Thesis for The Degree of Doctor of Philosophy

Groupoid-Valued Presheaf Models of Univalent Type Theory

Fabian Ruch

GÖTEBORGS
UNIVERSITET

Division of Computing Science
Department of Computer Science and Engineering
University of Gothenburg
Göteborg, Sweden 2022

Groupoid-Valued Presheaf Models of Univalent Type Theory
Fabian Ruch

Division of Computing Science
Department of Computer Science and Engineering
University of Gothenburg

Technical Report Number 227D

Printed by Chalmers Digitaltryck
Göteborg, Sweden 2022

Abstract

One main goal of this thesis is to study constructive models of type theory with one univalent universe that interpret types by "presheaves" of groupoids.

A starting point is the fact that the groupoid model can be defined in a constructive metatheory. Therefore, its definition relativizes to presheaf models over arbitrary small index categories. This way we obtain what we call "naive" presheaf groupoid models of type theory with one univalent universe and propositional truncation.

These naive presheaf groupoid models of univalent type theory can for instance be used to refute the principle of excluded middle. However, they seem inadequate for using univalent type theory as an internal language for groupoids varying over a category.

One inadequacy of these models is that levelwise surjections in general fail to be internally surjective in that the propositional truncations of their fibres are not contractible. The reason for this failure is that propositional truncation in these models captures global rather than levelwise inhabitation of a type.

To resolve the inadequacies of these models we refine their interpretation of types. The interpretation of types in the refined models will be restricted to presheaves of groupoids that satisfy a non-trivial *patch condition* to account for levelwise inhabited propositions being forced to be contractible.

That patch condition can be expressed as having an algebra structure for a particular kind of lex modality which we call a *descent data operation*. Such a lex modality is in particular a strictly functorial operation on types and terms that preserves unit and dependent sum types up to isomorphism.

In this thesis we develop the notion of descent data operation as an extension of type theory. In particular, we show that its algebras are closed under type formers so that they can be used in an internal model construction. We apply this construction to the concrete descent data operation on the naive presheaf groupoid models. Finally, we show that a map in the resulting models is indeed internally surjective if and only if it is levelwise surjective.

Acknowledgements

I am very grateful to Thierry, Peter, Simon, Víctor, Bassel, Christian, Carlos, Nachi and Andrea for their insight, their inspiration, the discussions, and the collaborations. I am especially grateful to Ana, Thierry, Peter, Simon and Christian for their support. Thank you.

Contents

Introduction

One main goal of this thesis is to study constructive models of type theory with one univalent universe that interpret types by "presheaves" of groupoids.

In this introduction we give some intuition behind the notion of presheaf as a *variable* collection, recall the truth- and set-valued presheaf models of intuitionistic logic and type theory, and outline the contents and contribution of this thesis.

1 Presheaves as variable collections

The notion of presheaf can be seen as any kind of object that varies with respect to a parameter.

We restrict our attention to collections and, more specifically, groupoids because groupoids form a relatively simple model of Martin-Löf type theory with a univalent universe. We also consider truth values and sets because truth- and set-valued presheaf models are well-established and groupoids can be seen as generalized sets, which in turn can be seen as generalized truth values.

We assume a parameter to be given by a preorder or, more generally, a category. This way a parameter does not only specify what its instances are but also in what ways any two of its instances may be related. It is those relations that put constraints on how an object may vary with respect to the parameter.

As a first example of variable collections and their parameters, consider how *the knowledge* of the truth of a proposition may vary with what else we know in a particular state [51]. Knowledge states w, v are naturally preordered by $w \geq v$ expressing that w includes the knowledge of v. Further knowledge does not lead to contradictions of truth nor do we ever forget knowledge so if the proposition is known to be true at v then it will still be known to be

true at $w \geq v$. As a second example, consider how the collection and identity of objects may vary with the extent of the "domain" on which a kind of object can be defined or exist. A collection of (sub)domains U, V is naturally partially ordered by inclusion $U \subseteq V$. Objects defined, existent or identified on a domain V are in particular defined, existent or identified on any subdomain $U \subseteq V$. As a third and final example, consider how the collection of objects that constitute a complex structure may vary with what kind of "shapes" we are looking for in the given structure. Shapes and transformations between them naturally constitute a category such as the category $s, t : V \rightrightarrows E$ [53] specifying two shapes V and E together with two ways s and t (for "vertex", "edge", "source" and "target", respectively) of the shape V appearing in the shape E, the category of simplices [27], or the category of cubes [47]. Any object of shape T can be transformed into objects of shapes S that appear in the shape T via transformations $f : S \to T$. For instance, an edge in a variable set that varies over $V \rightrightarrows E$, i.e. a (directed) graph, can be transformed into a vertex by picking either its source or its target.

An $\mathcal{S}$-valued presheaf A is a *contravariant* functor from a category $\mathscr{C}$ into a category $\mathcal{S}$ of objects. In the examples the preorders and categories of parameter instances X indeed naturally act contravariantly on families of objects $A(X)$—the notion $\mathcal{S}$ of object is either truth values or sets in the examples.

A concept might be more naturally formalized as a presheaf than as a constant object. In the examples, for instance, we might not know whether something is true or false or we might prefer not to assume that something is either true or false, there might be no canonical choice of domain on which to study a certain kind of object, or we might prefer to build up and analyze complex structures in terms of simpler ones.

On the other hand, working with (the illusion of) a constant object is often simpler because we do not have to keep track of parameters and the constraints they impose. This way of working can also be more conceptual because it can expose structure that is independent of a particular notion of parameter, parameter instance and encoding of variable objects.

Type theories make it possible to work with presheaves in a fixed context as if they were constant objects. This capability of type theory to be used as a (convenient) language for presheaves leads us to a logical interest in presheaves to be used as a rich source of models for type theory—rich because of the generality of the notion of parameter. Since—by a soundness theorem—principles that are not valid for presheaves cannot be derived in type theory without imposing further principles, presheaves or rather their parameters

provide countermodels to such principles. We will recall a few examples of applications of that sort now.

2 Set-valued presheaf semantics

The first logical application of presheaf semantics is Kripke [51]'s semantics of intuitionistic propositional logic (**IPL**). In this semantics, formulas are interpreted as variable *truth values*. More precisely, formulas ϕ denote *truth-valued* presheaves over a preorder $\mathscr{P}$, i.e. families of truth values $\phi(X)$ indexed by $X : \mathscr{P}$ such that $\phi(X)$ implies $\phi(Y)$ whenever $Y \geq X$. Equivalently, a variable truth value can be encoded as an upwards-closed subset of parameters, i.e. a subset $\phi \subseteq \{X : \mathscr{P}\}$ such that $X \in \phi$ implies $Y \in \phi$ whenever $Y \geq X$. Note that variable truth values would not be able to detect the difference between transformations $f, g : Y \to X$ and this is why—for the semantics of **IPL** —we restrict our attention to parameters that are given by a preorder. The interpretations of the logical operations then need to maintain the upwards closure or monotonicity of truth. For instance, for the negation $\neg\phi$ to be true at X the proposition ϕ must not become true at *any* $Y \geq X$. The nonconstructive principle of excluded middle ($\phi \vee \neg\phi$) is thereby refuted by the presheaf semantics of **IPL** because a proposition that is not true (as opposed to being false) at X might become true at some $Y \geq X$, in which case neither the proposition nor its negation are true at X.

The second logical application of presheaf semantics is Kripke [51]'s immediate extension of his semantics to intuitionistic first-order logic (**IFOL**). In anticipation of presheaf models of type theory, we assume **IFOL** to be typed and to include an identity relation symbol Id_A for each type A of individuals. In Kripke semantics of **IFOL**, types A are interpreted as variable *sets* and formulas ϕ, which can now mention free variables ranging over individuals, are interpreted as variable *families* of truth values, which can be seen as truth values varying not only with respect to the given parameter but also with respect to instantiations of the free variables. More precisely, types A denote *set-valued* presheaves over a category $\mathscr{C}$, i.e. families of sets $A(X)$ and functions $A(X) \to A(Y), a \mapsto a|f$ indexed by $X : \mathscr{C}$ and $f : Y \to X$ subject to functoriality laws, and formulas ϕ in a context Γ (a product of types) denote truth-valued presheaves over the category $\int \Gamma$ of elements of (the presheaf denoted by) Γ, i.e. families of truth values $\phi(\gamma)$ indexed by $\gamma \in \Gamma(X)$ such that $\phi(\gamma)$ implies $\phi(\gamma|f)$ for all $f : Y \to X$. Equivalently, a variable family ϕ of truth values over a variable set Γ can be encoded as a family of sub-

sets $\phi(X) \subseteq \Gamma(X)$ such that $\gamma \in \phi(X)$ implies $\gamma|f \in \phi(Y)$ for all $f : Y \to X$. For instance, the identity relation $\mathrm{Id}_A(t,u)$ for terms t and u of type A in context Γ is defined to be true at $\gamma \in \Gamma(X)$ if and only if t and u denote equal elements at γ, i.e. $t(\gamma) = u(\gamma)$. The identity of terms t and u is a particular formula for which decidability $(\mathrm{Id}(t,u) \vee \neg\mathrm{Id}(t,u))$ is refuted by the presheaf semantics of **IFOL** because elements that are not identified (as opposed to being distinct) at X might become identified by some $f : Y \to X$, in which case t and u are neither identical nor distinct at X.

Presheaf models extend to intuitionistic higher-order logic (**HOL**) [75, 54] because set-valued presheaves are also closed under function types $A \Rightarrow B$ and truth values form a presheaf Ω such that families of truth values over A correspond to maps $A \Rightarrow \Omega$. Ignoring predicativity issues, the type Ω is interpreted at X by the set of subsets $p \subseteq \{(Y : \mathscr{C}, f : Y \to X)\}$ such that $f \in p$ implies $f \circ g \in p$ for every $g : Z \to Y$. For each map $p : \Gamma \to \Omega$ we have the family of truth values $[p](\gamma) :\Leftrightarrow \mathrm{id}_X \in p(\gamma)$ indexed by $\gamma \in \Gamma(X)$. The presheaf semantics for **HOL** validates propositional extensionality $(([p] \Rightarrow [q]) \wedge ([q] \Rightarrow [p]) \Rightarrow \mathrm{Id}_\Omega(p,q))$ and refutes various choice principles.

Martin-Löf type theory (**MLTT**) can also be interpreted in presheaves [39]. Types, which can now *dependent* on other types, are interpreted as variable *families* of sets. More precisely, (dependent) types A in a context Γ denote set-valued presheaves over the category $\int \Gamma$, i.e. families of sets $A(\gamma)$ and functions $A(\gamma) \to A(\gamma|f), a \mapsto \gamma|f$ indexed by $\gamma \in \Gamma(X)$ and $f : Y \to X$ subject to functoriality laws. For instance, the identity *type* for terms t and u is interpreted by the family of subsingletons $\mathrm{Id}(t,u)(\gamma) \coloneqq \{0 \mid t(\gamma) = u(\gamma)\}$ indexed by $\gamma \in \Gamma(X)$. It follows that identity types validate equality reflection (given $\Gamma \vdash p : \mathrm{Id}_A(t,u)$ we have $\Gamma \vdash t = u : A$) so that, in fact, presheaves form a model of *extensional* **MLTT**.

The model of **MLTT** in presheaves covers a universe type [40]. Universe types U can be used to define and quantify over types of structures because dependent types over A correspond to functions $A \Rightarrow \mathrm{U}$. Universe types can thus be seen as generalizing types of truth values and their interpretation in presheaves is indeed analogous with sets in place of truth values. Ignoring size issues, the type U is interpreted at X by the set of presheaves p over the slice category $\mathscr{C}/X$ over X, i.e. families of sets $P(f)$ and functions $P(f) \to P(f \circ g), p \mapsto p|g$ indexed by $f : Y \to X$ and $g : Z \to Y$. For each map $P : \Gamma \to \mathrm{U}$ we have the family of sets $\mathrm{El}(P)(\gamma) \coloneqq P(\gamma)(\mathrm{id}_X)$ indexed by $\gamma \in \Gamma(X)$.

Besides the logical applications of providing complete semantics and inde-

pendence results, presheaves model various applications in computer science such as normalization by evaluation [19, 4, 5, 23], higher-order abstract syntax [30, 38, 28], homotopy type theory [49], cubical type theory [17], parametricity [7, 10, 67], guarded recursion [11, 59, 52], and directed homotopy type theory [84].

In contrast to the extensionality principle validated by the type Ω of truth values, bijective types $P, Q : \mathrm{U}$ are not identified by $\mathrm{Id}(P, Q)$ in the presheaf model of **MLTT**.

On the other hand, **MLTT** with a universe type that satisfies an extensionality principle corresponding to extensionality of propositions can be interpreted in groupoids [42, 41]. In this model, closed types A are interpreted as (constant) sets of points $a, b \in A$ together with (constant) families of sets of paths $\alpha : a \nrightarrow b$ equipped with a groupoid structure. The points of a groupoid are thought of as the elements of a type and the paths as the identifications between those elements. The identity type for terms t and u of type A is then interpreted by the set $\mathrm{Id}(t, u) \coloneqq \{\alpha : t \nrightarrow u\}$ of paths α between t and u. This way the groupoid model refutes the principle of uniqueness of identity proofs ($\prod_{x,y:A} \prod_{p,q:\mathrm{Id}(x,y)} \mathrm{Id}(p, q)$ for arbitrary types A)—for instance, for the groupoid $\mathbb{Z}/2\mathbb{Z}$ with a single point $\bullet$ and a single non-trivial loop i ($i \neq \mathrm{id}_\bullet$ and, necessarily, $i \cdot i = \mathrm{id}_\bullet$) the principle would say $i = \mathrm{id}_\bullet$ in contradiction to the non-triviality [42, 41]. The universe type can now be interpreted by the groupoid whose points are sets and whose paths are bijections. This way bijective sets (and, more generally, isomorphic structures) get identified in the groupoid model in the sense that the canonical maps $\mathrm{Id}_{\mathrm{U}}(M, N) \to \mathrm{Iso}(\mathrm{El}(M), \mathrm{El}(N))$ are equivalences, i.e. the interpretation of U validates Voevodsky's univalence axiom. Observe that the universe type denotes a particular groupoid that is not a set because two sets M and $N : \mathrm{U}$ can be in bijection in several distinct ways—for instance, the two-element set $\mathrm{Bool} = \{0, 1\} : \mathrm{U}$ is in bijection with itself not only via the identity function but also via the distinct swap function defined by $\neg 0 = 1$ and $\neg 1 = 0$ [41].

In contrast to the presheaf models of **MLTT**, the groupoid model satisfies the axiom of choice and the law of excluded middle if and only if they hold in the metatheory.

It is natural at this point to combine presheaf and groupoid models. For instance, we can obtain models of **MLTT** that validate the univalence axiom (for one universe) and refute classical principles in this way.

3 Towards groupoid-valued presheaf models

The groupoid model can be constructed in a constructive metatheory. In fact, it can be constructed in extensional **MLTT** [41]. Since extensional **MLTT** can be interpreted in presheaves we obtain a model of (intensional) **MLTT** in groupoid-valued functors. More precisely, in this model a type A is interpreted by a *family of groupoids* $A(X)$ and *functors* $A(f) : A(X) \to A(Y)$ indexed by $X : \mathscr{C}$ and $f : Y \to X$ subject to functoriality laws. In particular, the interpretation of a type not only specifies a variable set of elements $a, b \in A(X)$ together with restrictions $a|f \in A(Y)$ but also a variable family of *sets* of identifications $\alpha : a \nrightarrow b$ between those elements together with restrictions $\alpha|f : a|f \nrightarrow b|f$. For instance, a type A over the groupoid $\mathbb{Z}/2\mathbb{Z}$ is given by a single groupoid A together with a single non-trivial action $A \to A, a \mapsto a|i$ satisfying $a|i|i = a$ for all points $a \in A$ and similarly for paths.

The resulting model—which we will refer to as the *naive* groupoid-valued presheaf model—validates the univalence axiom. Moreover, it refutes the axiom of choice and the law of excluded middle whenever the set-valued presheaf model of the metatheory does. However, it can also exhibit two (related) features that are perhaps unexpected.

Firstly, the naive groupoid-valued presheaf model can have strictly more (homotopy) sets than the corresponding set-valued presheaf model, even up to equivalence and classically. More precisely, a type A in the naive presheaf model, i.e. a groupoid-valued functor, can be checked to be a *homotopy* set [69] if and only if any two elements a and $b \in A(X)$ are identified in at most one way $a \nrightarrow b$. Define a groupoid-valued functor A to be a *strict* set or *discrete* if and only if any two elements a and $b \in A(X)$ are identified by $\alpha : a \nrightarrow b$ only if $b = a$ and $\alpha = \mathrm{id}_a$. Then it turns out that there are index categories $\mathscr{C}$ such that the naive presheaf model over $\mathscr{C}$ has types that are homotopy sets but that are not equivalent to any set-valued presheaf considered as a discrete type. For instance, take the homotopy set L over $\mathbb{Z}/2\mathbb{Z}$ which has a single path $\lambda : s \nrightarrow t$ between two distinct points s and t and whose action $-|i : L \to L$ swaps the points and reverses the path. No map $m : L \to P$ to any discrete type P over $\mathbb{Z}/2\mathbb{Z}$ is invertible. The reason is that there can be no map to L from a type with an action that has a fixed point but any map $m : L \to P$ to a discrete type P implies the existence of a fixed point $p_0 \in P$. Indeed, if P is discrete then the path $m(\lambda) : m(s) \nrightarrow m(t)$ must be the identity and hence $m(s)|i = m(s|i) = m(t) = m(s)$ by naturality and definition of the action on L

so that $p_0 \coloneqq m(s)$ is a fixed point. Now, if there was any map $m' : P \to L$ then $m'(p_0)$ would be a fixed point by naturality. However, neither s nor t is a (strict) fixed point of the action on L so that there can be no map $P \to L$ and, in particular, $m : L \to P$ cannot be invertible. Note that s and t are *homotopy* fixed points in the sense that λ is a path $s|i \mathrel{\tilde{\to}} s$ that satisfies a coherence condition called the cocycle condition, and similarly for t.

Secondly, levelwise invertible maps $m : A \to B$, i.e. maps such that each functor $m(X) : A(X) \to B(X)$ is invertible, in general fail to have an inverse $B \to A$ in the naive presheaf models. Consider again the example L of a homotopy set over $\mathbb{Z}/2\mathbb{Z}$ that is not equivalent to any strict set. The unique map from the type L to the unit type 1 is levelwise invertible because the groupoid L (at the only level $\bullet$) is contractible. However, as we saw, this map cannot be invertible since the unit type is a strict set. In particular, the point $c \in L$ given by the levelwise inverse $1 \to L$ cannot satisfy the strict naturality equation $c|i = c$.

These two features of the naive presheaf models are related as follows. Since in the constant groupoid model types *are*[1] homotopy sets if and only if they are equivalent to a constant set considered as a discrete groupoid, in the naive presheaf models the homotopy sets are exactly those types that are *levelwise* equivalent to strict sets. The discrepancy between the homotopy and strict notions of set in the naive presheaf models over some index categories can thus be seen as an instance of the failure of invertibility of levelwise invertible maps.

The naive notion of type in a model of groupoid-valued functors can be refined in such a way that a map between types is indeed invertible if (and only if) it is levelwise invertible. The resulting *refined* groupoid-valued presheaf model then has up to equivalence the same sets and propositions as the corresponding set- and truth-valued presheaf models. This parallels the fact that up to isomorphism the propositions in a set-valued presheaf model of **IFOL** or **HOL** are exactly the propositions in the corresponding truth-valued presheaf model of **IPL**.

The refined notion of type can be characterized *inside* the naive presheaf model using a particular strict left-exact modality [70, 74] that maps a type A—strictly functorially—to the type $D(A)$ of "virtual" elements or descent data families of elements, i.e. *pseudo*natural (as opposed to strictly natural) families of elements $a(X) \in A(X)$ *together with* identifications $a(f) : a(X)|f \mathrel{\tilde{\to}} a(Y)$

[1] Classical reasoning is required to construct an inverse to the quotient map from a homotopy set to the discrete groupoid of connected components.

for each $f : Y \to X$ subject to the *cocycle* condition $a(f)|g \cdot a(g) = a(f \circ g)$. There is then a canonical inclusion map $A \to D(A)$ of "actual" elements and, as we hinted at above, for an arbitrary type A this inclusion map is not invertible—it can even happen that $D(A)$ is inhabited when A is not. Types for which the canonical map $A \to D(A)$ is invertible are called modal. Types of the form $D(A)$ and discrete types such as inductive types are modal, and maps between modal types are invertible if and only if they are levelwise invertible.

The construction of the refined presheaf model is an instance of the general fact that the modal types with respect to a left-exact modality form a new model of **MLTT** with a univalent universe [70]. This internal construction of new models using left-exact modalities in the setting of **MLTT** corresponds to the internal construction of new (complete) Heyting algebras using nuclei and new elementary toposes using left-exact idempotent monads in the setting of **IPL** (**IFOL**) and **HOL**, respectively.

Two questions that arise from the construction of the refined groupoid-valued presheaf models are how they relate to the model structure in groupoid-valued functors of Bordg [12] and how they can be generalized to groupoid-valued *sheaf* models of **MLTT** with (higher) inductive types. Versions of the descent data operation used to construct the refined groupoid-valued presheaf models were used in Coquand, Mannaa and Ruch [20] to construct sheaf models of **MLTT** with one univalent universe and (higher) inductive types over two special cases of sites. This construction has been extended to a hierarchy of univalent universes and (higher) inductive types over arbitrary sites in Coquand, Ruch and Sattler [22].

4 Outline and contribution

This thesis is split into two parts. The first part develops the notion of descent data operation. The second part applies the theory developed in the first part to the construction of the refined groupoid-valued presheaf model. In the first part we work inside type theory extended with an abstract descent data operation. In the second part we work inside constructive set theory.

We begin the first part of this thesis by studying the notion of lex operation. This notion is defined as an extension of **MLTT** without assuming a notion of identity type. Lex operations act on types and terms in a way stable under substitution as well as preserving dependent sum and unit types up to isomorphism. Lex operations are automatically pointed as endofunctors. In the presence of identity types lex operations are shown to preserve finite ho-

motopy limits and equivalences, and the notions of modality and modal type for pointed endofunctors are defined. We show that a lex operation is a (lex) modality if and only if it maps types to modal types and inverts the map of the pointedness structure. A descent data operation is defined to be a lex operation that is a lex modality. The universe of modal types with respect to a descent data operation is shown to be modal. Inductive types are usually not modal and applying the modality produces a "homotopy" inductive type that satisfies the computation rules only up to homotopy. We end the first part by showing how *higher* inductive types can be used to model (higher) inductive types with strict computation rules in the internal model [70] of modal types.

We begin the second part of this thesis by recalling the categories with families (cwf) presentation of **MLTT** [25, 39] as a generalized algebraic theory [15]. We give a presentation of lex and descent data operations in the same style, and show that pointed pseudoendomorphisms [48] of cwfs induce lex operations. Next we reformulate the groupoid model in an elementary way and with a *primitive* notion of "paths over paths"[2] inspired by the relational models of Martin-Löf [64], Tonelli [79] and Hofmann [37]. Either formulation of the groupoid model is constructive but the reformulation depends on a type of extensional propositions that are strictly closed under dependent sum and product types. The metatheory of the groupoid model can be taken to be extensional **MLTT**. After showing that types of extensional propositions can be lifted to the (set-valued) presheaf model of extensional **MLTT** we describe the naive groupoid-valued presheaf model. We then discuss the refutation of the principle of excluded middle by this model in a few simple cases of index categories. We also observe the perhaps surprising fact that the principle is even refuted in cases where (classically) the corresponding set-valued presheaf model validates it. In those cases the naive presheaf model contains homotopy propositions that are not equivalent to any strict proposition. In the last chapter we construct the refined groupoid-valued presheaf model by showing that sending a presheaf to the presheaf of descent data is a descent data operation. We show that the modal types for this descent data operation support propositional truncation and that levelwise equivalences between them are invertible. We conclude with a description of what descent data look like and how the descent data operation "completes" groupoid-valued presheaves

[2]The notion of dependent paths $a \simeq_{\mu} a'$ over paths $\mu : \gamma \simeq \gamma'$ in Γ can be *derived* for the original encoding [40] of families of groupoids as functors $\Gamma \to \mathrm{Gpd}$. However, this involves a choice between non-dependent paths $a \simeq \mu^{-} a'$ in the fibre over γ and paths $\mu^{+} a \simeq a'$ in the fibre over γ'. Here, $\mu^{-} a'$ and $\mu^{+} a$ denote the transports of a' and a along μ, respectively.

under virtual elements in a few simple cases of index categories.

The main contribution of this thesis is the construction of the refined groupoid-valued presheaf model of type theory with one univalent universe and propositional truncation in a constructive metatheory. This model construction consists in defining the concrete descent data operation on the naive groupoid-valued presheaf model, instantiating the internal model construction from an abstract descent data operation, and showing that the resulting refined model supports propositional truncation. The notion of descent data operation has appeared in Coquand, Ruch and Sattler [22], the naive groupoid-valued presheaf model has appeared in Coquand, Mannaa and Ruch [20], and the internal model construction is essentially taken from Quirin [70].

Part I

Lex operation in type theory

Chapter 1

Introduction

Many constructions of new toposes from old ones can be formulated in terms of (co)algebras for finite limit-preserving endofunctors, also called left exact or simply lex. Recall that a (co)algebra for an endofunctor F is given by an underlying object A together with a structure map $F(A) \to A$ (or $A \to F(A)$, respectively). If F is a (co)monad, then the structure map is required to satisfy (co)unit and (co)associativity laws.

If a monad on a topos is lex and idempotent, then the category of its algebras is again a topos, and if a comonad on a topos is lex (actually, preservation of pullbacks suffices [44, Remark A4.2.3]), then the category of its coalgebras is again a topos. See Wraith [86] for the statements and the references there to Kock and Wraith [50] for proofs.

Examples of constructions of toposes that can be formulated in this way include presheaves on an internal category [57, 44], and hence sheaves on an internal site and the slice over any object, as well as the gluing along a lex functor [87].

We are interested in constructing new models of type theory from old ones. To that end we introduce the notion of lex operation that generalizes the notion of lex functor from topos theory to type theory and study some of its properties.

The notion of lex operation can be defined in basic dependent type theory using only unit, pair and function types. If we further assume identity types and function extensionality, then any lex operation automatically acts on homotopies, and preserves both equivalences and finite homotopy limits.

Lex operations are automatically pointed, i.e. any lex operation F comes equipped with a natural transformation $\eta : \mathrm{Id} \to F$. We can then ask a giv-

en lex operation F to be idempotent in the sense that both $F\eta_A$ and $\eta F_A : F(A) \to F(F(A))$ are equivalences for all types A. A lex operation that is idempotent will be called a descent data operation. For a given descent data operation D we can define the property of a given type being a (homotopy) pointed algebra for D. An algebra for a descent data operation will also be called a patch algebra. If we further assume a univalent universe then its subuniverse of algebras is itself an algebra. If certain higher inductive types exist then algebras support (higher) inductive types. In fact, a lex operation is idempotent if and only if it defines a lex modality in the sense of Rijke, Shulman and Spitters [74] so that its algebras or modal types are also closed under unit, pair, function and identity types. As a result, the algebras for a descent data operation form a new model of type theory [70].

Examples of lex operations include exponentiation $X \mapsto X^T$ by a fixed type T. If T is a (homotopy) proposition, then the lex operation it defines is idempotent. Exponentiation by a proposition is also called an open modality [74, Example 1.7] and includes the identity lex operation by exponentiation with the unit type. A type is a sheaf if and only if it is modal for all the dense or connected propositions induced by a site or a nucleus. Note that here modalities are used to characterize a subclass of types we might be interested in rather than for programming or expressing logical statements.

In this part of the thesis we work in type theory. As usual, we leave contexts implicit so that by a type A we mean a type A *in some context* Γ which we omit, and we refer to a type P in the extended context $\Gamma.A$ as a type family over A or a family of types $P(a)$ indexed by $a : A$. A similar convention applies to elements a of a type A (in some context Γ) and sections p of a type family P over A, that is families of elements $p(a) : P(a)$ indexed by $a : A$. Reindexing of a type family P over A along a map $f : A' \to A$ to the type family $(P(f(a')))_{a':A'}$ over A' by substitution will sometimes be denoted by Pf. Similarly, pf will denote the section of Pf obtained by reindexing a section p of P.

Chapter 2

Lex operation

In this chapter we investigate the properties of functors in type theory that act not only on types but also on families of types and that preserve both unit and dependent sum types. Such functors will be called *lex operations*.

Examples of lex operations include exponentiation $X \mapsto X^R$ by a fixed type R, in particular the functors Match_c that send a type X to its type $X^{[c]}$ of matching families on a cover $c : \mathrm{J}$, where $[c]$ denotes the proposition corresponding to c seen as an element of the type of propositions Ω.

The notion of lex operation can be defined in type theory without assuming a notion of identity type and we begin this chapter by investigating the properties of lex operations in that setting in Section 2.1. In the presence of identity types lex operations satisfy additional properties, which we investigate in Section 2.2.

