositional resizing axiom to get a type $\Pi'(\mathtt{A}, \mathtt{B}) : \mathsf{Prop}(\mathsf{u})$ equivalent to $\Pi(\mathtt{A}, \mathtt{B})$. We have the introduction and elimination rules for $\Pi'(\mathtt{A}, \mathtt{B})$ transported from $\Pi(\mathtt{A}, \mathtt{B})$, but the equality rules hold only up to ==.

Suppose that u is an impredicative universe. Then the universe Prop(u) is also impredicative because propositions are closed under arbitrary Π-types, but this

does not imply the propositional resizing axiom. Nevertheless, we can find the best approximation of a proposition in $\widehat{\mathsf{u}}$ by a proposition in u. For a proposition $\mathtt{P} : \mathsf{Prop}(\widehat{\mathsf{u}})$, we define

$$\gamma(\mathtt{P}) := \forall_{\mathtt{X}:\mathsf{Prop}(\mathsf{u})}(\mathtt{P} \to \mathtt{X}) \to \mathtt{X}$$

which belongs to $\mathsf{Prop}(\mathsf{u})$ because $\mathsf{Prop}(\mathsf{u})$ is impredicative. We have a canonical function $\eta_{\mathtt{P}} : \mathtt{P} \to \gamma(\mathtt{P})$ defined by

$$\eta_{\mathtt{P}}(\mathtt{x}) = \lambda \mathtt{X}\mathtt{f}.\mathtt{f}(\mathtt{x}).$$

8.3.1. Proposition. *For an impredicative universe* u *and a larger universe* $\widehat{\mathsf{u}}$*, the function* $\mathsf{Prop}(\mathsf{u}) \to \mathsf{Prop}(\widehat{\mathsf{u}})$ *is an equivalence if and only if the function* $\eta_{\mathtt{P}} : \mathtt{P} \to \gamma(\mathtt{P})$ *is an equivalence for any* $\mathtt{P} : \mathsf{Prop}(\widehat{\mathsf{u}})$*.*

Proof:
Straightforward. □

The definition of the function $\eta_{\mathtt{P}}$ makes sense for any proposition $\mathtt{P}$ not necessarily in a larger universe, and thus we can formulate propositional resizing in the presence of an impredicative universe as follows.

8.3.2. Definition. We say an impredicative universe u *admits propositional resizing* if the function $\eta_{\mathtt{P}} : \mathtt{P} \to \gamma(\mathtt{P})$ is an equivalence for any proposition $\mathtt{P}$.

The impredicative universe PER in the model **Asm** of the good type theory admits propositional resizing. A proposition in **Asm** is a family of assemblies $P : A \to \mathbf{Asm}$ such that the underlying set of $P(a)$ has at most one element for any $a \in A$. Then, by definition, any proposition in **Asm** is a modest family of assemblies and thus belongs to PER.

8.3.3. Remark. The fact that PER admits propositional resizing implies that any monomorphism of assemblies is classified by some morphism into $\mathsf{Prop}(\mathrm{PER})$. This does not imply that $\mathsf{Prop}(\mathrm{PER})$ is a subobject classifier in the category **Asm** (there is no subobject classifier in **Asm** as it is not an elementary topos) because we do not have the uniqueness of a characteristic morphism. For example, both partial equivalence relations $\{(0,0)\}$ and $\{(1,1)\}$ classify the same assembly, the singleton $\{*\}$.

In this section, we show that the impredicative universe in **CAsm** does not admit propositional resizing, that is, we construct a proposition $\mathtt{P}$ valued in the impredicative universe such that the function $\eta_{\mathtt{P}} : \mathtt{P} \to \gamma(\mathtt{P})$ is not an equivalence.

8.3.4. Remark. Assuming Grothendieck universes, **CAsm** also has predicative universes, but it is unknown whether these predicative universes satisfy the propositional resizing axiom.

8.3.1 Uniform assemblies

The key idea to a counterexample to propositional resizing is the orthogonality of modest and *uniform* assemblies [176]: if B is modest and A is uniform and well-supported, then the morphism $b \mapsto \lambda a.b : B \to B^A$ is an isomorphism. Since the impredicative universe PER classifies modest assemblies, $\prod_{\mathtt{X}:\mathrm{PER}}(A \texttt{ -> } \mathtt{X}) \texttt{ -> } \mathtt{X}$ has a section for any uniform, well-supported assembly A. There is still a possibility that A does not have a section in which case A and $\prod_{\mathtt{X}:\mathrm{PER}}(A \texttt{ -> } \mathtt{X}) \texttt{ -> } \mathtt{X}$ are never equivalent.

We first extend the notion of uniformity for presheaves in **Asm**.

8.3.5. Definition. An assembly A is *uniform* if there exists an $n \in \mathbb{N}$ realizing all the elements of A, that is, $\bigcap_{a \in A} \mathrm{E}_A(a)$ is non-empty. We say a presheaf A over a category $\mathcal{C}$ in **Asm** is *uniform* if every $A(x)$ is uniform.

8.3.6. Definition. In the good type theory, we say a type $\mathtt{A}$ is *well-supported* if the truncation $\mathsf{Trunc}(\mathtt{A})$ is inhabited.

From the construction of propositional truncation in **Asm**, we have the following characterization of well-supported presheaves.

8.3.7. Proposition. *A presheaf A over a category $\mathcal{C}$ in* **Asm** *is well-supported (in the model $\mathbf{PSh}_{\mathcal{C}}^{\mathbf{Asm}}$) if and only if there exists a natural number e such that, for any $x \in \mathcal{C}_0$ and $n \in \mathrm{E}_{\mathcal{C}_0}(x)$, there exists an $a \in A(x)$ such that the application $e \cdot n$ is defined and belongs to $\mathrm{E}_A(a)$.*

By definition, a modest assembly cannot distinguish elements with a common realizer, while elements of a uniform assembly have a common realizer. Therefore, a modest assembly "believes a uniform assembly has at most one element". Formally, the following holds.

8.3.8. Proposition. *Let $\mathcal{C}$ be a category in* **Asm**. *For any uniform presheaf A over $\mathcal{C}$ and any presheaf B over $\mathcal{C}$ valued in* PER, *the precomposition function*

$$\mathsf{trunc}^* := \lambda \mathtt{f}.\mathtt{f} \circ \mathsf{trunc} : (\mathsf{Trunc}(A) \texttt{ -> } B) \texttt{ -> } (A \texttt{ -> } B)$$

is an isomorphism. In particular, if, in addition, A is well-supported, then the function $\lambda \mathtt{yx}.\mathtt{y} : B \texttt{ -> } (A \texttt{ -> } B)$ is an isomorphism.

Proof:
Since trunc is a regular epi, trunc^* is a monomorphism. We show that trunc^* is a regular epi witnessed by the code of the identity function. Let $x \in \mathcal{C}$ be an object and $f : \mathrm{Y}_{\mathcal{C}}\, x \times A \to B$ a morphism of presheaves tracked by e. We have to show that there exists a morphism $f' : \mathrm{Y}_{\mathcal{C}}\, x \times \mathsf{Trunc}(A) \to B$ tracked by e such that $f' \circ (\mathrm{Y}_{\mathcal{C}}\, x \times \mathsf{trunc}) = f$. By the construction of propositional

truncation, it suffices to show that $f(u, a_1) = f(u, a_2)$ for any arrow $u : x' \to x$ and any elements $a_1, a_2 \in A(x')$. Since B is modest, it suffices to find a realizer for both $f(u, a_1)$ and $f(u, a_2)$. Since A is uniform, $A(x')$ has a common realizer $m \in \bigcap_{a \in A(x')} \mathrm{E}_{A(x')}(a)$, and we have $e \cdot \langle n, m \rangle \in \mathrm{E}_{B(x')}(f(u, a_1)) \cap \mathrm{E}_{B(x')}(f(u, a_2))$ for some $n \in \mathrm{E}_{\mathrm{Hom}(x',x)}(u)$. □

We now have a sufficient condition for a failure of propositional resizing.

8.3.9. Theorem. *Let $A : \mathbf{CAsm}(\diamond) \to \mathbf{CAsm}(U)$ be a closed type and $B : \mathbf{CAsm}(\diamond)/\{A\} \to \mathbf{CAsm}(U)$ a type. Suppose that B is uniform and well-supported but does not have a section. Then the function $\eta : B$ `->` $\gamma(B)$ is not an equivalence.*

Proof:
By Proposition 8.3.8, we see that $\gamma(B) = \forall_{\mathtt{X:Prop(u)}}(B$ `->` $\mathtt{X})$ `->` $\mathtt{X}$ has a section while B does not by assumption. □

We consider constructing a counterexample as in Theorem 8.3.9 out of a type in **Asm**.

8.3.10. Theorem. *Let $A : \mathbf{Asm}(\diamond) \to \mathbf{Asm}(U)$ be a closed type and $B : \mathbf{Asm}(\diamond)/\{A\} \to \mathbf{Asm}(U)$ a type. Suppose that B is uniform and well-supported but does not have a section. Then $\nabla_A B$ is a proposition with respect to path types such that the function $\eta : \nabla_A B$ `->` $\gamma(\nabla_A B)$ is not an equivalence.*

Proof:
By Theorem 8.3.9, it suffices to show that the type $\nabla_A B : \mathbf{CAsm}(\diamond)/\{\Delta A\} \to \mathbf{CAsm}(U)$ is uniform and well-supported but does not have a section. Since $\mathrm{ev}_1(\nabla_A B)(x, a) \cong B(a)$, the presheaf $\nabla_A B$ cannot have a section as B does not.

To show the uniformity, let $x \in \mathcal{C}$ be an object and $a \in A$ an element. Since B is uniform, there exists a common realize $n \in \bigcap_{b \in B(a)} \mathrm{E}_{B(a)}(b)$. Then the code of the constant function at n realizes all the elements of $\nabla_A B(x, a) = \mathrm{Hom}(1, x)$ `->` $B(a)$.

For the well-supportedness, let e be a natural number such that, for any $a \in A$ and $n \in \mathrm{E}_A(a)$, there exists a $b \in B(a)$ such that $e \cdot n$ is defined and belongs to $\mathrm{E}_{A(a)}(b)$. Then the code f of the function mapping $\langle n, m \rangle$ to the code of the constant function at $e \cdot m$ realizes that $\nabla_A B$ is well-supported. Indeed, for any $x \in \mathcal{C}_0$, $n \in \mathrm{E}_{\mathcal{C}_0}(x)$, $a \in A$ and $m \in \mathrm{E}_A(a)$, the code $f \cdot \langle n, m \rangle$ realizes the constant function $\mathrm{Hom}(1, x) \ni u \mapsto b \in B(a)$ for some $b \in B(a)$ such that $e \cdot m \in \mathrm{E}_{B(a)}(b)$. □

8.3.2 The counterexample

We construct a closed type $A : \mathbf{Asm}(\diamond) \to \mathbf{Asm}(U)$ and a type $B : \mathbf{Asm}(\diamond)/\{A\} \to \mathbf{Asm}(U)$ satisfying the assumptions of Theorem 8.3.10 so that the codiscrete presheaf $\nabla_A B$ is a proposition with respect to path types such that the function $\eta : \nabla_A B \to \gamma(\nabla_A B)$ is not an equivalence. We define A to be the assembly

$$(\mathbb{N}, \mathrm{E}_A(n) = \{m \in \mathbb{N} \mid m > n\})$$

and $B(n)$ to be the assembly

$$\big(\{m \in \mathbb{N} \mid m > n\}, \mathrm{E}_{B(n)}(m) = \{n, m\}\big).$$

Then B is uniform because n realizes all the elements of $B(n)$. The code of the identity function realizes that B is well-supported. To see that B does not have a section, suppose that a section $f : \prod_{n \in A} B(n)$ is tracked by e. Then, for any $m > n$, we have $e \cdot m \in \{n, f(n)\}$. This implies that $m \leq e \cdot (m + 1) \leq f(0)$ for any m, a contradiction.

8.4 Markov's Principle

Constructive Recursive Mathematics is a form of constructivism in which the validity of a proposition is justified by the existence of an *algorithm* or *recursive function*. In Constructive Recursive Mathematics, *Markov's Principle*

$$\forall_{P:\mathbb{N}\to 2} \neg(\forall_{n:\mathbb{N}} \neg P(n)) \to \exists_{n:\mathbb{N}} P(n)$$

is accepted. Informally, this principle states that for an algorithm P that answers "yes" or "no" for each input, if it is not the case that P answers "no" for any input, then there exists an n for which P answers "yes". The validity of this principle is justified by an algorithm to find a witness n from a given algorithm P. Indeed, we can sequentially execute P for $0, 1, 2, \ldots$ until P answers "yes". The assumption that it is not the case that P answers "no" for any input guarantees that this algorithm terminates.

In a type theory with propositional truncation, $\exists_{\mathtt{n:N}} \mathtt{P(n)}$ is defined to be $\mathsf{Trunc}(\sum_{\mathtt{n:N}} \mathtt{P(n)})$, but Markov's Principle is equivalent to a proposition definable without propositional truncation. Indeed, for any $\mathtt{P} : \mathsf{N} \to 2$, the proposition $\exists_{\mathtt{n:N}} \mathtt{P(n)}$ is equivalent to the type

$$\sum_{\mathtt{n:N}} \mathtt{P(n)} \times \prod_{\mathtt{m:N}} \mathtt{P(m)} \to \mathtt{n} \leq \mathtt{m}$$

whose elements are the least natural numbers satisfying $\mathtt{P}$. This type is a proposition because the least element is unique. Therefore, Markov's Principle in a type theory with sufficient structure is equivalent to

$$\mathsf{MP} = \prod_{\mathtt{P:N\to 2}} ((\prod_{\mathtt{n:N}} \mathtt{P(n)} \to 0) \to 0) \to \sum_{\mathtt{n:N}} \mathtt{P(n)} \times \prod_{\mathtt{m:N}} \mathtt{P(m)} \to \mathtt{n} \leq \mathtt{m}.$$

8.4.1. Proposition. *For any category $\mathcal{C}$ in the model of the good type theory in assemblies for Kleene's first model $\mathcal{K}_1$, the presheaf model of the good type theory $\mathbf{PSh}_{\mathcal{C}}^{\mathbf{Asm}}$ satisfies Markov's Principle.*

Proof:
This is because **Asm** satisfies Markov's Principle and the constant presheaf functor $\mathbf{Asm} \to \mathbf{PSh}_{\mathcal{C}}^{\mathbf{Asm}}$ preserves type-theoretic structures. □

8.4.2. Corollary. *The model* **CAsm** *of CTT in cubical assemblies for Kleene's first model $\mathcal{K}_1$ satisfies Markov's Principle.*

8.5 Church's Thesis

Church's Thesis is one of the defining characteristics of Constructive Recursive Mathematics. It asserts that any function on natural numbers is computable:

$$\forall_{f:\mathbb{N}\to\mathbb{N}}\exists_{e:\mathbb{N}}\forall_{n:\mathbb{N}}\exists_{m:\mathbb{N}}\mathrm{T}(e,n,m) \wedge \mathrm{U}(m) = f(n)$$

where $\mathrm{T} : \mathbb{N} \times \mathbb{N} \times \mathbb{N} \to 2$ and $\mathrm{U} : \mathbb{N} \to \mathbb{N}$ are Kleene's computation predicate and result extraction function, respectively, meaning that $\mathrm{T}(e, n, m)$ is true if e is the code of a (deterministic) Turing machine, the computation of e terminates for input n, and m is the code of the computation, and that $\mathrm{U}(m)$ is the output of the computation.

Since T and U are primitive recursive, they are definable inside a type theory with natural numbers type. Recall that, in a type theory with propositional truncation, the existential quantifier $\exists_{\mathtt{x:A}}\mathtt{P}$ is defined to be $\mathsf{Trunc}(\sum_{\mathtt{x:A}}\mathtt{P})$. Since there is at most one natural number $\mathtt{m}$ satisfying $\mathrm{T}(\mathtt{e},\mathtt{n},\mathtt{m})$, the type $\sum_{\mathtt{m:N}}\mathrm{T}(\mathtt{e},\mathtt{n},\mathtt{m}) \wedge \mathrm{U}(\mathtt{m})$ == $\mathtt{f}(\mathtt{n})$ is already a proposition. Therefore, Church's Thesis in a type theory with sufficient structure is equivalent to

$$\mathsf{CT} = \prod_{\mathtt{f:N->N}} \mathsf{Trunc}(\sum_{\mathtt{e:N}}\prod_{\mathtt{n:N}}\sum_{\mathtt{m:N}}\mathrm{T}(\mathtt{e},\mathtt{n},\mathtt{m}) \times \mathrm{U}(\mathtt{m}) == \mathtt{f}(\mathtt{n})).$$

We show in Section 8.5.1 that the negation of Church's Thesis holds in any cubical model obtained by the method of Chapter 7. To obtain a model of univalent type theory satisfying Church's Thesis, we use the theory of *modalities* in homotopy type theory [146]. The idea is to collect those types who "believe that $\mathsf{Trunc}(\sum_{\mathtt{e:N}}\prod_{\mathtt{n:N}}\sum_{\mathtt{m:N}}\mathrm{T}(\mathtt{e},\mathtt{n},\mathtt{m}) \times \mathrm{U}(\mathtt{m}) == \mathtt{f}(\mathtt{n}))$ is contractible for any `f : N -> N`". Such types are called *null types*. We review properties of null types in Section 8.5.2. Finally, we prove in Section 8.5.3 that Church's Thesis holds in null types in the cubical assembly model. We also see that the same technique applies to obtain a model of univalent type theory satisfying Brouwer's Continuity Principle.