2.1 Definition of lex operation

Definition 1. A *strict functor* F is given by two operations:

1. a type $F_0(A)$ for each type A
2. a map $F_1(f) : F_0(A) \to F_0(B)$ for each map $f : A \to B$

such that the two *functoriality* equations

1. $F_1(\mathrm{id}_A) = \mathrm{id}_{F_0(A)}$ and
2. $F_1(g \circ f) = F_1(g) \circ F_1(f)$

hold. ■

Note that the two functoriality equalities are *strict*. In fact, the definition of a functor does not refer to identity types at all.

Example 1. *Let R be a type, then the operation $F(A) = (R \to A)$ on types can be extended to a functor by $F(f) = \lambda_{d:R\to A}\lambda_{r:R} f(d(r))$; this is the definition of the underlying functor of the so-called reader monad for environments of type R [45].*

Definition 2. An *action of a functor F on families of types* is given by two operations:

1. a family $\widetilde{F}_0(P)$ over $F_0(A)$ for each family P over some type A
2. a section $\widetilde{F}_1(p)$ of $\widetilde{F}_0(P)$ for each section p of some family P

such that they commute with reindexing in the sense that the equations

1. $\widetilde{F}_0(Pf) = \widetilde{F}_0(P)F_1(f)$ and
2. $\widetilde{F}_1(pf) = \widetilde{F}_1(p)F_1(f)$

hold for all maps $f : A' \to A$. ■

As in Definition 1, the equalities for an action of a functor on families are *strict*.

As in Example 1, we will usually omit the subscripts and just write $F(A)$ and $F(f)$ as well as $\widetilde{F}(P)$ and $\widetilde{F}(p)$.

Example 2. *The exponentiation functor F can be expanded with an action on families of types as follows:*

$$\widetilde{F}(P) = (\prod\nolimits_{r:R} P(d(r)))_{d:R\to A}$$
$$\widetilde{F}(p) = (\lambda_{r:R} p(d(r)))_{d:R\to A}$$

Recall that a *(Tarski) universe* [61, 63, 39, Section 2.1.6] is given by a type U together with a decoding operation from elements $a : \mathrm{U}$ to types $\mathrm{El}\, a$, and that a type A is called U-*small* if there is an element $a : \mathrm{U}$ such that $A = \mathrm{El}\, a$.

We say that the action of a functor on (families of) types preserves U-smallness if it restricts to a universe U in the following sense.

Definition 3. Let U be a universe, then a functor F *preserves* U-*small types* if there is a map $\widehat{F_0} : \mathrm{U} \to \mathrm{U}$ that reflects the type operation F_0, i.e. for each $a : \mathrm{U}$ there is $\widehat{F_0}(a) : \mathrm{U}$ such that $F_0(\mathrm{El}\, a) = \mathrm{El}\, \widehat{F_0}(a)$. ■

Definition 4. Let U be a universe, then an action of a functor F on families of types *preserves* U-*small families* if it preserves U-small types (Definition 3) *and* there is a map $\widehat{\widetilde{F_0}} : (A \to \mathrm{U}) \to (F_0(A) \to \mathrm{U})$ for each type A that reflects the family operation $\widetilde{F_0}$, i.e. for each $p : A \to \mathrm{U}$ there is $\widehat{\widetilde{F_0}}(p) : F_0(A) \to \mathrm{U}$ such that $\widetilde{F_0}(\mathrm{El}\, p) = \mathrm{El}\, \widehat{\widetilde{F_0}}(p)$ and $\widehat{\widetilde{F_0}}(p \circ f) = \widehat{\widetilde{F_0}}(p) \circ F_1(f)$. ■

As before, the equalities for preservation of smallness are *strict*. Moreover, note that the index type A of a U-small family p is *not* required to be U-small itself.

Example 3. *The running example of exponentiation by a type* R *preserves* U-*small types and families whenever* R *is* U-*small. This is because we assume universes to reflect function types, i.e. we assume a map* $\widehat{\prod}$ *such that* $\prod_{\mathrm{El}\, r} \mathrm{El}\, p = \mathrm{El}(\widehat{\prod}_r p)$*, and hence* $\prod_R P$ *is* U-*small if* R *and* P *are.*

Let F be a functor with an action on families of types, then for each family P over A the canonical map

$$s_P : F(\sum_{x:A} P(x)) \to \sum_{x:F(A)} \widetilde{F}(P)(x)$$

is defined as the pairing of the map $\pi_1 = F(\mathrm{fst})$ and the section $\pi_2 = \widetilde{F}(\mathrm{snd})$. The map s_P is a map over $F(A)$ in the sense that the diagram

$$\begin{array}{ccc} F(\sum_{x:A} P(x)) & \xrightarrow{\quad s_P \quad} & \sum_{x:F(A)} \widetilde{F}(P)(x) \\ & {\scriptstyle \pi_1} \searrow \quad \swarrow {\scriptstyle \mathrm{fst}} & \\ & F(A) & \end{array}$$

commutes *strictly*.

Definition 5. A *lex operation* is given by a functor F with an action on families of types such that the canonical maps

1. $F(\mathrm{Unit}) \to \mathrm{Unit}$ and
2. $F(\sum_{x:A} P(x)) \to \sum_{x:F(A)} \widetilde{F}(P)(x)$ for each family P over A

are (*strict*) isomorphisms. ■

Given a lex operation F, the inverses of the canonical maps induce elements

1. $\top : F(\mathrm{Unit})$ and
2. $\langle t, u\rangle : F(\sum_{x:A} P(x))$ for each $t : F(A)$ and $u : \widetilde{F}(P)(t)$

which satisfy the *strict* equations

1. $\top = t$ for each $t : F(\mathrm{Unit})$,
2. $\pi_1\langle t, u\rangle = t$,
3. $\pi_2\langle t, u\rangle = u$, and
4. $\langle \pi_1 t, \pi_2 t\rangle = t$ for each $t : F(\sum_{x:A} P(x))$.

so that $F(\mathrm{Unit})$ and $F(\sum_{x:A} P(x))$ have the structure of a unit type and a dependent sum type (of the family $\widetilde{F}(P)$ over $F(A)$), respectively.

Example 4. *We have seen that exponentiation by a type R is functorial and acts on families of types. That exponentiation preserves unit types, i.e. $(R \to \mathrm{Unit}) \cong \mathrm{Unit}$, follows from the η laws for function and unit types, and the fact that it also preserves dependent sum types, i.e.*

$$(R \to \sum_{x:A} P(x)) \cong \sum_{d:R\to A} \prod_{r:R} P(d(r)),$$

is an instance of the type-theoretic axiom of choice [63, 60].

Proposition 1. *The identity map* Id *is a lex operation, and the composite map $G \circ F$ is a lex operation whenever G and F are lex operations.*

Proof. We only discuss the preservation of dependent sum types by the composite $G \circ F$ of two lex operations G and F. The canonical morphism $s_{GF,P}$ is an isomorphism because it factors as follows:

$$\begin{array}{ccc} G(F(\sum_{x:A} P(x))) & \xrightarrow{\quad s_{GF,P} \quad} & \sum_{x:G(F(A))} \widetilde{G}(\widetilde{F}(P))(x) \\ & {\scriptstyle \cong}\searrow \quad \nearrow{\scriptstyle \cong} & \\ & G(\sum_{x:F(A)} \widetilde{F}(P)(x)) & \end{array}$$

by the universal property of $\sum_{x:G(F(A))} \widetilde{G}(\widetilde{F}(P))(x)$ and the assumption that G and F are lex operations. □

Proposition 2. *Let F be a functor. If F preserves terminal objects, then F is pointed. Moreover, the natural transformation $\eta : \mathrm{Id} \to F$ is uniquely determined in that case by $\eta_A(a) = F(\hat{a})(\top)$ where $\hat{a} = \mathrm{const}_a = \lambda_{x:\mathrm{Unit}} a$ and $\top = \eta_{\mathrm{Unit}}(\mathrm{tt}) : F(\mathrm{Unit})$.*[1]

Proof. Any natural transformation $\eta : \mathrm{Id} \to F$ is uniquely determined by the component $\eta_{\mathrm{Unit}} : \mathrm{Unit} \to F(\mathrm{Unit})$, that is the element $\eta_{\mathrm{Unit}}(\mathrm{tt})$. Indeed, for each $a : A$ we have

$$\eta_A(a) = \eta_A(\mathrm{const}_a(\mathrm{tt})) = F(\mathrm{const}_a)(\eta_{\mathrm{Unit}}(\mathrm{tt}))$$

by naturality. Moreover, the assignment $a \mapsto F(\mathrm{const}_a)(\eta_{\mathrm{Unit}}(\mathrm{tt}))$ is natural because for each $f : B \to A$ and $b : B$ we have

$$F(\mathrm{const}_{f(b)})(\eta_{\mathrm{Unit}}(\mathrm{tt})) = F(f)(F(\mathrm{const}_b)(\eta_{\mathrm{Unit}}(\mathrm{tt})))$$

In conclusion, if a functor F preserves terminal objects, then there is exactly one natural transformation $\mathrm{Id} \to F$ up to strict equality. □

In anticipation of their role in the presence of identity types and to avoid introducing another terminology like *display map* [77, 68, 43] only for the course of this section, we will start calling a map $p : T \to A$ that is isomorphic to the projection map $\mathrm{fst}_P : \sum_{x:A} P(x) \to A$ for some family P over A a *fibration* [31, 8] even though we do not assume a notion of equivalence of types at this point. We will refer to the pair (P, s) of a family P and an isomorphism $s : T \xrightarrow{\sim} \sum_{x:A} P(x)$ from p to fst_P over A as a *fibration structure* on the map $p : T \to A$.

The properties of fibrations we are interested in at this point are the existence of pullbacks along arbitrary maps $f : A' \to A$ as well as the closure under composition and pullbacks. These properties correspond to the fact that one can take the maps that are isomorphic to projection maps to be the class of fibrations of a *fibration category* [13] or *clan* [46, Definition 1.1.1] associated to the type theory. [8, Theorem 3.2.5]

A functor (between categories with a notion of fibration, like the fibration category or clan associated to a type theory) that preserves terminal objects, fibrations and pullbacks of fibrations corresponds to the notion of *clan morphism* [46, Definition 1.1.8] with the significant difference that preservation of fibrations is not merely a property of a functor here because being a fibration is not a property of but given by a fibration structure (P_f, s_f) on a

[1] We owe this observation to Dan Licata.

map $f : B \to A$. What we will show in the remainder of this section is that lex operations can be seen as endomorphisms on the associated clan.

Proposition 3. *Let F be a lex operation and (f, P_f, s_f) a fibration, then $F(f)$ has a fibration structure given by $s_{P_f} \circ F(s_f) : F(B) \to \sum_{x:F(A)} \widetilde{F}(P_f)$.*

Proof. Since functors preserve commuting diagrams and in particular isomorphisms, the isomorphism $s_{F(f)} = s_{P_f} \circ F(s_f)$ is a fibration structure for the map $F(f) : F(B) \to F(A)$. □

We recall from Avigad, Kapulkin and Lumsdaine [8] that canonical pullbacks of fibrations, i.e. maps isomorphic to a projection map, indeed exist and are again fibrations:

Proposition 4. *Let (f, P_f, s_f) be a fibration over A and $g : A' \to A$ any map, then the projection map $\sum_{x:A'} P_f(g(x)) \to A'$, which has a fibration structure, together with the map*

$$g^* : \sum_{x:A'} P_f(g(x)) \to B \qquad (a', p) \mapsto s_f^{-1}(g(a'), p)$$

is a pullback of f along g, that is the square

$$\begin{array}{ccc} \sum_{x:A'} P_f(g(x)) & \xrightarrow{g^*} & B \\ \downarrow{\scriptstyle \mathrm{fst}} & & \downarrow{\scriptstyle f} \\ A' & \xrightarrow{g} & A \end{array}$$

Diagram 1: Base change

is a pullback square. Moreover, the equations

1. $\mathrm{id}_A^* = s_f^{-1}$ *and*
2. $(g \circ h)^* = g^* \circ h^*$

hold.

Proof. See Avigad, Kapulkin and Lumsdaine [8, Lemmas 3.2.7 and 3.2.9] for the existence of pullbacks of projections and the preservation of projections under pullbacks.

□

It immediately follows that lex operations preserve pullbacks of fibrations and binary products.

Proposition 5. *Let F be a lex operation and (p, q) a pullback of a fibration (f, P_f, s_f) along a map $g : A' \to A$, then $(F(p), F(q))$ is a pullback of $F(f)$ along $F(g)$.*

Proof. Without loss of generality, we can assume $f = \mathrm{fst}$, $P_f = \sum_{x:A} P(x)$ and $s_f = \mathrm{id}$ for some family P over A. We show that the structure isomorphism s_{Pg} of the image of the canonical pullback of f along g is the canonical cone morphism from $F(\mathrm{fst})$ and $s_P \circ F(g^*)$ to $\mathrm{fst}_{\widetilde{F}(P)F(g)}$ and $F(g)^*$. Since $\widetilde{F}(Pg) = \widetilde{F}(P)F(g)$ we have that $s_{\widetilde{F}(Pg)}$ has the right type and $\mathrm{fst}_{\widetilde{F}(P)F(g)} \circ s_{Pg} = F(\mathrm{fst})$. To show $F(g)^* \circ s_{\widetilde{F}(Pg)} = s_P \circ F(g^*)$ we note:

$$\widetilde{F}(\mathrm{snd}_P) \circ F(g^*) = \widetilde{F}(\mathrm{snd}_P \circ g^*) = \widetilde{F}(\mathrm{snd}_{Pg})$$

□

Let F be a functor which preserves terminal objects and acts on families, then for a family P over some type A and an element $a : A$ the map

$$f_{P,a} : F(P(a)) \to \widetilde{F}(P)(\eta_A(a)) \qquad x \mapsto \pi_2(F((a, -))(x)) \tag{2.1}$$

factors the map

$$\tilde{\eta}_{P,a} : P(a) \to \widetilde{F}(P)(\eta_A(a)) \qquad p \mapsto \pi_2(\eta_{\sum_{x:A} P(x)}((a, p))) \tag{2.2}$$

through $\eta_{P(a)} : P(a) \to F(P(a))$.

From the preservation of pullbacks (and the unit type) it follows that lex operations preserve fibres of families in the following sense.

Lemma 6. *Let F be a lex operation, then F preserves (*strict*) fibres in the sense that the map*

$$f_{P,a} : F(P(a)) \to \widetilde{F}(P)(\eta_A(a))$$

from Equation 2.1 is an isomorphism. In particular, F preserves constant families in the sense that in the case of $P(x) = B$ for all $x : \mathrm{Unit}$ and $a = \mathrm{tt}$ this says that the map

$$f_B : F(B) \to \widetilde{F}((B)_{x:\mathrm{Unit}})(\top) \qquad x \mapsto \pi_2(F((\mathrm{tt}, -))(x)) \tag{2.3}$$

is an isomorphism.

Proof. For each $x : F(P(a))$ we have

$$\pi_2(F((a,-))(x)) = \pi_2(F(\hat{a}^*)(F((\mathrm{tt},-))(x))) = \pi_2(F((\mathrm{tt},-))(x)),$$

and both $F((\mathrm{tt},-)) : F(P(a)) \to F(\sum_{x:\mathrm{Unit}} P(a))$ and $\pi_2 : F(\sum_{x:\mathrm{Unit}} P(a)) \to \widetilde{F}((P(a))_{x:\mathrm{Unit}})(\top) = \widetilde{F}(P\hat{a})(\top) = \widetilde{F}(P)(\eta_A(a))$ are isomorphisms. □

We recall that the binary product $A \times B$ of types A and B can be defined either as a choice of pullback of the constant fibration $\sum_{x:\mathrm{Unit}} B \to \mathrm{Unit}$ along the terminal map $A \to \mathrm{Unit}$, which is how it is done in Joyal [46] for instance, or as the dependent sum $\sum_{x:A} B$ of the constant family $(B)_{x:A}$, which is how it is usually done in type theory [63].

Proposition 7. *Let F be a lex operation, then*

1. *$F(T)$ is a terminal object whenever T is a terminal object, and*
2. *$(F(p), F(q))$ is a product of $F(A)$ and $F(B)$ whenever (p, q) is a product of A and B.*

Concretely, the inverse of $!_{F(\mathrm{Unit})} : F(\mathrm{Unit}) \to \mathrm{Unit}$ is given by $x \mapsto \top$ and the inverse of $(F(\mathrm{fst}), F(\mathrm{snd})) : F(A \times B) \to F(A) \times F(B)$ is given by $(x, y) \mapsto \langle x, f_B(y)\rangle$. $\top$ and $\langle -,-\rangle$ are defined below Definition 5, and f_B is defined in Equation 2.3.

Proof. It is clear that F preserves terminal objects because being a terminal object is invariant under isomorphism, F preserves isomorphisms and F preserves the unit type (the canonical terminal object) by definition.

For a proof of the preservation of binary products given the preservation of pullbacks of fibrations (Proposition 5) see Joyal [46, Proposition 1.1.10]. □

The notion of a lex operation is also implicitly used in Coquand and Paulin [21] for a description of the notion of inductive type; a (generalized) inductive type A is given by a family of lex operations F^i indexed by some type I, a family of maps $\mathrm{intro}^i : F^i(A) \to A$ (corresponding to the introduction rules), and a section $\mathrm{elim}_{P,d}$ of each dependent type P over A with a family of maps $d^i : \sum_{x:F^i(A)} \widetilde{F}^i(P)(x) \to P(\mathrm{intro}^i x)$ (corresponding to the elimination rule) such that the strict equalities $\mathrm{elim}_{P,d}(\mathrm{intro}^i x) = d^i(x)(\widetilde{F}^i(\mathrm{elim}_{P,d})(x))$ are satisfied for each $i : I$ and $x : F^i(A)$.

Having defined the notion of functor in type theory and when a functor is a lex operation, we continue in the next section with a justification of the attribute *lex* by showing that in the presence of identity types and function extensionality lex operations preserve equivalences, identity types, and finite *homotopy* limits.

2.2 In the presence of identity types

From this moment on we assume identity types and function extensionality in our type theory.

This section is dedicated to showing that in this setting lex operations (see Definition 5) preserve equivalences, identity types, and finite *homotopy* limits.

With identity types at hand, we can now give the motivation for thinking of projection maps as fibrations: Gambino and Garner [31] showed that every map $f : A \to B$ can be factored as $i_f : A \to \sum_{x:B} \mathrm{fib}_f(x), a \mapsto (f(a), a, \mathrm{refl})$ followed by the fibration $\sum_{x:B} \mathrm{fib}_f(x) \to B$ such that i_f has the left lifting property against all fibrations, which we recall from Gambino and Garner [31] in the special case of $f = \mathrm{id}_A$:

Lemma 8. *The function* $\mathrm{refl} : A \to \sum_{A\times A} \mathrm{Id}_A$ *has the left lifting property against all fibrations.*

Proof. See Gambino and Garner [31, Lemma 11] for a proof. □

Towards the goal of this section, we will first show that functors as defined in Definition 1 always preserve equivalences by showing that they act on homotopies.

We recall the method of homotopy induction from the HoTT book [69], which is equivalent to function extensionality and corresponds to path induction at function type.

Theorem 9. *Let P be a family of types $P(f, g, H)$ indexed by functions $f, g : A \to B$ and homotopies $H : f \sim g$ for some types A and B. Given a section $b : \prod_{f:A\to B} P(f, f, \mathrm{hrefl}_f)$, there exists a section $s : \prod_{f,g:A\to B, H:f\sim g} P(f, g, H)$ together with paths $s(\mathrm{hrefl}_f) \equiv b(f)$ for each function $f : A \to B$.*

Proof. See Corollary 5.8.6 in [69] for a proof. □

Proposition 10. *Let F be a (pointed) functor and $H : f \sim g$ a homotopy, then there is a homotopy $F_2(H) : F_1(f) \sim F_1(g)$ such that*

$$F_2(\mathrm{hrefl}_f) \equiv \mathrm{hrefl}_{F_1(f)}$$
$$(F_2(H) \circ \eta \equiv \mathrm{ap}_\eta \circ H)$$

Proof. By homotopy induction it suffices to consider the case $H = \mathrm{hrefl}_f$. In this case we need to construct a homotopy $H' : F_1(f) \sim F_1(f)$ together with a path $H' \circ \eta_A \equiv \mathrm{ap}_{\eta_B} \circ \mathrm{hrefl}_f$ for each function $f : A \to B$, but the constant

function $\mathrm{hrefl}_{F_1(f)} = \lambda_{x:F_0(A)}\mathrm{refl}_{F_1(f)(x)}$ is such a homotopy between $F_1(f)$ and itself such that there is a path $\mathrm{hrefl}_{F_1(f)}(\eta_A(a)) = \mathrm{refl}_{\eta_B(f(a))} \equiv \mathrm{ap}_{\eta_B}\mathrm{refl}_{f(a)}$ for each $a : A$. Therefore, there is a section $F_2(H)$ of $F_1(f) \sim F_1(g)$ over f, $g : A \to B$ and $H : f \sim g$ such that $F_2(\mathrm{hrefl}_f) \equiv \mathrm{hrefl}_{F_1(f)}$ and $F_2(H) \circ \eta_A \equiv \mathrm{ap}_{\eta_B} \circ H$. □

Note that the previous proposition only says that functors preserve the identity homotopy $\mathrm{hrefl}_f : f \sim f$ at each function $f : A \to B$ *up to path equality.*

Proposition 11. *Let F be a functor and $f : A \to B$ an equivalence, then $F(f) : F(A) \to F(B)$ is an equivalence.*

Proof. Let $g : B \to A$ be a homotopy inverse of f with $H : g \circ f \sim \mathrm{id}_A$ and $I : f \circ g \sim \mathrm{id}_B$, then we have homotopies $F(g) \circ F(f) \sim \mathrm{id}_{F(A)}$ and $F(f) \circ F(g) \sim \mathrm{id}_{F(B)}$ by Proposition 10, that is a homotopy inverse of $F(f) : F(A) \to F(B)$. □

Remark 1. *The assumption of function extensionality is strictly necessary to prove that lex operations preserve equivalences.*

For the running example $A \mapsto (R \to A)$ of a lex operation preservation of equivalences unfolds to postcomposition $(R \to A) \to (R \to B)$ with any equivalence $f : A \to B$ for any pair of types A and B being an equivalence.

If exponentiation by a type R preserves equivalences then for all functions g and $g' : R \to C$ we have that $g \sim g'$ implies $g \equiv g'$ by applying the assumption to the equivalence $\sum_{c:C}\sum_{c':C} c \equiv c' \to C, (c, c', p) \mapsto c$. In fact, we have that for all dependent *functions g and $g' : \prod_{r:R} C(r)$ with a homotopy $g \sim g'$ there is a path $g \equiv g'$. This is enough to prove contractibility of the type $\sum_{h':\prod_{r:R} P(r)} h \sim h'$ for each function $h : \prod_{r:R} P(r)$ and hence function extensionality. [69, Exercise 2.16, Theorem 5.8.4]*

That lex operations not only preserve fibrations (see Proposition 3) but also equivalences (since all functors do) means that they correspond to endomorphisms on the *fibration category* [13] that Avigad, Kapulkin and Lumsdaine [8, Theorem 3.2.5] associate to the type theory.

The preservation of fibrations and equivalences implies the preservation of (families of) contractible types by lex operations:

Corollary 12. *Let F be a lex operation and $P(x)$ a family of contractible types over $x : A$, then $\widetilde{F}(P)(x)$ is a family of contractible types over $x : F(A)$.*

Proof. $\widetilde{F}(P)$ is a family of contractible types if and only if the first projection $\mathrm{fst}_{\widetilde{F}(P)} : \sum_{x:F(A)} \widetilde{F}(P)(x) \to F(A)$ is an equivalence, but $\mathrm{fst}_{\widetilde{F}(P)}$ is isomorphic to $F(\mathrm{fst}_P) : F(\sum_{x:A} P(x)) \to F(A)$, which is an equivalence by the assumption that P is a family of contractible types and the proposition that F as a functor preserves equivalences. Therefore, $\mathrm{fst}_{\widetilde{F}(P)}$ is an equivalence by 2–out–of–3. □

We now show that lex operations preserve a notion of identity type where the computation rule is only required to hold *up to path equality*. We recall its definition from the HoTT book [69]:

Definition 6. An *identity system* over a type A is given by a family $I(a,b)$ indexed by $a, b : A$ together with a section $r : \prod_{a:A} R(a,a)$ such that for every family $C(a,b,p)$ indexed by $a, b : A$, $p : I(a,b)$ and section $c : \prod_{a:A} C(a,a,r(a))$ there is a section $l(c) : \prod_{a,b:A,p:I(a,b)} C(a,b,p)$ together with a homotopy $L(c) : \prod_{a:A} l(c,a,a,r(a)) \equiv c(a)$. ■

Note that being an identity system (over a type A) is a homotopy property of the tuple (I,r), and that the family Id_A together with the section $\mathrm{refl}_A : \prod_{x:A} \mathrm{Id}_A(x,x)$ indeed is an identity system over A.

There is exactly one identity system up to equivalence, any two identity systems are equivalent. Given an identity system (I,r), denote the inverses of the transport maps $t_{a,b} : \mathrm{Id}_A(a,b) \to I(a,b)$ (cf. Theorem 5.8.4 (iv) in [69]) for each $a, b : A$ by

$$t_{a,b}^{-1} : I(a,b) \to \mathrm{Id}_A(a,b). \tag{2.4}$$

Note that there is a path $t_{a,a}^{-1}(r(a)) \equiv \mathrm{refl}_a$ for each $a : A$ because by definition $t_{a,a}(\mathrm{refl}_a) = r(a)$ for all $a : A$.

Proposition 13. *Let F be a lex operation, then for every type A the family $\widetilde{F}(\mathrm{Id}_A)$ together with the section $\widetilde{F}(\mathrm{refl}_A)$ is an identity system (see Definition 6) over the type $F(A)$.*

Proof. By Theorem 5.8.4 in [69] it suffices to show that for any $x : F(A)$, the type $\sum_{y:F(A)} \widetilde{F}(\mathrm{Id}_A)\langle x,y\rangle$ is contractible. This follows from Corollary 12 and the fact that Id_A together with refl_A is an identity system. □

Together with the preservation of finite products (see Proposition 7) the preservation of identity types implies the preservation of all finite homotopy limits by lex operations, in particular homotopy pullbacks whose definition we recall from Wellen [85, Definition 3.2.1]:

Definition 7. A homotopy $H : f \circ p \sim g \circ q : P \to C$ is a *homotopy pullback* of $f : A \to C$ along $g : B \to C$ if the *comparison map* $\mathrm{gap}_H : P \to P(f,g)$ into the *canonical homotopy pullback* $P(f,g) = \sum_{x:A,y:B} f(x) \equiv g(y)$ is an equivalence. ■

The previous definition of homotopy pullback is independent of the choice of identity system for the canonical homotopy pullback:

Lemma 14. *Let (I,r) be an identity system over a type C, then a homotopy $H : f \circ p \sim g \circ q : P \to C$ is a homotopy pullback if and only if the* comparison map

$$g_H : P \to \sum_{x:A,y:B} I(f(x), g(y)) \tag{2.5}$$

induced by H and $t_{a,b} : \mathrm{Id}_A(a,b) \to I(a,b)$ (see Equation 2.4) is an equivalence.

Proof. By definition, the map g_H factors through the canonical homotopy pullback as the composite

$$P \xrightarrow[\mathrm{gap}_H]{} P(f,g) = \textstyle\sum_{x:A,y:B} f(x) \equiv g(y) \xrightarrow[\mathrm{total}(r)]{\simeq} \sum_{x:A,y:B} I(f(x), g(y))$$

and, hence, g_H is an equivalence if and only if gap_H is an equivalence by 2–out–of–3, i.e. g_H is an equivalence if and only if H is a homotopy pullback. □

Preservation of identity types in the sense of Proposition 13 by a functor (not necessarily a lex operation) induces an action on homotopies

$$f \sim g \to F(f) \sim F(g)$$

which in the case of lex operations coincides with the one from Proposition 10 up to path equality:

Lemma 15. *Let F be a lex operation, then for every homotopy $H : f \sim g$ there are paths $F_2(H)(x) \equiv t^{-1}(\widetilde{F}(H)(x))$ (see Equation 2.4) for $x : F(A)$.*

Proof. By homotopy induction it suffices to consider the cases $H = \mathrm{hrefl}_f$ and give paths $F_2(\mathrm{hrefl}_f)(x) \equiv t^{-1}(\widetilde{F}(\mathrm{hrefl}_f)(x))$ for $x : F(A)$, which we have because by Proposition 10 and the remark below Equation 2.4 both sides of the equation are path equal to $\mathrm{refl}_{F(f)(x)}$. □

Proposition 16. *Let F be a lex operation. If $H : f \circ p \sim g \circ q$ is a homotopy pullback of $f : A \to B$ along $g : B \to C$, then $F_2(H) : F_1(f) \circ F_1(p) \sim F_1(g) \circ F_1(q)$ is a homotopy pullback of $F_1(f) : F_0(A) \to F_0(C)$ along $F_1(g) : F_0(B) \to F_0(C)$.*

Proof. The map $\mathrm{gap}_H : P \to P(f,g)$ is an equivalence by assumption and hence $F(\mathrm{gap}_H) : F(P) \to F(P(f,g))$ is an equivalence by Proposition 11. $\widetilde{F}(\mathrm{Id}_C)$ and $\widetilde{F}(\mathrm{refl}_C)$ form an identity system over $F(C)$ by Proposition 13 and the canonical map $u : F(P(f,g)) \to \sum_{x:F(A),y:F(B)} \widetilde{F}(\mathrm{Id}_C)\langle F(f)(x), F(g)(y)\rangle$ is an isomorphism because F preserves strict pullbacks (see Proposition 5). From this it follows that $g_{F_2(H)} \sim u \circ F(\mathrm{gap}_H)$ (see Lemma 15) is an equivalence by 2–out–of–3 and hence $F_2(H)$ is a homotopy pullback by Lemma 14. □

Corollary 17. *For F a lex operation, if p and q are a homotopy pullback (in the sense that $H =$ hrefl is a homotopy pullback) of f and g, then $F_1(p)$ and $F_1(q)$ are a homotopy pullback of $F_1(f)$ and $F_1(g)$.*

Proof. F preserves strictly commuting diagrams because it is a functor, and the identity homotopy hrefl by Proposition 10, so by Proposition 16 the homotopy $F_2(\mathrm{hrefl}) \equiv \mathrm{hrefl}$ is a homotopy pullback of $F_1(p)$ and $F_1(q)$. □

Corollary 18. *For F a lex operation and $f : A \to B$ an arbitrary function, the gap map $F(\mathrm{fib}_f(b)) \to \mathrm{fib}_{F(f)}(\eta_B(b))$ is an equivalence for each $b : B$. Moreover, the gap map factorizes the fibre map $\mathrm{fib}_f(b) \to \mathrm{fib}_{F(f)}(\eta_B(b))$ through $\eta_{\mathrm{fib}_f(b)} : \mathrm{fib}_f(b) \to F(\mathrm{fib}_f(b))$.*

Proof. The fibre of f over b is a homotopy pullback of the constant function $\hat{b} = \mathrm{const}_b : \mathrm{Unit} \to B$ along f such that the gap maps $X \to \mathrm{fib}_f(b)$ strictly commute with the projections:

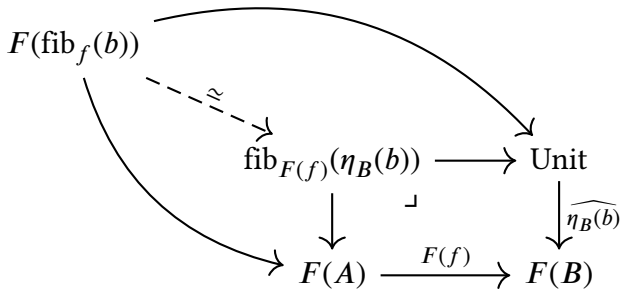
$$\begin{array}{ccc} \mathrm{fib}_f(b) & \xrightarrow{!} & \mathrm{Unit} \\ \downarrow{\scriptstyle \mathrm{fst}} & \lrcorner & \downarrow{\scriptstyle \hat{b}} \\ A & \xrightarrow{f} & B \end{array}$$

Diagram 2: *The fibre of a map over a point is the homotopy pullback of the constant map along the map.*

By preservation of homotopy pullbacks (see Proposition 16) $F_2(\mathrm{snd}) : F(f) \circ F(\mathrm{fst}) \sim F(\hat{b}) \circ F(!)$ is a homotopy pullback, and by preservation of the unit type and Wellen [85, Lemma 3.2.3 (d)] also the outer square, where $F(\hat{b})$ and $F(!)$ are replaced by the strictly isomorphic $\widehat{\eta_B(b)} = \eta_B \circ \hat{b} = F(\hat{b}) \circ \eta_{\mathrm{Unit}}$ and $! = ! \circ F(!)$, respectively, is a homotopy pullback:

$$\begin{array}{ccccc} F(\mathrm{fib}_f(b)) & \longrightarrow & F(\mathrm{Unit}) & \xleftarrow[\eta_{\mathrm{Unit}}]{\cong} & \mathrm{Unit} \\ \downarrow & \lrcorner & \downarrow & & \downarrow{\scriptstyle \hat{b}} \\ F(A) & \longrightarrow & F(B) & \xleftarrow[\eta_B]{} & B \end{array}$$

We hence have two homotopy pullback squares over the same cospan and an equivalence between them (the unique map from $F(\mathrm{fib}_f(b))$ to $\mathrm{fib}_{F(f)}(\eta_B(b))$ such that whiskering the inner square yields the outer square is an equivalence by Rijke [73, Lemma 10.2.3]):



□

Corollary 19. *Let F be a lex operation and $P(x)$ a family of mere propositions over $x : A$, then $\widetilde{F}(P)(x)$ is a family of mere propositions over $x : F(A)$.*

Proof. $\widetilde{F}(P)$ is a family of mere propositions if and only if $\mathrm{fst} : \sum_{F(A)} \widetilde{F}(P) \to F(A)$ is an embedding, or, equivalently, if the diagram

$$\begin{array}{ccc} \sum_{F(A)} \widetilde{F}(P) & = & \sum_{F(A)} \widetilde{F}(P) \\ \| & & \downarrow{\scriptstyle \mathrm{fst}} \\ \sum_{F(A)} \widetilde{F}(P) & \xrightarrow{\mathrm{fst}} & F(A) \end{array}$$

is a homotopy pullback. Now, this diagram is (isomorphic to) the image of the diagram

$$\begin{array}{ccc} \sum_A P & = & \sum_A P \\ \| & & \downarrow{\scriptstyle \mathrm{fst}} \\ \sum_A P & \xrightarrow{\mathrm{fst}} & A \end{array}$$

which is a homotopy pullback because $\mathrm{fst} : \sum_A P \to A$ is an embedding by assumption. Since F preserves homotopy pullbacks by Corollary 17, the claim follows. □

It turns out that our running example of a lex operation does not just preserve families of types of homotopy level $n < 0$ but of arbitrary homotopy level:

Example 5. *Let R be any type, then the lex operation F given by $F(A) = (R \to A)$ does not only preserve families of contractible types and families of mere propositions but F preserves families of n-truncated types $P(x)$ for every $n \geqslant -2$: $\widetilde{F}(P)(d) = \prod_{r:R} P(d(r))$ is n-truncated for each $d : F(A)$ whenever $P(x)$ is n-truncated for each $x : A$. [69, Theorem 7.1.9]*

However, we do not have a proof (or disproof) that lex operations preserve families of n-truncated types for arbitrary $n \geqslant -2$ in general.

Chapter 3

Descent data operation

In the previous chapter we looked at functorial operations on (families of) types and defined the notion of lex operation. We then showed that lex operations preserve finite products and pullbacks of fibrations as well as equivalences and finite homotopy limits (in the presence of identity types and function extensionality).

In this chapter we look at lex operations that are lex modalities in the sense of Rijke, Shulman and Spitters [74]. The interest in lex modalities lies in them giving rise to new models of type theory [70]. The resulting notion is that of a descent data operation.

We show that the algebras or modal types of a descent data operation are closed under dependent sums, and if the descent data operation acts on a univalent universe then they support a (univalent) universe. Moreover, we show that if certain higher inductive types exist then the submodel of modal types induced by a descent data operation supports (higher) inductive types.

3.1 Definition of descent data operation

Definition 8. A *descent data operation* is given by a lex operation D such that

1. the function $\eta_{D(A)} : D(A) \to D(D(A))$ is an equivalence, and
2. the function $D(\eta_A) : D(A) \to D(D(A))$ is an equivalence

for each type A. ■

Definition 9. A type A is called *modal* with respect to a descent data operation D if the function $\eta_A : A \to D(A)$ is an equivalence. ■

We write isModal(A) for isEquiv(η_A), which is a mere proposition. In particular, by the definition of a descent data operation, we have isModal($D(A)$) for any type A.

Lemma 20. *Let D be a descent data operation and $f, g : D(A) \to B$ parallel functions into a modal type B, then $f \circ \eta_A \sim g \circ \eta_A$ implies $f \sim g$.*

Proof. Since, by assumption, η_B is an equivalence, it suffices to show that $\eta_B \circ f \sim \eta_B \circ g$, which is the same as showing that $D(f) \circ \eta_{D(A)} \sim D(g) \circ \eta_{D(A)}$. But, since $\eta_{D(A)}$ is an equivalence, it actually suffices to show that $D(f) \sim D(g)$. Now, the assumption $f \circ \eta_A \sim g \circ \eta_A$ implies $D(f) \circ D(\eta_A) \sim D(g) \circ D(\eta_A)$ and, hence, $D(f) \sim D(g)$ because $D(\eta_A)$ is an equivalence. □

Lemma 21. *Let D be a descent data operation, then there is a homotopy*

$$w_A : \eta_{D(A)} \sim D(\eta_A)$$

for each type A.

Proof. Note that $D(D(A))$ is modal and that $\eta_{D(A)} \circ \eta_A = D(\eta_A) \circ \eta_A$. Therefore, the conclusion $\eta_{D(A)} \sim D(\eta_A)$ follows by Lemma 20. □

Definition 10. A *patch algebra* for a descent data operation D is given by a type A together with a *patching structure*:

1. a function $patch : D(A) \to A$, and
2. a homotopy $linv : patch \circ \eta_A \sim \mathrm{id}_A$.

■

Note that the type isPA(A) of patching structures on a type A is the same as the type leftInv(η_A) of homotopy left inverses of the function $\eta_A : A \to D(A)$.

Lemma 22. *Let D be a descent data operation, then* isPA(A) *is logically equivalent to* isModal(A) *for each type A.*

Proof. isModal(A) is logically equivalent to having an (left and right) inverse. It is therefore immediate that isModal(A) implies isPA(A). In the other direction, assume a function $patch : D(A) \to A$ such that $patch \circ \eta_A \sim \mathrm{id}_A$. It suffices to show that also $\eta_A \circ patch \sim \mathrm{id}_{D(A)}$. This is the same as showing that $D(patch) \circ \eta_{D(A)} \sim D(\mathrm{id}_A)$. By Lemma 21 we have $D(patch) \circ \eta_{D(A)} \sim D(patch) \circ D(\eta_A)$ and, by assumption, we have $D(patch) \circ D(\eta_A) \sim D(\mathrm{id}_A)$. In conclusion, we have $D(patch) \circ \eta_{D(A)} \sim D(\mathrm{id}_A)$ and hence, using the assumption that $patch$ is a left inverse of η_A, we have that A is modal. □

Corollary 23. *Let D be a descent data operation, then* isPA(A) *is a mere proposition for each type A.*

Proof. For showing that isPA(A) is a mere proposition, it suffices to show that isPA(A) implies isContr(isPA(A)). [69, Exercise 3.5] Now, assume isPA(A). Then we have isModal(A) by Lemma 22, which is the same as isEquiv(η_A). In turn, this implies that leftInv(η_A) is contractible [69, Lemma 4.2.9], which is the same as isContr(isPA(A)). □

Corollary 24. *Let D be a descent data operation, then the two types* isPA(A) *and* isModal(A) *are equivalent for each type A.*

Proof. The types isPA(A) and isModal(A) = isEquiv(η_A) are logically equivalent mere propositions by the previous two lemmas. □

Proposition 25. *Let D be a descent data operation, A a type and B a modal type, then precomposition with η_A is an equivalence between the types $D(A) \to B$ and $A \to B$.*

Proof. Let $patch : D(B) \to B$ be a patch function for the modal type B. We show that $(A \to B) \to (D(A) \to B), f \mapsto patch \circ D(f)$ is an inverse of $g \mapsto g \circ \eta_A$ by using the homotopy $\eta_{D(A)} \sim D(\eta_A)$ from Lemma 21: For a function $g : D(A) \to B$ we have $patch \circ D(g \circ \eta_A) \sim patch \circ D(g) \circ \eta_{D(A)} \sim patch \circ \eta_B \circ g \sim g$, and for a function $f : A \to B$ we have $patch \circ D(f) \circ \eta_A \sim patch \circ \eta_B \circ f \sim f$. □

Definition 11. A *dependent patch algebra* over a patch algebra $(A, patch, linv)$ for a descent data operation D is given by a dependent type P over A together with a dependent patching structure:

1. $\widetilde{patch} : \prod_{x:D(A)} (\widetilde{D}(P)(x) \to P(patch(x)))$, and
2. $\widetilde{linv} : \prod_{x:A, y:P(x)} \widetilde{patch}(\eta(x), \tilde{\eta}(y)) \equiv_{linv(x)} y$

where $\tilde{\eta}$ is the map $P(x) \to \widetilde{D}(P)(\eta(x))$ from Equation 2.2. ■

Lemma 26. *Let $(P, \widetilde{patch}, \widetilde{linv})$ be a dependent patch algebra over a patch algebra $(A, patch, linv)$ for a descent data operation D, then the total type $\sum_{x:A} P(x)$ has a patching structure.*

Proof. The type

$$\sum_{\widetilde{patch}:\prod_{x:D(A)}(\widetilde{D}(P)(x)\to P(patch_A(x)))} \prod_{x:A,y:P(x)} \widetilde{patch}(\eta(x),\tilde{\eta}(y)) \equiv_{linv_A(x)} y$$

of dependent patching structures on P is isomorphic to the type

$$\sum_{\widetilde{patch}:\prod_{z:D(\sum_{x:A}P(x))}\to P(patch_A(\pi_1 z))} \prod_{z:\sum_{x:A}P(x)} \widetilde{patch}(\eta(z)) \equiv_{linv_A(\mathrm{fst}\, z)} \mathrm{snd}\, z$$

which implies the type

$$\sum_{patch:\prod_{z:D(\sum_{x:A}P(x))}\to\sum_{x:A}P(x)} \prod_{z:\sum_{x:A}P(x)} patch(\eta(z)) \equiv z$$

of patching structures on $\sum_{x:A} P(x)$. [69, Theorem 2.7.2] □

Lemma 27. *Let D be a descent data operation, $(A, patch_A, linv_A)$ a patch algebra, and $(P(a), patch_{P(a)}, linv_{P(a)})$ a family of patch algebras indexed by $a : A$, then the dependent type P has a dependent patching structure over $(A, patch_A, linv_A)$.*

Proof. We need to construct an element of the type

$$\sum_{\widetilde{patch}:\prod_{x:D(A)}(\widetilde{D}(P)(x)\to P(patch_A(x)))} \prod_{a:A,y:P(a)} \widetilde{patch}(\eta_A(a),\tilde{\eta}(y)) \equiv_{linv_A(a)} y$$
$$= \sum_{\widetilde{patch}:\prod_{x:D(A)}(\widetilde{D}(P)(x)\to P(patch_A(x)))} \varphi(\widetilde{patch}\circ\eta_A)$$

where $\varphi(f) = \prod_{a:A,y:P(a)} f(a,\tilde{\eta}(y)) \equiv_{linv_A(a)} y$.

By Lemma 22, since A is a patch algebra, η_A and hence also precomposition

$$\prod_{x:D(A)} (\widetilde{D}(P)(x) \to P(patch_A(x))) \to \prod_{a:A} (\widetilde{D}(P)(\eta(a)) \to P(patch_A(\eta(a))))$$

with η_A is an equivalence. [69, Lemma 4.2.8] This means that the above type is equivalent to the type

$$\sum_{f:\prod_{a:A}(\widetilde{D}(P)(\eta(a))\to P(patch_A(\eta(a))))} \prod_{a:A,y:P(a)} f(a,\tilde{\eta}(y)) \equiv_{linv_A(a)} y$$

which by type-theoretic choice is isomorphic to the type

$$\begin{aligned}
&\prod_{a:A} \sum_{g:\widetilde{D}(P)(\eta(a))\to P(patch_A(\eta(a))))} \prod_{y:P(a)} g(\tilde{\eta}(y)) \equiv_{linv_A(a)} y \\
=&\prod_{a:A} \sum_{g:\widetilde{D}(P)(\eta(a))\to P(patch_A(\eta(a))))} \prod_{y:P(a)} \mathrm{tp}_P(linv_A(a), g(\tilde{\eta}(y))) \equiv y \\
=&\prod_{a:A} \sum_{g:\widetilde{D}(P)(\eta(a))\to P(patch_A(\eta(a))))} \psi(\mathrm{tp}_P(linv_A(a)) \circ g)
\end{aligned}$$

where $\psi(h) = \prod_{y:P(a)} h(\tilde{\eta}(y)) \equiv y$.

Since transport is an equivalence [69, Example 2.4.9], again, also

$$\mathrm{tp}_P(linv_A(a)) \circ - : [\widetilde{D}(P)(\eta(a)) \to P(patch_A(\eta(a)))] \to [\widetilde{D}(P)(\eta(a)) \to P(a)]$$

is an equivalence and the above type is equivalent to the type

$$\prod_{a:A} \sum_{h:\widetilde{D}(P)(\eta(a))\to P(a)} h(\tilde{\eta}(y)) \equiv y$$

By Lemma 6, there is an isomorphism $i : D(P(a)) \cong \widetilde{D}(P)(\eta(a))$ such that $i \circ \eta_{P(a)} = \tilde{\eta}_P(a)$ for each $a : A$. Thus, the last type is isomorphic to the type

$$\prod_{a:A} \sum_{patch:D(P(a))\to P(a)} \prod_{y:P(a)} patch(\eta(y)) \equiv y$$

of families of patching structures. In conclusion, there is an equivalence between the type of dependent patching structures on the dependent type P and the type of families of patching structures on the family of types $P(a)$. □

Corollary 28. *Let D be a descent data operation and P a dependent type over some type A, then $\sum_{x:A} P(x)$ has a patching structure if the types A and $P(a)$ for $a : A$ have.*

Proof. Immediate from the previous two lemmas. □

In summary, we have that a lex operation is a descent data operation if and only if it is a modality:

Definition 12. A *modality M* is given by

1. a type isModal(A) for each type A

2. a type $M(A)$ for each type A
3. a map $\eta_A : A \to M(A)$ for each type A

such that

1. isProp(isModal(A))
2. isModal(A) if Equiv(A, B) and isModal(B)
3. isModal($M(A)$) for each type A
4. isEquiv($M(A) \to B, A \to B, - \circ \eta_A$) if isModal($B$)
5. isModal($\sum_{x:A} P(x)$) if isModal(A) and isModal($P(a)$) for all $a : A$

■

A modality as in Definition 12 does not act on a universe type but on types and is called a *judgemental* modality in Rijke, Shulman and Spitters [74].

Proposition 29. *Let D be a descent data operation, then the family of mere propositions* isPA *together with the modal operator D and the modal unit η is a lex modality.*

Proof. The triple (isPA, D, η) is a Σ-closed reflective subuniverse because we have isProp(isPA(A)) by Corollary 23, isPA($D(A)$) by definition of a descent data operation, isEquiv($- \circ \eta_A$) by Proposition 25, and isPA($\sum_{x:A} P(x)$) whenever isPA(A) and isPA($P(a)$) for each $a : A$ by Corollary 28. The triple is a lex modality because it is a Σ-closed reflective subuniverse and pullback-preserving by Proposition 5. □

Proposition 30. *Let F be a lex operation. If* (isPA, F, η) *is a modality, then F is a descent data operation.*

Proof. We need to show that $\eta_{F(A)}$ and $F(\eta_A)$ are equivalences between $F(A)$ and $F(F(A))$, which are both modal by assumption. Actually, the two maps are path equal because their precompositions with η_A are (even definitionally) equal by naturality and precomposition with η_A is an equivalence by assumption, so that it suffices to show that say $\eta_{F(A)}$ is an equivalence. Indeed, the unique map $\mathrm{rec}(\mathrm{id}_{F(A)}) : F(F(A)) \to F(A)$ such that $\mathrm{rec}(\mathrm{id}_{F(A)}) \circ \eta_{F(A)}$ is path equal to $\mathrm{id}_{F(A)}$ is also a right inverse of $\eta_{F(A)}$ because the precompositions of $\eta_{F(A)} \circ \mathrm{rec}(\mathrm{id}_{F(A)})$ and $\mathrm{id}_{F(F(A))}$ with $\eta_{F(A)}$ are path equal. □

3.2 Descent data operation on a universe

From now on we assume a univalent universe U in our type theory. The goal of this section is to show that if a descent data operation D preserves small types (see Definition 3) then the subuniverse

$$\mathrm{U}_D = \sum_{x:\mathrm{U}} \mathrm{isModal}(x)$$

of modal types (see Definition 9) has a patching structure.

As a result, the internal model of modal types with respect to a descent data operation D will support a univalent universe whenever D preserves small types.

The subuniverse U_D is indeed univalent when U is:

Lemma 31. *Let D be a descent data operation, then the type U_D classifies families of D-modal* U*-small types and is univalent.*

Proof. This holds for any predicate P on a univalent universe U, in particular for $P = \mathrm{isModal}$, and is immediate by the assumption that any two elements of $P(A)$ for $A : \mathrm{U}$ are propositionally equal. □

Proposition 32. *Let D be a descent data operation that preserves types in a univalent universe* U*, then the subuniverse U_D has a patching structure.*

Proof. We have a function $L : D(\mathrm{U}_D) \to \mathrm{U}_D$ such that $\mathrm{fib}_{D(\mathrm{fst})}(x) \simeq \mathrm{El}_D\, L(x)$ for all $x : D(\mathrm{U}_D)$ because $D(\mathrm{fst}) : D(\tilde{\mathrm{U}}_D) \to D(\mathrm{U}_D)$ has (modal) U-small fibres by the assumption that D preserves small types. Moreover, we have $\mathrm{fib}_{D(\mathrm{fst})}(\eta(a)) \simeq D(\mathrm{fib}_{\mathrm{fst}}(a)) \simeq D(\mathrm{El}_D\, a) \simeq \mathrm{El}_D\, a$ for all $a : \mathrm{U}_D$. Thus, we have $\mathrm{El}_D\, L(\eta(a)) \simeq \mathrm{El}_D\, a$ and, by univalence, $L(\eta(a)) \equiv a$ for all $a : \mathrm{U}_D$. In conclusion, L is a patching function for U_D. □

Corollary 33. *Let D be a descent data operation that preserves types in a univalent universe* U*, then* $\mathrm{isModal}(\mathrm{U}_D)$.

Proof. Immediate by Lemma 22 and the proposition. □

3.3 On modal inductive types

We have seen that the modal types with respect to a descent data operation D are closed under the basic type formers (closure under unit, dependent product and identity types follows [70, 74] from D being a lex modality) and support univalent universes (whenever D preserves small types).

The goal of this section is to show that certain higher inductive types satisfy the rules for (higher) inductive types with respect to modal types.

3.3.1 Natural numbers patch algebra

Let D be a descent data operation.

While it is possible to show that, for instance, the (modal) type $D(\mathrm{Nat})$ is a *homotopy* initial natural numbers algebra with respect to modal types, we cannot show that it satisfies the computation rules strictly.

In the case of non-recursive inductive types like, for instance, binary sums, the types $D(A + B)$ can be shown to be a sum type of $D(A)$ and $D(B)$ (satisfying the computation rules strictly) *if* the modality that D gives rise to (see Proposition 29) is *strict* in the sense that each $- \circ \eta_A$ is a *strict* retraction. [70]

In this subsection we show that a *higher* inductive type Nat_D with three point constructors and one path constructor:

$$\begin{aligned} \mathrm{zero} &: \mathrm{Nat}_D \\ \mathrm{succ} &: \mathrm{Nat}_D \to \mathrm{Nat}_D \\ \mathrm{patch} &: D(\mathrm{Nat}_D) \to \mathrm{Nat}_D \\ \mathrm{linv} &: \prod_{n:\mathrm{Nat}_D} \mathrm{patch}(\eta(n)) \equiv_{\mathrm{Nat}_D} n \end{aligned}$$

is a natural numbers type with respect to modal types (if it exists).

The assumption that Nat_D is a higher inductive type generated by the given constructor implies that we also assume eliminators $\mathrm{elim}_{P,z,s,\widetilde{patch},\widetilde{linv}} : \prod_{n:\mathrm{Nat}_D} P(n)$ satisfying the strict computation rules

$$\begin{aligned} \mathrm{elim}_{P,z,s,\widetilde{patch},\widetilde{linv}}(\mathrm{zero}) &= z \\ \mathrm{elim}_{P,z,s,\widetilde{patch},\widetilde{linv}}(\mathrm{succ}(n)) &= s(n, \mathrm{elim}_{P,z,s,\widetilde{patch},\widetilde{linv}}(n)) \end{aligned}$$

for any dependent type P over Nat_D together with

$$\begin{aligned} z &: P(\mathrm{zero}) \\ s &: \prod_{n:\mathrm{Nat}_D} (P(n) \to P(\mathrm{succ}(n))) \\ \widetilde{patch} &: \prod_{x:\widetilde{D}(\mathrm{Nat}_D)} (\widetilde{D}(P)(x) \to P(\mathrm{patch}(x))) \\ \widetilde{linv} &: \prod_{n:\mathrm{Nat}_D, y:P(n)} \widetilde{patch}(\eta(n), \tilde{\eta}(y)) \equiv_{\mathrm{linv}(n)} y \end{aligned}$$

Note that we do *not* require Nat_D to satisfy any particular equations or come with any particular paths for the cases $\mathrm{elim}(\mathrm{patch}(x)) : P(\mathrm{patch}(x))$ and $\mathrm{elim}(\mathrm{linv}(n)) : \mathrm{elim}(\mathrm{patch}(\eta(n))) \equiv_{\mathrm{linv}(n)} \mathrm{elim}(n)$. However, we can prove them path equal to $\widetilde{patch}(x, \widetilde{D}(\mathrm{elim})(x))$ and $\widetilde{linv}(n, \mathrm{elim}(n))$, respectively.

Lemma 34. Nat_D *has a patching structure.*

Proof. As witnessed by the constructors patch and linv. □

Observe that, by definition, Nat_D can be eliminated into dependent types that have a dependent natural numbers algebra structure as well as a dependent patch algebra structure over $(\mathrm{Nat}_D, \mathrm{patch}, \mathrm{linv})$. We use Lemma 27 to eliminate Nat_D into dependent types with a dependent natural numbers algebra structure and whose fibres have a patching structure:

Proposition 35. *Let* $(P(n), patch_{P(n)}, linv_{P(n)})$ *be a family of patch algebras indexed by* $n : \mathrm{Nat}_D$ *together with*

$$\begin{aligned} z &: P(\mathrm{zero}) \\ s &: \prod_{n:\mathrm{Nat}_D} (P(n) \to P(\mathrm{succ}(n))) \end{aligned}$$

then there exists a section $\mathrm{elim} : \prod_{n:\mathrm{Nat}_D} P(n)$ *such that the equations*

1. $\mathrm{elim}(\mathrm{zero}) = z$, *and*
2. $\mathrm{elim}(\mathrm{succ}(n)) = s(n, \mathrm{elim}(n))$ *for all* $n : \mathrm{Nat}_D$

hold strictly.

Proof. Using Lemma 27 we construct a dependent patching structure on P over the patch algebra $(\mathrm{Nat}_D, \mathrm{patch}, \mathrm{linv})$. Then, by definition of the higher inductive type Nat_D there is a section $\prod_{n:\mathrm{Nat}_D} P(n)$ as required. □

3.3.2 Propositional truncation patch algebra

As an example of a higher inductive type, we state a proposition for propositional truncation analogous to Proposition 35 for natural numbers.

Let A be a type and $\|A\|_D$ be a higher inductive type with two point constructors and two path constructors:

$$\begin{aligned} \text{inc} &: A \to \|A\|_D \\ \text{squash} &: \prod_{x,y:\|A\|_D} x \equiv_{\|A\|_D} y \\ \text{patch} &: D(\|A\|_D) \to \|A\|_D \\ \text{linv} &: \prod_{x:\|A\|_D} \text{patch}(\eta(x)) \equiv_{\|A\|_D} x \end{aligned}$$

Note that, by virtue of being a mere proposition, the constructor patch would yield a patching function for $\|A\|_D$ even without requiring the path constructor linv. Indeed, $\text{squash}(\text{patch}(\eta(x)), x)$ is a path of the same type as $\text{linv}(x)$. However, we assume the path constructor linv to emphasize the general pattern of modal (higher) inductive types. In either case, note that $\|A\|_D$ has a patching structure irrespective of any patching structure on A.

Also note that, as for Nat_D, we do not require the eliminator for $\|A\|_D$, whose rule we have omitted, to satisfy any computation rules for the constructors patch and linv.

Proposition 36. *Let* $(P(x), \mathit{patch}_{P(x)}, \mathit{linv}_{P(x)})$ *be a family of patch algebras indexed by* $x : \|A\|_D$ *together with*

$$\begin{aligned} i &: \prod_{a:A} P(\text{inc}(a)) \\ s &: \prod_{x,y:\|A\|_D} \prod_{p:P(x),q:P(y)} p \equiv_{\text{squash}(x,y)} q \end{aligned}$$

then there exists a section $\text{elim} : \prod_{x:\|A\|_D} P(x)$ *such that the equations*

1. $\text{elim}(\text{inc}(a)) = i(a)$ *for all* $a : A$, *and*
2. $\text{elim}(\text{squash}(x, y)) = s(x, y, \text{elim}(x), \text{elim}(y))$ *for all* $x, y : \|A\|_D$

hold strictly.