8.5.1 Failure of Church's Thesis in internal cubical models

Let $\mathcal{M}$ be a model of the good type theory and construct a model $\left(\mathbf{PSh}_{\mathcal{C}}^{\mathcal{M}}\right)_{\text{fib}}$ of CTT as in Theorem 7.2.15 for a cube category $\mathcal{C}$. We also assume that $\mathcal{M}$ models W-types so that $\left(\mathbf{PSh}_{\mathcal{C}}^{\mathcal{M}}\right)_{\text{fib}}$ models propositional truncation.

8.5.1. Theorem. *The negation of Church's Thesis holds in* $\left(\mathbf{PSh}_{\mathcal{C}}^{\mathcal{M}}\right)_{\text{fib}}$.

We prepare a couple of lemmas.

8.5.2. Lemma. *The mapping* $\Delta : \mathcal{M} \to \left(\mathbf{PSh}_{\mathcal{C}}^{\mathcal{M}}\right)_{\text{fib}}$ *sending a type* A *to the constant presheaf at* A *preserves* Π*-types,* Σ*-types, identity types, and natural numbers type.*

Proof:
Straightforward. □

8.5.3. Lemma. *"Untruncated" Church's Thesis*

$$\prod_{\mathtt{f:N->N}} \sum_{\mathtt{e:N}} \prod_{\mathtt{n:N}} \sum_{\mathtt{m:N}} \mathrm{T}(\mathtt{e},\mathtt{n},\mathtt{m}) \times \mathrm{U}(\mathtt{m}) == \mathtt{f}(\mathtt{n})$$

is inconsistent with the type theory with Π*-types,* Σ*-types, and extensional identity types.*

Proof:
It is known [168] that Church's Thesis is inconsistent with the axiom of choice and function extensionality. Under the propositions-as-types interpretation, the type theory with Π-types and Σ-types satisfies the axiom of choice (the type-theoretic axiom of choice). Function extensionality holds in the type theory with Π-types and extensional identity types. Hence, untruncated Church's Thesis is inconsistent with the type theory with Π-types, Σ-types, and extensional identity types. □

8.5.4. Remark. The consistency of untruncated Church's Thesis with the type theory with Π-types, Σ-types, and *intensional* identity types has been an open problem since it was conjectured by Maietti and Sambin [122]. Ishihara et al. [88] proved that untruncated Church's Thesis is consistent with a variant of the type theory where the congruence rule for the λ-abstraction is dropped. Recently, Yamada [182] announced a proof of Maietti and Sambin's conjecture by a combination of realizability and game semantics. We also note that, as the proof of Lemma 8.5.3 shows, untruncated Church's Thesis is inconsistent with function extensionality. In particular, since univalence implies function extensionality [172, Section 4.9], untruncated Church's Thesis is inconsistent with the univalence axiom.

Proof of Theorem 8.5.1:
We define a type $A : (\mathtt{f} : () \to \mathsf{N} \to \mathsf{N}) \Rightarrow U$ by

$$A(\mathtt{f}) = \sum_{\mathtt{e}:\mathsf{N}} \prod_{\mathtt{n}:\mathsf{N}} \sum_{\mathtt{m}:\mathsf{N}} \mathrm{T}(\mathtt{e},\mathtt{n},\mathtt{m}) \times \mathrm{U}(\mathtt{n}) == \mathtt{f}(\mathtt{n})$$

so that $\mathsf{CT} = \prod_{\mathtt{f}:\mathsf{N}\to\mathsf{N}} \mathsf{Trunc}(A(\mathtt{x}))$. By Lemma 8.5.2, the constant presheaf functor $\Delta : \mathcal{M} \to \left(\mathbf{PSh}_{\mathcal{C}}^{\mathcal{M}}\right)_{\mathrm{fib}}$ preserves the interpretation of A. Therefore, Church's Thesis is interpreted in $\left(\mathbf{PSh}_{\mathcal{C}}^{\mathcal{M}}\right)_{\mathrm{fib}}$ as

$$\prod_{\mathtt{f}:\Delta(\mathcal{M}(\mathsf{N})\to\mathcal{M}(\mathsf{N}))} \mathsf{Trunc}(\Delta\,\mathcal{M}(A))(\mathtt{f}).$$

We construct the following two functions in $\left(\mathbf{PSh}_{\mathcal{C}}^{\mathcal{M}}\right)_{\mathrm{fib}}$:

- $\prod_{\mathtt{f}:\Delta(\mathcal{M}(\mathsf{N})\to\mathcal{M}(\mathsf{N}))} \mathsf{Trunc}(\Delta\,\mathcal{M}(A))(\mathtt{f}) \to \left(\nabla_{\mathcal{M}(\mathsf{N})\to\mathcal{M}(\mathsf{N})}\,\mathcal{M}(A)\right)(\mathtt{f})$;
- $\left(\prod_{\mathtt{f}:\Delta(\mathcal{M}(\mathsf{N})\to\mathcal{M}(\mathsf{N}))} \left(\nabla_{\mathcal{M}(\mathsf{N})\to\mathcal{M}(\mathsf{N})}\,\mathcal{M}(A)\right)(\mathtt{f})\right) \to \left(\mathbf{PSh}_{\mathcal{C}}^{\mathcal{M}}\right)_{\mathrm{fib}}(0)$.

Then we readily get a function $\left(\prod_{\mathtt{f}:\Delta(\mathcal{M}(\mathsf{N})\to\mathcal{M}(\mathsf{N}))} \mathsf{Trunc}(\Delta\,\mathcal{M}(A))(\mathtt{f})\right) \to 0$.

For the former function, it suffices to give a function

$$(\Delta\,\mathcal{M}(A))(\mathtt{f}) \to \left(\nabla_{\mathcal{M}(\mathsf{N})\to\mathcal{M}(\mathsf{N})}\,\mathcal{M}(A)\right)(\mathtt{f})$$

for all $\mathtt{f} : \Delta(\mathcal{M}(\mathsf{N}) \to \mathcal{M}(\mathsf{N}))$ because the codomain is a proposition by Proposition 7.2.20. By the adjunction $\mathrm{ev}_1 \dashv \nabla$ where ev_1 is the evaluation at the final object $1 : \mathcal{C}$, it suffices to give a function $\mathrm{ev}_1(\Delta\,\mathcal{M}(A))(\mathtt{f}) \to \mathcal{M}(A)(\mathtt{f})$ for all $\mathtt{f} : \mathcal{M}(\mathsf{N}) \to \mathcal{M}(\mathsf{N})$. Since $\mathrm{ev}_1(\Delta\,\mathcal{M}(A)) \cong \mathcal{M}(A)$, just give the identity.

For the latter function, observe that

$$\prod_{\mathtt{f}:\Delta(\mathcal{M}(\mathsf{N})\to\mathcal{M}(\mathsf{N}))} \left(\nabla_{\mathcal{M}(\mathsf{N})\to\mathcal{M}(\mathsf{N})}\,\mathcal{M}(A)\right)(\mathtt{f}) \simeq \nabla\left(\prod_{\mathtt{f}:\mathcal{M}(\mathsf{N})\to\mathcal{M}(\mathsf{N})} \mathcal{M}(A)(\mathtt{f})\right)$$

by checking that both sides have the same universal property and that

$$\left(\mathbf{PSh}_{\mathcal{C}}^{\mathcal{M}}\right)_{\mathrm{fib}}(0) \simeq \nabla\,\mathcal{M}(0).$$

Then apply $\nabla\,\mathcal{M}(-)$ to the function $\left(\prod_{\mathtt{f}:\mathsf{N}\to\mathsf{N}} A(\mathtt{f})\right) \to 0$ obtained from the inconsistency of untruncated Church's Thesis (Lemma 8.5.3). □

8.5.2 Null types

In the previous section, we have seen that the cubical assembly model does not satisfy Church's Thesis. To get a model of univalent type theory satisfying Church's Thesis, we use the theory of *modalities* and *localization* in homotopy type theory [146]. The idea is to construct a new model from the cubical assembly model in which some propositions are forced to be true. Recall that Church's Thesis is of the form

$$\forall_{\mathtt{x:A}}\mathsf{Trunc}(\mathtt{B}(\mathtt{x}))$$

for a type $\mathtt{B} : (\mathtt{x} : \mathtt{A}) \to U$. We cannot directly force Church's Thesis itself to be true because its negation holds in the cubical assembly model. However, the type $\mathsf{Trunc}(\mathtt{B}(\mathtt{x}))$ for each $\mathtt{x} : \mathtt{A}$ is non-empty in the cubical assembly model because the assembly model of extensional type theory satisfies Church's Thesis. Hence, there is a chance to force $\mathsf{Trunc}(\mathtt{B}(\mathtt{x}))$ to be true for any $\mathtt{x} : \mathtt{A}$, and then $\forall_{\mathtt{x:A}}\mathsf{Trunc}(\mathtt{B}(\mathtt{x}))$ becomes true.

In this thesis, we use a special case of localization called *nullification*.

8.5.5. Definition. We define a type $\mathsf{IsNull} : ([\mathtt{A} : () \to U], \mathtt{P} : (\mathtt{x} : \mathtt{A}) \to U, \mathtt{B} : () \to U) \Rightarrow U$ in univalent type theory by

$$\mathsf{IsNull}(\mathtt{P}, \mathtt{B}) \equiv \forall_{\mathtt{x}}\mathsf{IsEquiv}(\lambda(\mathtt{y} : \mathtt{B}).\lambda(_ : \mathtt{P}(\mathtt{x})).\mathtt{y}).$$

A term of $\mathsf{IsNull}(\mathtt{P}, \mathtt{B})$ is called a $\mathtt{P}$-*null structure on* $\mathtt{B}$. By a $\mathtt{P}$-*null type*, we mean a type equipped with a $\mathtt{P}$-null structure. Although the notion of a $\mathtt{P}$-null structure makes sense for any family of types $\mathtt{P}$, in this thesis, we only consider the case when $\mathtt{P}$ is a proposition. A $\mathtt{P}$-*nullification operator* is the following structure.

$$\begin{aligned}
&\mathcal{L} : (\mathtt{B} : () \to U) \Rightarrow U\\
&_ : (\mathtt{B} : () \to U) \Rightarrow \mathsf{IsNull}(\mathtt{P}, \mathcal{L}(\mathtt{B}))\\
&\eta : ([\mathtt{B} : () \to U], \mathtt{b} : () \to \mathtt{B}) \Rightarrow \mathcal{L}(\mathtt{B})\\
&_ : (\mathtt{B} : () \to U, \mathtt{C} : () \to U, _ : () \to \mathsf{IsNull}(\mathtt{P}, \mathtt{C})) \Rightarrow \mathsf{IsEquiv}(\lambda(\mathtt{f} : \mathcal{L}(\mathtt{B}) \texttt{ -> } \mathtt{C}).\mathtt{f} \circ \eta)
\end{aligned}$$

Intuitively, a $\mathtt{P}$-null type is a type who "believes $\mathtt{P}(\mathtt{x})$ is true for any $\mathtt{x} : \mathtt{A}$", and thus we expect that $\mathtt{P}$-null types form a model in which $\mathtt{P}$ is true. We review properties of null types.

8.5.6. Proposition. *Null types are closed under* Π*-types indexed over an arbitrary type,* Σ*-types, unit type, and intensional identity types.*

Proof:
The closure property under Π-types and unit type is immediate. For Σ-types, see [146, Theorem 2.19]. For identity types, let $\mathtt{B}$ be a $\mathtt{P}$-null type. We show that the function

$$\lambda \mathtt{pz.p} : \mathtt{y}_1 \texttt{ == } \mathtt{y}_2 \texttt{ -> } (\mathtt{P}(\mathtt{x}) \texttt{ -> } \mathtt{y}_1 \texttt{ == } \mathtt{y}_2)$$

is an equivalence for any $\mathtt{x} : \mathtt{A}$ and $\mathtt{y}_1, \mathtt{y}_2 : \mathtt{B}$. It suffices to show that the total function

$$(\sum_{\mathtt{y}_2:\mathtt{B}} \mathtt{y}_1 == \mathtt{y}_2) \to (\sum_{\mathtt{y}_2:\mathtt{B}} \mathtt{P}(\mathtt{x}) \to \mathtt{y}_1 == \mathtt{y}_2)$$

is an equivalence for any $\mathtt{x} : \mathtt{A}$ and $\mathtt{y}_1 : \mathtt{B}$. Since the domain is contractible, it suffices to show that the codomain is contractible, but we have

$$\sum_{\mathtt{y}_2:\mathtt{B}} \mathtt{P}(\mathtt{x}) \to \mathtt{y}_1 == \mathtt{y}_2 \simeq \sum_{\mathtt{y}_2:\mathtt{P}(\mathtt{x})\to\mathtt{B}} \prod_{\mathtt{z}:\mathtt{P}(\mathtt{x})} \mathtt{y}_1 == \mathtt{y}_2(\mathtt{z}) \qquad (\mathtt{B} \text{ is } \mathtt{P}\text{-null})$$

$$\simeq \mathtt{P}(\mathtt{x}) \to \sum_{\mathtt{y}_2:\mathtt{B}} \mathtt{y}_1 == \mathtt{y}_2 \qquad (\text{type-theoretic axiom of choice})$$

and $\mathtt{P}(\mathtt{x}) \to \sum_{\mathtt{y}_2:\mathtt{B}} \mathtt{y}_1 == \mathtt{y}_2$ is contractible. □

For a universe $\mathtt{u}$, we define a subuniverse $\mathtt{u}_\mathtt{P} \subset \mathtt{u}$ by

$$\mathtt{u}_\mathtt{P} \equiv \{\mathtt{C} : \mathtt{u} \mid \mathsf{IsNull}(\mathtt{P}, \mathtt{C})\}.$$

8.5.7. Proposition. *If* $\mathtt{P}$ *is* $\mathtt{u}$*-small, the universe* $\mathtt{u}_\mathtt{P}$ *has a* $\mathtt{P}$*-null structure.*

Proof:
Assuming that a $\mathtt{P}$-nullification operator exists, this follows from [146, Theorem 3.11] because the nullification at a family of propositions induces a lex modality by [146, Corollary 3.12], but we can extract from their proof an explicit inverse of the function $\lambda\mathtt{C}\mathtt{z}.\mathtt{C} : \mathtt{u}_\mathtt{P} \to (\mathtt{P}(\mathtt{x}) \to \mathtt{u}_\mathtt{P})$ without reference to the nullification. For $\mathtt{C} : \mathtt{P}(\mathtt{x}) \to \mathtt{u}_\mathtt{P}$, the type $\prod_{\mathtt{z}:\mathtt{P}(\mathtt{x})} \mathtt{C}(\mathtt{z})$ belongs to $\mathtt{u}_\mathtt{P}$ by Proposition 8.5.6 and satisfies $(\prod_{\mathtt{z}:\mathtt{P}(\mathtt{x})} \mathtt{C}(\mathtt{z})) \simeq \mathtt{C}(\mathtt{z}')$ for any $\mathtt{z}' : \mathtt{P}(\mathtt{x})$ because $\mathtt{P}(\mathtt{x})$ is a proposition. We also have $(\mathtt{P}(\mathtt{x}) \to \mathtt{C}) \simeq \mathtt{C}$ for any $\mathtt{C} : \mathtt{u}_\mathtt{P}$ by definition, and thus $\lambda\mathtt{C}.\prod_{\mathtt{z}:\mathtt{P}(\mathtt{x})} \mathtt{C}(\mathtt{z})$ is the inverse of $\lambda\mathtt{C}\mathtt{z}.\mathtt{C}$. □

8.5.8. Proposition. *Any nullification operator preserves propositions.*

Proof:
This follows from [146, Corollary 3.9]. □

The following illustrates that a $\mathtt{P}$-nullification operator "forces $\mathtt{P}$ to be true".

8.5.9. Corollary. *For any* $\mathtt{P}$*-nullification operator* $\mathcal{L}$ *and any* $\mathtt{a} : \mathtt{A}$*, the type* $\mathcal{L}(\mathtt{P}(\mathtt{a}))$ *is contractible.*

Proof:
By Proposition 8.5.8, $\mathcal{L}(\mathtt{P}(\mathtt{a}))$ is a proposition, and thus it suffices to show that it is inhabited. Since $\mathcal{L}(\mathtt{P}(\mathtt{a}))$ is P-null, it is enough to give a function $\mathtt{P}(\mathtt{a})\ \mathtt{->}\ \mathcal{L}(\mathtt{P}(\mathtt{a}))$, but we already have $\eta : \mathtt{P}(\mathtt{a})\ \mathtt{->}\ \mathcal{L}(\mathtt{P}(\mathtt{a}))$. □

8.5.10. Corollary. *For any* P*-nullification operator* $\mathcal{L}$ *and any* P*-null type* B*, the* P*-null proposition* $\mathcal{L}(\mathsf{Trunc}(\mathtt{B}))$ *is the propositional truncation of* B *in* P*-null types, that is, for any* P*-null proposition* Q*, the canonical function* $(\mathcal{L}(\mathsf{Trunc}(\mathtt{B}))\ \mathtt{->}\ \mathtt{Q})\ \mathtt{->}\ (\mathtt{B}\ \mathtt{->}\ \mathtt{Q})$ *is an equivalence.*

Proof:
By definition. □

Let $\mathcal{M}$ be a model of univalent type theory, $A : \mathcal{M}(\diamond) \to \mathcal{M}(U)$ a closed type and $P : \mathcal{M}(\diamond)/\{A\} \to \mathcal{M}(U)$ a proposition. We define $\mathcal{M}_P(U)$ to be the discrete fibration of P-null types in $\mathcal{M}$, that is, the pullback

$$\begin{array}{ccc} \mathcal{M}_P(U) & \longrightarrow & \mathcal{M}(E) \\ \downarrow & \lrcorner & \downarrow \\ \mathcal{M}(U) & \xrightarrow[\mathsf{IsNull}(P,-)]{} & \mathcal{M}(U). \end{array}$$

Let $\mathcal{M}_P(E)$ denote the pullback of $\mathcal{M}(E)$ along the map $\mathcal{M}_P(U) \to \mathcal{M}(U)$. From the closure properties of null types, we have the following.