Proof. We construct a dependent patching structure $(\widetilde{\mathit{patch}}, \widetilde{\mathit{linv}})$ on P over the patch algebra $(\|A\|_D, \text{patch}, \text{linv})$ using Lemma 27, and then use the eliminator for $\|A\|_D$ to construct a section $\text{elim} : \prod_{x:\|A\|_D} P(x)$ as required. □

This concludes the first part of the thesis. In the second part we present a particular model of type theory that supports a univalent universe and comes equipped with a descent data operation. We then use the result of this part to obtain a new model of univalent type theory with a single univalent universe.

Part II

Groupoid-valued presheaf models of type theory

Chapter 4

Categories with families

Type theory has a convenient syntax for reasoning and programming informally. Importantly, type theory can also be presented as a (generalized) algebraic theory by moving from metasubstitution and named variables to explicit substitutions and de Bruijn indices. By virtue of the algebraic presentation using only sorts, (first-order) operations and equations, type theory has an elementary model theory and universal results like the existence of initial models apply.

We begin this chapter by recalling the notion of generalized algebraic theory [15] (without sort equations). Examples of generalized algebraic theories include the theory of small categories and even the theory of small categories with finite limits. Then we recall the presentation of type theory as a generalized algebraic theory [25, 39] and extensions by operations and equations corresponding to extra type structure. In particular, a universe type structure can be axiomatized in this way, but we will also give an equivalent higher-order definition that will be convenient to employ in model constructions.

The notion of homomorphism between models of type theory presented as a generalized algebraic theory was generalized by Kaposi, Huber and Sattler [48] to preserve the basic substitution structure strictly but the structure defined by universal properties (like context extension and extra type structure) only up to canonical isomorphism. The resulting notion of *pseudo*morphism was used to generalize the gluing construction from topos theory to type theory [48].

We conclude this chapter by presenting the notions of lex operation and descent data operation from Chapters 2 and 3 as extensions of the generalized algebraic theory of type theory. We can then show that pointed pseudoen-

domorphisms induce lex operations and that they give rise to submodels of modal types when they are idempotent.

4.1 Generalized algebraic theories

The metatheory of this work is a *constructive* set theory with a countable hierarchy of inaccessible sets $(\mathcal{U}_i)_{i\in\mathbb{N}}$ [71] as axiomatized in the system CZF by Aczel [2]. In particular, we work *without* the law of excluded middle and *without* the axiom of choice. The sets $\mathcal{U}_i$ play the role of Grothendieck universes [78, 58] in classical set theory and will be called set-theoretic universes in order to distinguish them from the universes of the object theory.

The notion of model of type theory we consider is that of the generalized algebraic theory (GAT, Cartmell [15]) of categories with families (cwf, Dybjer [25]). The reason why these two notions of model should coincide is that the GAT of cwfs can be seen as a name-free version of type theory with explicit substitutions [65].

The notion of GAT is a generalization of the notion of many-sorted algebraic theory [35] where the sorts may vary over or depend on a context of variables each with a specified sort; it can also be seen as a notion of type theory without binders and without operations on types but with arbitrary equational axioms for types and terms.

The algebraic presentation of type theory is of a more restricted form of GAT than originally introduced by Cartmell [15] where the equational axioms are only between terms of the GAT.

A GAT (of the restricted form without axioms for types) is hence given by

1. a set of *sort symbols* each with the context it varies over or depends on,

2. a set of *operator symbols* each with the context of its arguments and the sort of its result, and

3. a set of *term equations* each with the context of the two terms it equates and their type.

The significance of an algebraic presentation of type theory is that such a presentation is itself a mathematical object and comes with a precise notion of model, which, moreover, is amenable to general algebraic techniques.

As in the case of many-sorted algebraic theories, the specific syntactic form of a GAT, namely *first-order* expressions and *equational* axioms, determine

1. what a *set-theoretic* model of that GAT is, and 2. what a homomorphism between two models is.

When defining algebraic structures, instead of giving their axiomatization as a GAT and defining a structure of their kind to be a model of that GAT, we will usually leave the GAT to be inferred from the definition of its models.

The typical and first example of a mathematical structure that is axiomatized by a GAT which falls outside the fragment of many-sorted algebraic theories is that of a (small) category with its partial composition operation:

Definition 13. A *small category* $\mathscr{C}$ is given by

1. a set Obj of *objects*, and
2. a family assigning sets $\mathrm{Hom}(X, Y)$ of *morphisms* to $X, Y \in \mathrm{Obj}$, which we also denote by $X \to Y$

together with operations assigning

1. *identity* morphisms $\mathrm{id}_X \in X \to X$ to $X \in \mathrm{Obj}$, and
2. *composite* morphisms $g \circ f \in X \to Z$ to $g \in Y \to Z$ and $f \in X \to Y$

such that the equations

1. $f \circ \mathrm{id}_X = f$ and $\mathrm{id}_Y \circ f = f$ (*left and right unity*), and
2. $(h \circ g) \circ f = h \circ (g \circ f)$ (*associativity*)

are satisfied. ■

The definition of the structure of a small category can be seen as an unfolding of the definition of a model of the GAT of small categories:

Definition 14. The *GAT of small categories* is given by two sort symbols:

1. O
2. $\mathrm{H}(X, Y)$ for $X, Y : \mathrm{O}$,

two operator symbols:

1. $\mathrm{i}(X) : \mathrm{H}(X, X)$ for $X : \mathrm{O}$
2. $\mathrm{c}(X, Y, Z, g, f) : \mathrm{H}(X, Z)$ for $X, Y, Z : \mathrm{O}$, $g : \mathrm{H}(Y, Z)$, $f : \mathrm{H}(X, Y)$

and three term equations:

1. $\mathrm{c}(X, X, Y, f, \mathrm{i}(X)) = f$ for $X, Y : \mathrm{O}$, $f : \mathrm{H}(X, Y)$
2. $\mathrm{c}(X, Y, Y, \mathrm{i}(Y), f) = f$ for $X, Y : \mathrm{O}$, $f : \mathrm{H}(X, Y)$
3. $\mathrm{c}(X, Y, Z', \mathrm{c}(Y, Z, Z', h, g), f) = \mathrm{c}(X, Z, Z', h, \mathrm{c}(X, Y, Z, g, f))$ for $X, Y, Z : \mathrm{O}$, $h : \mathrm{H}(Z, Z')$, $g : \mathrm{H}(Y, Z)$, $f : \mathrm{H}(X, Y)$

■

Definition 14 spells out the whole context of each symbol and each equation, although, for instance, the first three arguments of the quinary operator symbol c can be inferred from the types of its other two arguments. For the sake of readability or conciseness, we often leave inferrable arguments implicit and do not mention them in our notation.

Typical examples of mathematical structures that are axiomatized in a way that does *not* fit into the framework of GATs are those of topological spaces and locales because they involve higher-order sorts (the sort of opens) or infinitary operations (the join operation).

Extensions of the GAT of small categories that are relevant for the algebraic presentation of type theory are that of presheaves:

Definition 15. A *presheaf* P over a small category $\mathscr{C}$ is given by an additional family assigning

$$\text{sets } P(X) \text{ of } \textit{elements over } X \text{ to } X \in \mathrm{Obj}$$

and a third operation assigning

$$\textit{restrictions } x|f \in P(Y) \text{ to } f \in Y \to X \text{ and } x \in P(X)$$

such that the two additional equations

1. $x|\mathrm{id}_X = x$, and
2. $x|(f \circ g) = (x|f)|g$

satisfied. ■

and that of terminal objects:

Definition 16. A *terminal object structure* 1 on a small category $\mathscr{C}$ is given by two more operations

1. picking the *terminal object* $1 \in \mathrm{Obj}$, and
2. assigning *terminal morphisms* $!_X \in X \to 1$ to $X \in \mathrm{Obj}$

such that the two additional equations

1. $!_1 = \mathrm{id}_1$, and
2. $!_Y \circ f = !_X$

are satisfied. ■

Note that the usual category-theoretic definition of a terminal object by its universal property is not purely equational and that a morphism between small categories with a terminal object in the category-theoretic sense is not required to map one choice of a terminal object to the other but instead it is only required to map any object satisfying the universal property to another. Small categories with Cartesian products and equalizers can also be axiomatized purely equationally and similar remarks about (homo)morphisms between categories with that extra property (structure) apply, in particular a homomorphism between small categories with finite limits as models of the corresponding GAT is required to preserve the choice of limit for each (finite) diagram *strictly*.

4.2 Type theory as an algebraic theory

We now recall the algebraic presentation of type theory by the GAT of cwfs as originally introduced by Dybjer [25]:

Definition 17. A *category with families* is given by

1. a set Ctx of *contexts*,
2. a family of sets $\mathrm{Sub}(\Delta, \Gamma)$ of *substitutions* from context Δ to context Γ,
3. a family of sets $\mathrm{Ty}(\Gamma)$ of *types* in context Γ, and
4. a family of sets $\mathrm{Tm}(\Gamma, A)$ of *terms* of type A in context Γ

together with ten operations assigning

1. *identity* substitutions $\mathrm{id}_\Gamma \in \mathrm{Sub}(\Gamma, \Gamma)$ to $\Gamma \in \mathrm{Ctx}$,
2. *composite* substitutions $\sigma \circ \tau \in \mathrm{Sub}(\mathrm{E}, \Gamma)$ to $\sigma \in \mathrm{Sub}(\Delta, \Gamma)$ and $\tau \in \mathrm{Sub}(\mathrm{E}, \Delta)$,
3. *terminal* context $[] \in \mathrm{Ctx}$,
4. *terminal* substitutions $\langle\rangle_\Gamma \in \mathrm{Sub}(\Gamma, [])$ to $\Gamma \in \mathrm{Ctx}$,
5. *substituted* types $A\sigma \in \mathrm{Ty}(\Delta)$ to $\sigma \in \mathrm{Sub}(\Delta, \Gamma)$ and $A \in \mathrm{Ty}(\Gamma)$,
6. *substituted* terms $t\sigma \in \mathrm{Tm}(\Delta, A\sigma)$ to $\sigma \in \mathrm{Sub}(\Delta, \Gamma)$ and $t \in \mathrm{Tm}(\Gamma, A)$,

7. *extended* contexts $\Gamma.A \in \mathrm{Ctx}$ to $\Gamma \in \mathrm{Ctx}$ and $A \in \mathrm{Ty}(\Gamma)$,
8. *display* substitutions $\mathrm{p}_A \in \mathrm{Sub}(\Gamma.A, \Gamma)$ to $A \in \mathrm{Ty}(\Gamma)$,
9. *variable* terms $\mathrm{q}_A \in \mathrm{Tm}(\Gamma.A, A\mathrm{p}_A)$ to $A \in \mathrm{Ty}(\Gamma)$, and
10. *pairing* substitutions $\langle \sigma, t \rangle \in \mathrm{Sub}(\Delta, \Gamma.A)$ to $\sigma \in \mathrm{Sub}(\Delta, \Gamma)$ and $t \in \mathrm{Tm}(\Delta, A\sigma)$

such that the three *composition structure* equations

1. $\sigma \circ \mathrm{id}_\Delta = \sigma$,
2. $\mathrm{id}_\Gamma \circ \sigma = \sigma$, and
3. $(\sigma \circ \tau) \circ \upsilon = \sigma \circ (\tau \circ \upsilon)$,

the two *terminal context structure* equations

1. $\langle\rangle_{[]} = \mathrm{id}_{[]}$, and
2. $\langle\rangle_\Gamma \circ \sigma = \langle\rangle_\Delta$,

the four *substitution structure* equations

1. $A\mathrm{id}_\Gamma = A$,
2. $A(\sigma \circ \tau) = (A\sigma)\tau$,
3. $t\mathrm{id}_\Gamma = t$, and
4. $t(\sigma \circ \tau) = (t\sigma)\tau$,

and the four *comprehension structure* equations

1. $\mathrm{p}_A \circ \langle \sigma, t \rangle = \sigma$,
2. $\mathrm{q}_A \langle \sigma, t \rangle = t$,
3. $\langle \mathrm{p}_A, \mathrm{q}_A \rangle = \mathrm{id}_{\Gamma.A}$, and
4. $\langle \sigma \circ \tau, t\tau \rangle = \langle \sigma, t \rangle \circ \tau$

are satisfied. ■

Note that there are two *distinct* notions of context involved here: 1. the *contexts* of the types and terms of the GAT of cwfs in contrast to 2. the *terms* of the sort Ctx, as well as two *distinct* notions of type: 1. the *types* of the GAT in contrast to 2. the *terms* of the sort Ty, and, finally, two *distinct* notions of term: 1. the *terms* of the GAT in contrast to 2. the *terms* of the sort Tm.

Note also that the argument A of the operation $\langle \sigma, t \rangle$ is in general not actually inferrable from the types of the arguments σ and t because the operation $A\sigma$ is in general not injective, but leaving it implicit nonetheless in our

notation should not lead to confusion — we will not mention this kind of issues anymore below.

Lastly, note that the cwf axioms are of the restricted form where no equations between the types of the GAT are postulated.

The definition of a cwf can also be organized in the following way: The set Ctx together with the family Sub, the operation [] and the composition structure is the same as a small category $\mathscr{C}$ with a terminal object; the family Ty together with the operation $A\sigma$ is the same as a presheaf U over $\mathscr{C}$; since a presheaf over the category of elements of a presheaf Q is equivalently a presheaf over Q, the family Tm together with the operation $t\sigma$, which is the same as a presheaf over the category $\int U$ of elements of U, corresponds to a presheaf $\widetilde{U}$ over U, that is a natural transformation $p : \widetilde{U} \to U$ of presheaves. As observed by Fiore [29] and Awodey [9], the comprehension structure of a cwf can then be captured by saying that p is *representable*, which means that its pullbacks along generalized elements with representable domain are representable.

The notion of cwf is closely related to the notion of category with attributes [14, 66, 68] where the display substitutions p_A instead of having pullbacks satisfy a comprehension schema [55]. Indeed, what the comprehension structure gives us for $\sigma \in \mathrm{Sub}(\Delta, \Gamma)$ and $A \in \mathrm{Ty}(\Gamma)$ is both a natural bijection between the sets $\mathrm{Tm}(\Delta, A\sigma)$ of terms of type $A\sigma$ and $\mathrm{Sub}_\Gamma(\sigma, \mathrm{p}_A) = \{\tau \in \mathrm{Sub}(\Delta, \Gamma.A) \mid \mathrm{p}_A \circ \tau = \sigma\}$ of morphisms from σ to p_A over Γ, and a Cartesian morphism from $\mathrm{p}_{A\sigma}$ to p_A over σ.

We now introduce notation for two *derived* substitution operations assigning

1. $\langle\sigma\rangle = \langle\sigma \circ \mathrm{p}_{A\sigma}, \mathrm{q}_{A\sigma}\rangle \in \mathrm{Sub}(\Delta.A\sigma, \Gamma.A)$ to $\sigma \in \mathrm{Sub}(\Delta, \Gamma)$, and
2. $[a] = \langle \mathrm{id}_\Gamma, a\rangle \in \mathrm{Sub}(\Gamma, \Gamma.A)$ to $a \in \mathrm{Ty}(\Gamma, A)$.

From the equations for the operation $\langle\sigma, t\rangle$ we can then show that the derived operations $\langle\sigma\rangle$ and $[a]$ satisfy the following equations:

1. $\mathrm{p}_A \circ [a] = \mathrm{id}_\Gamma$
2. $\mathrm{q}_A[a] = a$
3. $[a] \circ \sigma = \langle\sigma\rangle \circ [a\sigma]$
4. $\mathrm{p}_A \circ \langle\sigma\rangle = \sigma \circ \mathrm{p}_{A\sigma}$
5. $\mathrm{q}_A\langle\sigma\rangle = \mathrm{q}_{A\sigma}$
6. $\langle\sigma \circ \tau\rangle = \langle\sigma\rangle \circ \langle\tau\rangle$
7. $\langle \mathrm{p}_A\rangle \circ [\mathrm{q}_A] = \mathrm{id}_{\Gamma.A}$

For instance, we have $[a] \circ \sigma = \langle \mathrm{id}_\Gamma, a\rangle \circ \sigma$ by definition, $\langle \mathrm{id}_\Gamma, a\rangle \circ \sigma = \langle \mathrm{id}_\Gamma \circ \sigma, a\sigma\rangle$ by the equation $\langle \sigma, t\rangle \circ \tau = \langle \sigma \circ \tau, t\tau\rangle$, $\langle \mathrm{id}_\Gamma \circ \sigma, a\sigma\rangle = \langle \sigma \circ \mathrm{id}_\Delta, a\sigma\rangle$ by left and right unity, $\langle \sigma \circ \mathrm{id}_\Delta, a\sigma\rangle = \langle (\sigma \circ \mathrm{p}_{A\sigma}) \circ \langle \mathrm{id}_\Delta, a\sigma\rangle, \mathrm{q}_{A\sigma}\langle \mathrm{id}_\Delta, a\sigma\rangle\rangle$ by associativity and the equations $\mathrm{p}_A \circ \langle \sigma, t\rangle = \sigma$ and $\mathrm{q}_A\langle \sigma, t\rangle = t$, and, hence, in conclusion, $[a] \circ \sigma = \langle \sigma\rangle \circ [a\sigma]$ by another application of the equation $\langle \sigma, t\rangle \circ \tau = \langle \sigma \circ \tau, t\tau\rangle$ and the definitions.

More generally, given $\boldsymbol{B} \in \mathrm{Ty}(\Gamma.\boldsymbol{A})$ and $b \in \mathrm{Tm}(\Gamma, \boldsymbol{B}[a])$, we can then write $[a, b]$ for $\langle [a], b\rangle \in \mathrm{Sub}(\Gamma, \Gamma.\boldsymbol{A}.\boldsymbol{B})$, and given $\boldsymbol{C} \in \mathrm{Ty}(\Gamma.\boldsymbol{A}.\boldsymbol{B})$ and $c \in \mathrm{Tm}(\Gamma, \boldsymbol{C}[a, b])$, we can write $[a, b, c]$ for $\langle [a, b], c\rangle \in \mathrm{Sub}(\Gamma, \Gamma.\boldsymbol{A}.\boldsymbol{B}.\boldsymbol{C})$ etc. so that $\mathrm{p}^i \circ [a_n, \ldots, a_0] = [a_n, \ldots, a_i]$ and $(\mathrm{qp}^i)[a_n, \ldots, a_0] = a_i$.

In fact, we could define another GAT with $\langle \sigma\rangle$ and $[a]$ as primitive operations and their equations as axioms, and instead derive the operation $\langle \sigma, t\rangle$ with its equations by setting $\langle \sigma, t\rangle = \langle \sigma\rangle[t]$. That GAT, which would essentially be the one defined by Ehrhard [26], would be equivalent to the GAT of cwfs as defined in Definition 17 because the operation $\langle \sigma, t\rangle$ (and hence the operations $\langle \sigma\rangle$ and $[a]$) is uniquely determined. This illustrates that the choice of operations and axioms for presenting type theory algebraically is not canonical.

Next, we define extra structure a cwf might carry and which corresponds to the additional type and term formers present in intensional intuitionistic type theory [61, 62] and homotopy type theory [69].

Note that this extra structure can be axiomatized as an extension of the GAT of cwfs without introducing new sorts, that is by only introducing new operations and term equations for those new operations. In fact, the new operations do not introduce new elements of the sort Ctx but act only on the sorts Ty and Tm.

Definition 18. A *unit type structure* on a cwf is given by two operations:

1. $\mathrm{Unit} \in \mathrm{Ty}([])$
2. $\mathrm{tt} \in \mathrm{Tm}([], \mathrm{Unit})$

satisfying the equation

$$t = \mathrm{tt}\langle\rangle_\Gamma$$

for $t \in \mathrm{Tm}(\Gamma, \mathrm{Unit}\langle\rangle_\Gamma)$. ■

Definition 18 corresponds to the "pragmatist", "negative" or "record" type view [6, 33, 88] on the unit type, where an element is uniquely determined by its behaviour under the canonical eliminations (of which there are none here).

Definition 19. A *Σ type structure* on a cwf is given by four operations:

1. $\Sigma_A B \in \mathrm{Ty}(\Gamma)$ for $A \in \mathrm{Ty}(\Gamma)$ and $B \in \mathrm{Ty}(\Gamma.A)$
2. $(a, b) \in \mathrm{Tm}(\Gamma, \Sigma_A B)$ for $a \in \mathrm{Tm}(\Gamma, A)$ and $b \in \mathrm{Tm}(\Gamma, B[a])$
3. $\mathrm{fst} p \in \mathrm{Tm}(\Gamma, A)$ for $p \in \mathrm{Tm}(\Gamma, \Sigma_A B)$
4. $\mathrm{snd} p \in \mathrm{Tm}(\Gamma, B[\mathrm{fst} p])$ for $p \in \mathrm{Tm}(\Gamma, \Sigma_A B)$

satisfying the equations

1. $(\Sigma_A B)\sigma = \Sigma_{A\sigma} B\langle\sigma\rangle$.
2. $(a, b)\sigma = (a\sigma, b\sigma)$,
3. $(\mathrm{fst} p)\sigma = \mathrm{fst}\, p\sigma$,
4. $(\mathrm{snd} p)\sigma = \mathrm{snd}\, p\sigma$,
5. $\mathrm{fst}(a, b) = a$,
6. $\mathrm{snd}(a, b) = b$, and
7. $p = (\mathrm{fst} p, \mathrm{snd} p)$.

■

If a cwf has a unit and Σ type structure, then the categories $\mathrm{Ty}(\Gamma)$ of types and functions in context Γ have a terminal context and comprehension structure which is preserved by the substitution functors $\mathrm{Ty}(\sigma) : \mathrm{Ty}(\Gamma) \to \mathrm{Ty}(\Delta)$ for $\sigma \in \mathrm{Sub}(\Delta, \Gamma)$. The converse that a cwf has a unit and Σ type structure if the categories $\mathrm{Ty}(\Gamma)$ have terminal context and comprehension structure which is preserved by substitution holds as well. The resulting cwf and morphisms of cwfs are denoted by $\mathscr{C}_\Gamma$ and $\mathscr{C}_\sigma$, respectively. They will be used to characterize a lex operation structure on $\mathscr{C}$ as a (family of) pseudomorphisms on the (family of) cwfs $\mathscr{C}_\Gamma$.

Definition 20. A *Π type structure* on a cwf is given by three operations:

1. $\Pi_A B \in \mathrm{Ty}(\Gamma)$ for $A \in \mathrm{Ty}(\Gamma)$ and $B \in \mathrm{Ty}(\Gamma.A)$
2. $\lambda b \in \mathrm{Tm}(\Gamma, \Pi_A B)$ for $b \in \mathrm{Tm}(\Gamma.A, B)$
3. $\mathrm{app}(t, a) \in \mathrm{Tm}(\Gamma.B[a])$ for $t \in \mathrm{Tm}(\Gamma, \Pi_A B)$ and $a \in \mathrm{Tm}(\Gamma, A)$

satisfying the equations

1. $(\Pi_A B)\sigma = \Pi_{A\sigma} B\langle\sigma\rangle$,
2. $(\lambda b)\sigma = \lambda\, b\langle\sigma\rangle$,
3. $\mathrm{app}(t, a)\sigma = \mathrm{app}(t\sigma, a\sigma)$,
4. $\mathrm{app}(\lambda b, a) = b[a]$, and
5. $t = \lambda\, \mathrm{app}(t \mathrm{p}_A\, \mathrm{q}_A)$.

■

Let $\mathscr{C}$ be a cwf with Π type structure. As usual, write $A \to B \in \mathrm{Ty}(\Gamma)$ for $\Pi_A B\mathrm{p}_A$ when $B \in \mathrm{Ty}(\Gamma)$. Note that the sets $\mathrm{Tm}(\Gamma, A' \to A)$ and $\mathrm{Tm}(\Gamma, \Pi_A B)$ are in bijection with the sets $\mathrm{Sub}_{\mathscr{C}_\Gamma}(A', A)$ and $\mathrm{Tm}_{\mathscr{C}_\Gamma}(A, B)$, respectively, so that a Π type structure provides us with (a family of) internal homs on the categories $\mathrm{Ty}(\Gamma)$. A natural family of endofunctors on $\mathrm{Ty}(\Gamma)$ for each $\Gamma \in \mathrm{Ctx}$ will automatically be a family of internal functors in that case. Moreover, the functors will be pointed in the presence of a unit type structure.

Definition 21. An *identity type structure* on a cwf is given by three operations:

1. $\mathrm{Id}_A(a, b) \in \mathrm{Ty}(\Gamma)$ for $A \in \mathrm{Ty}(\Gamma)$, $a \in \mathrm{Tm}(\Gamma, A)$ and $b \in \mathrm{Tm}(\Gamma, A)$
2. $\mathrm{refl}_a \in \mathrm{Tm}(\Gamma, \mathrm{Id}_A(a, a))$ for $a \in \mathrm{Tm}(\Gamma, A)$
3. $\mathrm{J}(a, d, b, p) \in \mathrm{Tm}(\Gamma, C[b, p])$ for $a \in \mathrm{Tm}(\Gamma, A)$,
   $C \in \mathrm{Ty}(\Gamma.A.\mathrm{Id}_{A\mathrm{p}_A}(a\mathrm{p}_A, \mathrm{q}_A))$,
   $d \in \mathrm{Tm}(\Gamma, C[a, \mathrm{refl}_a])$,
   $b \in \mathrm{Tm}(\Gamma, A)$ and
   $p \in \mathrm{Tm}(\Gamma, \mathrm{Id}_A(a, b))$

satisfying the equations

1. $\mathrm{Id}_A(a, b)\sigma = \mathrm{Id}_{A\sigma}(a\sigma, b\sigma)$,
2. $\mathrm{refl}_a\sigma = \mathrm{refl}_{a\sigma}$,
3. $\mathrm{J}(a, d, b, p)\sigma = \mathrm{J}(a\sigma, d\sigma, b\sigma, p\sigma)$, and
4. $\mathrm{J}(a, d, a, \mathrm{refl}_a) = d$.

■

Definition 22. An *empty type structure* on a cwf is given by two operations:

1. $\mathrm{Empty} \in \mathrm{Ty}([])$
2. $\mathrm{elim}(C, e) \in \mathrm{Tm}(\Gamma, C[e])$ for $C \in \mathrm{Ty}(\Gamma.\mathrm{Empty}\langle\rangle_\Gamma)$ and
   $e \in \mathrm{Tm}(\Gamma, \mathrm{Empty}\langle\rangle_\Gamma)$

satisfying the equation

$$\mathrm{elim}(e)\sigma = \mathrm{elim}(e\sigma).$$

■

Note that, unlike for other elimination term operations, we do not drop the type argument C in the term operation $\mathrm{elim}(C, e)$ because there is no argument sort in which it is even mentioned.

This definition corresponds to the "verificationist", "positive" or "data" type view [6, 33, 88] on the empty type, where functions are determined by their behaviour on the canonical introductions (of which there are none here) but not necessarily uniquely so.

Definition 23. A *Booleans type structure* on a cwf is given by four operations:

1. $\mathrm{Bool} \in \mathrm{Ty}([])$
2. $\mathrm{true} \in \mathrm{Tm}([], \mathrm{Bool})$
3. $\mathrm{false} \in \mathrm{Tm}([], \mathrm{Bool})$
4. $\mathrm{elim}(c_0, c_1, b) \in \mathrm{Tm}(\Gamma, C[b])$ for $c_0 \in \mathrm{Tm}(\Gamma, C[\mathrm{true}\langle\rangle_\Gamma])$, $c_1 \in \mathrm{Tm}(\Gamma, C[\mathrm{false}\langle\rangle_\Gamma])$ and $b \in \mathrm{Tm}(\Gamma, \mathrm{Bool}\langle\rangle_\Gamma)$

satisfying the equations

1. $\mathrm{elim}(c_0, c_1, b)\sigma = \mathrm{elim}(c_0\sigma, c_1\sigma, b\sigma)$,
2. $\mathrm{elim}(c_0, c_1, \mathrm{true}\langle\rangle) = c_0$, and
3. $\mathrm{elim}(c_0, c_1, \mathrm{false}\langle\rangle) = c_1$.