8.5.11. Proposition. *For any closed type* $A : \mathcal{M}(\diamond) \to \mathcal{M}(U)$ *and any proposition* $P : \mathcal{M}(\diamond)/\{A\} \to \mathcal{M}(U)$*, the representable map* $\mathcal{M}_P(E) \to \mathcal{M}_P(U)$ *is part of a model of type theory* $\mathcal{M}_P$ *with* Π*-types,* Σ*-types, unit type, intensional identity types, and univalent universes. Moreover, the forgetful map* $\mathcal{M}_P \to \mathcal{M}$ *preserves* Π*-types,* Σ*-types, unit type, and intensional identity types. If* $\mathcal{M}$ *has a* P*-nullification operator, then* $\mathcal{M}_P$ *has propositional truncation.*

Null types in internal cubical models

Let $\mathcal{M}$ be a model of qCTT. We show additional closure properties of P-null types in $\mathcal{M}_{\text{fib}}$ when P is a proposition in $\mathcal{M}_{\text{fib}}$ and well-supported in $\mathcal{M}$.

8.5.12. Proposition. *If* P *is well-supported in* $\mathcal{M}$*, then any discrete closed type* $B : \mathcal{M}_{\text{fib}}(\diamond) \to \mathcal{M}_{\text{fib}}(U)$ *has a* P*-null structure.*

Proof:
We show that for any $\mathtt{a} : A$, the function $\mathtt{f} := \lambda\mathtt{b}.\lambda\mathtt{x}.\mathtt{b} : B\ \mathtt{->}\ (P(\mathtt{a})\ \mathtt{->}\ B)$ is an isomorphism in $\mathcal{M}$, and then f is, in particular, an equivalence in $\mathcal{M}_{\text{fib}}$. Since

P is well-supported, $\mathtt{f}$ is injective. To prove surjectivity, let $\mathtt{g} : P(\mathtt{a}) \to B$ be an arbitrary function. By the well-supportedness of P, there exists some element $\mathtt{x} : P(\mathtt{a})$. We show that $\mathtt{f}(\mathtt{g}(\mathtt{x})) == \mathtt{g}$, and then $\mathtt{f}$ is surjective witnessed by $\mathtt{g}(\mathtt{x})$. By function extensionality, it suffices to show that $\mathtt{f}(\mathtt{g}(\mathtt{x}))(\mathtt{x}') == \mathtt{g}(\mathtt{x}')$ for any $\mathtt{x}' : P(\mathtt{a})$. By definition, $\mathtt{f}(\mathtt{g}(\mathtt{x}))(\mathtt{x}') == \mathtt{g}(\mathtt{x})$. Since P is a proposition in $\mathcal{M}_p$, we have a path $\mathtt{p} : \mathbb{I} \to P(\mathtt{a})$ between $\mathtt{x}$ and $\mathtt{x}'$. By the discreteness of B, the path $\mathtt{g} \circ \mathtt{p} : \mathbb{I} \to B$ is constant, which implies that $\mathtt{g}(\mathtt{x}) == \mathtt{g}(\mathtt{x}')$. □

8.5.13. Corollary. *If P is well-supported in $\mathcal{M}$, then the type $\mathcal{M}_{\mathrm{fib}}(\mathtt{0})$ has a P-null structure. Therefore, if $\mathcal{M}$ is non-trivial (that is, the empty type does not have a section), then the model of univalent type theory $(\mathcal{M}_{\mathrm{fib}})_P$ is also non-trivial.*

8.5.14. Corollary. *If P is well-supported in $\mathcal{M}$, then the type $\mathcal{M}_{\mathrm{fib}}(\mathtt{N})$ has a P-null structure.*

8.5.15. Corollary. *If P is well-supported in $\mathcal{M}$, then P-null types are closed under binary coproducts.*

Proof:
Let B_0 and B_1 be two P-null types in $\mathcal{M}_{\mathrm{fib}}$. Then the coproduct $B_0 + B_1$ is equivalent to $\sum_{\mathtt{x}:2} B(\mathtt{x})$, where B is defined by $B(0) \equiv B_0$ and $B(1) \equiv B_1$, and is P-null by Propositions 7.1.11, 8.5.6 and 8.5.12. □

8.5.16. Corollary. *If $\mathcal{M}$ models a countable chain of universes and P is well-supported in $\mathcal{M}$, then $(\mathcal{M}_{\mathrm{fib}})_P$ is a model of univalent type theory. Moreover, the forgetful map $(\mathcal{M}_{\mathrm{fib}})_P \to \mathcal{M}_{\mathrm{fib}}$ preserves Π-types, Σ-types, unit type, identity types, finite coproducts, and natural numbers type.*

To construct a P-nullification operator, we assume that $\mathcal{M}$ models W-types with reductions with respect to cofibrations. Rijke, Shulman, and Spitters [146] gave a general construction of a localization operator. Recall that a type B is P-null if the function

$$\lambda\mathtt{y}.\lambda_.\mathtt{y} : B \to (P(\mathtt{x}) \to B) \tag{8.2}$$

an equivalence for any $\mathtt{x} : A$. Therefore, to obtain the P-nullification of B, we have to adjoin inverses of the functions (8.2). We first construct a higher inductive

type $\mathcal{J}_P(B)$ which adjoint right inverses of the functions (8.2).

$$\begin{aligned}
\mathcal{J}_P &: (\mathtt{B} : () \to U) \Rightarrow U \\
\alpha &: ([\mathtt{B} : () \to U], \mathtt{b} : () \to \mathtt{B}) \Rightarrow \mathcal{J}_P(\mathtt{B}) \\
\mathsf{ext} &: ([\mathtt{B} : () \to U], [\mathtt{a} : () \to A], \mathtt{f} : (\mathtt{z} : P(\mathtt{a})) \to \mathcal{J}_P(\mathtt{B})) \Rightarrow \mathcal{J}_P(\mathtt{B}) \\
\mathsf{isext} &: ([\mathtt{B} : () \to U], [\mathtt{a} : () \to A], \mathtt{f} : (\mathtt{z} : P(\mathtt{a})) \to \mathcal{J}_P(\mathtt{B}), \mathtt{c} : () \to P(\mathtt{a}), \mathtt{i} : \mathbb{I}) \\
&\quad \Rightarrow \mathcal{J}_P(\mathtt{B}) \\
_ &: (\mathtt{B} : () \to U, \mathtt{a} : () \to A, \mathtt{f} : (\mathtt{z} : P(\mathtt{a})) \to \mathcal{J}_P(\mathtt{B}), \mathtt{c} : () \to P(\mathtt{a})) \\
&\quad \Rightarrow \mathsf{isext}(\mathtt{f}, \mathtt{c}, 0_{\mathbb{I}}) \equiv \mathsf{ext}(\mathtt{f}) : \mathcal{J}_P(\mathtt{B}) \\
_ &: (\mathtt{B} : () \to U, \mathtt{a} : () \to A, \mathtt{f} : (\mathtt{z} : P(\mathtt{a})) \to \mathcal{J}_P(\mathtt{B}), \mathtt{c} : () \to P(\mathtt{a})) \\
&\quad \Rightarrow \mathsf{isext}(\mathtt{f}, \mathtt{c}, 1_{\mathbb{I}}) \equiv \mathtt{f}(\mathtt{c}) : \mathcal{J}_P(\mathtt{B})
\end{aligned}$$

Then the P-nullification operator is defined to be $\mathcal{J}_{\hat{P}}$ where $\hat{P} : \mathcal{M}_{\text{fib}}(\diamond)/A + A \to \mathcal{M}_{\text{fib}}(U)$ is a type defined as follows:

- $\hat{P}(\mathsf{inl}(\mathtt{x})) \simeq P(\mathtt{x})$;
- $\hat{P}(\mathsf{inr}(\mathtt{x}))$ is the following pushout in $\mathcal{M}_{\text{fib}}$.

$$\begin{array}{ccc}
P(\mathtt{x}) & \longrightarrow & 1 \\
\downarrow & \ulcorner & \downarrow \\
1 & \longrightarrow & \hat{P}(\mathsf{inr}(\mathtt{x}))
\end{array} \tag{8.3}$$

See [146, Section 2.2] for details. Since the pushout (8.3) is equivalent to the suspension $\mathsf{Susp}(P(\mathtt{x}))$ which is constructed in Example 7.1.5, all we have to do is to construct the higher inductive type $\mathcal{J}_P$.

8.5.17. Definition. For a type $\mathtt{A}$ in $\mathcal{M}$, we define $\mathsf{Cone}(\mathtt{A})$ to be the following pushout in $\mathcal{M}$.

$$\begin{array}{ccc}
\mathtt{A} & \longrightarrow & 1 \\
{\scriptstyle (\mathrm{id}_{\mathtt{A}}, 0_{\mathbb{I}})}\downarrow & \ulcorner & \downarrow{\scriptstyle \mathsf{inl}} \\
\mathtt{A} \times \mathbb{I} & \xrightarrow[\mathsf{inr}]{} & \mathsf{Cone}(\mathtt{A})
\end{array}$$

Let B be a type in $\mathcal{M}_{\text{fib}}$. We define $\mathcal{J}_P(B)$ to be the W-type with reductions for the sum of the following three polynomials with reductions $(\mathtt{A}_i, \mathtt{B}_i, \mathtt{P}_i, \mathtt{f}_i)$.

- $(\mathtt{A}_0, \mathtt{B}_0, \mathtt{P}_0, \mathtt{f}_0)$ is as in Definition 7.1.4. This part adds to $\mathcal{J}_P(B)$ a homogeneous composition structure.
- $\mathtt{A}_1 \equiv B$, $\mathtt{B}_1(\mathtt{y}) \equiv \mathsf{0}$, and $\mathtt{P}_1(\mathtt{y}) \equiv \bot$. This part adds to $\mathcal{J}_P(B)$ the constructor $\alpha : B \texttt{ -> } \mathcal{J}_P(B)$.

- $\mathtt{A}_2 \equiv \sum_{\mathtt{x}:A} \mathsf{Cone}(P(\mathtt{x}))$ and $\mathtt{B}_2(\mathtt{x},\mathtt{z}) \equiv P(\mathtt{x})$. We define $\mathtt{P}_2 : \mathtt{A}_2 \to \mathsf{Cof}$ by $\mathtt{P}_2(\mathtt{x},\mathsf{inl}(*)) \equiv \bot$ and $\mathtt{P}_2(\mathtt{x},\mathsf{inr}(\mathtt{z},\mathtt{i})) \equiv (\mathtt{i} == 1_{\mathbb{I}})$ which is well-defined since $\bot == (0_{\mathbb{I}} == 1_{\mathbb{I}})$ by propositional extensionality. We define $\mathtt{f}_2 : (\mathtt{x} : \mathtt{A}_2) \to \mathtt{P}_2(\mathtt{x}) \to \mathtt{B}_2(\mathtt{x})$ by $\mathtt{f}_2(\mathtt{x},\mathsf{inr}(\mathtt{z},1_{\mathbb{I}})) \equiv \mathtt{z}$.

From the last part we have a constructor

$$\mathsf{sup}_2 : (\mathtt{a} : () \to \mathtt{A}_2, \mathtt{d} : (\mathtt{z} : \mathtt{B}_2(\mathtt{a})) \to \mathcal{J}_P(B)) \Rightarrow \mathcal{J}_P(B)$$

which splits into two parts

$$\mathsf{sup}_{\mathsf{inl}} : ([\mathtt{a} : () \to A], \mathtt{d} : (\mathtt{z} : P(\mathtt{x})) \to \mathcal{J}_P(B)) \Rightarrow \mathcal{J}_P(B)$$
$$\mathsf{sup}_{\mathsf{inr}} : ([\mathtt{a} : () \to A], \mathtt{c} : () \to P(\mathtt{a}), \mathtt{i} : () \to \mathbb{I}, \mathtt{d} : (\mathtt{z} : P(\mathtt{x})) \to \mathcal{J}_P(B)) \Rightarrow \mathcal{J}_P(B)$$

satisfying

$$\begin{aligned} _ : (\mathtt{a} : () \to A, \mathtt{c} : () \to P(\mathtt{a}), \mathtt{d} : (\mathtt{z} : P(\mathtt{x})) \to \mathcal{J}_P(B)) \\ \Rightarrow \mathsf{sup}_{\mathsf{inl}}(\mathtt{d}) \equiv \mathsf{sup}_{\mathsf{inr}}(\mathtt{c}, 0_{\mathbb{I}}, \mathtt{d}) : \mathcal{J}_P(B). \end{aligned}$$

The reduction is then given by

$$\begin{aligned} _ : (\mathtt{a} : () \to A, \mathtt{c} : () \to P(\mathtt{a}), \mathtt{d} : (\mathtt{z} : P(\mathtt{x})) \to \mathcal{J}_P(B)) \\ \Rightarrow \mathsf{sup}_{\mathsf{inr}}(\mathtt{c}, 1_{\mathbb{I}}, \mathtt{d}) \equiv \mathtt{d}(\mathtt{c}) : \mathcal{J}_P(B). \end{aligned}$$

Thus, $\mathsf{sup}_{\mathsf{inl}}$ and $\mathsf{sup}_{\mathsf{inr}}$ define the constructors ext and isext, respectively.

By construction, $\mathcal{J}_P$ is the inductive type with the required constructors and a homogeneous composition structure. It remains to define a transport structure. Suppose that A, P, and B are parameterized by $\mathtt{i} : \mathbb{I}$ and constant over a cofibration $\mathtt{Q}$. Let $D(\mathtt{i}) = \mathcal{J}_{P(\mathtt{i})}(B(\mathtt{i}))$. For an element $\mathtt{d}_0 : D(0_{\mathbb{I}})$, the transport

$$\mathsf{transp}(D, \mathtt{Q}, \mathtt{d}_0) : D(1_{\mathbb{I}})$$

is defined by induction on $\mathtt{d}_0$. Recall that the constructors for $\mathcal{J}_P$ are hcomp, α, ext, and isext. The first two cases are straightforward. Suppose that $\mathtt{d}_0 \equiv \mathsf{ext}(\mathtt{f}_0)$ for $\mathtt{a}_0 : A(0_{\mathbb{I}})$ and $\mathtt{f}_0 : P(0_{\mathbb{I}}, \mathtt{a}_0) \to D(0_{\mathbb{I}})$. By the Kan filling operation [41, Section 4.4], we have a path $\mathtt{a} : (\mathtt{i} : \mathbb{I}) \to A(\mathtt{i})$ such that $\mathtt{a}(0_{\mathbb{I}}) \equiv \mathtt{a}_0$ and $(_ : \mathtt{Q}, \mathtt{i} : \mathbb{I}) \to \mathtt{a}(\mathtt{i}) \equiv \mathtt{a}_0$. We then define

$$\mathsf{transp}(D, \mathtt{Q}, \mathsf{ext}(\mathtt{f}_0)) \equiv \mathsf{ext}(\mathtt{f}_1)$$

where

$$\begin{aligned} \mathtt{f}_1 : (\mathtt{z} : P(1_{\mathbb{I}}, \mathtt{a}(1_{\mathbb{I}}))) \to D(1_{\mathbb{I}}) \\ \mathtt{f}_1(\mathtt{z}) \equiv \mathsf{transp}(D, \mathtt{Q}, \mathtt{f}_0(\mathsf{transp}(\langle \mathtt{i} \rangle P(1 - \mathtt{i}, \mathtt{a}(1 - \mathtt{i})), \mathtt{Q}, \mathtt{z}))) \end{aligned}$$

The case when $\mathtt{d}_0 \equiv \mathsf{isext}(\mathtt{f}_0, \mathtt{c}_0, \mathtt{j})$ for $\mathtt{a}_0 : A(0_\mathbb{I})$, $\mathtt{f}_0 : P(0_\mathbb{I}, \mathtt{a}_0) \to D(0_\mathbb{I})$, $\mathtt{c}_0 : P(0_\mathbb{I}, \mathtt{a}_0)$, and $\mathtt{j} : \mathbb{I}$ is not immediate. The first attempt is to define $\mathsf{transp}(D, \mathtt{Q}, \mathsf{ext}(\mathtt{f}_0, \mathtt{c}_0, \mathtt{j}))$ to be $\mathsf{isext}(\mathtt{f}_1, \mathtt{c}_1, \mathtt{j})$ where $\mathtt{f}_1$ is defined in the same way as the case of ext and $\mathtt{c}_1 \equiv \mathsf{transp}(\langle \mathtt{i} \rangle P(\mathtt{i}, \mathtt{a}(\mathtt{i})), \mathtt{Q}, \mathtt{c}_0)$. However, $\mathsf{isext}(\mathtt{f}_1, \mathtt{c}_1, \mathtt{j})$ does not satisfy $\mathsf{isext}(\mathtt{f}_1, \mathtt{c}_1, 1_\mathbb{I}) \equiv \mathsf{transp}(D, \mathtt{Q}, \mathtt{f}_0(\mathtt{c}_0))$ which is one of the boundary conditions for the recursion principle for the higher inductive type $D(0_\mathbb{I}) = \mathcal{J}_{P(0_\mathbb{I})}(B(0_\mathbb{I}))$. The problem is that $\mathsf{transp}(\langle \mathtt{i} \rangle P(1 - \mathtt{i}, \mathtt{a}(1 - \mathtt{i})))$ is only a homotopy inverse to $\mathsf{transp}(\langle \mathtt{i} \rangle P(\mathtt{i}, \mathtt{a}(\mathtt{i})))$. We fix this using the hcomp constructor, following the construction of pushouts in [44, Section 2.3]. We define a cofibration $\mathtt{Q}'$ to be $\mathtt{Q} \vee (\mathtt{j} == 0_\mathbb{I}) \vee (\mathtt{j} == 1_\mathbb{I})$ and then define

$$\begin{aligned} \mathtt{d}' &: (_ : \mathtt{Q}', \mathtt{i} : \mathbb{I}) \to D(1_\mathbb{I}) \\ \mathtt{d}'_0 &: () \to D(1_\mathbb{I}) \end{aligned}$$

by

$$\begin{aligned} \mathtt{d}'_0 &\equiv \mathsf{isext}(\mathtt{f}_1, \mathtt{c}_1, \mathtt{j}) \\ \mathtt{d}'(\mathtt{i}) &\equiv \mathsf{isext}(\mathtt{f}_0, \mathtt{c}_0, \mathtt{j}) && (\text{if } \mathtt{Q}) \\ \mathtt{d}'(\mathtt{i}) &\equiv \mathsf{ext}(\mathtt{f}_1) && (\text{if } \mathtt{j} == 0_\mathbb{I}) \\ \mathtt{d}'(\mathtt{i}) &\equiv \mathsf{transp}(D, \mathtt{Q}, \mathtt{f}_0(\mathtt{p}(\mathtt{i}))) && (\text{if } \mathtt{j} == 1_\mathbb{I}) \end{aligned}$$

where $\mathtt{p}$ is a canonical path from $\mathsf{transp}(\langle \mathtt{i} \rangle P(1 - \mathtt{i}, \mathtt{a}(1 - \mathtt{i})), \mathtt{Q}, \mathsf{transp}(\langle \mathtt{i} \rangle P(\mathtt{i}, \mathtt{a}(\mathtt{i})), \mathtt{Q}, \mathtt{c}_0))$ to $\mathtt{c}_0$ constant over $\mathtt{Q}$. We then define

$$\mathsf{transp}(D, \mathtt{Q}, \mathsf{isext}(\mathtt{f}_0, \mathtt{c}_0, \mathtt{j})) \equiv \mathsf{hcomp}(D(1_\mathbb{I}), \mathtt{Q}', \mathtt{d}', \mathtt{d}'_0).$$

Since $\mathtt{p}(1_\mathbb{I}) \equiv \mathtt{c}_0$, we have the required boundary condition

$$\mathsf{hcomp}(D(1_\mathbb{I}), \mathtt{Q}', \mathtt{d}', \mathtt{d}'_0) \equiv \mathsf{transp}(D, \mathtt{Q}, \mathtt{f}_0(\mathtt{c}_0))$$

when $\mathtt{j} == 1_\mathbb{I}$.