■

Definition 24. A *natural numbers type structure* on a cwf is given by four operations:

1. $\mathrm{Nat} \in \mathrm{Ty}([])$
2. $\mathrm{zero} \in \mathrm{Tm}([], \mathrm{Nat})$
3. $\mathrm{succ}(n) \in \mathrm{Tm}(\Gamma, \mathrm{Nat}\langle\rangle_\Gamma)$ for $n \in \mathrm{Tm}(\Gamma, \mathrm{Nat}\langle\rangle_\Gamma)$
4. $\mathrm{elim}(c_0, c_1, n) \in \mathrm{Tm}(\Gamma, C[n])$ for $c_0 \in \mathrm{Tm}(\Gamma, C[\mathrm{zero}\langle\rangle_\Gamma])$, $c_1 \in \mathrm{Tm}(\Gamma.\mathrm{Nat}\langle\rangle_\Gamma.C, C[\mathrm{succ}(\mathrm{q}_{\mathrm{Nat}\langle\rangle_\Gamma})]\mathrm{p}_C)$ and $n \in \mathrm{Tm}(\Gamma, \mathrm{Nat}\langle\rangle_\Gamma)$

satisfying the equations

1. $\mathrm{succ}(n)\sigma = \mathrm{succ}(n\sigma)$,
2. $\mathrm{elim}(c_0, c_1, n)\sigma = \mathrm{elim}(c_0\sigma, c_1\langle\langle\sigma\rangle\rangle, n\sigma)$,
3. $\mathrm{elim}(c_0, c_1, \mathrm{zero}\langle\rangle) = c_0$, and
4. $\mathrm{elim}(c_0, c_1, \mathrm{succ}(n)) = c_1[\mathrm{elim}(c_0, c_1, n)]$.

■

Definition 25. A *T-universe type structure* on a cwf with unit, Π, Σ, identity, empty, Booleans and natural numbers type structures is given by nine operations:

1. $\mathrm{U} \in \mathrm{Ty}([])$
2. $\mathrm{El}\, a \in \mathrm{Ty}(\Gamma)$ for $a \in \mathrm{Tm}(\Gamma, \mathrm{U}\langle\rangle)$
3. $\widehat{\mathrm{Unit}} \in \mathrm{Tm}([], \mathrm{U})$, $\widehat{\pi}(a, b)$, $\widehat{\sigma}(a, b)$, $\widehat{\mathrm{Id}}(a, x, y) \in \mathrm{Tm}(\Gamma, \mathrm{U}\langle\rangle)$, $\widehat{\mathrm{Empty}}$, $\widehat{\mathrm{Bool}}$, $\widehat{\mathrm{Nat}} \in \mathrm{Tm}([], \mathrm{U})$ for $a \in \mathrm{Tm}(\Gamma, \mathrm{U}\langle\rangle)$, x, $y \in \mathrm{Tm}(\Gamma, \mathrm{El}\, a)$ and $b \in \mathrm{Tm}(\Gamma.\mathrm{El}\, a, \mathrm{U}\langle\rangle)$

satisfying the equations

1. $(\mathrm{El}\, a)\sigma = \mathrm{El}\, a\sigma$,
2. $\widehat{\pi}(a, b)\sigma = \widehat{\pi}(a\sigma, b\langle\sigma\rangle)$,
3. $\widehat{\sigma}(a, b)\sigma = \widehat{\sigma}(a\sigma, b\langle\sigma\rangle)$,
4. $\widehat{\mathrm{Id}}(a, x, y)\sigma = \widehat{\mathrm{Id}}(a\sigma, x\sigma, y\sigma)$,
5. $\mathrm{El}\, \widehat{\mathrm{Unit}} = \mathrm{Unit}$,
6. $\mathrm{El}\, \widehat{\pi}(a\, b) = \Pi_{\mathrm{El}\, a} \mathrm{El}\, b$,
7. $\mathrm{El}\, \widehat{\sigma}(a\, b) = \Sigma_{\mathrm{El}\, a} \mathrm{El}\, b$,
8. $\mathrm{El}\, \widehat{\mathrm{Id}}(a\, x\, y) = \mathrm{Id}_{\mathrm{El}\, a}(x, y)$,
9. $\mathrm{El}\, \widehat{\mathrm{Empty}} = \mathrm{Empty}$,
10. $\mathrm{El}\, \widehat{\mathrm{Bool}} = \mathrm{Bool}$, and
11. $\mathrm{El}\, \widehat{\mathrm{Nat}} = \mathrm{Nat}$.

■

This definition of a cwf with a universe type corresponds to type theory with a Tarski-style universe [61, 63, 39, Section 2.1.6].

When constructing a cwf with a T-universe type structure it can be convenient (because of the reduced number of operations) to construct the following structure which defines a universe type to be a universal type with respect to a set of small types instead:

Definition 26. A *V-universe type structure* on a cwf with unit, Π, Σ, identity, empty type, Booleans type and natural numbers type structures is given by

$$\text{a subset } \mathrm{ty}(\Gamma) \subseteq \mathrm{Ty}(\Gamma) \text{ of } \textit{small types} \text{ for } \Gamma \in \mathrm{Ctx}$$

and three operations

1. a type $\mathrm{U} \in \mathrm{Ty}([])$
2. a small type $\mathrm{El}\, a \in \mathrm{ty}(\Gamma)$ for every code $a \in \mathrm{Tm}(\Gamma, \mathrm{U}\langle\rangle_\Gamma)$
3. a code $\mathrm{In}A \in \mathrm{Tm}(\Gamma, \mathrm{U}\langle\rangle_\Gamma)$ for every small type $A \in \mathrm{ty}(\Gamma)$

such that

1. $A\sigma \in \mathrm{ty}(\Delta)$ whenever $A \in \mathrm{ty}(\Gamma)$,
2. $(\mathrm{El}\, a)\sigma = \mathrm{El}\, a\sigma$,
3. $(\mathrm{In} A)\sigma = \mathrm{In} A\sigma$,
4. $\mathrm{El}(\mathrm{In} A) = A$,
5. $\mathrm{In}(\mathrm{El}\, a) = a$,
6. $\mathrm{Unit} \in \mathrm{ty}([])$,
7. $\Pi_A B \in \mathrm{ty}(\Gamma)$ whenever $A \in \mathrm{ty}(\Gamma)$ and $B \in \mathrm{ty}(\Gamma.A)$,
8. $\Sigma_A B \in \mathrm{ty}(\Gamma)$ whenever $A \in \mathrm{ty}(\Gamma)$ and $B \in \mathrm{ty}(\Gamma.A)$,
9. $\mathrm{Id}_A(a, b) \in \mathrm{ty}(\Gamma)$ whenever $A \in \mathrm{ty}(\Gamma)$,
10. $\mathrm{Empty} \in \mathrm{ty}([])$,
11. $\mathrm{Bool} \in \mathrm{ty}([])$, and
12. $\mathrm{Nat} \in \mathrm{ty}([])$.

■

Note that this definition does *not* fit into the framework of GATs because it talks about a "subsort" $\mathrm{ty}(\Gamma) \subseteq \mathrm{Ty}(\Gamma)$ and operations having more than one sort.

This definition of a universe type structure corresponds to a type theory with a universe type like in Voevodsky [83] which differs from the universe type in Martin-Löf [61, 62] in that there is a rule to derive the equality judgement $a = b$ from the premise $\mathrm{El}\, a = \mathrm{El}\, b$.

Proposition 37. *A cwf with a V-universe type structure has a T-universe type structure.*

Proof. Define the coding operations of a T-universe type structure (see Definition 25) by setting $\widehat{\pi}(a, b) = \mathrm{In}(\Pi_{\mathrm{El}\, a} \mathrm{El}\, b)$ etc. □

Proposition 38. *A cwf with a T-universe type structure where the decoding operation is injective has a V-universe type structure.*

Proof. Given a cwf with a universe type structure where $\mathrm{El}\, a$ is injective, that is $a = b$ whenever $\mathrm{El}\, a = \mathrm{El}\, b$, we can define a universe structure (Definition 26) by taking $\mathrm{ty}(\Gamma) = \{\mathrm{El}\, a \in \mathrm{Ty}(\Gamma) \mid a \in \mathrm{Tm}(\Gamma, \mathrm{U}\langle\rangle_\Gamma)\}$ and $\mathrm{In}(\mathrm{El}\, a) = a$, which is well-defined by the injectivity assumption, so that $\Pi_{\mathrm{El}\, a} \mathrm{El}\, b \in \mathrm{ty}(\Gamma)$ because $\Pi_{\mathrm{El}\, a} \mathrm{El}\, b = \mathrm{El}\, \widehat{\pi}(a\, b)$ etc. □

Definition 27. A *propositional truncation type structure* on a cwf with identity type structure is given by four operations:

1. $\|A\| \in \mathrm{Ty}(\Gamma)$ for $A \in \mathrm{Ty}(\Gamma)$

2. $\mathrm{inc}(a) \in \mathrm{Tm}(\Gamma, \|A\|)$ for $a \in \mathrm{Tm}(\Gamma, A)$
3. $\mathrm{squash}(t,u) \in \mathrm{Tm}(\Gamma, \mathrm{Id}_{\|A\|}(t,u))$ for $t, u \in \mathrm{Tm}(\Gamma, \|A\|)$
4. $\mathrm{rec}(c,p,t) \in \mathrm{Tm}(\Gamma, C)$ for $C \in \mathrm{Ty}(\Gamma)$,
   $c \in \mathrm{Tm}(\Gamma.A, C\mathrm{p})$,
   $p \in \mathrm{Tm}(\Gamma.C.C\mathrm{p}, \mathrm{Id}_{C\mathrm{p}^2}(\mathrm{qp}^1, \mathrm{qp}^0))$ and
   $t \in \mathrm{Tm}(\Gamma, \|A\|)$

satisfying the equations

1. $\|A\|\sigma = \|A\sigma\|$,
2. $\mathrm{inc}(a)\sigma = \mathrm{inc}(a\sigma)$,
3. $\mathrm{squash}(t,u)\sigma = \mathrm{squash}(t\sigma, u\sigma)$,
4. $\mathrm{rec}(c,p,t)\sigma = \mathrm{rec}(c\langle\sigma\rangle, p\langle\langle\sigma\rangle\rangle, t\sigma)$, and
5. $\mathrm{rec}(c,p,\mathrm{inc}(a)) = c[a]$.

■

Usually, a universe type reflects all the basic types and is closed under all type formers, but this is not always the case in all models as we will see with propositional truncation and a univalent universe in the groupoid model. Note that, in order to avoid a paradox [32, 62], for a universe type to reflect or be closed under a universe type means for it to contain a "smaller" universe rather than itself.

4.3 Lex and descent data operation structure

We end this chapter by extending the GAT of cwfs with operations corresponding to a lex operation (Definition 5) and a descent data operation (Definition 8), and building a new cwf from a cwf which supports them.

Recall that a descent data operation is a lex operation that is a modality and that a lex operation is an operation on types, type families and functions that preserves both unit and dependent sum types, commutes with reindexing and is functorial. Thus, a lex operation structure is in particular given by an underlying functor structure:

Definition 28. An *internal functor structure* on a cwf with Π type structure is given by:

1. $F_0(A) \in \mathrm{Ty}(\Gamma)$ for $A \in \mathrm{Ty}(\Gamma)$
2. $F_1(t) \in \mathrm{Tm}(\Gamma, F_0(A) \to F_0(B))$ for $t \in \mathrm{Tm}(\Gamma, A \to B)$

satisfying the stability equations

1. $F_0(A)\sigma = F_0(A\sigma)$ and
2. $F_1(t)\sigma = F_1(t\sigma)$,

as well as the functoriality equations

1. $F_1(\mathrm{id}_A) = \mathrm{id}_{F_0(A)}$ and
2. $F_1(u \circ t) = F_1(u) \circ F_1(t)$.

■

This is essentially Definition 1 with explicit contexts and (stability) axioms expressing that constructions in type theory are invariant under substitution.

We can then show that an internal functor structure on a cwf is the same as a family of endofunctors on the categories of types and functions so that, as noted after Definition 1, the use of the Π type structure in this definition is inessential:

Proposition 39. *Let $\mathscr{C}$ be a cwf with Π type structure, then a family of operations*

1. $F_0(\Gamma) : \mathrm{Ty}(\Gamma) \to \mathrm{Ty}(\Gamma)$ *and*
2. $F_1(A, B) : \mathrm{Tm}(\Gamma, A \to B) \to \mathrm{Tm}(\Gamma, F_0(\Gamma, A) \to F_0(\Gamma, B))$

for $\Gamma \in \mathrm{Ctx}$, $A, B \in \mathrm{Ty}(\Gamma)$ is an internal functor structure on $\mathscr{C}$ if and only if it is a natural family of endofunctors $F_\Gamma : \mathrm{Ty}(\Gamma) \to \mathrm{Ty}(\Gamma)$, that is $\mathrm{Ty}(\sigma) \circ F_\Gamma = F_\Delta \circ \mathrm{Ty}(\sigma)$ for $\sigma \in \mathrm{Sub}(\Delta, \Gamma)$.

Proof. Unfolding definitions. □

Recall that when a cwf $\mathscr{C}$ has a unit and Σ type structure, the categories $\mathrm{Ty}(\Gamma)$ of types and functions canonically extend to the cwfs $\mathscr{C}_\Gamma$. We can then observe that to extend an internal functor structure on $\mathscr{C}$ to a lex operation structure is the same as giving a *pseudo*morphism of cwfs structure on the corresponding family of functors $F_\Gamma : \mathscr{C}_\Gamma \to \mathscr{C}_\Gamma$.

Definition 29. A *lex operation structure* on an internal functor structure F on a cwf with unit, Σ and Π type structure is given by:

1. $\widetilde{F}_0(B) \in \mathrm{Ty}(\Gamma.F_0(A))$ for $B \in \mathrm{Ty}(\Gamma.A)$
2. $\widetilde{F}_1(b) \in \mathrm{Tm}(\Gamma.F_0(A), \widetilde{F}_0(B))$ for $b \in \mathrm{Tm}(\Gamma.A, B)$

such that the equations

1. $\widetilde{F}_0(B)\langle\sigma\rangle = \widetilde{F}_0(B\langle\sigma\rangle)$
2. $\widetilde{F}_1(b)\langle\sigma\rangle = \widetilde{F}_1(b\langle\sigma\rangle)$

as well as the equations

1. $\widetilde{F}_0(B)\langle\Gamma.F_1(t)\rangle = \widetilde{F}_0(B\langle\Gamma.t\rangle)$
2. $\widetilde{F}_1(b)\langle\Gamma.F_1(t)\rangle = \widetilde{F}_1(b\langle\Gamma.t\rangle)$

are satisfied *and* the canonical comparison functions $u \in \mathrm{Tm}(\Gamma, F_0(\mathrm{Unit}) \to \mathrm{Unit})$ and $p(B) \in \mathrm{Tm}(\Gamma, F_0(\Sigma_A B) \to \Sigma_{F_0(A)}\widetilde{F}_0(B))$ are isomorphisms, respectively, that is additionally a lex operation structure is given by

1. $\top \in \mathrm{Tm}(\Gamma, F_0(\mathrm{Unit}))$
2. $\langle a, b\rangle \in \mathrm{Tm}(\Gamma, F_0(\Sigma_A B))$ for $a \in \mathrm{Tm}(\Gamma, F_0(A))$ and $b \in \mathrm{Tm}(\Gamma, \widetilde{F}_0(B)[a])$

such that the equations

1. $\top = t$ for $t \in \mathrm{Tm}(\Gamma, F_0(\mathrm{Unit}))$
2. $\pi_1\langle a, b\rangle = a$ and $\pi_2\langle a, b\rangle = b$
3. $\langle\pi_1 p, \pi_2 p\rangle = p$ for $p \in \mathrm{Tm}(\Gamma, F_0(\Sigma_A B))$,

where π_1 and π_2 denote the projections of $p(B)$, are satisfied. ■

We recall the notion of pseudomorphism of cwfs from Kaposi, Huber and Sattler [48]:

Definition 30. Let $\mathscr{C}$ and $\mathscr{D}$ be cwfs. A *pseudomorphism* structure on a functor F from $\mathscr{C}$ to $\mathscr{D}$ is given by

1. $\widetilde{F}_0(A) \in \mathrm{Ty}_{\mathscr{D}}(F_0(\Gamma))$ for $A \in \mathrm{Ty}_{\mathscr{C}}(\Gamma)$
2. $\widetilde{F}_1(t) \in \mathrm{Tm}_{\mathscr{D}}(F_0(\Gamma), \widetilde{F}_0(A))$ for $t \in \mathrm{Tm}_{\mathscr{C}}(\Gamma, A)$

such that the equations

1. $\widetilde{F}_0(A\sigma) = \widetilde{F}_0(A)F_1(\sigma)$
2. $\widetilde{F}_1(t\sigma) = \widetilde{F}_1(t)F_1(\sigma)$

are satisfied *and* the comprehension structure is preserved *up to isomorphism* in the sense that the canonical substitutions $n \in \mathrm{Sub}_{\mathscr{D}}(F_0([]), [])$ and $p(A) \in \mathrm{Sub}_{\mathscr{D}}(F_0(\Gamma.A), F_0(\Gamma).\widetilde{F}_0(A))$ are isomorphisms, that is additionally a pseudomorphism is given by

1. $t \in \mathrm{Sub}_{\mathscr{D}}([], F_0([]))$
2. $s(A) \in \mathrm{Sub}_{\mathscr{D}}(F_0(\Gamma).\widetilde{F}_0(A), F_0(\Gamma.A))$ for $A \in \mathrm{Ty}_{\mathscr{C}}(\Gamma)$

such that the equations

1. $n \circ t = \mathrm{id}_{[]}$
2. $t \circ n = \mathrm{id}_{F_0([])}$
3. $p(A) \circ s(A) = \mathrm{id}_{F_0(\Gamma).\widetilde{F}_0(A)}$
4. $s(A) \circ p(A) = \mathrm{id}_{F_0(\Gamma.A)}$

are satisfied. ■

Note that homomorphisms of cwfs as models of the GAT of cwfs are exactly the pseudomorphisms F of cwfs for which the canonical substitutions $\mathrm{Sub}_{\mathscr{D}}(F_0([]), [])$ and $\mathrm{Sub}_{\mathscr{D}}(F_0(\Gamma.A), F_0(\Gamma).\widetilde{F}_0(A))$ are the respective identity substitutions. Those pseudomorphisms are also called *strict morphisms*.

Also note that a pseudomorphism is in particular an internal functor p-reserving context extension but not necessarily the context in the sense that $F_0(A)$ need not be the identity functor. If we think of a cwf structure as a displayed fibration (that is discrete and comes with a displayed functor corresponding to the comprehension structure) then we can think of a pseudomorphism as a displayed Cartesian functor (that preserves the comprehension (and terminal context) structure).

Proposition 40. *The identity map* $\mathrm{Id}_{\mathscr{C}}$ *for a cwf* $\mathscr{C}$ *and the composite map* $G \circ F$ *for pseudomorphisms* $G : \mathscr{D} \to \mathscr{E}$ *and* $F : \mathscr{C} \to \mathscr{D}$ *are pseudomorphisms, and hence pseudomorphisms form a category.*

Proof. By the identity and composite map we mean the identity and composite functions on the data of a cwf.

It is clear that the identity and composite maps are homomorphisms with respect to the composition and substitution structure of a cwf. What remains to be shown is that they preserve the comprehension structure weakly.

The unique substitution $\mathrm{Sub}(\mathscr{E}, G_0(F_0([])), [])$ is an isomorphism because it is necessarily equal to the composite of the isomorphisms $G_0(F_0([])) \to G_0([])$ and $G_0([]) \to []$. Similarly, the canonical substitution $G_0(F_0(\Gamma.A)) \to G_0(F_0(\Gamma)).\widetilde{G}_0(\widetilde{F}_0(A))$ is the composite of the isomorphisms $G_0(F_0(\Gamma.A)) \to G_0(F_0(\Gamma).\widetilde{F}_0(A))$ and $G_0(F_0(\Gamma).\widetilde{F}_0(A)) \to G_0(F_0(\Gamma)).\widetilde{G}_0(\widetilde{F}_0(A))$. □

Proposition 41. *Let* $\mathscr{C}$ *be a cwf with unit,* Σ*, and* Π *type structures. For each* $\Gamma \in \mathrm{Ctx}$*, denote the induced cwf whose set of contexts is* $\mathrm{Ty}(\Gamma)$ *by* $\mathscr{C}_\Gamma$*, and, for*

each $\sigma \in \mathrm{Sub}(\Delta, \Gamma)$, denote the induced pseudomorphism $\mathscr{C}_\Gamma \to \mathscr{C}_\Delta$ whose action on contexts is substitution $\mathrm{Ty}(\Gamma) \to \mathrm{Ty}(\Delta)$ by $\mathscr{C}_\sigma$. Then, a family of operations

1. $F_0(\Gamma) : \mathrm{Ty}(\Gamma) \to \mathrm{Ty}(\Gamma)$,
2. $F_1(A', A) : \mathrm{Tm}(\Gamma, A' \to A) \to \mathrm{Tm}(\Gamma, F_0(\Gamma, A') \to F_0(\Gamma, A'))$,
3. $\widetilde{F}_0(A) : \mathrm{Ty}(\Gamma.A) \to \mathrm{Ty}(\Gamma.F_0(\Gamma, A))$, *and*
4. $\widetilde{F}_1(B) : \mathrm{Tm}(\Gamma.A, B) \to \mathrm{Tm}(\Gamma.F_0(\Gamma, A), \widetilde{F}_0(A, B))$

is a lex operation structure on $\mathscr{C}$ if and only if it is a natural family of pseudomorphisms $F_\Gamma : \mathscr{C}_\Gamma \to \mathscr{C}_\Gamma$, that is $\mathscr{C}_\sigma \circ F_\Gamma = F_\Delta \circ \mathscr{C}_\sigma$ for $\sigma \in \mathrm{Sub}(\Delta, \Gamma)$.

Proof. Unfolding definitions. □

Definition 31. Let F and G be pseudomorphisms between cwfs $\mathscr{C}$ and $\mathscr{D}$. A *natural transformation* $\eta : F \Rightarrow G$ between F and G is given by

$$\eta_\Gamma \in \mathrm{Sub}(F_0(\Gamma), G_0(\Gamma)) \text{ for } \Gamma \in \mathrm{Ctx}$$

such that the equation

$$G_1(\sigma) \circ \eta_\Gamma = \eta_\Delta \circ F_1(\sigma)$$

is satisfied for $\sigma \in \mathrm{Sub}(\Delta, \Gamma)$. ■

The descent data operation we will consider in Section 7.1 more naturally arises as a pseudoendomorphism and the following corollary will be convenient to obtain the induced lex operation to which we can apply the development in Chapters 2 and 3:

Corollary 42. *Let $\mathscr{C}$ be a cwf with unit and Σ type structures. Given a pseudoendomorphism $F : \mathscr{C} \to \mathscr{C}$ and a natural transformation $\eta : \mathrm{Id}_\mathscr{C} \to F$, the family of pseudoendomorphisms*

$$\mathscr{C}_{\eta_\Gamma} \circ \widetilde{F}_0(\Gamma) : \mathscr{C}_\Gamma \to \mathscr{C}_\Gamma \text{ for } \Gamma \in \mathrm{Ctx}$$

induces a lex operation structure on $\mathscr{C}$.

Proof. F induces the family of pseudomorphisms $\widetilde{F}_0(\Gamma)$ from $\mathscr{C}_\Gamma$ to $\mathscr{C}_{F_0(\Gamma)}$, η the family of pseudomorphisms $\mathscr{C}_{\eta_\Gamma}$ from $\mathscr{C}_{F_0(\Gamma)}$ to $\mathscr{C}_\Gamma$, and families of pseudomorphisms are closed under componentwise composition. Apply Proposition 41. □

When the closed types correspond exactly to the contexts, which is called *democracy* in Clairambault and Dybjer [16], then conversely lex operations induce pseudomorphisms, but we will not need this.

Definition 32. Let $\mathscr{C}$ and $\mathscr{D}$ be cwfs with injective universe type structure. A pseudomorphism F from $\mathscr{C}$ to $\mathscr{D}$ *preserves small types* if $\widetilde{F}_0(\mathrm{El}\, a)$ is small for $a \in \mathrm{Tm}_{\mathscr{C}}(\Gamma, \mathrm{U})$, that is there exists a (unique) $\widehat{f}(a) \in \mathrm{Tm}_{\mathscr{D}}(\widetilde{F}_0(\Gamma), \mathrm{U})$ such that $\mathrm{El}\, \widehat{f}(a) = \widetilde{F}_0(\mathrm{El}\, a)$ for $a \in \mathrm{Tm}_{\mathscr{C}}(\Gamma, \mathrm{U})$. ■

Definition 33. A *descent data operation structure* on a cwf $\mathscr{C}$ with unit, Σ, Π and identity type structure is given by a smallness-preserving lex operation structure D on $\mathscr{C}$ together with:

1. a path w_A between $D_1(\eta_A)$ and $\eta_{D_0(A)}$
2. a function $patch_{D_0(A)}$ from $D_0(D_0(A))$ to $D_0(A)$
3. a homotopy $linv_{D_0(A)}$ between $patch_{D_0(D_0(A))} \circ \eta_{D_0(A)}$ and $\mathrm{id}_{D_0(A)}$

for $A \in \mathrm{Ty}(\Gamma)$ such that the equations

1. $w_A\sigma = w_{A\sigma}$
2. $patch_{D_0(A)}\sigma = patch_{D_0(A)\sigma}$
3. $linv_{D_0(A)}\sigma = linv_{D_0(A)\sigma}$

are satisfied. ■

4.4 Submodel of modal types

Given a descent data operation structure D on a cwf $\mathscr{C}$ we build a new cwf $\mathscr{C}_D$ with the same underlying category of substitutions as $\mathscr{C}$ but with types in context Γ instead given by types equipped with patching structures:

Definition 34. A *patching structure* on a type $A \in \mathrm{Ty}(\Gamma)$ with respect to a descent data operation structure D is given by

1. a function $patch_A \in \mathrm{Tm}(\Gamma, D_0(A) \to A)$
2. a homotopy $linv_A \in \mathrm{Tm}(\Gamma, patch_A \circ \eta_A \sim \mathrm{id}_A)$

■

Note that the types $D_0(A) \in \mathrm{Ty}_{\mathscr{C}}(\Gamma)$ have a patching structure by the definition of a descent data operation structure. Also recall that a patching structure on a type is essentially unique. (Corollary 23)

We will call a type together with a patching structure a *patch algebra*. We will usually use the same notation for elements of $\mathrm{Ty}_{\mathscr{C}_D}(\Gamma)$, i.e. patch algebras, as for elements of $\mathrm{Ty}_{\mathscr{C}}(\Gamma)$, but sometimes also the notation $|A|$ to clearly distinguish the underlying type from the patch algebra A.

The terms of a patch algebra $A \in \mathrm{Ty}_{\mathscr{C}_D}(\Gamma)$ are given by the terms of its underlying type, that is $\mathrm{Tm}_{\mathscr{C}_D}(\Gamma, A) = \mathrm{Tm}_{\mathscr{C}}(\Gamma, |A|)$.

By the stability equations for the underlying internal functor structure, the substitutions $patch_A\sigma$ and $linv_A\sigma$ of a patching structure for A by $\sigma \in \mathrm{Sub}(\Delta, \Gamma)$ is a patching structure for $A\sigma$ so that componentwise substitution induces a substitution structure on $\mathscr{C}_D$.

The display substitution and variable term of a type with a patching structure is taken to be the display substitution and variable term of the underlying type and this determines a comprehension structure on $\mathscr{C}_D$.

Note that to build the basic cwf structure $\mathscr{C}_D$ we are only using the underlying internal (pointed) functor structure of D.

We can interpret the type-theoretic proofs in Part I as patching structures for unit, Σ, Π, and identity types (these make use of the lex operation structure) as well as univalent universe types (this makes use of the descent data operation structure) so that we have the following proposition:

Proposition 43. *A cwf $\mathscr{C}$ with a descent data operation structure D gives rise to a cwf $\mathscr{C}_D$ with unit, Σ, Π, identity, and univalent universe type structures as well as a strict pseudomorphism $A \mapsto |A|$ from $\mathscr{C}_D$ to $\mathscr{C}$ which is the identity functor on the composition structure.*

Moreover, $\mathscr{C}_D$ of Proposition 43 supports a natural numbers type if $\mathscr{C}$ supports a natural numbers patch algebra (cf. Subsection 3.3.1):

Definition 35. A *natural numbers patch algebra structure* on a cwf with a descent data operation structure D is given by six operations:

1. $\mathrm{Nat}_D(\Gamma) \in \mathrm{Ty}(\Gamma)$ for each $\Gamma \in \mathrm{Ctx}$
2. $\mathrm{zero}_D(\Gamma) \in \mathrm{Tm}(\Gamma, \mathrm{Nat}_D)$ for each $\Gamma \in \mathrm{Ctx}$
3. $\mathrm{succ}_D(n) \in \mathrm{Tm}(\Gamma, \mathrm{Nat}_D)$ for each $n \in \mathrm{Tm}(\Gamma, \mathrm{Nat}_D)$
4. $patch_D(x) \in \mathrm{Tm}(\Gamma, \mathrm{Nat}_D)$ for each $x \in \mathrm{Tm}(\Gamma, D_0(\mathrm{Nat}_D))$
5. $linv_D(n) \in \mathrm{Tm}(\Gamma, \mathrm{Id}(patch_D(\eta(n)), n))$ for each $n \in \mathrm{Tm}(\Gamma, \mathrm{Nat}_D)$

6. $\mathrm{elim}(z, s, \widetilde{patch}, \widetilde{linv}, n) \in \mathrm{Tm}(\Gamma, C[n])$ for each $C \in \mathrm{Ty}(\Gamma.\mathrm{Nat}_D)$,
   $z \in \mathrm{Tm}(\Gamma, C[\mathrm{zero}_D])$,
   $s \in \mathrm{Tm}(\Gamma, \Pi_{n:\mathrm{Nat}_D}(C \to C[\mathrm{succ}_D(n)]))$,
   $\widetilde{patch} \in \mathrm{Tm}(\Gamma, \Pi_{x:D_0(\mathrm{Nat}_D)}(\widetilde{D}_0(C) \to C[patch_D(x)]))$,
   $\widetilde{linv} \in \mathrm{Tm}(\Gamma, \Pi_{n:\mathrm{Nat}_D, c:C}\mathrm{Id}_{C, linv_D(n)}(\widetilde{patch}(\eta(n), \tilde{\eta}(n, c)), c))$,
   $n \in \mathrm{Tm}(\Gamma, \mathrm{Nat}_D)$

satisfying the equations

1. $\mathrm{Nat}_D(\Gamma)\sigma = \mathrm{Nat}_D(\Delta)$,
2. $\mathrm{zero}_D(\Gamma)\sigma = \mathrm{zero}_D(\Delta)$,
3. $\mathrm{succ}_D(n)\sigma = \mathrm{succ}_D(n\sigma)$,
4. $patch_D(x)\sigma = patch_D(x\sigma)$,
5. $linv_D(n)\sigma = linv_D(n\sigma)$,
6. $\mathrm{elim}(z, s, \widetilde{patch}, \widetilde{linv}, n)\sigma = \mathrm{elim}(z\sigma, s\sigma, \widetilde{patch}\sigma, \widetilde{linv}\sigma, n\sigma)$,
7. $\mathrm{elim}(z, s, \widetilde{patch}, \widetilde{linv}, \mathrm{zero}_D) = z$, and
8. $\mathrm{elim}(z, s, \widetilde{patch}, \widetilde{linv}, \mathrm{succ}_D(n)) = s(n, \mathrm{elim}(z, s, \widetilde{patch}, \widetilde{linv}, n))$.