8.5.3 Church's Thesis in null types

Recall that Church's Thesis is a proposition of the form

$$\forall_{\mathtt{x}:\mathtt{A}} \mathsf{Trunc}(\mathtt{B}(\mathtt{x}))$$

where $\mathtt{A}$ is a closed type and $\mathtt{B}$ is a type over $\mathtt{A}$, both defined only using Π-types, Σ-types, unit type, identity types, finite coproducts, and natural numbers type.

8.5.18. THEOREM. *Let* $\mathtt{A}$ *be a closed type and* $\mathtt{B}$ *a type over* $\mathtt{A}$ *both definable only using* Π*-types,* Σ*-types, unit type, identity types, finite coproducts, and natural numbers type. For a model* $\mathcal{M}$ *of qCTT with* W*-types with reductions and a countable chain of universes, if* $\mathcal{M}(\mathtt{B})$ *is well-supported, then the proposition*

$$\forall_{\mathtt{x}:\mathtt{A}} \mathsf{Trunc}(\mathtt{B}(\mathtt{x}))$$

holds in the model of univalent type theory $(\mathcal{M}_{\mathrm{fib}})_P$ *where* $P = \mathsf{Trunc}(\mathcal{M}_{\mathrm{fib}}(\mathtt{B}))$.

Proof:
By Proposition 7.1.11, $\mathtt{A}$ and $\mathtt{B}$ are discrete and the forgetful map $\mathcal{M}_{\mathrm{fib}} \to \mathcal{M}$ preserves the interpretation of $\mathtt{A}$ and $\mathtt{B}$ including the interpretation of identity types which are interpreted in $\mathcal{M}_{\mathrm{fib}}$ as path types and interpreted in $\mathcal{M}$ as extensional identity types. Since $\mathcal{M}(\mathtt{B})$ is well-supported by assumption, the truncation $P = \mathsf{Trunc}(\mathcal{M}_{\mathrm{fib}}(\mathtt{B}))$ is well-supported in $\mathcal{M}$. Then $\mathcal{M}_{\mathrm{fib}}(\mathtt{A})$ and $\mathcal{M}_{\mathrm{fib}}(\mathtt{B})$ are P-null types by Proposition 8.5.12. The proposition $\forall_{\mathtt{x:A}}\mathsf{Trunc}(\mathtt{B}(\mathtt{x}))$ is then interpreted in $(\mathcal{M}_{\mathrm{fib}})_P$ as

$$\forall_{\mathtt{x}:\mathcal{M}_{\mathrm{fib}}(\mathtt{A})}\mathcal{L}(\mathsf{Trunc}(\mathcal{M}_{\mathrm{fib}}(\mathtt{B})(\mathtt{x})))$$

by Corollary 8.5.10, where $\mathcal{L}$ is a P-nullification operator. This type is equal to $\forall_{\mathtt{x}:\mathcal{M}_{\mathrm{fib}}(\mathtt{A})}\mathcal{L}(P(\mathtt{x}))$ and is inhabited by Corollary 8.5.9. □

8.5.19. Corollary. *Let* $\mathtt{A}$ *and* $\mathtt{B}$ *be as in Theorem 8.5.18. For a model* $\mathcal{M}$ *of the good type theory with* W*-types and a countable chain of universes, if* $\forall_{\mathtt{x:A}}\mathsf{Trunc}(\mathtt{B}(\mathtt{x}))$ *holds in* $\mathcal{M}$*, then it also holds in the model of univalent type theory* $\big(\big(\mathbf{PSh}_{\mathbf{B}}^{\mathcal{M}}\big)_{\mathrm{fib}}\big)_P$ *where* $P = \mathsf{Trunc}\big(\big(\mathbf{PSh}_{\mathbf{B}}^{\mathcal{M}}\big)_{\mathrm{fib}}(\mathtt{B})\big)$.

Proof:
The assumption that $\forall_{\mathtt{x:A}}\mathsf{Trunc}(\mathtt{B}(\mathtt{x}))$ holds in $\mathcal{M}$ is equivalent to that $\mathcal{M}(\mathtt{B})$ is well-supported. $\mathtt{B}$ is interpreted in $\mathbf{PSh}_{\mathbf{B}}^{\mathcal{M}}$ as the constant presheaf at $\mathcal{M}(\mathtt{B})$, and thus $\mathbf{PSh}_{\mathbf{B}}^{\mathcal{M}}(\mathtt{B})$ is also well-supported. Then apply Theorem 8.5.18. □

8.5.20. Example. We can apply Corollary 8.5.19 for Church's Thesis

$$\prod_{\mathtt{f:N->N}} \mathsf{Trunc}\Big(\sum_{\mathtt{e:N}} \prod_{\mathtt{n:N}} \sum_{\mathtt{m:N}} \mathrm{T}(\mathtt{e},\mathtt{n},\mathtt{m}) \times \mathrm{U}(\mathtt{m}) == \mathtt{f}(\mathtt{n})\Big).$$

Since the model of the good type theory in assemblies for Kleene's first model $\mathcal{K}_1$ satisfies Church's Thesis, we obtain a model of univalent type theory satisfying Church's Thesis.

We now prove that univalent type theory is consistent with the main principles of Recursive Constructive Mathematics.

8.5.21. Theorem. *Martin-Löf type theory remains consistent when all of the following extra structure and axioms are added.*

1. *Propositional truncation.*
2. *The axiom of univalence.*
3. *Church's Thesis.*

4. Markov's Principle.

Proof:
By constructing a non-trivial model. We begin with the model **CAsm** of CTT in cubical assemblies for Kleene's first model $\mathcal{K}_1$. Let $\mathbf{CAsm}'$ be the model in null types of univalent type theory satisfying Church's Thesis obtained in Example 8.5.20. Since **CAsm** satisfies Markov's Principle by Corollary 8.4.2 and since null types are closed under the type constructors used in the definition of Markov's Principle by Corollary 8.5.16, $\mathbf{CAsm}'$ also satisfies Markov's Principle. Since **CAsm** is non-trivial, so is $\mathbf{CAsm}'$ by Corollary 8.5.13. □

We can use Theorem 8.5.18 and Corollary 8.5.19 for other principles.

8.5.22. Example. *Brouwer's Principle* is written as follows.

$$\forall_{\mathtt{F:(N->N)->N}}\forall_{\mathtt{f:N->N}}\exists_{\mathtt{n:N}}\forall_{\mathtt{g:N->N}}(\forall_{\mathtt{m:N}}\mathtt{m} < \mathtt{n} \mathtt{\ ->\ } \mathtt{f(m)} == \mathtt{g(m)}) \mathtt{\ ->\ } \mathtt{F(f)} == \mathtt{F(g)}$$

Thus, by Corollary 8.5.19, we can construct a model of univalent type theory satisfying Brouwer's Principle out of a model of the good type theory satisfying Brouwer's Principle.

It is known that Brouwer's Principle holds in the effective topos [176, Proposition 3.1.6] and thus in assemblies for Kleene's first model $\mathcal{K}_1$. We thus obtain a new proof of the following result originally proved by Coquand using cubical stacks [42].

8.5.23. Theorem. *Martin-Löf type theory remains consistent when all of the following extra structure and axioms are added.*

1. *Propositional truncation.*
2. *The axiom of univalence.*
3. *Brouwer's Principle.*

We can moreover combine those principles.

8.5.24. Example. Let $(\mathtt{A}_1, \mathtt{B}_1)$ and $(\mathtt{A}_2, \mathtt{B}_2)$ be two types as in Theorem 8.5.18. Let $\mathtt{A} \equiv \mathtt{A}_1 + \mathtt{A}_2$ and define $\mathtt{B}$ by $\mathtt{B}(\mathsf{inl}(\mathtt{x}_1)) \equiv \mathtt{B}_1(\mathtt{x}_1)$ and $\mathtt{B}(\mathsf{inr}(\mathtt{x}_2)) \equiv \mathtt{B}_2(\mathtt{x}_2)$. Then we have

$$\forall_{\mathtt{x:A}}\mathsf{Trunc}(\mathtt{B}(\mathtt{x})) \leftrightarrow (\forall_{\mathtt{x}_1:\mathtt{A}_1}\mathsf{Trunc}(\mathtt{B}_1(\mathtt{x}_1))) \wedge (\forall_{\mathtt{x}_2:\mathtt{A}_2}\mathsf{Trunc}(\mathtt{B}_2(\mathtt{x}_2))).$$

Therefore, if a model $\mathcal{M}$ of the good type theory with W-types and a countable chain of universes satisfies both $\forall_{\mathtt{x}_1:\mathtt{A}_1}\mathsf{Trunc}(\mathtt{B}_1(\mathtt{x}_1))$ and $\forall_{\mathtt{x}_2:\mathtt{A}_2}\mathsf{Trunc}(\mathtt{B}_2(\mathtt{x}_2))$, then so does the model $\big(\big(\mathbf{PSh}_{\mathbf{B}}^{\mathcal{M}}\big)_{\mathrm{fib}}\big)_P$ of univalent type theory where $P = \mathsf{Trunc}\big(\big(\mathbf{PSh}_{\mathbf{B}}^{\mathcal{M}}\big)_{\mathrm{fib}}(\mathtt{B})\big)$.

Therefore, any finite set of propositions of the form $\forall_{\mathtt{x:A}}\mathsf{Trunc}(\mathtt{B}(\mathtt{x}))$, where $\mathtt{A}$ and $\mathtt{B}$ are as in Theorem 8.5.18, satisfied by **Asm** is consistent with the univalence axiom. By the compactness argument, we see that the set of all such propositions is consistent with the univalence axiom.

Some important principles of Constructive Recursive Mathematics are not covered by Corollary 8.5.19. The *axiom of countable choice*

$$(\forall_{n:\mathbb{N}}\exists_{a:A}P(n,a)) \to \exists_{f:\mathbb{N}\to A}\forall_{n:\mathbb{N}}P(n,f(n))$$

is valid in the category of assemblies, but it is uncertain if it holds in cubical assemblies. We cannot apply Corollary 8.5.19 for two reasons: A ranges over all types and P ranges over all propositions; the statement is not of the form $\forall_{\mathtt{x:A}}\mathsf{Trunc}(\mathtt{B}(\mathtt{x}))$. *Extended Church's Thesis* is important since it characterizes Kleene realizability over Heyting Arithmetic [167]. It roughly asserts that certain partial functions on natural numbers are computable. A problem here is that it is not clear what a good notion of a partial function is in univalent type theory. Escardó and Knapp [52] studied partial functions in univalent type theory, but a form of countable choice is needed for their definition to work well.

Bibliography

[1] J. Adámek, J. Rosický, and E. M. Vitale. *Algebraic Theories*. Vol. 184. Cambridge Tracts in Mathematics. Cambridge University Press, 2011. DOI: 10.1017/CBO9780511760754.

[2] J. Adámek and J. Rosický. *Locally Presentable and Accessible Categories*. Vol. 189. London Mathematical Society Lecture Note Series. Cambridge University Press, 1994. DOI: 10.1017/CBO9780511600579.

[3] T. Altenkirch, P. Capriotti, and N. Kraus. "Extending Homotopy Type Theory with Strict Equality". In: *25th EACSL Annual Conference on Computer Science Logic (CSL 2016)*. Ed. by J.-M. Talbot and L. Regnier. Vol. 62. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2016, 21:1–21:17. DOI: 10.4230/LIPIcs.CSL.2016.21.

[4] C. Angiuli. "Computational Semantics of Cartesian Cubical Type Theory". PhD thesis. Carnegie Mellon University, 2019. URL: https://www.cs.cmu.edu/~cangiuli/thesis/thesis.pdf.

[5] C. Angiuli, G. Brunerie, T. Coquand, K.-B. Hou (Favonia), R. Harper, and D. R. Licata. *Syntax and Models of Cartesian Cubical Type Theory*. 2019. URL: https://github.com/dlicata335/cart-cube.

[6] C. Angiuli, K.-B. Hou (Favonia), and R. Harper. "Cartesian Cubical Computational Type Theory: Constructive Reasoning with Paths and Equalities". In: *27th EACSL Annual Conference on Computer Science Logic (CSL 2018)*. Ed. by D. Ghica and A. Jung. Vol. 119. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018, 6:1–6:17. DOI: 10.4230/LIPIcs.CSL.2018.6.

[7] D. Annenkov, P. Capriotti, N. Kraus, and C. Sattler. *Two-Level Type Theory and Applications*. 2019. arXiv: 1705.03307v3.

[8] P. Arndt and K. Kapulkin. "Homotopy-Theoretic Models of Type Theory". In: *Typed Lambda Calculi and Applications: 10th International Conference, TLCA 2011, Novi Sad, Serbia, June 1-3, 2011. Proceedings*. Ed. by L. Ong. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 45–60. DOI: 10.1007/978-3-642-21691-6_7.

[9] J. Avigad, K. Kapulkin, and P. L. Lumsdaine. "Homotopy limits in type theory". In: *Mathematical Structures in Computer Science* 25.5 (2015), pp. 1040–1070. DOI: 10.1017/S0960129514000498.

[10] S. Awodey. "A cubical model of homotopy type theory". In: *Annals of Pure and Applied Logic* 169.12 (2018). Logic Colloquium 2015, pp. 1270–1294. DOI: 10.1016/j.apal.2018.08.002.

[11] S. Awodey. *Category Theory*. Oxford Logic Guides. Oxford University Press, 2010.

[12] S. Awodey. "Natural models of homotopy type theory". In: *Mathematical Structures in Computer Science* 28.2 (2018), pp. 241–286. DOI: 10.1017/S0960129516000268.

[13] S. Awodey, J. Frey, and S. Speight. "Impredicative Encodings of (Higher) Inductive Types". In: *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*. LICS '18. Oxford, United Kingdom: ACM, 2018, pp. 76–85. DOI: 10.1145/3209108.3209130.

[14] S. Awodey and C. Newstead. *Polynomial pseudomonads and dependent type theory*. 2018. arXiv: 1802.00997v1.

[15] S. Awodey and M. A. Warren. "Homotopy theoretic models of identity types". In: *Mathematical Proceedings of the Cambridge Philosophical Society* 146.1 (2009), pp. 45–55. DOI: 10.1017/S0305004108001783.

[16] S. Awodey and A. Bauer. "Propositions As [Types]". In: *J. Log. and Comput.* 14.4 (Aug. 2004), pp. 447–471. DOI: 10.1093/logcom/14.4.447.

[17] R. Balbes and P. Dwinger. *Distributive lattices*. University of Missouri Press, Columbia, Mo., 1974, pp. xiii+294.

[18] H. P. Barendregt. "Lambda Calculi with Types". In: *Handbook of Logic in Computer Science (Vol. 2)*. Ed. by S. Abramsky, D. M. Gabbay, and S. E. Maibaum. New York, NY, USA: Oxford University Press, Inc., 1992, pp. 117–309. URL: https://hdl.handle.net/2066/17231.

[19] A. Bauer, J. Gross, P. L. Lumsdaine, M. Shulman, M. Sozeau, and B. Spitters. "The HoTT Library: A Formalization of Homotopy Type Theory in Coq". In: *Proceedings of the 6th ACM SIGPLAN Conference on Certified Programs and Proofs*. CPP 2017. Paris, France: ACM, 2017, pp. 164–172. DOI: 10.1145/3018610.3018615.