■

Proposition 44. *Let $\mathscr{C}$ be a cwf with a descent data operation D and a natural numbers patch algebra structure, then the cwf $\mathscr{C}_D$ has a natural numbers type structure.*

Proof. Since, by definition, Nat_D has a patch algebra structure that is stable under substitution, we have $\mathrm{Nat}(\Gamma) \in \mathrm{Ty}_{\mathscr{C}_D}(\Gamma)$ for each context Γ such that $\mathrm{Nat}(\Gamma)\sigma = \mathrm{Nat}(\Delta)$. Moreover, since the terms of a patch algebra $A \in \mathrm{Ty}_{\mathscr{C}_D}(\Gamma)$ are defined to be the terms of its underlying type $|F| \in \mathrm{Ty}_{\mathscr{C}}(\Gamma)$, that is $\mathrm{Tm}_{\mathscr{C}_D}(\Gamma, A) = \mathrm{Tm}_{\mathscr{C}}(\Gamma, |A|)$ for each $A \in \mathrm{Ty}_{\mathscr{C}_D}(\Gamma)$, we also have $\mathrm{zero}(\Gamma) \in \mathrm{Tm}_{\mathscr{C}_D}(\Gamma, \mathrm{Nat})$ for each context Γ such that $\mathrm{zero}(\Gamma)\sigma = \mathrm{zero}(\Delta)$ and $\mathrm{succ}(n) \in \mathrm{Tm}_{\mathscr{C}_D}(\Gamma, \mathrm{Nat})$ for each $n \in \mathrm{Tm}_{\mathscr{C}_D}(\Gamma, \mathrm{Nat})$ such that $\mathrm{succ}(n)\sigma = \mathrm{succ}(n\sigma)$.

What remains to be shown for Nat to be a natural numbers type structure on $\mathscr{C}_D$ is that we have $\mathrm{elim}(z, s, n) \in \mathrm{Tm}_{\mathscr{C}_D}(\Gamma, C[n])$ for each $C \in \mathrm{Ty}_{\mathscr{C}_D}(\Gamma.\mathrm{Nat})$ and $n \in \mathrm{Tm}_{\mathscr{C}_D}(\Gamma, \mathrm{Nat})$, $z \in \mathrm{Tm}_{\mathscr{C}_D}(\Gamma, C[\mathrm{zero}])$ and $s \in \mathrm{Tm}_{\mathscr{C}_D}(\Gamma, \Pi_{n:\mathrm{Nat}}(C \to C[\mathrm{succ}(n)]))$ such that $\mathrm{elim}(z, s, n)\sigma = \mathrm{elim}(z\sigma, s\sigma, n\sigma)$, $\mathrm{elim}(z, s, \mathrm{zero}) = z$ and $\mathrm{elim}(z, s, \mathrm{succ}(n)) = s(\mathrm{elim}(z, s, n))$. Apply Proposition 35. □

And similarly for propositional truncation:

Definition 36. A *propositional truncation patch algebra structure* on a cwf with a descent data operation structure D is given by six operations:

1. $\|A\|_D \in \mathrm{Ty}(\Gamma)$ for $A \in \mathrm{Ty}(\Gamma)$
2. $\mathrm{inc}(a) \in \mathrm{Tm}(\Gamma, \|A\|_D)$ for $a \in \mathrm{Tm}(\Gamma, A)$
3. $\mathrm{squash}(x, y) \in \mathrm{Tm}(\Gamma, \mathrm{Id}(x, y))$ for $x, y \in \mathrm{Tm}(\Gamma, \|A\|_D)$
4. $\mathrm{patch}(x) \in \mathrm{Tm}(\Gamma, \|A\|_D)$ for $x \in \mathrm{Tm}(\Gamma, D(\|A\|_D))$
5. $\mathrm{linv}(x) \in \mathrm{Tm}(\Gamma, \widetilde{\mathrm{Id}}(\mathrm{patch}(\eta(x)), x))$ for $x \in \mathrm{Tm}(\Gamma, \|A\|_D)$
6. $\mathrm{elim}(i, s, \widetilde{patch}, \widetilde{linv}, x) \in \mathrm{Tm}(\Gamma, C[x])$ for $C \in \mathrm{Ty}(\Gamma.\|A\|_D)$,
   $i \in \mathrm{Tm}(\Gamma.A, C[\mathrm{inc}])$,
   $s \in \mathrm{Tm}(\Gamma.\|A\|_D.C.\|A\|_D\mathrm{p}^2.C\langle \mathrm{p}^2\rangle, \mathrm{Id}_{\mathrm{squash}(\mathrm{qp}^3,\mathrm{qp}^1)}(\mathrm{qp}^2, \mathrm{qp}^0))$,
   $\widetilde{patch} \in \mathrm{Tm}(\Gamma.D(\|A\|_D).\widetilde{D}(C), C\langle\Gamma.\mathrm{patch}\rangle\mathrm{p})$,
   $\widetilde{linv} \in \mathrm{Tm}(\Gamma.\|A\|_D, \mathrm{Id}_C(\mathrm{patch}(\eta(\mathrm{qp}^0)), \mathrm{qp}^0))$,
   $x \in \mathrm{Tm}(\Gamma, \|A\|_D)$

satisfying the equations

1. $\|A\|_D\sigma = \|A\sigma\|_D$,
2. $\mathrm{inc}(a)\sigma = \mathrm{inc}(a\sigma)$,
3. $\mathrm{squash}(x, y)\sigma = \mathrm{squash}(x\sigma, y\sigma)$,
4. $\mathrm{patch}(x)\sigma = \mathrm{patch}(x\sigma)$,
5. $\mathrm{linv}(x)\sigma = \widetilde{linv}(x\sigma)$,
6. $\mathrm{elim}(i, s, \widetilde{patch}, \widetilde{linv}, x)\sigma = \mathrm{elim}(i\langle\sigma\rangle, s\langle\langle\langle\langle\sigma\rangle\rangle\rangle\rangle, \widetilde{patch}\langle\langle\sigma\rangle\rangle, \widetilde{linv}\langle\sigma\rangle, x\sigma)$,
7. $\mathrm{elim}(i, s, \widetilde{patch}, \widetilde{linv}, \mathrm{inc}(a)) = i(a)$, and
8. $\mathrm{elim}(i, s, \widetilde{patch}, \widetilde{linv}, \mathrm{squash}(x, y)) = s(\mathrm{elim}(i, s, \widetilde{patch}, \widetilde{linv}, x), \mathrm{elim}(i, s, \widetilde{patch}, \widetilde{linv}, y))$.

■

Proposition 45. *Let $\mathscr{C}$ be a cwf with a descent data operation D and a propositional truncation patch algebra structure, then the cwf $\mathscr{C}_D$ has a propositional truncation structure.*

Proof. By Definition 36, the assignment of patch algebras $\|A\| = (\|\|A\|\|_D, \mathrm{patch}, \mathrm{linv}) \in \mathrm{Ty}_{\mathscr{C}_D}(\Gamma)$ to $A \in \mathrm{Ty}_{\mathscr{C}_D}(\Gamma)$ is stable under substitution, that is $\|A\|\sigma = \|A\sigma\|$. By Proposition 36 it has an eliminator elim into $B \in \mathrm{Ty}_{\mathscr{C}_D}(\Gamma)$ that are mere propositions. It follows that $(\|A\|, \mathrm{inc}, \mathrm{squash}, \mathrm{elim})$ is a propositional truncation for each A, that is $\mathscr{C}_D$ has a propositional truncation structure. □

Chapter 5

Groupoid model

Hofmann and Streicher [42, 41] answered the question of uniqueness of identity proofs *negatively* by constructing a model of intensional type theory in which there is a type T_0 with elements t, u of T_0 and p, q of $\mathrm{Id}_{T_0}(t, u)$ such that there is *no* element of $\mathrm{Id}_{\mathrm{Id}_{T_0}(t,u)}(p, q)$. In fact, the identity type $\mathrm{Id}(p, q)$ is empty, i.e. the type $\mathrm{Id}(p, q) \to \bot$ is inhabited, and hence T_0 is *not* a (homotopy) set, i.e. the type $\mathrm{isSet}(T_0) \to \bot$ is inhabited.

The model of Hofmann and Streicher also shows that type theory identifies isomorphic structures in the sense that if Σ is some algebraic signature, A and B are two isomorphic Σ-structures (in the set-theoretic model of type theory), and P is a predicate on Σ-structures expressed using the language of type theory then A satisfies P if and only if B satisfies P.

Actually, the universe U is an example of a type T_0 that is *not* a homotopy set in the model of Hofmann and Streicher, i.e. their model inhabits the type $\mathrm{isSet}(\mathrm{U}) \to \bot$. Moreover, one can construct a proof of this in type theory extended with an axiom that asserts that the canonical maps $\mathrm{Id}_{\mathrm{U}}(A, B) \to \mathrm{Equiv}(A, B)$ are equivalences [41]. This axiom is validated in the model [41] and entails that the isomorphic Σ-structures A and B are identified as elements of the type U_Σ of Σ-structures. The invariance of type-theoretic predicates under isomorphism can then be proved using transport.

The model of Hofmann and Streicher interprets contexts as categories in which every morphism is invertible (groupoids), type families $A \in \mathrm{Ty}(\Gamma)$ as functors $A : \Gamma \to \mathrm{Gpd}$, and terms $t \in \mathrm{Tm}(\Gamma, A)$ as sections $t : \Gamma \to \Gamma.A$ of the projection $\mathrm{p}_A : \Gamma.A \to \Gamma$. Identity types Id_A have as points the paths in A. The functors $A(g) : A(\gamma) \to A(\gamma')$ over paths $g : \gamma \xrightarrow{\sim} \gamma' : \Gamma$ interpret the eliminator J for identity types and the functor law $A(\mathrm{id}_\gamma) = \mathrm{Id}_{A(\gamma)}$ is needed to

validate its computation law $\mathrm{J}(d, a, \mathrm{refl}) = d$. The universe type U has (small) *discrete* groupoids as points and isomorphisms between them as paths.

Inspired by the setoid model of type theory formulated with heterogeneous equivalence relations as in Martin-Löf [64], Tonelli [79] and Hofmann [37], in this chapter we recall the groupoid model in a slightly different formulation. Seeing groupoids as proof-relevant setoids with coherence laws for the congruence proofs, contexts are interpreted as groupoids and type families are interpreted as displayed categories [3] fibred in groupoids, in particular a type family specifies a family of sets $A(g, a, a')$ of paths *over* paths $g : \gamma \nrightarrow \gamma' : \Gamma$.

The displayed categories formulation allows for both a simplification of definitions involving dependent paths (when compared to the formulation based on functors) and a strict model (when compared to a formulation based on categories fibred in groupoids).

However, there arises an issue with the strictness of the universe, the law $\mathrm{El}(\Pi_A B) = \Pi_{\mathrm{El}(A)}\mathrm{El}(B)$ for small types A, $B : \mathrm{U}$ for instance. The issue is solved by assuming subsingletons $P \subseteq 1 = \{0\}$ to be closed under dependent products and sums, in particular $\prod_{x \in X} P(x) \subseteq 1$ for any set X and family P of subsingletons $P(x) \subseteq 1$ for $x \in X$. An encoding of functions that has this property was devised by Aczel [1] to interpret a type of impredicative and proof-irrelevant propositions in set theory.

We stress the groupoid model's constructive and elementary nature, which we will make use of in Chapter 6 to relativize the groupoid model to presheaves over an arbitrary base category.

5.1 Basic cwf structure

Recall from Definition 17 in Chapter 4 that a cwf is given by a 1. composition structure on the sets of contexts and substitutions, 2. a substitution structure on the sets of types and terms, and 3. a comprehension structure relating types and terms to substitutions. We organize the presentation of the groupoid model accordingly.

The elements of the *sets* of contexts and types in the groupoid model are structures, namely groupoids and families of groupoids. Correspondingly, the universe type structure on the groupoid model is given by a structure of structures. Since we need a universe in order to define a *set* of all structures of some kind, let $\mathscr{V}$ be an arbitrary but fixed set-theoretic universe. In addition to that, since we need a universe in $\mathscr{V}$ in order to define a set *in* $\mathscr{V}$ of all $\mathscr{V}$-small structures of a certain kind, let $\mathscr{U}$ be a set-theoretic universe that is

an element of $\mathscr{V}$. This means that the construction of the groupoid model is parameterized by two set-theoretic universes.

5.1.1 Contexts and substitutions

We start the construction of the groupoid model by giving the interpretations of the sort symbols Ctx and Sub.

Definition 37. A *context* Γ is given by

1. a $\mathscr{V}$-small set of *points*, which we denote by $\mathrm{Pt}(\Gamma)$ or Γ, and
2. a family assigning $\mathscr{V}$-small sets of *paths* to $\gamma, \gamma' \in \Gamma$, which we denote by $\mathrm{Path}(\Gamma, \gamma, \gamma')$ or $\gamma \nrightarrow \gamma'$

together with operations assigning

1. *identity* paths $\mathrm{id}_\gamma \in \gamma \nrightarrow \gamma$ to $\gamma \in \Gamma$,
2. *composite* paths $\mu \cdot \mu' \in \gamma \nrightarrow \gamma''$ to $\mu \in \gamma \nrightarrow \gamma'$ and $\mu' \in \gamma' \nrightarrow \gamma''$, and
3. *inverse* paths $\mu^{-1} \in \gamma' \nrightarrow \gamma$ to $\mu \in \gamma \nrightarrow \gamma'$

such that the equations

1. $\mathrm{id}_\gamma \cdot \mu = \mu$ and $\mu \cdot \mathrm{id}_{\gamma'} = \mu$ (*left and right unity*),
2. $(\mu \cdot \mu') \cdot \mu'' = \mu \cdot (\mu' \cdot \mu'')$ (*associativity*), and
3. $\mu^{-1} \cdot \mu = \mathrm{id}_{\gamma'}$ and $\mu \cdot \mu^{-1} = \mathrm{id}_\gamma$ (*left and right invertibility*)

are satisfied. ■

Note that, like for groups, the identity and inverse operations are uniquely determined by the axioms, but we include them to emphasize the algebraic nature of the definition. A context in the groupoid model is hence the same as a $\mathscr{V}$-small groupoid, that is a $\mathscr{V}$-small category in which every morphism is invertible.

Definition 38. A *substitution* σ between contexts Δ and Γ is given by operations assigning

1. $\sigma(\delta) \in \Gamma$ to $\delta \in \Delta$, and
2. $\sigma(\nu) \in \sigma(\delta) \nrightarrow \sigma(\delta')$ to $\nu \in \delta \nrightarrow \delta'$

such that the equations

1. $\sigma(\mathrm{id}_\delta) = \mathrm{id}_{\sigma(\delta)}$,

2. $\sigma(\nu \cdot \nu') = \sigma(\nu) \cdot \sigma(\nu')$, and
3. $\sigma(\nu^{-1}) = \sigma(\nu)^{-1}$

are satisfied. ■

Note that, like for homomorphisms of monoids, the inverse preservation equation is derivable because the property of being an inverse is preserved whenever identity and composite preservation hold, and inverses are uniquely determined. In fact, like for homomorphisms of groups, even the identity preservation equation is derivable. However, we include the equation to emphasize that the definition of a substitution matches the notion of homomorphism between contexts seen as algebraic structures with two sorts and three operations. A substitution in the groupoid model is hence the same as a functor.

Proposition 46. *The operation* $\sigma \circ \tau \in \mathrm{Sub}(\mathrm{E}, \Gamma)$ *defined by*

1. $(\sigma \circ \tau)(\varepsilon) = \sigma(\tau(\varepsilon))$ *for* $\varepsilon \in \mathrm{E}$*, and*
2. $(\sigma \circ \tau)(o) = \sigma(\tau(o))$ *for* $o \in \varepsilon \nrightarrow \varepsilon'$

for $\sigma \in \mathrm{Sub}(\Delta, \Gamma)$ *and* $\tau \in \mathrm{Sub}(\mathrm{E}, \Delta)$ *is associative and has units.*

Proof. Just like for functors between categories and homomorphisms between algebraic structures. □

Proposition 47. *The element* $[] \in \mathrm{Ctx}$ *defined by*

1. $\mathrm{Pt} = \{0\}$*, and*
2. $\mathrm{Path}(x, y) = \{0\}$ *for* $x, y \in \mathrm{Pt}$

is a terminal context.

Proof. Just like for categories and algebraic structures. □

5.1.2 Types and terms

We continue the construction of the groupoid model by giving the interpretations of the (dependent) sort symbols Ty and Tm.

Definition 39. A *type* A in context Γ is given by families assigning

1. $\mathscr{V}$-small sets of *points over* γ to $\gamma \in \Gamma$, which we denote by $\mathrm{Pt}(A, \gamma)$ or $A(\gamma)$, and

2. $\mathscr{V}$-small sets of *paths over* μ to $\mu \in \gamma \nrightarrow \gamma'$, $a \in A(\gamma)$, $a' \in A(\gamma')$, which we denote by $\mathrm{Path}(A, \mu, a, a')$ or $a \nrightarrow_\mu a'$

together with operations assigning

1. *identity* paths $\mathrm{id}_a \in a \nrightarrow_{\mathrm{id}_\gamma} a$ to $a \in A(\gamma)$,
2. *composite* paths $\alpha \cdot \alpha' \in a \nrightarrow_{\mu\cdot\mu'} a''$ to $\alpha \in a \nrightarrow_\mu a'$ and $\alpha' \in a' \nrightarrow_{\mu'} a''$,
3. *inverse* paths $\alpha^{-1} \in a' \nrightarrow_{\mu^{-1}} a$ to $\alpha \in a \nrightarrow_\mu a'$, and
4. *transports* $\mu^+ a \in A(\gamma')$ and *lifts* $\mu^\rightarrow a \in a \nrightarrow_\mu \mu^+ a$ to $\mu \in \gamma \nrightarrow \gamma'$ and $a \in A(\gamma)$

such that the equations

1. $\mathrm{id}_a \cdot \alpha = \alpha$ and $\alpha \cdot \mathrm{id}_{a'} = \alpha$ (*left and right unity*),
2. $(\alpha \cdot \alpha') \cdot \alpha'' = \alpha \cdot (\alpha' \cdot \alpha'')$ (*associativity*),
3. $\alpha^{-1} \cdot \alpha = \mathrm{id}_{a'}$ and $\alpha \cdot \alpha^{-1} = \mathrm{id}_a$ (*left and right invertibility*),
4. $\mathrm{id}_\gamma^+ a = a$ and $\mathrm{id}_\gamma{}^\rightarrow a = \mathrm{id}_a$ (*normality*), and
5. $(\mu \cdot \mu')^+ a = \mu'^+ (\mu^+ a)$ and $(\mu \cdot \mu')^\rightarrow a = \mu^\rightarrow a \cdot \mu'^\rightarrow (\mu^+ a)$ (*splitness*)

are satisfied. ■

Note that, as for contexts, the identity and inverse operations are uniquely determined.

We can think of a type in context Γ in the groupoid model as a displayed category over Γ [3] in which every morphism is invertible together with a *split* (op-)lifting operation.

For a point $\gamma \in \Gamma$, the set $\mathrm{Path}(A, \mathrm{id}_\gamma, a_0, a_1)$ of *vertical* paths over γ is also denoted by $a_0 \nrightarrow_\gamma a_1$ and, by abuse of notation, the fibre category of A over γ whose objects are the points over γ and whose morphisms are the vertical paths over γ is also denoted by $A(\gamma)$. The operations of the dual lifting structure induced by the invertibility of the paths in Γ is denoted by $\mu^- a' \in A(\gamma)$ and $\mu^\leftarrow a' \in \mu^- a' \nrightarrow_\mu a'$ for $a' \in A(\gamma')$ and $\mu \in \gamma \nrightarrow \gamma'$.

A type $A \in \mathrm{Ty}(\Gamma)$ will be called *small* if it is $\mathscr{U}$*-small* and *set-like*: all point sets $A(\gamma) \in \mathscr{U}$ and path sets $A(\mu, a, a')$ are in $\mathscr{U}$, and $A(\mu, a, a') \subseteq \{0\}$ and the transports $\mu^+ a$ along paths $\mu : \gamma \nrightarrow \gamma'$ are the unique points $a' \in A(\gamma')$ such that there is a (necessarily unique) path $a \nrightarrow_\mu a'$. Note that small types are in particular discrete [36, 76], i.e. we have $a' = a$ and $\alpha = \mathrm{id}_a$ for all vertical paths $\alpha : a \nrightarrow_\gamma a'$. It follows immediately that the subsets $\mathrm{ty}(\Gamma) \subseteq \mathrm{Ty}(\Gamma)$ of small types are closed under substitution and we will see that they

are also closed under Π, Σ, and identity types. Moreover, two small types are the same whenever their point sets and transport operations coincide. As a result, we will see that small types are classified by the type of ($\mathscr{U}$-small) sets and bijections between them.

Definition 40. A *term* t of type A in context Γ is given by operations assigning

1. $t(\gamma) \in A(\gamma)$ to $\gamma \in \Gamma$, and
2. $t(\mu) \in t(\gamma) \xrightarrow{\sim}_{\mu} t(\gamma')$ to $\mu \in \gamma \xrightarrow{\sim} \gamma'$

such that the equations

1. $t(\mathrm{id}_\gamma) = \mathrm{id}_{t(\gamma)}$,
2. $t(\mu \cdot \mu') = t(\mu) \cdot t(\mu')$, and
3. $t(\mu^{-1}) = t(\mu)^{-1}$

hold. ■

Note that, as for substitutions, the equations $t(\mathrm{id}_\gamma) = \mathrm{id}_{t(\gamma)}$ and $t(\mu^{-1}) = t(\mu)^{-1}$ are actually derivable. Also note that we do *not* require the equation $t(\mu) = \mu^{\rightarrow} t(\gamma)$ to hold.

Proposition 48. *The operations* $A\sigma \in \mathrm{Ty}(\Delta)$ *and* $t\sigma \in \mathrm{Tm}(\Delta, A\sigma)$ *defined by*

1. $\mathrm{Pt}(A\sigma, \delta) = \mathrm{Pt}(A, \sigma(\delta))$, *and*
2. $\mathrm{Path}(A\sigma, \nu, a, a') = \mathrm{Path}(A, \sigma(\nu), a, a')$,

and

1. $t\sigma(\delta) = t(\sigma(\delta))$, *and*
2. $t\sigma(\nu) = t(\sigma(\nu))$

for $\sigma \in \mathrm{Sub}(\Delta, \Gamma)$, $A \in \mathrm{Ty}(\Gamma)$, *and* $t \in \mathrm{Tm}(\Gamma, A)$ *are functorial in* σ, *that is* $-\mathrm{id}_\Gamma$ *is equal to the identity function and* $-(\sigma \circ \tau)$ *is equal to the composite function* $-\tau \circ -\sigma$.

Proof. Unfolding the definitions. □

Before we continue the construction of the groupoid model by giving the comprehension structure, we define some derived structure on types that will be used for the Π type structure and which draws a connection to other notions of type in a model of groupoids.

Bifibration structure The derived structure can be seen to be induced by the fact that the notion of type in the groupoid model is a special case of the notion of (op-)fibration [72, 82, 76] where every morphism is (op-)Cartesian. This is the case simply because paths in a type — in particular the lifts — are invertible by definition so that for $\alpha \in a \simeq_{\mu} a'$ and $\alpha' \in a \simeq_{\mu \cdot \mu'} a''$ the composite path $\alpha^{-1} \cdot \alpha' \in a' \simeq_{\mu'} a''$ is the unique path such that the triangle

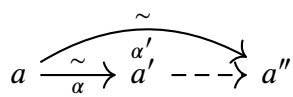


Diagram 3: *Paths in a type in the groupoid model are op-Cartesian*

commutes. The Cartesianness of an arbitrary path α can be seen analogously.

Transport functor structure For $\mu \in \gamma \simeq \gamma'$, the lifts $\mu^{\rightarrow} a \in a \simeq_{\mu} \mu^{+} a$ for $a \in A(\gamma)$ seen as op-Cartesian morphisms give rise to a canonical functor $A(\mu) : A(\gamma) \to A(\gamma')$ mapping $a \in A(\gamma)$ to the transport $\mu^{+} a \in A(\gamma')$ and $\alpha \in a_0 \simeq_{\gamma} a_1$ to the unique vertical path $\mu^{+} \alpha \in \mu^{+} a_0 \simeq_{\gamma'} \mu^{+} a_1$ which makes the square

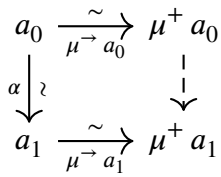


Diagram 4: *Transport in a type in the groupoid model is functorial*

commute. Note that this assignment of $\mu^{+} \alpha$ to $\alpha \in a_0 \simeq_{\gamma} a_1$ is functorial because $\mathrm{id}_{\mu^{+} a}$ makes the square for id_a commute and $\mu^{+} \alpha_0 \cdot \mu^{+} \alpha_1$ makes the square for $\alpha_0 \cdot \alpha_1$ commute so that by uniqueness we get $\mu^{+} \mathrm{id}_a = \mathrm{id}_{\mu^{+} a}$ and $\mu^{+} (\alpha_0 \cdot \alpha_1) = \mu^{+} \alpha_0 \cdot \mu^{+} \alpha_1$.

The assignment of the canonical functors $A(\mu) : A(\gamma) \to A(\gamma')$ just defined to $\mu \in \gamma \simeq \gamma'$ is functorial because the lift operation is (normal and) split by definition. The mapping of types over Γ to groupoid-valued functors on Γ recovers the notion of type in the original formulation of the groupoid model [41].

Giraud–Conduché structure For $\mu \in \gamma \simeq \gamma'$ and $\mu' \in \gamma \simeq \gamma'$, the lifts $\mu^{\rightarrow} a \in a \simeq_{\mu} \mu^{+} a$ for $a \in A(\gamma)$ seen as op-Cartesian morphisms give

rise to canonical factorizations $\mathrm{left}(\mu, \mu', \psi) \cdot \mathrm{right}(\mu, \mu', \psi) = \psi$ of paths $\psi \in a \xrightarrow{\sim}_{\mu \cdot \mu'} a''$ given by $\mathrm{left}(\mu, \mu', \psi) = \mu^{\rightarrow}\, a$ and the unique path $\mathrm{right}(\mu, \mu', \psi) \in \mu^{+}\, a \xrightarrow{\sim}_{\mu'} a''$ which makes the triangle

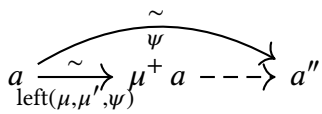
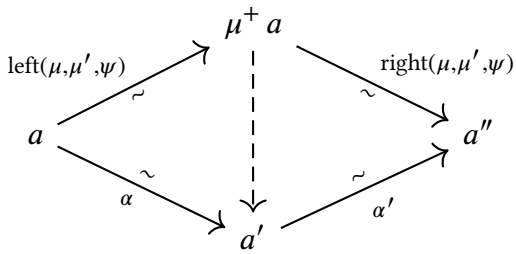


Diagram 5: *Paths over a composite path in a type in the groupoid model can be factored*

commute.

The notion of type in the groupoid model is a special case of the notion of Giraud–Conduché fibration [34, 18] where every pair of factorizations is isomorphic. Given any factorization $\alpha \cdot \alpha' = \psi$ of ψ into morphisms $\alpha \in a \xrightarrow{\sim}_{\mu} a'$ and $\alpha' \in a' \xrightarrow{\sim}_{\mu'} a''$ there is a vertical path such that the diagram



commutes.

5.1.3 Comprehension structure

We end the construction of the basic cwf structure of the groupoid model by giving the interpretations of the operation symbols $\Gamma.A$, p_A, q_A, and $\langle \sigma, t \rangle$.

Proposition 49. *For $A \in \mathrm{Ty}(\Gamma)$, define $\Gamma.A$ as the total category of A, $\mathrm{p}_A \in \mathrm{Sub}(\Gamma.A, \Gamma)$ as its forgetful functor, and $\mathrm{q}_A \in \mathrm{Tm}(\Gamma.A, A\mathrm{p}_A)$ by*

1. $\mathrm{q}_A(\gamma, a) = a$, *and*
2. $\mathrm{q}_A(\mu, \alpha) = \alpha$

Then, the operation $\langle \sigma, t \rangle \in \mathrm{Sub}(\Delta, \Gamma.A)$ defined by

1. $\langle \sigma, t \rangle(\delta) = (\sigma(\delta), t(\delta))$ *for $\delta \in \Delta$, and*
2. $\langle \sigma, t \rangle(\nu) = (\sigma(\nu), t(\nu))$ *for $\nu \in \delta \xrightarrow{\sim} \delta'$*

for $\sigma \in \mathrm{Sub}(\Delta, \Gamma)$ *and* $t \in \mathrm{Tm}(\Delta, A\sigma)$ *produces the unique substitution* $\tau \in \mathrm{Sub}(\Delta, \Gamma.A)$ *such that* $\mathrm{p}_A \circ \tau = \sigma$ *and* $\mathrm{q}_A\tau = t$.

Proof. By definition of (the components of) $\Gamma.A$ as a dependent sum, and p_A and q_A its projections. □

This completes the description of the basic cwf structure of the groupoid model.

5.2 Extra type structure

5.2.1 Encoding of functions and pairs in the metatheory

When embedding sets in (discrete) groupoids we need to make a choice of how to represent the unique path between any pair of points. In principle, that choice is arbitrary as long as the embedding preserves Π and Σ types or, in other words, as long as that choice is preserved by Π and Σ. From this it follows that, in order for small types to be closed under Π and Σ types, we need to be careful what *exactly* we pick as the sets $\prod_{i\in I} X(i)$ and $\sum_{i\in I} X(i)$ of dependent functions and pairs in our metatheory since they will be used to define the path sets of Π and Σ types (see Subsections 5.2.2 and 5.2.3) and small types are required to have path sets that are subsets of $1 = \{0\}$.

Aczel [1] shows how to encode functions (rather than as functional relations) in such a way that the collection $\mathrm{P}(1) = \{X \mid X \subseteq 1\}$ of all subsets of 1 is closed under dependent products indexed by arbitrary sets.

Without assuming P(1) to exist as a set, we can use the function encoding of Aczel [1] to show that the set $\mathscr{P} = \{x \in \mathscr{U} \mid x \subseteq 1\}$ of $\mathscr{U}$*-small* subsets of 1 is closed under dependent products indexed by $\mathscr{U}$-small sets:

Lemma 50. *Let* I *be a set and* X *be a family of sets* $X(i)$ *indexed by* $i \in I$*, then the set*

$$\prod_{i\in I} X(i) \coloneqq \{F \subseteq I \times \bigcup_{i\in I, x\in X(i)} x \mid \forall i \in I.\, \mathrm{app}(F, i) \in X(i)\}$$

together with the function $\mathrm{app}(F, i) \coloneqq \{z \in \bigcup_{i\in I, x\in X(i)} x \mid (i, z) \in F\}$ *is a dependent product of* X *such that*

1. $I \in \mathscr{U}$ *and* $X(i) \in \mathscr{U}$ *for all* $i \in I$ *implies* $\prod_{i\in I} X(i) \in \mathscr{U}$*, and*
2. $X(i) \subseteq 1$ *for all* $i \in I$ *implies* $\prod_{i\in I} X(i) \subseteq 1$.

Property 1 holds for any set-theoretic universe, in particular $\mathscr{V}$ is also closed under these dependent products.

The encoding of pairs such that $\mathscr{P}$ is closed under dependent sums can be reduced to that of functions:

Lemma 51. *Let I be a set and X a family of sets $X(i)$ indexed by $i \in I$, then for $C(0) \coloneqq I$ and $C(1) \coloneqq \bigcup_{i\in I} X(i)$ the set*

$$\sum_{i\in I} X(i) \coloneqq \{P \in \prod_{z\in 2} C(z) \mid \mathrm{fst}(P) \in I \wedge \mathrm{snd}(P) \in X(\mathrm{fst}(P))\}$$

(note that the condition $\mathrm{fst}(P) \in I$ is technically redundant) together with the functions $\mathrm{fst}(P) \coloneqq \mathrm{app}(P, 0)$ and $\mathrm{snd}(P) \coloneqq \mathrm{app}(P, 1)$ is a dependent sum of X such that

1. *$I \in \mathscr{U}$ and $X(i) \in \mathscr{U}$ for all $i \in I$ implies $\sum_{i\in I} X(i) \in \mathscr{U}$, and*
2. *$I \subseteq 1$ and $X(i) \subseteq 1$ for all $i \in I$ implies $\sum_{i\in I} X(i) \subseteq 1$.*

Proof. Since small sets and subsingletons are closed under union, C is in $\mathscr{U}$ if I and X are, and likewise $C(0)$ and $C(1)$ are subsets of 1 if I and $X(i)$ for all $i \in I$ are, so that 1 and 2 hold by Lemma 50. □

As for dependent products, any set-theoretic universe, including $\mathscr{V}$, is closed under these dependent sums.

We remark that, while essential for the encoding of Aczel [1], for our *use* of $\mathscr{P}$ to embed sets in small types it is inessential what singleton precisely we take elements $X \in \mathscr{P}$ to be subsets of, only that it is some fixed singleton so that two elements are equal whenever they are (logically) equivalent. Furthermore, we will not depend on $\mathscr{P}$ containing all $\mathscr{U}$-subsingletons but (in addition to closure under dependent products and sums) closure under the (extensional) identity type, i.e. the sets $\{0 \mid m = m'\}$ for any given $m, m' \in M \in \mathscr{U}$, would suffice. Note that these closure properties seem essential for a groupoid to strictly classify a subset of groupoid families as defined in Definition 39.

5.2.2 Σ type structure

We define the Σ type structure on the groupoid model by giving the interpretations of the operation symbols $\Sigma_A B$, (a, b), fst p and snd p.

Let A be a type in a context Γ and B a type in the context $\Gamma.A$.

We begin with the definition of the type $\Sigma_A B \in \mathrm{Ty}(\Gamma)$. We define the set of points over $\gamma \in \Gamma$ as $(\Sigma_A B)(\gamma) = \sum_{a \in A(\gamma)} B(\gamma, a)$ and the sets of paths over $\mu \in \gamma \nrightarrow \gamma'$ as $(\Sigma_A B)(\mu, (a, b), (a', b')) = \sum_{\alpha \in A(\mu, a, a')} B((\mu, \alpha), b, b')$. Note that $\Sigma_A B$ is set-like if both A and B are because then $\sum_{\alpha \in A(\mu,a,a')} B((\mu, \alpha), b, b') \subseteq 1$, and first a' and α and then b' and β are uniquely determined by a and b by the discreteness of A and B. The composition, invertibility and lifting structures on the points and paths are then given componentwise. The naturality of the operation $\Sigma_A B$ with respect to the type substitution operation can be verified by unfolding the definitions.

If A and B are $\mathscr{U}$-small then $\Sigma_A B$ is too. If they are moreover set-like then $\Sigma_A B$ is too because $\mathscr{P}$ is closed under dependent sums and it can be checked that $\Sigma_A B$ is discrete when A and B are. In particular, $\Sigma_A B$ is small if A and B are.

We continue with the definitions of the terms (a, b), $\mathrm{fst}\, p$ and $\mathrm{snd}\, p$. Given terms $a \in \mathrm{Tm}(\Gamma, A)$ and $b \in \mathrm{Tm}(\Gamma, B[a])$, define the term $(a, b) \in \mathrm{Tm}(\Gamma, \Sigma_A B)$ by $(a, b)(\gamma) = (a(\gamma), b(\gamma))$ and $(a, b)(\mu) = (a(\mu), b(\mu))$. Given a term $p \in \mathrm{Tm}(\Gamma, \Sigma_A B)$, define the terms $\mathrm{fst}\, p \in \mathrm{Tm}(\Gamma, A)$ and $\mathrm{snd}\, p \in \mathrm{Tm}(\Gamma, B[\mathrm{fst}\, p])$ by $\mathrm{fst}\, p(\gamma) = \pi_1\, p(\gamma)$ and $\mathrm{fst}\, p(\mu) = \pi_1\, p(\mu)$, and $\mathrm{snd}\, p(\gamma) = \pi_2\, p(\gamma)$ and $\mathrm{snd}\, p(\mu) = \pi_2\, p(\mu)$. These definitions satisfy the preservation equations of terms because the terms a, b, and p do. The naturality of the operations (a, b), $\mathrm{fst}\, p$, and $\mathrm{snd}\, p$ with respect to the term substitution operation as well as the β and η equations $\mathrm{fst}\,(a, b) = a$, $\mathrm{snd}\,(a, b)$ and $p = (\mathrm{fst}\, p, \mathrm{snd}\, p)$ can be verified by unfolding the definitions.

5.2.3 Π type structure

We define the Π type structure on the cwf constructed at the beginning of this chapter by giving the interpretations of the operation symbols $\Pi_A B$, $\lambda\, b$ and $\mathrm{app}(t, u)$.

Let A be a type in a context Γ and B a type in the context $\Gamma.A$. We begin with the definition of the type $\Pi_A B \in \mathrm{Ty}(\Gamma)$.

A point $s \in (\Pi_A B)(\gamma)$ over $\gamma \in \Gamma$ is given by operations assigning

1. $s(a) \in B(\gamma, a)$ to $a \in A(\gamma)$, and
2. $s(\alpha) \in s(a) \nrightarrow_{(\mathrm{id}_\gamma, \alpha)} s(a')$ to $\alpha \in a \nrightarrow_\gamma a'$

such that the preservation equations

1. $s(\mathrm{id}_a) = \mathrm{id}_{s(a)}$, and

2. $s(\alpha_0 \cdot \alpha_1) = s(\alpha_0) \cdot s(\alpha_1)$

hold. Seeing γ as a substitution $\gamma \in \mathrm{Sub}([], \Gamma)$, the set $(\Pi_A B)(\gamma)$ is hence the same as the set $\mathrm{Tm}(A(\gamma), B\langle\gamma\rangle)$ of terms of type $B\langle\gamma\rangle$ in context $A(\gamma)$.

A path $\omega \in s \mathrel{\overset{\sim}{\to}}_\mu s'$ over $\mu \in \gamma \mathrel{\overset{\sim}{\to}} \gamma'$ is given by an operation assigning

$$\omega(\alpha) \in s(a) \mathrel{\overset{\sim}{\to}}_{(\mu,\alpha)} s'(a')$$

to $\alpha \in a \mathrel{\overset{\sim}{\to}}_\mu a'$ such that the naturality equations

1. $s(\alpha_0) \cdot \omega(\alpha) = \omega(\alpha_0 \cdot \alpha)$ for vertical paths α_0 over γ, and
2. $\omega(\alpha) \cdot s(\alpha_0') = \omega(\alpha \cdot \alpha_0')$ for vertical paths α_0' over γ'

are satisfied.

Note that $\Pi_A B$ is set-like when B is since then $\mathrm{Path}(B, (\mu, \alpha), s(a), s'(a')) \subseteq 1$ and hence $\mathrm{Path}(\Pi_A B, \mu, s, s') \subseteq \prod_{\alpha \in a \mathrel{\overset{\sim}{\to}}_\mu a'} \mathrm{Path}(B, (\mu, \alpha), s(a), s'(a')) \subseteq 1$; the condition that for $\omega \in s \mathrel{\overset{\sim}{\to}}_\gamma s'$ and $\omega' \in s \mathrel{\overset{\sim}{\to}}_\gamma s''$ we have $s'' = s'$ (and $\omega' = \omega$) by discreteness of B can also be checked easily.

Let us define the composition structure on $\Pi_A B$. For paths $\omega \in s \mathrel{\overset{\sim}{\to}}_\mu s'$ and $\omega' \in s' \mathrel{\overset{\sim}{\to}}_{\mu'} s''$, using the canonical factorization given by the Giraud–Conduché structure (Subsection 5.1.2), define the composite path $\omega \cdot \omega' \in s \mathrel{\overset{\sim}{\to}}_{\mu \cdot \mu'} s''$ by $(\omega \cdot \omega')(\alpha) = \omega(\mathrm{left}(\mu, \mu', \alpha)) \cdot \omega'(\mathrm{right}(\mu, \mu', \alpha))$ for $\alpha \in a \mathrel{\overset{\sim}{\to}}_{\mu \cdot \mu'} a''$.

The definition of the composite path is independent of the chosen factorization:

Lemma 52. *For $\omega \in s \mathrel{\overset{\sim}{\to}}_\mu s'$, $\omega' \in s' \mathrel{\overset{\sim}{\to}}_{\mu'} s''$ and $\alpha \in a \mathrel{\overset{\sim}{\to}}_\mu a'$, $\alpha' \in a' \mathrel{\overset{\sim}{\to}}_{\mu'} a''$, the equation $(\omega \cdot \omega')(\alpha \cdot \alpha') = \omega(\alpha) \cdot \omega(\alpha')$ holds.*

Proof. Using the Giraud–Conduché structure (Subsection 5.1.2), the canonical factorization is isomorphic to the given one, that is there is a vertical path α_0' over γ' such that $\mathrm{left}(\mu, \mu', \alpha \cdot \alpha') \cdot \alpha_0' = \alpha$ and $\alpha_0' \cdot \alpha' = \mathrm{right}(\mu, \mu', \alpha \cdot \alpha')$. Thus, by definition of the composite path $\omega \cdot \omega'$ and naturality of the paths ω and ω', we have

$$\begin{aligned}(\omega \cdot \omega')(\alpha \cdot \alpha') &= \omega(\mathrm{left}(\mu, \mu', \alpha \cdot \alpha')) \cdot \omega'(\mathrm{right}(\mu, \mu', \alpha \cdot \alpha')) \\ &= \omega(\mathrm{left}(\mu, \mu', \alpha \cdot \alpha')) \cdot s'(\alpha_0') \cdot \omega'(\alpha') \\ &= \omega(\alpha) \cdot \omega'(\alpha')\end{aligned}$$

□

The definition of the composite path is natural: First, notice that, using the Giraud–Conduché structure, it suffices to check $(\omega \cdot \omega')(\alpha_0 \cdot \alpha \cdot \alpha') = s(\alpha_0) \cdot (\omega \cdot \omega')(\alpha \cdot \alpha')$ for a vertical path α_0 over γ, a path α over μ, and a path α' over μ'. Then, using the key observation, we have $(\omega \cdot \omega')(\alpha_0 \cdot \alpha \cdot \alpha') = \omega(\alpha_0 \cdot \alpha) \cdot \omega'(\alpha')$. And, by naturality of paths and another application of the key observation, we have $(\omega \cdot \omega')(\alpha_0 \cdot \alpha \cdot \alpha') = s(\alpha_0) \cdot \omega(\alpha \cdot \alpha')$. Naturality on the right is analogous.

The composition operation is associative: Notice again that, using the Giraud–Conduché structure, it suffices to check $(\omega \cdot (\omega' \cdot \omega''))(\alpha \cdot \alpha' \cdot \alpha'') = ((\omega \cdot \omega') \cdot \omega'')(\alpha \cdot \alpha' \cdot \alpha'')$ for paths α, α', and α'' over μ, μ', and μ'', respectively, but this follows from the key observation and associativity of composition in $\boldsymbol{A}$ and $\boldsymbol{B}$.

The composition operation has units: The definition of the unit path $\mathrm{id}_s \in s \mathrel{\overset{\sim}{\to}}_{\gamma} s$ by $\mathrm{id}_s(\alpha) = s(\alpha)$ is natural because the point s preserves composition, and id_s is indeed a unit because $(\mathrm{id}_s \cdot \omega)(\alpha_0 \cdot \alpha) = s(\alpha_0) \cdot \omega(\alpha) = \omega(\alpha_0 \cdot \alpha)$. Unity on the right is analogous.

Lastly, the composition operation has inverses: The definition of the inverse path $\omega^{-1} \in s' \mathrel{\overset{\sim}{\to}}_{\mu^{-1}} s$ by $\omega^{-1}(\alpha) = \omega(\alpha^{-1})^{-1}$ is natural because we have $\omega((\alpha_0 \cdot \alpha)^{-1}) \cdot s(\alpha_0) = \omega(\alpha^{-1})$ and hence $\omega^{-1}(\alpha_0 \cdot \alpha) = s(\alpha_0) \cdot \omega^{-1}(\alpha)$, and analogously on the right; and, ω^{-1} is indeed an inverse of ω because we have $s(\alpha \cdot \alpha') \cdot \omega(\alpha'^{-1}) = \omega(\alpha)$ and hence $(\omega \cdot \omega^{-1})(\alpha \cdot \alpha') = s(\alpha \cdot \alpha')$. Invertibility on the right is analogous.

Let us define the lifting structure on $\Pi_{\boldsymbol{A}} \boldsymbol{B}$. For a point $s \in (\Pi_{\boldsymbol{A}} \boldsymbol{B})(\gamma)$ and a path $\mu \in \gamma \mathrel{\overset{\sim}{\to}} \gamma'$, define the transport $\mu^+ s \in (\Pi_{\boldsymbol{A}} \boldsymbol{B})(\gamma')$ by

1. $(\mu^+ s)(a') = \mu^+ s(\mu^- a')$ for $a' \in A(\gamma')$, and
2. $(\mu^+ s)(\alpha') = \mu^+ s(\mu^- \alpha')$ for $\alpha' \in a'_0 \mathrel{\overset{\sim}{\to}}_{\gamma'} a'_1$

where we write $\mu^+ b \in \boldsymbol{B}(\gamma', \mu^+ a)$ and $\mu^+ \beta \in \mu^+ b \mathrel{\overset{\sim}{\to}}_{(\mathrm{id}_{\gamma'}, \mu^+ \alpha)} \mu^+ b'$ for the components $\pi_2(\mu^+ (a, b))$ and $\pi_2(\mu^+ (\alpha, \beta))$, respectively, when $b \in \boldsymbol{B}(\gamma, a)$ and $\beta \in b \mathrel{\overset{\sim}{\to}}_{(\mathrm{id}_{\gamma}, \alpha)} b'$. This definition is functorial because $\mu^+ \beta$ is.

The transports $\mu^+ b$ of points b over γ along μ are the endpoints of the associated lifts $\mu^{\rightarrow} b = \pi_2(\mu^{\rightarrow} (a, b)) \in b \mathrel{\overset{\sim}{\to}}_{(\mu, \mu^{\rightarrow} \alpha)} \mu^+ b$.

The lift $\mu^{\rightarrow} s \in s \mathrel{\overset{\sim}{\to}}_{\mu} \mu^+ s$ is uniquely determined by $(\mu^{\rightarrow} s)(\mu^{\leftarrow} a') = \mu^{\rightarrow} s(\mu^- a')$ for $a' \in A(\gamma')$ because we can make the following observation:

Lemma 53. *Let $s \in (\Pi_{\boldsymbol{A}} \boldsymbol{B})(\gamma)$ and $s' \in (\Pi_{\boldsymbol{A}} \boldsymbol{B})(\gamma')$. Given an arbitrary family $\omega_a \in s(a) \mathrel{\overset{\sim}{\to}}_{(\mu, \mu^{\rightarrow} a)} s'(\mu^+ a)$ for $a \in A(\gamma)$, there is a unique path $\omega \in s \mathrel{\overset{\sim}{\to}}_{\mu} s'$ such that $\omega(\mu^{\rightarrow} a) = \omega_a$ for every $a \in A(\gamma)$.*

Proof. For $\alpha \in a \nrightarrow_{\mu} a'$, by the bifibration structure (5.1.2), there exists a unique vertical path $\mu{\downarrow}\,\alpha$ over γ' such that $\mu^{\rightarrow}\, a \cdot \mu{\downarrow}\,\alpha = \alpha$ and we define $\omega(\alpha) = \omega_a \cdot s'(\mu{\downarrow}\,\alpha) \in s(a) \nrightarrow_{(\mu,\alpha)} s'(a')$. This definition is natural because

$$\omega(\alpha \cdot \alpha_0') = \omega_a \cdot s'(\mu{\downarrow}\,(\alpha \cdot \alpha_0')) = \omega_a \cdot s'(\mu{\downarrow}\,\alpha \cdot \alpha_0') = \omega(\alpha) \cdot s'(\alpha_0')$$

for vertical paths α_0' over γ'. □

We can then verify the normality and splitness equations for the lifting structure because they are satisfied by $\mu^{+}\, b$, $\mu^{+}\, \beta$ and $\mu^{\rightarrow}\, b$.

The naturality of the operation $\Pi_A B \in \mathrm{Ty}(\Gamma)$ with respect to the type substitution operation can be verified by unfolding the definitions.

If A and B are $\mathscr{U}$-small then $\Pi_A B$ is too. If B is moreover set-like then $\Pi_A B$ is too because $\mathscr{P}$ is closed under dependent products indexed by $\mathscr{U}$-small sets and it can be checked that $\Pi_A B$ is discrete when B is. In particular, $\Pi_A B$ is small if A and B are.

We end the construction of the Π type structure on the groupoid model by giving the interpretations of the operation symbols λ and app.

Given $b \in \mathrm{Tm}(\Gamma.A, B)$, define $\lambda\, b \in \mathrm{Tm}(\Gamma, \Pi_A B)$ by

1. $(\lambda\, b)(\gamma)(a) = b(\gamma, a)$,
2. $(\lambda\, b)(\gamma)(\alpha) = b(\mathrm{id}_{\gamma}, \alpha)$, and
3. $(\lambda\, b)(\mu)(\alpha) = b(\mu, \alpha)$.

This definition satisfies the preservation equations of points in $\Pi_A B$ and terms in the groupoid model because b satisfies them and the composition structure on $\Gamma.A$ is defined componentwise.

Given $t \in \mathrm{Tm}(\Gamma, \Pi_A B)$ and $a \in \mathrm{Tm}(\Gamma, A)$, define $\mathrm{app}(t, a) \in \mathrm{Tm}(\Gamma, B[a])$ by

1. $\mathrm{app}(t, a)(\gamma) = t(\gamma)(a(\gamma))$, and
2. $\mathrm{app}(t, a)(\mu) = t(\mu)(a(\mu))$.

This definition satisfies the preservation equations because t and a satisfy them.

The naturality of the operations $\lambda\, b$ and $\mathrm{app}(t, a)$ with respect to the term substitution operation as well as the β and η equations $\mathrm{app}(\lambda\, b, a) = b[a]$ and $t = \lambda\, \mathrm{app}(t \mathrm{p}_A\, \mathrm{q}_A)$ can be verified by unfolding the definitions.

5.2.4 Identity type structure

We define the operation $\mathrm{Id}_A \in \mathrm{Ty}(\Gamma.(A \times A))$, where $A \times A$ denotes the type $\Sigma_A A\mathrm{p}_A$, as the set-like family given by the sets $\mathrm{Id}_A(\gamma, a, a') = A(\mathrm{id}_\gamma, a, a')$ and the sets $\mathrm{Id}_A((\mu, \alpha, \alpha'), \alpha_0, \alpha_1) = \{0 \mid \alpha_0 \cdot \alpha' = \alpha \cdot \alpha_1\}$. Given $\alpha_0 \in \mathrm{Id}_A(\gamma, a_0, a_1)$ and $\alpha \in a_0 \nrightarrow_\mu a_0'$ and $\alpha' \in a_1 \nrightarrow_\mu a_1'$ we have $(g, \alpha, \alpha')^+ \alpha_0 = \alpha^{-1} \cdot \alpha_0 \cdot \alpha'$ in $\mathrm{Id}_A(\gamma', a_0', a_1')$. We can then check that Id_A is stable under substitution.

For each term $a \in \mathrm{Tm}(\Gamma, A)$ we define the term $\mathrm{refl}_a \in \mathrm{Tm}(\Gamma, \mathrm{Id}_A(a, a))$ by $\mathrm{refl}_a(\gamma) = \mathrm{id}_{a(\gamma)}$. This definition is stable under substitution.

Since Id_A is set-like by construction (for any A), note that the paths

$$(\mu, (\alpha, \alpha'), x) \in \mathrm{Path}(\Gamma.(A \times A).\mathrm{Id}_A, (\gamma, (a, b), p), (\gamma', (a', b'), p'))$$

are determined by the components μ and (α, α'). So, for each family $C \in \mathrm{Ty}(\Gamma.(A \times A).\mathrm{Id}_A)$, terms $a,\ b \in \mathrm{Tm}(\Gamma, A)$, $d \in \mathrm{Tm}(\Gamma, C((a, a), \mathrm{refl}_a))$, and $p \in \mathrm{Tm}(\Gamma, \mathrm{Id}_A(a, b))$ we can define the term $\mathrm{J}(d, p) \in \mathrm{Tm}(\Gamma, C((a, b), p))$ by assigning

1. $\mathrm{J}(d, p)(\gamma) = (\mathrm{id}_\gamma, (\mathrm{id}_{a(\gamma)}, p(\gamma)))^+ d(\gamma)$ to $\gamma \in \Gamma$, and
2. $\mathrm{J}(d, p)(\mu) = (\mathrm{id}_\gamma, (\mathrm{id}_{a(\gamma)}, p(\gamma)))\downarrow (d(\mu) \cdot (\mathrm{id}_{\gamma'}, (\mathrm{id}_{a(\gamma')}, p(\gamma')))^{\rightarrow} d(\gamma'))$ to $\mu \in \gamma \nrightarrow \gamma'$

This definition is stable under substitution and we can verify the β equation $\mathrm{J}(d, \mathrm{refl}_a) = d$ by unfolding the definitions.

If A is $\mathscr{U}$-small then Id_A is too. Moreover, Id_A is set-like by construction so that Id_A is small whenever A is.

Let $A \in \mathrm{Ty}(\Gamma)$. Call A a *(homotopy) proposition* if the type $\mathrm{isProp}(A) = \Pi_{x,y:A}\mathrm{Id}(x, y)$ [69, Definition 3.3.1] has a section. Say that A is *codiscrete* if for all points a and $a' \in A(\gamma)$ there exists a unique path $a \nrightarrow_\gamma a'$. In fact, if A is codiscrete then for all $\mu : \gamma \nrightarrow \gamma'$, $a \in A(\gamma)$ and $a' \in A(\gamma')$ there exists a unique path $a \nrightarrow_\mu a'$. Say that A is a *strict proposition* if all vertical paths are constant, i.e. for all paths $\alpha : a \nrightarrow_\gamma a'$ we have $a' = a$ and $\alpha = \mathrm{id}_a$.

We end this subsection with the following pointwise characterization of homotopy propositions as the codiscrete types, and classically the types that are equivalent to strict propositions.

Lemma 54. *If $A \in \mathrm{Ty}(\Gamma)$ is codiscrete then A is a homotopy proposition. The converse holds as well.*

Proof. Let $A \in \mathrm{Ty}(\Gamma)$ be codiscrete. This means that we have a (unique) path $p(\gamma, a, a') : a \nrightarrow_\gamma a'$ for each $\gamma \in \Gamma$, $a,\ a' \in A(\gamma)$. This extends

to a section p of isProp(A) since by discreteness of Id it suffices to check $p(\gamma, a_0, a_0') \simeq_{\mu,\alpha,\alpha'} p(\gamma', a_1, a_1')$ for all $\mu : \gamma \simeq \gamma'$, $\alpha : a_0 \simeq_\mu a_1$ and $\alpha' : a_0' \simeq_\mu a_1'$. But we have $p(\gamma, a_0, a_0') \cdot \alpha' = \alpha \cdot p(\gamma', a_1, a_1') : a_0 \simeq_\mu a_1'$ by codiscreteness of A.

Conversely, if $A \in \mathrm{Ty}(\Gamma)$ is a homotopy proposition then we have in particular a section of isSet(A) [69, Lemma 3.3.4] and thus by discreteness of Id the path sets $A(\gamma, a, a')$ are singletons, i.e. A is codiscrete. □

Lemma 55. *In a classical metatheory, if $A \in \mathrm{Ty}(\Gamma)$ is a homotopy proposition then A is equivalent to a strict proposition.*

Proof. Let $A \in \mathrm{Ty}(\Gamma)$. Define a strict proposition $C(A) \in \mathrm{Ty}(\Gamma)$ by $C(A)(\gamma) = \{0 \mid \exists a.\, a \in A(\gamma)\}$. We show that, in a classical metatheory, the unique map $c : A \to C(A)$ is invertible when A is a homotopy proposition. Let A be a homotopy proposition, and hence codiscrete by Lemma 54. By an application of the axiom of choice there is a function $c' : \prod_{\gamma\in\Gamma, x\in\{0 \mid \exists a.\, a\in A(\gamma)\}} A(\gamma)$, which extends to a map $c' : C(A) \to A$ by codiscreteness of A. The maps $c' : C(A) \to A$ and $c : A \to C(A)$ are inverses because A and $C(A)$ are homotopy propositions [69, Lemma 3.3.3]. □

The converse of this lemma holds because, even in a constructive metatheory, being a homotopy proposition is invariant under equivalence [69, Corollary 7.1.5].