[20] A. Bauer, P. G. Haselwarter, and P. L. Lumsdaine. *A general definition of dependent type theories*. 2020. arXiv: 2009.05539v1.

[21] M. J. Beeson. *Foundations of constructive mathematics*. Vol. 6. Ergebnisse der Mathematik und ihrer Grenzgebiete (3) [Results in Mathematics and Related Areas (3)]. Springer-Verlag, Berlin, 1985, pp. xxiii+466. DOI: 10.1007/978-3-642-68952-9.

[22] M. Bezem, T. Coquand, and S. Huber. "A Model of Type Theory in Cubical Sets". In: *19th International Conference on Types for Proofs and Programs (TYPES 2013)*. Ed. by R. Matthes and A. Schubert. Vol. 26. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2014, pp. 107–128. DOI: 10.4230/LIPIcs.TYPES.2013.107.

[23] M. Bezem, T. Coquand, and S. Huber. "The Univalence Axiom in Cubical Sets". In: *Journal of Automated Reasoning* 63.2 (Aug. 2019), pp. 159–171. DOI: 10.1007/s10817-018-9472-6.

[24] M. Bezem, T. Coquand, and E. Parmann. "Non-Constructivity in Kan Simplicial Sets". In: *13th International Conference on Typed Lambda Calculi and Applications (TLCA 2015)*. Ed. by T. Altenkirch. Vol. 38. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2015, pp. 92–106. DOI: 10.4230/LIPIcs.TLCA.2015.92.

[25] M. E. Bidlingmaier. *An interpretation of dependent type theory in a model category of locally cartesian closed categories*. 2020. arXiv: 2007.02900v1.

[26] L. Birkedal, R. Clouston, B. Mannaa, R. Ejlers Møgelberg, A. M. Pitts, and B. Spitters. "Modal dependent type theory and dependent right adjoints". In: *Mathematical Structures in Computer Science* 30.2 (2020), pp. 118–138. DOI: 10.1017/S0960129519000197.

[27] R. Bocquet. *Coherence of strict equalities in dependent type theories*. 2020. arXiv: 2010.14166v1.

[28] J. Bourke. "Accessible aspects of 2-category theory". In: *J. Pure Appl. Algebra* 225.3 (2021), pp. 106519, 43. DOI: 10.1016/j.jpaa.2020.106519.

[29] J. Bourke, S. Lack, and L. Vokřínek. *Adjoint functor theorems for homotopically enriched categories*. 2020. arXiv: 2006.07843v1.

[30] K. S. Brown. "Abstract homotopy theory and generalized sheaf cohomology". In: *Transactions of the American Mathematical Society* 186 (1973), pp. 419–458. DOI: 10.2307/1996573.

[31] G. Brunerie, K.-B. Hou (Favonia), E. Cavallo, J. Cockx, C. Sattler, C. Jeris, M. Shulman, et al. *Homotopy Type Theory in Agda*. URL: https://github.com/HoTT/HoTT-Agda.

[32] M. Buckley. "Fibred 2-categories and bicategories". In: *Journal of Pure and Applied Algebra* 218.6 (2014), pp. 1034–1074. DOI: 10.1016/j.jpaa.2013.11.002.

[33] P. Capriotti. "Models of Type Theory with Strict Equality". PhD thesis. University of Nottingham, 2016. arXiv: 1702.04912v1.

[34] A. Carboni, S. Lack, and R. F. C. Walters. "Introduction to extensive and distributive categories". In: *Journal of Pure and Applied Algebra* 84.2 (1993), pp. 145–158. DOI: 10.1016/0022-4049(93)90035-R.

[35] J. W. Cartmell. "Generalised algebraic theories and contextual categories". PhD thesis. Oxford University, 1978.

[36] E. Cavallo and A. Mörtberg. *A unifying cartesian cubical type theory*. 2019. URL: https://github.com/mortberg/gen-cart/blob/master/unifying-cartesian.pdf.

[37] E. Cavallo, A. Mörtberg, and A. W. Swan. "Unifying Cubical Models of Univalent Type Theory". In: *28th EACSL Annual Conference on Computer Science Logic (CSL 2020)*. Ed. by M. Fernández and A. Muscholl. Vol. 152. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2020, 14:1–14:17. DOI: 10.4230/LIPIcs.CSL.2020.14.

[38] D.-C. Cisinski. *Higher Categories and Homotopical Algebra*. Cambridge Studies in Advanced Mathematics. Cambridge University Press, 2019. DOI: 10.1017/9781108588737.

[39] P. Clairambault and P. Dybjer. "The Biequivalence of Locally Cartesian Closed Categories and Martin-Löf Type Theories". In: *Typed Lambda Calculi and Applications*. Ed. by L. Ong. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 91–106. DOI: 10.1007/978-3-642-21691-6_10.

[40] P. Clairambault and P. Dybjer. "The biequivalence of locally cartesian closed categories and Martin-Löf type theories". In: *Mathematical Structures in Computer Science* 24.6 (2014), e240606. DOI: 10.1017/S0960129513000881.

[41] C. Cohen, T. Coquand, S. Huber, and A. Mörtberg. "Cubical Type Theory: A Constructive Interpretation of the Univalence Axiom". In: *21st International Conference on Types for Proofs and Programs (TYPES 2015)*. Ed. by T. Uustalu. Vol. 69. Leibniz International Proceedings

in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018, 5:1–5:34. DOI: 10.4230/LIPIcs.TYPES.2015.5.

[42] T. Coquand. *Cubical stacks*. 2018. URL: http://www.cse.chalmers.se/~coquand/stack.pdf.

[43] T. Coquand. *Internal version of the uniform Kan filling condition*. 2015. URL: http://www.cse.chalmers.se/~coquand/shape.pdf.

[44] T. Coquand, S. Huber, and A. Mörtberg. "On Higher Inductive Types in Cubical Type Theory". In: *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*. LICS '18. Oxford, United Kingdom: ACM, 2018, pp. 255–264. DOI: 10.1145/3209108.3209197.

[45] T. Coquand and G. Huet. "The Calculus of Constructions". In: *Information and Computation* 76.2 (1988), pp. 95–120. DOI: 10.1016/0890-5401(88)90005-3.

[46] R. L. Crole. *Categories for Types*. Cambridge University Press, 1994. DOI: 10.1017/CBO9781139172707.

[47] P.-L. Curien. "Substitution up to Isomorphism". In: *Fundam. Inform.* 19.1/2 (1993), pp. 51–85.

[48] P.-L. Curien, R. Garner, and M. Hofmann. "Revisiting the categorical interpretation of dependent type theory". In: *Theoretical Computer Science* 546 (2014), pp. 99–119. DOI: 10.1016/j.tcs.2014.03.003.

[49] T. de Jong and M. H. Escardó. *Predicative Aspects of Order Theory in Univalent Foundations*. 2021. arXiv: 2102.08812v3.

[50] J. W. Duskin. "Simplicial matrices and the nerves of weak n-categories. I. Nerves of bicategories". In: *Theory Appl. Categ.* 9 (2002), pp. 198–308. URL: http://www.tac.mta.ca/tac/volumes/9/n10/9-10abs.html.

[51] P. Dybjer. "Internal Type Theory". In: *Types for Proofs and Programs: International Workshop, TYPES '95 Torino, Italy, June 5–8, 1995 Selected Papers*. Ed. by S. Berardi and M. Coppo. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996, pp. 120–134. DOI: 10.1007/3-540-61780-9_66.

[52] M. H. Escardó and C. M. Knapp. "Partial Elements and Recursion via Dominances in Univalent Type Theory". In: *26th EACSL Annual Conference on Computer Science Logic (CSL 2017)*. Ed. by V. Goranko and M. Dam. Vol. 82. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017, 21:1–21:16. DOI: 10.4230/LIPIcs.CSL.2017.21.

[53] A. Felty. "Encoding the calculus of constructions in a higher-order logic". In: *[1993] Proceedings Eighth Annual IEEE Symposium on Logic in Computer Science*. 1993, pp. 233–244. DOI: 10.1109/LICS.1993.287584.

[54] M. Fiore. *Discrete Generalised Polynomial Functors*. Talk at ICALP 2012. 2012. URL: http://www.cl.cam.ac.uk/~mpf23/talks/ICALP2012.pdf.

[55] M. Fiore and C.-K. Hur. "Second-Order Equational Logic (Extended Abstract)". In: *Computer Science Logic*. Ed. by A. Dawar and H. Veith. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 320–335. DOI: 10.1007/978-3-642-15205-4_26.

[56] M. Fiore and O. Mahmoud. *Functorial Semantics of Second-Order Algebraic Theories*. 2014. arXiv: 1401.4697v1.

[57] M. Fiore and O. Mahmoud. "Second-Order Algebraic Theories". In: *Mathematical Foundations of Computer Science 2010*. Ed. by P. Hliněný and A. Kučera. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 368–380. DOI: 10.1007/978-3-642-15155-2_33.

[58] P. Freyd. *Abelian categories*. Reprint available at http://www.emis.de/journals/TAC/reprints/articles/3/tr3abs.html. Harper and Row, 1964.

[59] P. Freyd. "Aspects of topoi". In: *Bulletin of the Australian Mathematical Society* 7.01 (Aug. 1972), pp. 1–76. DOI: 10.1017/S0004972700044828.

[60] D. Frumin and B. van den Berg. "A homotopy-theoretic model of function extensionality in the effective topos". In: *Mathematical Structures in Computer Science* (2018), pp. 1–27. DOI: 10.1017/S0960129518000142.

[61] Y. Fu. "Categorical properties of logical frameworks". In: *Mathematical Structures in Computer Science* 7.1 (1997), pp. 1–47. DOI: 10.1017/S0960129596002058.

[62] P. Gabriel and F. Ulmer. *Lokal präsentierbare Kategorien*. Vol. 221. Lecture Notes in Mathematics. Springer-Verlag, Berlin, Heidelberg, 1971. DOI: 10.1007/BFb0059396.

[63] N. Gambino and S. Henry. *Towards a constructive simplicial model of Univalent Foundations*. 2021. arXiv: 1905.06281v3.

[64] N. Gambino, S. Henry, C. Sattler, and K. Szumiło. *The effective model structure and ∞-groupoid objects*. 2021. arXiv: 2102.06146v2.

[65] N. Gambino and J. Kock. "Polynomial functors and polynomial monads". In: *Mathematical Proceedings of the Cambridge Philosophical Society* 154.1 (2013), pp. 153–192. DOI: 10.1017/S0305004112000394.

[66] N. Gambino and C. Sattler. "The Frobenius condition, right properness, and uniform fibrations". In: *Journal of Pure and Applied Algebra* 221.12 (2017), pp. 3027–3068. DOI: 10.1016/j.jpaa.2017.02.013.

[67] N. Gambino, C. Sattler, and K. Szumiło. *The constructive Kan-Quillen model structure: two new proofs*. 2019. arXiv: 1907.05394v1.

[68] R. Garner. "Combinatorial structure of type dependency". In: *Journal of Pure and Applied Algebra* 219.6 (2015), pp. 1885–1914. DOI: 10.1016/j.jpaa.2014.07.015.

[69] D. Gepner and J. Kock. "Univalence in locally cartesian closed ∞-categories". In: *Forum Mathematicum* 29.3 (Jan. 2017). DOI: 10.1515/forum-2015-0228.

[70] H. Geuvers. "Induction Is Not Derivable in Second Order Dependent Type Theory". In: *Typed Lambda Calculi and Applications*. Ed. by S. Abramsky. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 166–181. DOI: 10.1007/3-540-45413-6_16.

[71] J.-Y. Girard. "Linear logic". In: *Theoretical Computer Science* 50.1 (1987), pp. 1–101. DOI: 10.1016/0304-3975(87)90045-4.

[72] J.-Y. Girard. "Une Extension De L'Interpretation De Gödel a L'Analyse, Et Son Application a L'Elimination Des Coupures Dans L'Analyse Et La Theorie Des Types". In: *Proceedings of the Second Scandinavian Logic Symposium*. Ed. by J. Fenstad. Vol. 63. Studies in Logic and the Foundations of Mathematics. Elsevier, 1971, pp. 63–92. DOI: 10.1016/S0049-237X(08)70843-7.

[73] J.-Y. Girard, Y. Lafont, and P. Taylor. *Proofs and Types*. Vol. 7. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1989. URL: http://www.paultaylor.eu/stable/Proofs+Types.html.

[74] D. Gratzer and J. Sterling. *Syntactic categories for dependent type theory: sketching and adequacy*. 2021. arXiv: 2012.10783v2.

[75] R. Harper, F. Honsell, and G. Plotkin. "A Framework for Defining Logics". In: *J. ACM* 40.1 (Jan. 1993), pp. 143–184. DOI: 10.1145/138027.138060.

[76] S. Henry. *A constructive account of the Kan-Quillen model structure and of Kan's Ex^∞ functor*. 2019. arXiv: 1905.06160v1.

[77] C. Hermida. "Fibrations, Logical Predicates and Indeterminates". PhD thesis. University of Edinburgh, 1993. URL: https://www.lfcs.inf.ed.ac.uk/reports/93/ECS-LFCS-93-277/.

[78] M. Hofmann. "On the interpretation of type theory in locally cartesian closed categories". In: *Computer Science Logic*. Ed. by L. Pacholski and J. Tiuryn. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995, pp. 427–441. DOI: 10.1007/BFb0022273.

[79] M. Hofmann. "Syntax and Semantics of Dependent Types". In: *Semantics and Logics of Computation*. Ed. by A. M. Pitts and P. Dybjer. Publications of the Newton Institute. Cambridge University Press, 1997, pp. 79–130. DOI: 10.1017/CBO9780511526619.004.

[80] M. Hofmann and T. Streicher. *Lifting Grothendieck Universes*. 1997. URL: http://www.mathematik.tu-darmstadt.de/~streicher/NOTES/lift.pdf.

[81] M. Hofmann and T. Streicher. "The groupoid interpretation of type theory". In: *Twenty-five years of constructive type theory (Venice, 1995)*. Vol. 36. Oxford Logic Guides. New York: Oxford Univ. Press, 1998, pp. 83–111. DOI: 10.1093/oso/9780198501275.003.0008.

[82] M. Hovey. *Model categories*. Vol. 63. Mathematical Surveys and Monographs. American Mathematical Society, Providence, RI, 1999, pp. xii+209.

[83] S. Huber. "Canonicity for Cubical Type Theory". In: *J. Autom. Reason.* 63.2 (2019), pp. 173–210. DOI: 10.1007/s10817-018-9469-1.

[84] J. M. E. Hyland. "A small complete category". In: *Annals of Pure and Applied Logic* 40.2 (1988), pp. 135–165. DOI: 10.1016/0168-0072(88)90018-8.

[85] J. M. E. Hyland. "The Effective Topos". In: *The L. E. J. Brouwer Centenary Symposium*. Ed. by A. Troelstra and D. van Dalen. Vol. 110. Studies in Logic and the Foundations of Mathematics. Elsevier, 1982, pp. 165–216. DOI: 10.1016/S0049-237X(09)70129-6.

[86] *Initiality Project*. URL: https://ncatlab.org/nlab/show/Initiality+Project.

[87] V. Isaev. *Algebraic Presentations of Dependent Type Theories*. 2018. arXiv: 1602.08504v3.

[88] H. Ishihara, M. E. Maietti, S. Maschio, and T. Streicher. "Consistency of the intensional level of the Minimalist Foundation with Church's thesis and axiom of choice". In: *Archive for Mathematical Logic* (Jan. 2018). DOI: 10.1007/s00153-018-0612-9.

[89] B. Jacobs. *Categorical Logic and Type Theory*. 1st. Elsevier Science, 1999.

[90] B. Jacobs. "Comprehension categories and the semantics of type dependency". In: *Theoretical Computer Science* 107.2 (1993), pp. 169–207. DOI: 10.1016/0304-3975(93)90169-T.

[91] B. Jacobs and T. Melham. "Translating dependent type theory into higher order logic". In: *Typed Lambda Calculi and Applications*. Ed. by M. Bezem and J. F. Groote. Berlin, Heidelberg: Springer Berlin Heidelberg, 1993, pp. 209–229. DOI: 10.1007/BFb0037108.

[92] P. T. Johnstone. *Sketches of an Elephant : A Topos Theory Compendium Volume 1*. Vol. 43. Oxford Logic Guides. Oxford University Press, 2002.

[93] P. T. Johnstone. *Sketches of an Elephant : A Topos Theory Compendium Volume 2*. Vol. 44. Oxford Logic Guides. Oxford University Press, 2002.

[94] A. Joyal. *Notes on clans and tribes*. 2017. arXiv: 1710.10238v1.

[95] A. Joyal. *Notes on quasi-categories*. 2008. URL: http://www.math.uchicago.edu/~may/IMA/Joyal.pdf.

[96] A. Joyal. *What is an elementary higher topos?* Talk at Reimagining The Foundations Of Algebraic Topology, April, 2014. 2014. URL: https://www.msri.org/workshops/689/schedules/18227.

[97] A. Kaposi and A. Kovács. "A Syntax for Higher Inductive-Inductive Types". In: *3rd International Conference on Formal Structures for Computation and Deduction (FSCD 2018)*. Ed. by H. Kirchner. Vol. 108. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018, 20:1–20:18. DOI: 10.4230/LIPIcs.FSCD.2018.20.

[98] A. Kaposi and A. Kovács. "Signatures and Induction Principles for Higher Inductive-Inductive Types". In: *Log. Methods Comput. Sci.* 16.1 (2020). DOI: 10.23638/LMCS-16(1:10)2020.

[99] K. Kapulkin. "Locally cartesian closed quasi-categories from type theory". In: *Journal of Topology* 10.4 (2017), pp. 1029–1049. DOI: 10.1112/topo.12031.

[100] K. Kapulkin and P. L. Lumsdaine. "The homotopy theory of type theories". In: *Adv. Math.* 337 (2018), pp. 1–38. DOI: 10.1016/j.aim.2018.08.003.

[101] K. Kapulkin and P. L. Lumsdaine. *The law of excluded middle in the simplicial model of type theory*. 2020. arXiv: 2006.13694v2.

[102] K. Kapulkin and P. L. Lumsdaine. "The simplicial model of Univalent Foundations (after Voevodsky)". In: *Journal of the European Mathematical Society* (2021). DOI: 10.4171/JEMS/1050.

[103] K. Kapulkin and K. Szumiło. "Internal languages of finitely complete $(\infty, 1)$-categories". In: *Selecta Math. (N.S.)* 25.2 (2019), Art. 33, 46. DOI: 10.1007/s00029-019-0480-0.

[104] L. H. Kauffman. "De Morgan algebras—completeness and recursion". In: *Proceedings of the Eighth International Symposium on Multiple-Valued Logic (Rosemont, Ill., 1978)*. IEEE, Long Beach, Calif., 1978, pp. 82–86.

[105] S. C. Kleene. "On the Interpretation of Intuitionistic Number Theory". In: *The Journal of Symbolic Logic* 10.4 (1945), pp. 109–124. DOI: 10.2307/2269016.

[106] J. Lambek and P. J. Scott. *Introduction to Higher Order Categorical Logic*. Cambridge University Press, 1986.

[107] M. Lambert. *Discrete 2-Fibrations*. 2020. arXiv: 2001.11477v1.

[108] F. W. Lawvere. "Functorial Semantics of Algebraic Theories". Reprint available at http://www.tac.mta.ca/tac/reprints/articles/5/tr5abs.html. PhD thesis. Columbia University, 1963.

[109] T. Leinster. *Basic Category Theory*. Vol. 143. Cambridge Studies in Advanced Mathematics. Cambridge University Press, 2014. DOI: 10.1017/CBO9781107360068.

[110] D. Licata, M. Riley, and M. Shulman. *A Fibrational Framework for Substructural and Modal Dependent Type Theories*. Talk at HoTTEST. Mar. 2019. URL: https://www.youtube.com/watch?v=-DP0wY2FBs4.

[111] D. R. Licata, I. Orton, A. M. Pitts, and B. Spitters. "Internal Universes in Models of Homotopy Type Theory". In: *3rd International Conference on Formal Structures for Computation and Deduction (FSCD 2018)*. Ed. by H. Kirchner. Vol. 108. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018, 22:1–22:17. DOI: 10.4230/LIPIcs.FSCD.2018.22.

[112] D. R. Licata and M. Shulman. "Calculating the Fundamental Group of the Circle in Homotopy Type Theory". In: *Proceedings of the 2013 28th Annual ACM/IEEE Symposium on Logic in Computer Science*. LICS '13. Washington, DC, USA: IEEE Computer Society, 2013, pp. 223–232. DOI: 10.1109/LICS.2013.28.

[113] D. R. Licata, M. Shulman, and M. Riley. "A Fibrational Framework for Substructural and Modal Logics". In: *2nd International Conference on Formal Structures for Computation and Deduction (FSCD 2017)*. Ed. by D. Miller. Vol. 84. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017, 25:1–25:22. DOI: 10.4230/LIPIcs.FSCD.2017.25.

[114] G. Longo and E. Moggi. "Constructive natural deduction and its 'ω-set' interpretation". In: *Mathematical Structures in Computer Science* 1.2 (1991), pp. 215–254. DOI: 10.1017/S0960129500001298.

[115] P. L. Lumsdaine. *What are we thinking when we present a type theory?* Talk at the HoTTEST Conference. 2020. URL: https://www.youtube.com/watch?v=kQe0knDuZqg.

[116] P. L. Lumsdaine and M. A. Warren. "The Local Universes Model: An Overlooked Coherence Construction for Dependent Type Theories". In: *ACM Trans. Comput. Logic* 16.3 (July 2015), 23:1–23:31. DOI: 10.1145/2754931.

[117] J. Lurie. *Higher Topos Theory*. Vol. 170. Annals of Mathematics Studies. Princeton University Press, 2009. URL: https://www.math.ias.edu/~lurie/papers/HTT.pdf.

[118] S. Mac Lane. *Categories for the Working Mathematician*. 2nd. Vol. 5. Graduate Texts in Mathematics. New York, NY: Springer New York, 1998. DOI: 10.1007/978-1-4757-4721-8.

[119] S. Mac Lane and I. Moerdijk. *Sheaves in Geometry and Logic*. New York, NY: Springer New York, 1992. DOI: 10.1007/978-1-4612-0927-0.

[120] M. E. Maietti. "Modular correspondence between dependent type theories and categories including pretopoi and topoi". In: *Mathematical Structures in Computer Science* 15.6 (2005), pp. 1089–1149. DOI: 10.1017/S0960129505004962.

[121] M. E. Maietti. "The Internal Type Theory of a Heyting Pretopos". In: *Types for Proofs and Programs, International Workshop TYPES'96, Aussois, France, December 15-19, 1996, Selected Papers*. Ed. by E. Giménez and C. Paulin-Mohring. Vol. 1512. Lecture Notes in Computer Science. Springer, 1996, pp. 216–235. DOI: 10.1007/BFb0097794.

[122] M. E. Maietti and G. Sambin. "Toward a minimalist foundation for constructive mathematics". In: *From Sets and Types to Topology and Analysis*. Ed. by L. Crosilla and P. Schuster. Oxford University Press, 2005. DOI: 10.1093/acprof:oso/9780198566519.003.0006.

[123] M. Makkai and G. E. Reyes. *First Order Categorical Logic. Model-Theoretical Methods in the Theory of Topoi and Related Categories*. Vol. 611. Lecture Notes in Mathematics. Springer-Verlag Berlin Heidelberg, 1977. DOI: 10.1007/BFb0066201.

[124] P. Martin-Löf. "An Intuitionistic Theory of Types: Predicative Part". In: *Studies in Logic and the Foundations of Mathematics* 80 (1975), pp. 73–118. DOI: 10.1016/S0049-237X(08)71945-1.

[125] P. Martin-Löf. *Intuitionistic type theory*. Vol. 1. Studies in Proof Theory. Lecture Notes. Notes by Giovanni Sambin. Bibliopolis, Naples, 1984, pp. iv+91.

[126] N. P. Mendler. "Quotient types via coequalizers in Martin-Löf type theory". In: *Proceedings of the Logical Frameworks Workshop*. 1990, pp. 349–361.

[127] E. Moggi. "Notions of computation and monads". In: *Information and Computation* 93.1 (1991). Selections from 1989 IEEE Symposium on Logic in Computer Science, pp. 55–92. DOI: 10.1016/0890-5401(91)90052-4.

[128] G. L. Monaco. "Dependent products and 1-inaccessible universes". In: *Theory and Applications of Categories* 37.5 (2021), pp. 107–143. URL: http://www.tac.mta.ca/tac/volumes/37/5/37-05abs.html.

[129] C. Newstead. "Algebraic models of dependent type theory". PhD thesis. Carnegie Mellon University, 2018. URL: https://sites.math.northwestern.edu/~newstead/thesis-clive-newstead.pdf.

[130] H. K. Nguyen and T. Uemura. *∞-type theories*. in preparation.

[131] S. B. Niefield. "Cartesianness: topological spaces, uniform spaces, and affine schemes". In: *Journal of Pure and Applied Algebra* 23.2 (1982), pp. 147–167. DOI: 10.1016/0022-4049(82)90004-4.

[132] B. Nordström, K. Petersson, and J. M. Smith. "Martin-Löf's Type Theory". In: *Handbook of Logic in Computer Science*. Ed. by S. Abramsky, D. M. Gabbay, and T. S. E. Maibaum. Vol. 5. Oxford University Press, 2001.

[133] B. Nordström, K. Petersson, and J. M. Smith. *Programming in Martin-Löf's Type Theory: An Introduction*. Oxford University Press, 1990. URL: http://www.cse.chalmers.se/research/group/logic/book/.

[134] I. Orton and A. M. Pitts. "Axioms for Modelling Cubical Type Theory in a Topos". In: *25th EACSL Annual Conference on Computer Science Logic (CSL 2016)*. Ed. by J.-M. Talbot and L. Regnier. Vol. 62. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2016, 24:1–24:19. DOI: 10.4230/LIPIcs.CSL.2016.24.

[135] I. Orton and A. M. Pitts. "Axioms for Modelling Cubical Type Theory in a Topos". In: *Logical Methods in Computer Science* 14 (4 Dec. 2018). DOI: 10.23638/LMCS-14(4:23)2018.

[136] F. Pfenning and R. Davies. "A judgmental reconstruction of modal logic". In: *Mathematical Structures in Computer Science* 11.4 (2001), pp. 511–540. DOI: 10.1017/S0960129501003322.

[137] W. Phoa. *An introduction to fibrations, topos theory, the effective topos and modest sets*. Tech. rep. ECS-LFCS-92-208. The University of Edinburgh, 2006. URL: http://www.lfcs.inf.ed.ac.uk/reports/92/ECS-LFCS-92-208/.

[138] A. M. Pitts. "Categorical Logic". In: *Handbook of Logic in Computer Science: Volume 5: Logic and Algebraic Methods*. USA: Oxford University Press, Inc., 2001, pp. 39–123.

[139] A. Rădulescu-Banu. *Cofibrations in Homotopy Theory*. 2009. arXiv: `math/0610009v4`.

[140] N. Rasekh. *A Theory of Elementary Higher Toposes*. 2018. arXiv: `1805.03805v2`.

[141] N. Rasekh. *Complete Segal Objects*. 2018. arXiv: `1805.03561v1`.

[142] N. Rasekh. *Univalence in Higher Category Theory*. 2021. arXiv: `2103.12762v1`.

[143] J. C. Reynolds. "Towards a theory of type structure". In: *Programming Symposium*. Ed. by B. Robinet. Berlin, Heidelberg: Springer Berlin Heidelberg, 1974, pp. 408–425. DOI: `10.1007/3-540-06859-7_148`.

[144] E. Riehl. *Category Theory in Context*. Dover Books, 2017. URL: `https://math.jhu.edu/~eriehl/context.pdf`.

[145] E. Riehl and D. Verity. *Elements of ∞-Category Theory*. 2021. URL: `http://www.math.jhu.edu/~eriehl/elements.pdf`.

[146] E. Rijke, M. Shulman, and B. Spitters. "Modalities in homotopy type theory". In: *Log. Methods Comput. Sci.* 16.1 (2020), Paper No. 2, 79. DOI: `10.23638/LMCS-16(1:2)2020`.

[147] C. Sattler. *The Equivalence Extension Property and Model Structures*. 2017. arXiv: `1704.06911v4`.

[148] R. A. G. Seely. "Categorical Semantics for Higher Order Polymorphic Lambda Calculus". In: *The Journal of Symbolic Logic* 52.4 (1987), pp. 969–989. DOI: `10.2307/2273831`.

[149] R. A. G. Seely. "Locally cartesian closed categories and type theory". In: *Math. Proc. Cambridge Philos. Soc.* 95.1 (1984), pp. 33–48. DOI: `10.1017/S0305004100061284`.

[150] R. A. G. Seely. "Hyperdoctrines, natural deduction and the Beck condition". In: *Zeitschrift für mathematische Logik und Grundlagen der Mathematik* 29.10 (1983), pp. 505–542. DOI: `10.1002/malq.19830291005`.

[151] M. Shulman. *All $(\infty, 1)$-toposes have strict univalent universes*. 2019. arXiv: `1904.07004v2`.

[152] M. Shulman. "Brouwer's fixed-point theorem in real-cohesive homotopy type theory". In: *Mathematical Structures in Computer Science* 28.6 (2018), pp. 856–941. DOI: `10.1017/S0960129517000147`.

[153] M. Shulman. *Towards elementary infinity-toposes*. Talk at Vladimir Voevodsky Memorial Conference, IAS, September 2018. 2018. URL: https://www.youtube.com/watch?v=ld4YL787dAk.

[154] M. Shulman. "Univalence for inverse diagrams and homotopy canonicity". In: *Mathematical Structures in Computer Science* 25.05 (2015), pp. 1203–1277. DOI: 10.1017/s0960129514000565.

[155] M. Shulman. *Higher Inductive Types via Impredicative Polymorphism*. 2011. URL: https://homotopytypetheory.org/2011/04/25/higher-inductive-types-via-impredicative-polymorphism/.

[156] S. Speight. "Impredicative Encodings of Inductive Types in Homotopy Type Theory". MA thesis. Carnegie Mellon University, 2017. URL: http://www.cs.ox.ac.uk/people/sam.speight/publications/sams-hott-thesis.pdf.

[Stacks] The Stacks Project Authors. *Stacks Project*. 2020. URL: https://stacks.math.columbia.edu.

[157] W. P. Stekelenburg. *Constructive Simplicial Homotopy*. 2016. arXiv: 1604.04746v1.

[158] T. Streicher. *Fibred Categories à la Jean Bénabou*. 2020. arXiv: 1801.02927v11.

[159] T. Streicher. "Independence of the induction principle and the axiom of choice in the pure calculus of constructions". In: *Theoretical Computer Science* 103.2 (1992), pp. 395–408. DOI: 10.1016/0304-3975(92)90021-7.

[160] T. Streicher. *Semantics of type theory. Correctness, Completeness and Independence Results*. Progress in Theoretical Computer Science. Birkhäuser Basel, 1991, pp. xii+298. DOI: 10.1007/978-1-4612-0433-6.

[161] A. W. Swan. *W-Types with Reductions and the Small Object Argument*. 2018. arXiv: 1802.07588v1.

[162] A. W. Swan. "An Algebraic Weak Factorisation System on 01-Substitution Sets: A Constructive Proof". In: *Journal of Logic & Analysis* 8 (2016), pp. 1–35. DOI: 10.4115/jla.2016.8.1.

[163] A. W. Swan and T. Uemura. *On Church's Thesis in Cubical Assemblies*. 2019. arXiv: 1905.03014v1.

[164] K. Szumiło. "Homotopy Theory of Cofibration Categories". In: *Homology, Homotopy & Applications* 18.2 (2016), pp. 345–357. DOI: 10.4310/HHA.2016.v18.n2.a19.

[165] K. Szumiło. "Two Models for the Homotopy Theory of Cocomplete Homotopy Theories". PhD thesis. University of Bonn, Nov. 2014. arXiv: 1411.0303v1.

[166] P. Taylor. "Internal completeness of categories of domains". In: *Category Theory and Computer Programming: Tutorial and Workshop, Guildford, U.K. September 16–20, 1985 Proceedings*. Ed. by D. Pitt, S. Abramsky, A. Poigné, and D. Rydeheard. Berlin, Heidelberg: Springer Berlin Heidelberg, 1986, pp. 449–465. DOI: 10.1007/3-540-17162-2_137.

[167] A. S. Troelstra and D. van Dalen. *Constructivism in Mathematics, Volume I*. Vol. 121. Studies in Logic and the Foundations of Mathematics. Elsevier Science Publishers B.V., 1988.

[168] A. S. Troelstra and D. van Dalen. *Constructivism in Mathematics, Volume II*. Vol. 123. Studies in Logic and the Foundations of Mathematics. Elsevier Science Publishers B.V., 1988.

[169] T. Uemura. *A General Framework for the Semantics of Type Theory*. 2019. arXiv: 1904.04097v2.

[170] T. Uemura. "Cubical Assemblies, a Univalent and Impredicative Universe and a Failure of Propositional Resizing". In: *24th International Conference on Types for Proofs and Programs (TYPES 2018)*. Ed. by P. Dybjer, J. E. Santo, and L. Pinto. Vol. 130. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019, 7:1–7:20. DOI: 10.4230/LIPIcs.TYPES.2018.7.

[171] T. Uemura. *The Universal Exponentiable Arrow*. 2020. arXiv: 2001.09940v1.

[172] The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. Institute for Advanced Study, 2013. URL: http://homotopytypetheory.org/book/.

[173] B. van den Berg. "Predicative topos theory and models for constructive set theory". PhD thesis. University of Utrecht, 2006. URL: https://staff.fnwi.uva.nl/b.vandenberg3/thesis/mythesis.pdf.

[174] B. van den Berg. "Univalent polymorphism". In: *Ann. Pure Appl. Logic* 171.6 (2020), pp. 102793, 29. DOI: 10.1016/j.apal.2020.102793.

[175] B. van den Berg and E. Faber. *Effective Kan fibrations in simplicial sets*. 2020. arXiv: 2009.12670v1.

[176] J. van Oosten. *Realizability: an introduction to its categorical side*. Vol. 152. Studies in Logic and the Foundations of Mathematics. Elsevier B. V., Amsterdam, 2008, pp. xvi+310.

[177] V. Voevodsky. *A universe polymorphic type system*. 2012. URL: https://ncatlab.org/ufias2012/files/Universe+polymorphic+type+sytem.pdf.

[178] V. Voevodsky. *B-systems*. 2014. arXiv: 1410.5389v1.

[179] V. Voevodsky. *Subsystems and regular quotients of C-systems*. 2016. arXiv: 1406.7413v3.

[180] F. Waldhausen. "Algebraic K-theory of spaces". In: *Algebraic and geometric topology: proceedings of a conference held at Rutgers Univ., New Brunswick, USA, July 6 - 13, 1983*. Ed. by A. Ranicki, N. Levitt, and F. Quinn. Springer, 1985, pp. 318–419. DOI: 10.1007/BFb0074449.

[181] M. Weber. "Polynomials in categories with pullbacks". In: *Theory and Applications of Categories* 30.16 (2015), pp. 533–598. URL: http://www.tac.mta.ca/tac/volumes/30/16/30-16abs.html.

[182] N. Yamada. *Game semantics of Martin-Löf type theory, part III: its consistency with Church's thesis*. 2020. arXiv: 2007.08094v2.

Acronym

$(\infty,1)$**-CwR** $(\infty,1)$-category with representable maps. 159, 162

CTT cubical type theory. 94, 189

CwR category with representable maps. 31, 32, 41

DTT basic dependent type theory. 83

qCTT quasi-cubical type theory. 190–192

SOGAT second-order generalized algebraic theory. 44, 47, 83

Notation

Δ	Diagonal map. 24
$\Gamma^\dagger$	Metavariable replacement of Γ. 78
$\Gamma\vert_{\mathsf{term}}$	Underlying variable signature of Γ. 59
Ω	Instantiation associated to metavariable replacement. 78
Ω	Subobject classifier. 116
$\Phi\vert_{\mathsf{term}}$	Underlying metavariable signature of Φ. 60
$\gamma^\dagger$	Metavariable replacement of γ. 78
ω	Substitution associated to metavariable replacement. 78
$\sqrt{}$	Right adjoint of the exponentiation by interval. 193
$0_{\mathbb{I}}$	End point. 94, 192
$1_{\mathbb{I}}$	End point. 94, 192
$\int_{\mathcal{C}} A$	Category of elements. 35, 200
$\nabla\, A$	Codiscrete presheaf at A. 208
$\Delta\, A$	Constant presheaf at A. 208
$A_\perp$	Partial map classifier. 116
$\{a\}$	Context comprehension of a. 36
$a \in \alpha$	Membership relation. 116
$\nabla_A\, B$	Codicsrete presheaf at B over $\Delta\, A$. 208
A -> B	Function type. 85
$A \rightharpoonup B$	Type of partial maps. 116
$\Gamma \to a : K$	Type judgment. 63
Asm	Assembly model. 213
Asm	Category of assemblies. 213
Asm	Assumption introduction rule. 67
$a \cdot u$	Right action of u. 25, 35, 200

$\{a\}_u$	Context comprehension of a with respect to u. 141
Axiom	Axiom introduction rule. 67
$\mathbf{B}$	Category of powers of 2. 206
$\mathcal{C}^{\to}$	Category of arrows of $\mathcal{C}$. 24
$\widehat{\mathbf{C}}$	Category of larger objects. 23
$\mathcal{C}^{\triangleright}$	Right cone. 143, 162
$\mathcal{C}^{\underline{\Delta}}$	Category of sections. 133, 176
$\mathcal{C}^{\swarrow\searrow}$	Category of spans in $\mathcal{C}$. 125
$\mathcal{C}_{\mathrm{r}}$	Category of representable objects in $\mathcal{C}$. 151
CAsm	Cubical assembly model. 213
Cat	$(2,1)$-Category of categories. 25
Cl	Syntactic category. 112
Cof	Universe of cofibrations. 95, 192
CwR	Category of categories representable maps. 41
$(\infty,1)$-**CwR**	$(\infty,1)$-category of $(\infty,1)$-categories with representable maps. 162
$\mathbb{D}^{\Pi}$	Dependent type theory with Π-types. 43
$\mathbb{D}$	Basic dependent type theory. 43, 176
$\mathrm{Deriv}(\mathscr{P},\mathcal{J})$	Set of derivations from $\mathscr{P}$ to $\mathcal{J}$. 70
DFib	Category of discrete fibrations. 34
$\mathbf{DFib}_{\mathcal{C}}$	Category of discrete fibrations over $\mathcal{C}$. 34
$\mathbb{E}_n$	Extensional $(n+1)$-type theory. 170
$\mathbb{E}_{\infty}^{\Pi}$	Extensional ∞-type theory with Π-types. 171
$\mathbb{E}_{\infty}$	Extensional ∞-type theory. 168
$\mathbb{E}$	Extensional type theory. 173
$[\![e]\!]$	Interpretation. 118
$\mathrm{E}_A(a)$	Set of realizers. 212
$\Gamma \to e\ c$	Well-formedness judgment. 65
$\Gamma \to e\ \mathsf{ok}$	Well-formedness judgment. 65
$\mathrm{Eqv}(u)$	Right fibration of equivalences. 164
$\underline{\mathrm{Eqv}}(u)$	Representing object for $\underline{\mathrm{Eqv}}(u)$. 164
$\overline{\mathrm{Eqv}}(x,y)$	Right fibration of equivalences. 164
$\underline{\mathrm{Eqv}}(x,y)$	Representing object for $\mathrm{Eqv}(x,y)$. 164
$\overline{\mathrm{Expr}}_{\Sigma,\mu}(\gamma,c)$	Set of expressions. 54
$\underline{\mathrm{Expr}}_{\mathcal{X}}(\gamma,c)$	Presheaf of semantic expressions. 117
$\overline{F_!}$	Left adjoint of F^*. 174
F^*	Precomposition functor. 174
$\Gamma' \to f_1 \equiv f_2 : \Gamma \setminus \Gamma_0$	Partially trivial substitution equality judgment. 74
$\Gamma \to f_1 \equiv f_2 : \Delta$	Substitution equality judgment. 64
$\Gamma' \to f : \Gamma \setminus \Gamma_0$	Partially trivial substitution judgment. 73

$\Gamma \to f : \Delta$	Substitution judgment. 64
$\Sigma, \mu \vdash f : \gamma \to \delta$	A substitution. 54
$f(a)\downarrow$	Application is defined. 116
$\mathbf{Fun}(\mathcal{C}, \mathcal{D})$	Category of functors $\mathcal{C} \to \mathcal{D}$. 24
$\Gamma \to \mathcal{H}$	Judgment. 63
$H : \Gamma \to P$	Assumption. 61
$H : \Phi \Rightarrow P$	Axiom. 62
Hyp	Hypothesis introduction rule. 66
$\Sigma, \mu \vdash I : \gamma \Rightarrow \nu$	An instantiation. 54
$\mathbb{I}$	Intensional type theory. 173
$\mathbb{I}$	Interval. 94, 192
$\Gamma \to I_1 \equiv I_2 : \Psi$	Instantiation equality judgment. 65
$\Gamma \to I : \Psi$	Instantiation judgment. 65
inl	Left inclusion of coproduct. 24
inr	Right inclusion of coproduct. 24
$\mathrm{Inst}_{\Sigma,\mu}(\gamma, \nu)$	Set of instantiations of ν over γ. 54
$\underline{\mathrm{Inst}}_{\mathcal{X}}(\gamma, \nu)$	Presheaf of semantic instantiations. 117
$\mathrm{I}(\mathcal{T})$	Initial model of $\mathcal{T}$. 151
$\underline{\mathrm{Itpr}}_{\mathcal{X}}(-)$	Presheaf of interpretations. 117, 120, 121
$\mathrm{Itpr}(T, \mathcal{C})$	Groupoid of interpretations. 128
$\varphi \models \mathcal{J}$	φ satisfies $\mathcal{J}$. 120
$\mathcal{K}_1$	Kleene's first model. 213
$\mathrm{k}(\mathcal{C})$	The largest groupoid contained in $\mathcal{C}$. 24
$\mathcal{L}$	Nullification. 223
LAdj	Category of left adjoints. 27
$(\infty, 1)$-**LCCC**	$(\infty, 1)$-category of $(\infty, 1)$-LCCCs. 171
$\mathsf{let}\ \mathsf{u} \Leftarrow a\ \mathsf{in}\ b$	Let notation. 116
$\mathbf{Lex}^{(\Xi)}$	Category of categories with limits of shape Ξ. 27
Lex	Category of categories with finite limits. 27
$(n, 1)$-**Lex**	$(n+1, 1)$-category of finitely complete $(n, 1)$-categories. 170
$(\infty, 1)$-**Lex**	$(\infty, 1)$-category of finitely complete $(\infty, 1)$-categories. 167
$\mathbf{Lex}(\mathcal{C}, \mathcal{D})$	Category of left exact functors $\mathcal{C} \to \mathcal{D}$. 24
$\mathrm{L}(\mathcal{M})$	Internal language of $\mathcal{M}$. 138, 163
$\mathcal{M}^{\heartsuit}$	Heart of $\mathcal{M}$. 142
$\mathrm{Map}(x, y)$	Right fibration of maps. 164
$\underline{\mathrm{Map}}(x, y)$	Representing object for $\mathrm{Map}(x, y)$. 164
$\mathcal{M}_{\mathrm{fib}}$	Model consisting of fibrations. 193
$\mathbf{Mod}(\mathcal{T})$	Category of models of $\mathcal{T}$. 138, 163
$\mathbf{Mod}^{\mathrm{dem}}(\mathcal{T})$	Category of democratic models of $\mathcal{T}$. 141, 163

$\mathcal{M}_P$	Model consisting of P-null types. 225
$\mathcal{M}(\mathsf{u})_{\mathcal{C}}$	Universe lifting. 201
$\equiv$-MVar	Metavariable congruence rule. 67
MVar	Metavariable introduction rule. 67
$\rho_{\mathcal{C}} : \widetilde{\mathrm{O}}_{\mathcal{C}} \to \mathrm{O}_{\mathcal{C}}$	Representable map classifier. 166
$\mathcal{O}_{\mathcal{C}}^{(\mathrm{all})}$	Right fibration of all maps. 164
$\mathcal{O}_{\mathcal{C}}$	Codomain fibration. 164
$\mathcal{P}\,A$	Power object. 116
$\Gamma \to P$	Proposition judgment. 63
P_u	Polynomial functor. 29
$\gamma(P)$	Reflection of proposition P into an impredicative universe. 216
η_P	Unit P -> $\gamma(P)$. 216
PER	Universe of partial equivalence relations. 214
PropConv	Proposition conversion rule. 68
$\mathbf{Pr}_{\omega}^{\mathrm{R}}$	$(2,1)$-category of compactly generated categories. 27
$\mathbf{PSh}_{\mathcal{C}}^{\mathcal{M}}$	Presheaf model in $\mathcal{M}$. 201
$\mathrm{p}_u(a)$	First context projection. 37
$\mathrm{q}_u(a)$	Second context projection. 37
$\mathrm{R}_{\mathcal{C}}$	Class of representable maps in $\mathcal{C}$. 41
$\mathbb{R}_\infty$	∞-type theory for ∞-type theories. 172
Refl	Reflexivity rule. 69
$\mathrm{S}(\Phi)$	Syntactic model generated by Φ. 151
$s_1 \,.\, s_2$	Extension of declaration. 59
$S : \mu \Rightarrow c$	Symbol of syntactic class c. 53
$S : \Phi \Rightarrow c$	Sort symbol. 62
Set	Category of sets. 23
$\mathbf{Sig}_A$	Category of signatures valued in A. 54
$S : \Phi \Rightarrow K$	Term symbol. 62
Smtry	Symmetry rule. 69
Space	$(\infty,1)$-category of spaces. 25
$\mathrm{Subst}_{\Sigma,\mu}(\gamma,\delta)$	Set of substitutions of δ in γ. 54
$\underline{\mathrm{Subst}}_{\mathcal{X}}(\gamma,\delta)$	Presheaf of semantic substitutions. 117
$\equiv$-Subst	Equality substitution rule. 69
Subst	Substitution rule. 69
$\equiv$-Sym	Symbol congruence rule. 67
Sym	Symbol introduction rule. 67
T	Kleene's computation predicate. 220
$T\vert_{\mathsf{expr}}$	Underlying symbol signature of T. 61
$\mathcal{T}[\Phi]$	Extension of type theory by global sections. 155

$\mathbf{Th}(T)$	Category of T-theories. 108
$\mathbf{Th}(\mathcal{T})$	Category of $\mathcal{T}$-theories. 138, 163
$T, \Phi \vdash \mathcal{J}$	$\mathcal{J}$ is derivable over T and Φ. 70
$T, \Phi \vdash \Gamma\ \mathsf{ok}$	Γ is a well-formed context over T and Φ. 71
$T \vdash \Phi\ \mathsf{ok}$	Φ is a well-formed environment over T. 71
Trans	Transitivity rule. 69
TT	Category of type theories. 42
∞-**TT**	$(\infty, 1)$-category of ∞-type theories. 162
TypeConv	Type conversion rule. 67
TypeConvEq	Equality conversion rule. 67
U	Kleene's result extraction function. 220
u^*	Pullback along u. 28
$u_!$	Postcomposition with u. 28
u_*	Pushforward along u. 29
Var	Variable introduction rule. 66
$v \otimes u$	Composition of polynomials. 30
x^*	Pullback along $x \to 1$. 28
$x_!$	Forgetful functor $\mathcal{C}/x \to \mathcal{C}$. 28
$x : a$	Entry of a signature. 52
$(\mathtt{x} : A)$ -> B	Dependent function type. 85
$(\mathtt{x} : A) \times B$	Dependent pair type. 86
$\{\mathtt{x} : A \mid P\}$	Set comprehension. 116
$x : b$	Entry of a declaration. 59
$\lambda\mathtt{x}.b$	Lambda abstraction. 85
$\langle \mathtt{x}_1, \ldots, \mathtt{x}_n \rangle e$	$\mathtt{x}_1, \ldots, \mathtt{x}_n$ are bound in e. 55
$[\mathtt{X} : \Gamma \to K]$	Implicit argument. 83
$\mathtt{X} : \Gamma \to K$	Metavariable. 61
Y	Yoneda embedding. 25
$(\mathtt{y}_1 := f_1, \ldots, \mathtt{y}_n := f_n)$	Substitution. 55
$(\mathtt{Y}_1 := I_1, \ldots, \mathtt{Y}_n := I_n)$	Instantiation. 55

Index

2-morphism
– of models of a type theory, 138

adjoint functor theorem, 28
arrow
exponentiable –, 29
univalent –, 164
assembly, 212
cubical –, 213
modest –, 214
uniform –, 217
assumption, 60
axiom, 61

Beck-Chevalley condition, 24
Brouwer's Principle, 231

category, 23, 200
– of elements, 35, 200
– with representable maps, 41
syntactic –, 112
– with weak equivalences and cofibrations, 177
compactly generated, 26
comprehension –, 175
cube –, 207
filtered –, 26
Church's Thesis, 220, 230
untruncated –, 221
circle, 91, 101
class
pullback-stable –, 29
closure rule, 69
closure system, 70
cofibration, 95, 177, 192
– of $\mathbb{D}$-theories, 178
generating –, 178
– of $\mathbb{I}$-theories
generatinc –, 184
colimit
filtered, 26
strictly extensive –, 92
compact
– object, 26
– theory over a type theory, 154
composition, 97, 193
homogeneous –, 97, 193
conclusion, 66, 70
connected, 192
context, 59
finite –, 83
well-formed –, 71
well-ordered –, 81
context comprehension, 37, 141
contextual completeness, 79
coproduct, 87
disjoint –, 90
strictly extensive –, 90
cubical type theory, 94

declaration, 59
relative –, 59
sub-, 59
declare, 84

define, 84
derivation, 70
discrete fibration, 34, 147
    representable –, 36
display map, 169

endpoint, 94
environment, 60
    finitary –, 83
    finite –, 83
    well-formed –, 71
    well-ordered –, 81
exponentiable
    – arrow, 29
    – object, 28
expression, 55
    proposition –, 55
    representable proposition –, 55
    representable type –, 55
    sort –, 55
    term –, 55
    type –, 55
extension
    – of a declaration, 59

filtered
    – category, 26
finitary
    – environment, 83
    – pretheory, 83
finite
    – context, 83
    – environment, 83
functor
    localization –, 174
    right exact –, 178

generalized algebraic theory, 105, 108
glueing, 100, 195
good type theory, 190

heart
    – of a model of a type theory, 142
higher inductive type, 91, 101
hypothesis, 59

inference rule, 66
∞-category
    – with representable maps, 162
∞-type theory, 162
instantiation, 54
internal language
    – of a model of a type theory, 138
    – of a topos, 115
interpretation, 128
interval, 94, 192, 203
isomorphism extension structure, 192

judgment, 63
    derivable –, 70
judgment head, 63
    proposition –, 63
    type –, 62

Kleene's first model, 213

left approximation property, 178
local fibrant replacement, 197
localization, 177

Markov's Principle, 219
Martin-Löf type theory, 84
metavariable, 52
metavariable replacement
    – of a context, 78
    – of a variable signature, 78
model
    – of a type theory, 42
      democratic –, 141
      initial –, 151
      syntactic –, 151
    – of an ∞-type theory, 162
morphism
    – of assemblies, 213
    – of categories with representable maps, 41
    – of declarations, 59
    – of ∞-categories with representable maps, 162

– of ∞-type theories, 162
– of models of a type theory, 42
– of models of an ∞-type theory, 163
– of signatures, 54
– of theories over a second-order generalized algebraic theories, 108
– of type theories, 42

n-type theory, 162
natural model, 33
(n, i)-category, 25
null structure, 223
nullification, 223

object
cofibrant –, 177
compact –, 26
contextual –, 141
exponentiable –, 28

partial equivalence relation, 214
partial map classifier, 116
polynomial
– with reductions, 93
polynomial functor, 29
premise, 66, 69
presentation
– of an ∞-type theory by a 1-type theory, 174
presheaf, 25, 200
codiscrete –, 208
constant –, 208
representable –, 25
uniform –, 217
presupposition, 66
pretheory, 61
finitary –, 83
well-formed –, 71
well-ordered –, 81
proposition, 92, 215
decidable –, 204
locally decidable –, 205
propositional extensionality, 192
propositional resizing axiom, 215, 216
propositional truncation, 92, 103, 198
pseudo-judgment, 64
pushforward, 29
pushout, 92

quasi-cubical type theory, 192

realizer, 213
representable
– discrete fibration, 36
– presheaf, 25
representable map
– in a category with representable maps, 41
– in an ∞-category with representable maps, 162
– of discrete fibrations, 36, 147
– of environments, 113
– of presheaves, 33
generic –, 166
representable map classifier, 166
right fibration, 147

second-order algebraic theory, 103
second-order generalized algebraic theory, 83
internal –, 133
section
– of a discrete fibration, 36
– of a presheaf, 25
signature, 52
metavariable –, 52
symbol –, 53
variable –, 52
strictly extensive
– colimit, 92
– coproduct, 90
strong generator, 26
substitution, 54
jointly partially trivial –, 74
partially trivial –, 73
sum

– of polynomials with reductions, 94
suspension, 102, 197
symbol, 53
proposition –, 53
representable proposition –, 53
representable type –, 53
term –, 53
type –, 53
syntactic class, 52

theory
– over a second-order generalized algebraic theory, 108
– over a type theory, 138
compact –, 154
– over an ∞-type theory, 163
tracker, 213
transport, 97
trivial cofibration, 177
trivial fibration
– of $\mathbb{D}$-theories, 179
two-pullbacks lemma, 24
type
dependent function –, 21
dependent pair –, 21
dependent product –, 21, 85
dependent sum –, 21, 85
discrete –, 198
empty –, 22, 87
Id-, 167
identity –, 22, 86, 195
extensional –, 22, 87
intensional –, 22, 87
natural numbers –, 23, 88
null –, 223
path –, 98, 195
Π-, 21, 85, 171, 172
Σ-, 21, 85, 167
unit –, 87, 167
W-, 88
– with reductions, 93
well-supported –, 217
type theory, 42

univalence axiom, 91
univalent type theory, 190
universe, 88
finitely complete –, 168
impredicative –, 214
weakly closed –, 89

variable, 52

weak equivalence, 177
weak representable map classifier, 116
well-formed
– context, 71
– environment, 71
– pretheory, 71
well-ordered
– context, 81
– environment, 81
– pretheory, 81

Yoneda embedding, 26
Yoneda lemma, 26

Samenvatting

In dit proefschrift bestuderen we zowel abstracte als concrete typentheorieën. Naast een abstract begrip van een typentheorie om algemene resultaten in de semantiek van typentheorie te kunnen bewijzen, introduceren we ook een manier om een typentheorie syntactisch te presenteren. Dit laatste gebruiken we om een concrete typentheorie te kunnen bestuderen en daarvoor consistentie- en onafhankelijksresultaten te bewijzen.

In Hoofdstuk 3 ontwikkelen we een abstract begrip van een typentheorie. Met Awodey's theorie van natuurlijke modellen van een typentheorie, waarin modellen van een typentheorie worden beschreven in termen van representeerbare afbeeldingen tussen discrete vezelingen, als uitgangspunt, geven we een abstracte definitie van een typentheorie als een categorie met representeerbare afbeeldingen. Een model van een typentheorie wordt dan gedefinïeerd als een structuurbehoudende functor naar een categorie van discrete vezelingen.

In Hoofdstuk 4 introduceren we "tweede-orde gegeneraliseerde algebraïsche theorieën": deze geven een syntactische presentatie van een typentheorie en zijn geïnspireerd door een definitie van een algemene typentheorie afkomstig van Bauer, Haselwarter en Lumsdaine. Deze presentatie komt overeen met de traditionele presentatie van een typentheorie gegeven door een grammatica en afleidingsregels. We laten zien dat elke tweede-orde gegeneraliseerde algebraïsche theorie een typentheorie genereert met een passende universele eigenschap en dat, omgekeerd, elke typentheorie wordt voortgebracht door een tweede-orde gegeneraliseerde algebraïsche theorie.

In Hoofdstuk 5 ontwikkelen we de semantiek van typentheorie gebaseerd op de definities die we in Hoofdstuk 3 hebben ingevoerd. Opmerkelijk genoeg kunnen de resultaten in dit hoofdstuk zuiver categorisch worden bewezen en wordt nergens gebruik gemaakt van de syntactische presentatie die we in Hoofdstuk 4 hebben geïntroduceerd. Het belangrijkste resultaat is dat er voor elke typentheorie een correspondentie bestaat tussen theorieën en modellen van die typentheorie. Dit resultaat geeft een formele rechtvaardiging voor het gebruik van de interne taal

van een model van een typentheorie.

De definitie van een typentheorie als een categorie met extra structuur stelt ons in staat om hoger-dimensionale generalisaties te definiëren. In Hoofdstuk 6 bestuderen we ∞-typentheorieën, de $(\infty, 1)$-categorische generalisatie van een typentheorie. ∞-typentheorieën blijken een nuttig hulpmiddel te zijn om die coherentieproblemen te begrijpen en aan te pakken die ontstaan in de (hogere) categorische semantiek van typentheorie. We laten zien dat zogeheten niet-gespleten modellen van een typentheorie op een natuurlijke manier als modellen van een ∞-typentheorie kunnen worden opgevat en dat coherentieproblemen in de (hogerdimensionale) categorische semantiek kunnen worden geformuleerd in de taal van de ∞-typentheorieën en aanverwante begrippen. We passen dit toe door een bewijs te schetsen van een vermoeden van Kapulkin en Lumsdaine dat zegt dat typentheorie met intensionele identiteitstypen een interne taal is voor $(\infty, 1)$-categorieën met eindige limieten.

Vanaf Hoofdstuk 7 richten we onze aandacht op een concrete typentheorie, namelijk univalente typentheorie. Deze typentheorie kan worden gepresenteerd als een tweede-orde gegeneraliseerde algebraïsche theorie en daarmee als een typentheorie in de zin van Hoofdstuk 3. Hoofdstuk 7 is voor het grootste gedeelte gewijd aan het herformuleren van de interne constructie van modellen van een univalente typentheorie gegeven door Orton en Pitts en Licata et al. in termen van ons begrip van een model van een typentheorie.

Tenslotte bestuderen we in Hoofdstuk 8 concrete modellen van univalente typentheorie. Door de constructie uit Hoofdstuk 7 toe te passen op het *assembly model* van extensionele typentheorie krijgen we het c*ubical assembly model*, een model van univalente typentheorie. We gebruiken dit *cubical assembly model* om consistentie- en onafhankelijkheidsresultaten te bewijzen. We laten zien dat het *cubical assembly model* een univalent en impredicatief universum bevat en dat het univalentie-axioma daarom consistent is met het bestaan van een impredicatief universum. Dit impredicatieve universum voldoet niet aan het "*propositional resizing axiom*" in dat we in het *cubical assembly model* een propositie kunnen construeren dat niet equivalent is aan een propositie in het impredicatieve universum. Dit laat zien dat het "*propositional resizing axiom*" niet bewijsbaar is in de univalente typentheorie. Verder bestuderen we de relatie tussen het *cubical assembly model* en Constructieve Recursieve Wiskunde. We laten zien dat het *cubical assembly model* aan het Principe van Markov voldoet, maar niet aan de These van Church. Door gebruik te maken van de theorie van modaliteiten in de homotopietypentheorie die door Rijke, Shulman en Spitters is ontwikkeld, kunnen we een reflectief deeluniversum van het *cubical assembly model* construeren waarin zowel het Principe van Markov als de These van Church gelden. Daarmee laten we zien dat deze principes van de Constructieve Recursieve Wiskunde consistent zijn met univalente typentheorie.

Abstract

In this thesis, we study abstract and concrete type theories. We introduce an abstract notion of a type theory to obtain general results in the semantics of type theories, but we also provide a syntactic way of presenting a type theory to allow us a further investigation into a concrete type theory to obtain consistency and independence results.

In Chapter 3, we introduce an abstract notion of a type theory. After reviewing the theory of natural models of a type theory introduced by Awodey where models of a type theory are described in terms of representable maps of discrete fibrations, we introduce a notion of a category with representable maps as an abstract definition of a type theory. A model of a type theory is defined to be a structure-preserving functor to a category of discrete fibrations.

In Chapter 4, we introduce a syntactic presentation of a type theory called a second-order generalized algebraic theory, inspired by the definition of general type theories given by Bauer, Haselwarter, and Lumsdaine. This presentation coincides with the traditional presentation of a type theory by a grammar and inference rules. We show that every second-order generalized algebraic theory generates a type theory with an appropriate universal property and that, conversely, every type theory is generated by some second-order generalized algebraic theory.

In Chapter 5, we develop the semantics of type theories based on the definitions introduced in Chapter 3. Remarkably, the results in this chapter are theorems in pure category theory and do not depend on the syntactic presentations introduced in Chapter 4. The main result is the correspondence between theories and models for any type theory. This formally justifies the use of the internal language of a model of a type theory.

Once a type theory is defined as a category with certain structure, its higher dimensional generalization makes sense. We study an $(\infty, 1)$-categorical generalization of a type theory called an ∞-type theory in Chapter 6. It turns out that ∞-type theories are a useful tool for understanding and tackling coherence problems which arise in the categorical and higher categorical semantics of type

theories. We see that so-called non-split models of a type theory are naturally regarded as models of an ∞-type theory and that coherence problems in both categorical and higher categorical semantics of type theories can be formulated in the language of ∞-type theories and related concepts. As an application, we sketch a solution to Kapulkin and Lumsdaine's conjecture that the dependent type theory with intensional identity types provides internal languages for finitely complete $(\infty, 1)$-categories.

From Chapter 7, we turn our attention to a concrete type theory, univalent type theory. This type theory can be presented by a second-order generalized algebraic theory and thus defined as a type theory in the sense introduced in Chapter 3. Chapter 7 is mostly devoted to rephrasing the internal construction of models of univalent type theory given by Orton and Pitts and Licata et al. in terms of our notion of a model of a type theory.

Finally, in Chapter 8, we study concrete models of univalent type theory. Applying the construction explained in Chapter 7 to the assembly model of the extensional type theory, we have a model of univalent type theory called the cubical assembly model. We use the cubical assembly model to obtain consistency and independence results. We show that the cubical assembly model has a univalent and impredicative universe, and thus univalence is consistent with an impredicative universe. This impredicative universe does not satisfy the propositional resizing axiom in that we can construct a proposition in the cubical assembly model that is not equivalent to any proposition in the impredicative universe. Hence, the propositional resizing axiom is not provable over univalent type theory. We then study the cubical assembly model in relation to Constructive Recursive Mathematics. We show that the cubical assembly model satisfies Markov's Principle but does not satisfy Church's Thesis. Using the theory of modalities in homotopy type theory developed by Rijke, Shulman, and Spitters, we construct a reflective subuniverse of the cubical assembly model in which both Markov's Principle and Church's Thesis hold. Thus, these principles of Constructive Recursive Mathematics are consistent with univalent type theory.

Titles in the ILLC Dissertation Series:

ILLC DS-2016-01: **Ivano A. Ciardelli**
Questions in Logic

ILLC DS-2016-02: **Zoé Christoff**
Dynamic Logics of Networks: Information Flow and the Spread of Opinion

ILLC DS-2016-03: **Fleur Leonie Bouwer**
What do we need to hear a beat? The influence of attention, musical abilities, and accents on the perception of metrical rhythm

ILLC DS-2016-04: **Johannes Marti**
Interpreting Linguistic Behavior with Possible World Models

ILLC DS-2016-05: **Phong Lê**
Learning Vector Representations for Sentences - The Recursive Deep Learning Approach

ILLC DS-2016-06: **Gideon Maillette de Buy Wenniger**
Aligning the Foundations of Hierarchical Statistical Machine Translation

ILLC DS-2016-07: **Andreas van Cranenburgh**
Rich Statistical Parsing and Literary Language

ILLC DS-2016-08: **Florian Speelman**
Position-based Quantum Cryptography and Catalytic Computation

ILLC DS-2016-09: **Teresa Piovesan**
Quantum entanglement: insights via graph parameters and conic optimization

ILLC DS-2016-10: **Paula Henk**
Nonstandard Provability for Peano Arithmetic. A Modal Perspective

ILLC DS-2017-01: **Paolo Galeazzi**
Play Without Regret

ILLC DS-2017-02: **Riccardo Pinosio**
The Logic of Kant's Temporal Continuum

ILLC DS-2017-03: **Matthijs Westera**
Exhaustivity and intonation: a unified theory

ILLC DS-2017-04: **Giovanni Cinà**
Categories for the working modal logician

ILLC DS-2017-05: **Shane Noah Steinert-Threlkeld**
Communication and Computation: New Questions About Compositionality

ILLC DS-2017-06: **Peter Hawke**
The Problem of Epistemic Relevance

ILLC DS-2017-07: **Aybüke Özgün**
Evidence in Epistemic Logic: A Topological Perspective

ILLC DS-2017-08: **Raquel Garrido Alhama**
Computational Modelling of Artificial Language Learning: Retention, Recognition & Recurrence

ILLC DS-2017-09: **Miloš Stanojević**
Permutation Forests for Modeling Word Order in Machine Translation

ILLC DS-2018-01: **Berit Janssen**
Retained or Lost in Transmission? Analyzing and Predicting Stability in Dutch Folk Songs

ILLC DS-2018-02: **Hugo Huurdeman**
Supporting the Complex Dynamics of the Information Seeking Process

ILLC DS-2018-03: **Corina Koolen**
Reading beyond the female: The relationship between perception of author gender and literary quality

ILLC DS-2018-04: **Jelle Bruineberg**
Anticipating Affordances: Intentionality in self-organizing brain-body-environment systems

ILLC DS-2018-05: **Joachim Daiber**
Typologically Robust Statistical Machine Translation: Understanding and Exploiting Differences and Similarities Between Languages in Machine Translation

ILLC DS-2018-06: **Thomas Brochhagen**
Signaling under Uncertainty

ILLC DS-2018-07: **Julian Schlöder**
Assertion and Rejection

ILLC DS-2018-08: **Srinivasan Arunachalam**
Quantum Algorithms and Learning Theory

ILLC DS-2018-09: **Hugo de Holanda Cunha Nobrega**
Games for functions: Baire classes, Weihrauch degrees, transfinite computations, and ranks

ILLC DS-2018-10: **Chenwei Shi**
Reason to Believe

ILLC DS-2018-11: **Malvin Gattinger**
New Directions in Model Checking Dynamic Epistemic Logic

ILLC DS-2018-12: **Julia Ilin**
Filtration Revisited: Lattices of Stable Non-Classical Logics

ILLC DS-2018-13: **Jeroen Zuiddam**
Algebraic complexity, asymptotic spectra and entanglement polytopes

ILLC DS-2019-01: **Carlos Vaquero**
What Makes A Performer Unique? Idiosyncrasies and commonalities in expressive music performance

ILLC DS-2019-02: **Jort Bergfeld**
Quantum logics for expressing and proving the correctness of quantum programs

ILLC DS-2019-03: **András Gilyén**
Quantum Singular Value Transformation & Its Algorithmic Applications

ILLC DS-2019-04: **Lorenzo Galeotti**
The theory of the generalised real numbers and other topics in logic

ILLC DS-2019-05: **Nadine Theiler**
Taking a unified perspective: Resolutions and highlighting in the semantics of attitudes and particles

ILLC DS-2019-06: **Peter T.S. van der Gulik**
Considerations in Evolutionary Biochemistry

ILLC DS-2019-07: **Frederik Möllerström Lauridsen**
Cuts and Completions: Algebraic aspects of structural proof theory

ILLC DS-2020-01: **Mostafa Dehghani**
Learning with Imperfect Supervision for Language Understanding

ILLC DS-2020-02: **Koen Groenland**
Quantum protocols for few-qubit devices

ILLC DS-2020-03: **Jouke Witteveen**
Parameterized Analysis of Complexity

ILLC DS-2020-04: **Joran van Apeldoorn**
A Quantum View on Convex Optimization

ILLC DS-2020-05: **Tom Bannink**
Quantum and stochastic processes

ILLC DS-2020-06: **Dieuwke Hupkes**
Hierarchy and interpretability in neural models of language processing

ILLC DS-2020-07: **Ana Lucia Vargas Sandoval**
On the Path to the Truth: Logical & Computational Aspects of Learning

ILLC DS-2020-08: **Philip Schulz**
Latent Variable Models for Machine Translation and How to Learn Them

ILLC DS-2020-09: **Jasmijn Bastings**
A Tale of Two Sequences: Interpretable and Linguistically-Informed Deep Learning for Natural Language Processing

ILLC DS-2020-10: **Arnold Kochari**
Perceiving and communicating magnitudes: Behavioral and electrophysiological studies

ILLC DS-2020-11: **Marco Del Tredici**
Linguistic Variation in Online Communities: A Computational Perspective

ILLC DS-2020-12: **Bastiaan van der Weij**
Experienced listeners: Modeling the influence of long-term musical exposure on rhythm perception

ILLC DS-2020-13: **Thom van Gessel**
Questions in Context

ILLC DS-2020-14: **Gianluca Grilletti**
Questions & Quantification: A study of first order inquisitive logic

ILLC DS-2020-15: **Tom Schoonen**
Tales of Similarity and Imagination. A modest epistemology of possibility

ILLC DS-2020-16: **Ilaria Canavotto**
Where Responsibility Takes You: Logics of Agency, Counterfactuals and Norms

ILLC DS-2021-01: **Yfke Dulek**
Delegated and Distributed Quantum Computation

ILLC DS-2021-02: **Elbert J.Booij**
The Things Before Us: On What it Is to Be an Object

ILLC DS-2021-03: **Seyyed Hadi Hashemi**
Modeling Users Interacting with Smart Devices

ILLC DS-2021-04: **Sophie Arnoult**
Adjunction in Hierarchical Phrase-Based Translation

ILLC DS-2021-05: **Cian Guilfoyle Chartier**
A Pragmatic Defense of Logical Pluralism

ILLC DS-2021-06: **Zoi Terzopoulou**
Collective Decisions with Incomplete Individual Opinions

ILLC DS-2021-07: **Anthia Solaki**
Logical Models for Bounded Reasoners

ILLC DS-2021-08: **Michael Sejr Schlichtkrull**
Incorporating Structure into Neural Models for Language Processing