The statement can be generalized to the fact that, in a classical metatheory, if $A \in \mathrm{Ty}(\Gamma)$ is a homotopy set then A is equivalent to the *discrete* type that identifies points in the same connected component.

5.2.5 Booleans and natural numbers type structures

The set-like types Bool $\in$ Ty([]) and Nat $\in$ Ty([]) are given by taking the sets $2 = \{0, 1\}$ of Booleans and $\mathbb{N}$ of natural numbers, respectively, as the set of points.

We have Bool $\in$ ty([]) and Nat $\in$ ty([]) because they are set-like by construction and $\mathscr{U}$-small because the sets 2 and $\mathbb{N}$ are.

The operations true and false $\in \mathrm{Tm}([], \mathrm{Bool})$ as well as zero $\in \mathrm{Tm}([], \mathrm{Nat})$ and $\mathrm{succ}(n) \in \mathrm{Tm}(\Gamma, \mathrm{Nat}\langle\rangle)$ for $n \in \mathrm{Tm}(\Gamma, \mathrm{Nat}\langle\rangle)$ are given by the corresponding set-theoretic operations. The operations $\mathrm{elim}(c_0, c_1, b) \in \mathrm{Tm}(\Gamma, C[b])$ and $\mathrm{elim}(d_0, d_1, n) \in \mathrm{Tm}(\Gamma, D[n])$ are defined by induction on $b(\gamma) \in \mathrm{Bool} = 2$ and $n(\gamma) \in \mathrm{Nat} = \mathbb{N}$ for $\gamma \in \Gamma$, respectively. The equations for stability under substitution and computation hold by definition.

5.2.6 Propositional truncation type structure

Let $A \in \mathrm{Ty}(\Gamma)$. We define $\|A\| \in \mathrm{Ty}(\Gamma)$ by $\|A\|(\gamma) = A(\gamma)$ and $\|A\|(\mu, a, a') = \{0\}$. The transport of $a \in A(\gamma)$ over $\mu : \gamma \nrightarrow \gamma'$ is defined as in A. The groupoid and lifting structure are then trivial. It follows directly that the substitution law $\|A\sigma\| = \|A\|\sigma$ holds.

We define $\mathrm{inc}(a) \in \mathrm{Tm}(\Gamma, \|A\|)$ for $a \in \mathrm{Tm}(\Gamma, A)$ by $\mathrm{inc}(a)(\gamma) = a(\gamma)$. The action on paths is trivial and $\mathrm{inc}(a\sigma) = \mathrm{inc}(a)\sigma$ follows immediately. It is also trivial to define $\mathrm{squash}(t, u) \in \mathrm{Tm}(\Gamma, \mathrm{Id}_{\|A\|}(t, u))$ and check $\mathrm{squash}(t\sigma, u\sigma) = \mathrm{squash}(t, u)\sigma$ for t and $u \in \mathrm{Tm}(\Gamma, \|A\|)$.

Let $C \in \mathrm{Ty}(\Gamma)$, $p \in \mathrm{Tm}(\Gamma, \mathrm{isProp}(C))$ and $c \in \mathrm{Tm}(\Gamma.A, C\mathrm{p})$. We define $\mathrm{rec}(c, p, t) \in \mathrm{Tm}(\Gamma, C)$ for $t \in \mathrm{Tm}(\Gamma, \|A\|)$ by $\mathrm{rec}(c, p, t)(\gamma) = c(\gamma, t(\gamma))$. The action on paths is trivial because the existence of p implies that C is codiscrete, i.e. $C(\mu, x, y)$ is a singleton for all $x \in C(\gamma)$ and $y \in C(\gamma')$. The computation law $\mathrm{rec}(c, p, \mathrm{inc}(a)) = c[a]$ follows immediately.

We end with the observation that the set-like types satisfy the axiom of choice constructively, in particular the axiom of countable choice is validated constructively. We also observe that the argument used in [69, Theorem 10.1.14] to prove Diaconescu's theorem [24] that choice implies the principle of excluded middle does not apply because the set-like types are not closed under suspension for a similar reason that they are not closed under propositional truncation. It is therefore that having choice for families indexed by set-like types constructively does not contradict the fact that the groupoid model does not validate the law of excluded middle unless it also holds in the metatheory.

5.2.7 Universe type structure

We have seen that the subsets $\mathrm{ty}(\Gamma)$ of small types in context Γ are closed under the operations $\Pi_A B$, $\Sigma_A B$ and Id_A. To give a universe type structure via Proposition 37 it remains to construct a universal small family $\mathrm{El} \in \mathrm{Ty}([].\mathrm{U})$ in the sense of Definition 26.

Define the closed type $\mathrm{U} \in \mathrm{Ty}([])$ by the point set $\mathrm{U} = \mathscr{U}$ and the path set $\mathrm{U}(M, N) = \{k \in M \to N \mid k \text{ bijection}\}$ for each pair of $\mathscr{U}$-small sets M and N. The composition operation defined by $k \cdot k' = k' \circ k$ is associative, and has both units and inverses. The lifting structure is trivial because U is closed.

Define the $\mathscr{U}$-small and set-like family $\mathrm{El} \in \mathrm{Ty}([].\mathrm{U})$ over U by the point set $\mathrm{El}(M) = M$ for each $\mathscr{U}$-small set M, the path set $\mathrm{El}(M)(k, m, n) = \{0 \mid n = k(m)\}$ for each bijection $k : M \to N$ and pair of elements $m \in M$ and $n \in N$, and the transport operation $k^+ m = k(m)$, which is functorial because the iden-

tity and composite paths in U are given by the identity functions and function composition. The composition structure and lift operation are trivial because El is set-like.

Proposition 56. *The operation* $\mathrm{In}(A) \in \mathrm{Tm}(\Gamma, \mathrm{U}\langle\rangle_\Gamma)$ *for* $A \in \mathrm{ty}(\Gamma)$ *defined by*

1. $\mathrm{In}(A)(\gamma) = A(\gamma)$, *and*
2. $\mathrm{In}(A)(\mu) = \{a \mapsto \mu^+ a\}$, *which is bijective because* μ^+ *is an isomorphism,*

produces the unique term $a \in \mathrm{Tm}(\Gamma, \mathrm{U}\langle\rangle_\Gamma)$ *such that* $\mathrm{El}(a) = A$, *in particular we have* $\mathrm{In}(A\sigma) = \mathrm{In}(A)\sigma$ *and* $\mathrm{In}(\mathrm{El}\, a) = a$ *for* $a \in \mathrm{Tm}(\Gamma, \mathrm{U}\langle\rangle_\Gamma)$.

Proof. Using the fact that set-like type families are equal if and only if their point sets and transport operations are equal. □

This concludes the description of the cwf structure of the groupoid model:

Theorem 57. *Groupoids form a cwf with unit, Σ, Π, identity, univalent universe, and propositional truncation type structures.*

5.3 Towards a groupoid-valued presheaf model

To conclude this chapter, we observe that like [37, 41] the original presentation of the groupoid model the presentation above can be constructed in (a model of) type theory with a universe $\mathscr{V}$ that contains both a universe $\mathscr{U}$ and a universe $\mathscr{P}$ of propositions such that identity types satisfy equality reflection:

$$\frac{p : \mathrm{Id}_A(a, b)}{a = b : A}$$

Furthermore, we require the universe $\mathscr{P}$ to contain the unit type, to be closed under identity types of types in $\mathscr{U}$ as well as dependent products of families indexed by types in $\mathscr{U}$, and to satisfy propositional extensionality:

$$\frac{}{\mathrm{Unit} : \mathscr{P}} \qquad \frac{A : \mathscr{U}}{\mathrm{Id}_A(a, b) : \mathscr{P}} \qquad \frac{A : \mathscr{U} \qquad P : A \to \mathscr{P}}{\Pi_A P : \mathscr{P}}$$

$$\frac{P, Q : \mathscr{P} \qquad f : P \to Q \qquad g : Q \to P}{\mathrm{propext}(f, g) : \mathrm{Id}_{\mathscr{P}}(P, Q)}$$

Recall that equality reflection implies uniqueness of identity proofs and function extensionality. Moreover, note that propositional extensionality implies that types in $\mathscr{P}$ are in fact propositions.

$\mathscr{V}$ is used to define the sorts of the cwf structure, and $\mathscr{U}$ and $\mathscr{P}$ to give the universe structure. The closure and extensionality properties are crucial for the strictness of the universe structure.

To obtain the groupoid model as presented above we can interpret this type theory in sets by assuming two set-theoretic universes $\mathscr{V}$ and $\mathscr{U}$, encoding functions as in Aczel [1] and setting $\mathscr{P} \coloneqq \{x \in \mathscr{U} \mid \forall y \in \mathscr{V}.\, y \in x \implies y \in 1\}$. Importantly, we can also interpret this type theory in presheaves because the usual lifting of the type structure [39, 40] preserves the properties we identify here. In this way we end up with a groupoid-valued presheaf model of intensional type theory, which we present in the next chapter.

A similar universe of propositions structure (strictly closed under certain function and pair spaces) is required in Licata et al. [56, Figure 2] and can be constructed in Licata et al.'s intended presheaf model by lifting Aczel [1]'s particular encoding of dependent functions.

Chapter 6

A naive groupoid-valued presheaf model

The groupoid model of intensional type theory validates the axiom of countable choice regardless of whether the axiom holds in the metatheory, which can be taken to be extensional type theory. Countable choice can be shown to be independent of higher-order logic [81] using sheaf models, and hence it can be shown to be independent of extensional type theory. Towards extending this independence proof and others to univalent type theory we introduce groupoid-valued presheaf models. However, we will not actually present an extended independence proof for countable choice here but only for the law of excluded middle since presheaves suffice for this [80].

We recalled the groupoid model of Hofmann and Streicher [41] in the previous chapter (albeit in a slightly different formulation) to stress its constructive and elementary nature. The metatheory we employed can be interpreted in presheaf models [39]. For this reason, we intuit the existence of groupoid-valued presheaf models to be obtained by replacing sets by presheaves and families of sets by families of presheaves throughout the construction of the groupoid model.

Recall that a presheaf P over a (small) category $\mathscr{C}$ is given by a family of sets $P(X)$ for each object $X \in \mathrm{Obj}(\mathscr{C})$ together with restriction functions $-|f : P(X) \to P(Y)$ for each morphism $f \in \mathrm{Hom}_{\mathscr{C}}(Y, X)$ satisfying the functoriality laws, that a family Q of presheaves over a presheaf P is given by a family of sets $Q(p)$ for each $p \in P(X)$ together with restriction functions $-|f : Q(p) \to Q(p|f)$ for each $f \in \mathrm{Hom}_{\mathscr{C}}(Y, X)$ functorial in f, and that a section q of such a family Q of presheaves is given by a family of

elements $q(p) \in Q(p)$ for each $p \in P(X)$ natural in X.

A subtle feature of the reformulation of the groupoid model with a primitive notion of dependent paths is that it relies on a universe $\mathscr{P}$ of propositions whose construction in the set model of the metatheory relies on an even more subtle encoding of functions due to Aczel [1]. After showing that any set-theoretic universe of propositions can be lifted to presheaf models of extensional type theory, we can compose the groupoid interpretation of intensional type theory with the presheaf interpretation of extensional type theory to obtain a *naive* groupoid-valued presheaf interpretation of intensional type theory.

Under the resulting interpretation a context Γ denotes a family of groupoids $\Gamma(X)$ and functors $\Gamma(f) : \Gamma(X) \to \Gamma(Y)$, and a type family A over Γ denotes a family of groupoid families $A(X)$ over $\Gamma(X)$ and maps $A(f) : A(X) \to A(Y)$ over $\Gamma(f) : \Gamma(X) \to \Gamma(Y)$ that preserve the lifting structure strictly. Note that this is equivalent to what we obtain by composition of the two interpretations, namely a family of *presheaves* $\Gamma_1(\gamma, \gamma')$ of paths indexed by a *presheaf* Γ_0 of points γ and γ' in the case of contexts Γ—a presheaf of groupoids is the same thing as a groupoid internal to a category of presheaves.

We end this chapter with the application of presheaf models to the law of excluded middle. This leads us to the observation that due to the fact that maps of presheaves are required to preserve the restriction structure strictly, there are maps $\sigma : \Delta \to \Gamma$ between *naive* groupoid-valued presheaves that are levelwise equivalences $\sigma(X) : \Delta(X) \simeq \Gamma(X)$ but that do *not* have an inverse $\Gamma \to \Delta$. In other words, there are pairs of types (even propositions) that look levelwise essentially the same in the sense of groupoid equivalence but that are not essentially the same in the sense of naive groupoid-valued presheaf equivalence. Note that this is in contrast to the fact that set-valued presheaves are identified by an isomorphism once a map identifies them levelwise. We refine the notion of groupoid-valued presheaf in Chapter 7 so that levelwise equivalences between them *are* invertible.

6.1 Lifting of a universe of propositions

Let $\mathscr{U}$ and $\mathscr{V}$ be two fixed set-theoretic universes such that $\mathscr{U} \in \mathscr{V}$, $\mathscr{P}$ the set $\{P \in \mathscr{U} \mid \forall x \in \mathscr{U}.\, x \in P \implies x \in 1\}$ of $\mathscr{U}$-small subsets of the singleton 1, and $\mathscr{C}$ a $\mathscr{U}$-small category. We denote the liftings [40] of the set-theoretic universes $\mathscr{P}$, $\mathscr{U}$, and $\mathscr{V}$ to presheaves over $\mathscr{C}$ by $\widehat{\mathscr{P}}$, $\widehat{\mathscr{U}}$, and $\widehat{\mathscr{V}}$, respectively. Then, $\widehat{\mathscr{P}}$ is a universe of propositions in the sense of Section 5.3

because the type families it classifies, i.e. presheaves $P \in \widehat{\int \Gamma}$ such that $P(\gamma) \in \mathscr{P}$ for all $\gamma \in \Gamma(X)$, do not only satisfy propositional extensionality but are also closed under dependent products $\widehat{\prod}$ and sums $\widehat{\sum}$ of presheaf families:

Lemma 58. *Let $\prod$ and $\sum$ be such that $\mathscr{P}$-small set families are closed under them as in Lemmas 50 and 51, then $\widehat{\mathscr{P}}$-small presheaf families are closed under $\widehat{\prod}$ and $\widehat{\sum}$, i.e. if A is $\widehat{\mathscr{U}}$-small, P is $\widehat{\mathscr{P}}$-small and $Q(p)$ is $\widehat{\mathscr{P}}$-small for all $p : P$ then $\widehat{\prod}_A Q$ and $\widehat{\sum}_P Q$ are $\widehat{\mathscr{P}}$-small.*

Proof. Since $(\widehat{\prod}_A B)(\gamma) \subseteq \prod_{Y:\mathscr{C}, f:Y\to X, a\in A(\gamma|f)} B(\gamma|f, a)$ by the usual construction of dependent products of presheaf families using dependent products of set families, we have that $\widehat{\prod}_A Q$ is $\widehat{\mathscr{P}}$-small by Lemma 50. Similarly, since $(\widehat{\sum}_A B)(\gamma) = \sum_{a\in A(\gamma)} B(\gamma, a)$ by the usual construction of dependent sums of presheaf families using dependent sums of set families, we have that $\widehat{\sum}_P Q$ is $\widehat{\mathscr{P}}$-small by Lemma 51. □

In the following subsections, with the descriptions of the groupoid and presheaf models in mind, we give a description of the groupoid-valued model over $\mathscr{C}$. This can be seen as an unfolding of the composition of the interpretation of intensional type theory in groupoids with the interpretation of extensional type theory in presheaves that interprets the universe of propositions as the presheaf $\widehat{\mathscr{P}}$.

6.2 Basic cwf structure

6.2.1 Contexts and substitutions

A *context* $\Gamma \in \mathrm{Ctx}$ is given by a family of $\mathscr{V}$-small sets $\Gamma(X)$ of *points* for each object $X \in \mathrm{Obj}$ together with a functorial family of *restriction* functions $-|f : \Gamma(X) \to \Gamma(Y)$ for each morphism $f \in \mathrm{Hom}(Y, X)$, a family of $\mathscr{V}$-small sets $\Gamma(\gamma, \gamma')$ of *paths* for each pair of points γ and $\gamma' \in \Gamma(X)$ together with a functorial family of restriction functions $-|f : \Gamma(\gamma, \gamma') \to \Gamma(\gamma|f, \gamma'|f)$ for each $f \in \mathrm{Hom}(Y, X)$, a natural and associative *composition* operation $\mu \cdot \mu' \in \Gamma(\gamma, \gamma'')$ on paths $\mu \in \Gamma(\gamma, \gamma')$ and $\mu' \in \Gamma(\gamma', \gamma'')$, a natural *inverse* operation $\mu^{-1} \in \Gamma(\gamma', \gamma)$ on paths $\mu \in \Gamma(\gamma, \gamma')$, and a natural family of *identity* paths $\mathrm{id}_\gamma \in \Gamma(\gamma, \gamma)$ for each point $\gamma \in \Gamma(X)$.

Note that a context Γ can also be seen as a functorial family of *groupoids* $\Gamma(X)$ and *functors* $\Gamma(f) : \Gamma(X) \to \Gamma(Y)$, i.e. a *contravariant* functor from $\mathscr{C}$ to the category of $\mathscr{V}$-small groupoids.

As before, we will also write $\mu : \gamma \nrightarrow \gamma'$ for $\mu \in \Gamma(\gamma, \gamma')$.

A *substitution* $\sigma \in \mathrm{Sub}(\Delta, \Gamma)$ is given by a pair of natural families of functions $\sigma_0(X) : \Delta(X) \to \Gamma(X)$ on points and $\sigma_1(X) : \Delta(\delta, \delta') \to \Gamma(\sigma_0(\delta), \sigma_0(\delta'))$ on paths preserving the groupoid structure for each $X \in \mathrm{Obj}$.

A substitution $\sigma \in \mathrm{Sub}(\Delta, \Gamma)$ can then also be seen as a *natural transformation* between the contexts Δ and Γ seen as functors.

As before, we will also write $\sigma : \Delta \to \Gamma$ for $\sigma \in \mathrm{Sub}(\Delta, \Gamma)$, and denote $\sigma_0(\delta)$ and $\sigma_1(\nu)$ by $\sigma(\delta)$ and $\sigma(\nu)$, respectively.

We use the composition structure on natural transformations to define a composition structure on substitutions, and the terminal functor as the terminal context [].

6.2.2 Types and terms

A *type* $A \in \mathrm{Ty}(\Gamma)$ over a context Γ is given by a family of $\mathscr{V}$-small sets $A(\gamma)$ of *points* over each point $\gamma \in \Gamma(X)$ together with a functorial family of *restriction* functions $-|f : A(\gamma) \to A(\gamma|f)$ for each morphism $f \in \mathrm{Hom}(Y, X)$, a family of $\mathscr{V}$-small sets $A(\mu, a, a')$ of *paths* over each path $\mu \in \Gamma(\gamma, \gamma')$ for each pair of points $a \in A(\gamma)$ and $a' \in A(\gamma')$ together with a functorial family of restriction functions $-|f : A(\mu, a, a') \to A(\mu|f, a|f, a'|f)$, a natural and associative *composition* operation $\alpha \cdot \alpha' \in A(\mu \cdot \mu', a, a'')$ on paths $\alpha \in A(\mu, a, a')$ and $\alpha' \in A(\mu', a', a'')$, a natural *inverse* operation $\alpha^{-1} \in A(\mu^{-1}, a', a)$ on paths $\alpha \in A(\mu, a, a')$, a natural family of *identity* paths $\mathrm{id}_a \in A(\mathrm{id}_\gamma, a, a)$ for each point $a \in A(\gamma)$, and natural and functorial families of *transports* $\mu^+ a \in A(\gamma')$ and *lifts* $\mu^\rightarrow a \in A(\mu, a, \mu^+ a)$ for each path $\mu \in \Gamma(\gamma, \gamma')$ and point $a \in A(\gamma)$.

Note that a type A over a context Γ can also be seen as a functorial family of split displayed categories $A(X)$ fibred in groupoids over $\Gamma(X)$ seen as groupoids and displayed functors $A(f) : A(X) \to_{\Gamma(f)} A(Y)$ over $\Gamma(f) : \Gamma(X) \to \Gamma(Y)$ such that each $A(f)$ preserves the lifting structure.

We use the reindexing structure on displayed categories and displayed functors along functors to define the functorial *substitution* operation $A\sigma$ on types $A \in \mathrm{Ty}(\Gamma)$ along substitutions $\sigma \in \mathrm{Sub}(\Delta, \Gamma)$. It can be checked that this preserves the lifting structure so that indeed $A\sigma \in \mathrm{Ty}(\Delta)$ is again a type.

As before, we will also write $\alpha : a \nrightarrow_\mu a'$ for paths $\alpha \in A(\mu, a, a')$, and

$\alpha : a \xrightarrow{\sim}_{\gamma} a'$ for vertical paths $\alpha \in A(\mathrm{id}_\gamma, a, a')$.

A type $A \in \mathrm{Ty}(\Gamma)$ will be called *small* if it is $\mathscr{U}$*-small* and *set-like*: $A(\gamma) \in \mathscr{U}$ for all $\gamma \in \Gamma(X)$, $A(\mu, a, a') \in \mathscr{P} \subseteq \mathscr{U}$ for all $\mu : \gamma \xrightarrow{\sim} \gamma'$, $a \in A(\gamma)$, $a' \in A(\gamma')$, and for all $a \in A(\gamma)$ it is the case that $\mu^+ a \in A(\gamma')$ is the unique point $a' \in A(\gamma')$ such that there is a path $\alpha : a \xrightarrow{\sim}_{\mu} a'$, which is necessarily unique. Note that small types are discrete, i.e. we have $a' = a$ and $\alpha = \mathrm{id}_a$ for all vertical paths $\alpha : a \xrightarrow{\sim}_{\gamma} a'$. It follows immediately that the subsets $\mathrm{ty}(\Gamma) \subseteq \mathrm{Ty}(\Gamma)$ of small types over contexts Γ are closed under substitution and we will see that they are also closed under Π, Σ and identity types once we have defined those. Moreover, we will see that small types are classified by a type of ($\mathscr{U}$-small) set-valued presheaves, which is the reason for the adjective "set-like".

A *term* $t \in \mathrm{Tm}(\Gamma, A)$ of a type A over a context Γ is given by a natural family of points $t(\gamma) \in A(\gamma)$ over each point $\gamma \in \Gamma(X)$ and a natural and functorial family of paths $t(\mu) \in A(\mu, t(\gamma), t(\gamma'))$ over each path $\mu \in \gamma \xrightarrow{\sim} \gamma'$. Note that terms are *not* required to be compatible with the lifting structure on A.

The *substitution* operation $t\sigma$ on terms $t \in \mathrm{Tm}(\Gamma, A)$ along substitutions $\sigma : \Delta \to \Gamma$ is defined by $t\sigma(\delta) = t(\sigma(\delta))$ for $\delta \in \Delta(X)$ and $t\sigma(\nu) = t(\sigma(\nu))$ for $\nu : \delta \xrightarrow{\sim} \delta'$. It can be checked that this indeed defines a term $t\sigma \in \mathrm{Tm}(\Delta, A\sigma)$ and that it is functorial.

6.2.3 Comprehension structure

The *extended context* $\Gamma.A$ for $A \in \mathrm{Ty}(\Gamma)$ is defined by $\Gamma.A(X) = \sum_{\gamma \in \Gamma(X)} A(\gamma)$ for $X \in \mathrm{Obj}$ and $\Gamma.A((\gamma, a), (\gamma', a')) = \sum_{\mu \in \Gamma(\gamma, \gamma')} A(\mu, a, a')$ for each γ, $\gamma' \in \Gamma(X)$, $a \in A(\gamma)$, $a' \in A(\gamma')$ together with componentwise restriction and composition structures. It can be checked that this indeed defines a context $\Gamma.A \in \mathrm{Ctx}$.

The *display substitution* $\mathrm{p}_A \in \mathrm{Sub}(\Gamma.A, \Gamma)$, the *variable term* $\mathrm{q}_A \in \mathrm{Tm}(\Gamma.A, A\mathrm{p}_A)$, and the *pairing substitution* $\langle \sigma, t \rangle \in \mathrm{Sub}(\Delta, \Gamma.A)$ for $\sigma : \Delta \to \Gamma$ and $t \in \mathrm{Tm}(\Delta, A\sigma)$ are defined by $\mathrm{p}_A(\gamma, a) = \gamma$ and $\mathrm{p}_A(\mu, \alpha) = \mu$, $\mathrm{q}_A(\gamma, a) = a$ and $\mathrm{q}_A(\mu, \alpha) = \alpha$, and $\langle \sigma, t \rangle(\delta) = (\sigma(\delta), t(\delta))$ and $\langle \sigma, t \rangle(\nu) = (\sigma(\nu), t(\nu))$, respectively. It can be checked that these satisfy the laws of a comprehension structure.

This concludes the description of the basic cwf structure of the groupoid-valued presheaf model. Next, we describe additional type structure.

6.3 Extra type structure

6.3.1 Σ type structure

The operation $\Sigma_A B$ for $A \in \mathrm{Ty}(\Gamma)$ and $B \in Ty(\Gamma.A)$ is given by taking pairs (a, b) of points $a \in A(\gamma)$ and $b \in B(\gamma, a)$ as the points over $\gamma \in \Gamma(X)$, pairs (α, β) of paths $\alpha : a \nrightarrow_{\mu} a'$ and $\beta : b \nrightarrow_{(\mu,\alpha)} b'$ as the paths from (a, b) to (a', b') over $\mu : \gamma \nrightarrow \gamma'$, and defining the presheaf, groupoid and lifting structures componentwise. The substitution law $(\Sigma_A B)\sigma = \Sigma_{A\sigma} B\langle\sigma\rangle$ follows then directly, and it can be checked that this indeed defines a type over the context Γ.

Since the set universes $\mathscr{U}$ and $\mathscr{P}$ are closed under dependent sums, it is the case that $\Sigma_A B$ is small whenever A and B are.

The operations $\mathrm{fst}(p)$ and $\mathrm{snd}(p)$ for $p \in \mathrm{Tm}(\Gamma, \Sigma_A B)$, and (a, b) for $a \in \mathrm{Tm}(\Gamma, A)$ and $b \in \mathrm{Tm}(\Gamma, B[a])$ are defined pointwise by the corresponding operations for dependent sums of set families. The substitution, computation ($\mathrm{fst}(a, b) = a$, $\mathrm{snd}(a, b) = b$) and extensionality ($p = (\mathrm{fst}(p), \mathrm{snd}(p))$) laws follow then directly, and it can be checked that these indeed define terms of type A, $B[\mathrm{fst}(p)]$ and $\Sigma_A B$, respectively.

6.3.2 Π type structure

We give the operation $\Pi_A B$ for $A \in \mathrm{Ty}(\Gamma)$ and $B \in \mathrm{Ty}(\Gamma.A)$. The points s over $\gamma \in \Gamma(X)$ are given by pairs (s_0, s_1) of natural families of points $s_0(f, a) \in B(\gamma|f, a)$ and paths $s_1(f, \alpha) : s_0(f, a) \nrightarrow_{(\mathrm{id}_{\gamma|f},\alpha)} s_0(f, a')$ for $f \in \mathrm{Hom}(Y, X)$, a and $a' \in A(\gamma|f)$, and $\alpha : a \nrightarrow_{\gamma|f} a'$ such that s_1 preserves the groupoid structure:

1. $s_0(f \circ g, a|g) = s_0(f, a)|g$
2. $s_1(f \circ g, \alpha|g) = s_1(f, \alpha)|g$
3. $s_1(f, \mathrm{id}_a) = \mathrm{id}_{s_0(f,a)}$
4. $s_1(f, \alpha \cdot \alpha') = s_1(f, \alpha) \cdot s_1(f, \alpha')$

We will denote $s_0(f, a)$ and $s_1(f, \alpha)$ also by $s(f, a)$ and $s(f, \alpha)$, respectively. The paths ω from s to s' over $\mu : \gamma \nrightarrow \gamma'$ are given by a natural family of paths $\omega(f, \alpha) : s_(f, a) \nrightarrow_{(\mu|f,\alpha)} s'_(f, a')$ for each $\alpha : a \nrightarrow_{\mu|f} a'$ such that the following naturality laws hold:

1. $\omega(f, \alpha_0 \cdot \alpha) = s(f, \alpha_0) \cdot \omega(f, \alpha)$
2. $\omega(f, \alpha \cdot \alpha'_0) = \omega(f, \alpha) \cdot s'(f, a'_0)$

The restriction of points and paths along morphisms $f \in \mathrm{Hom}(Y, X)$ is given by restricting the families to the morphisms $f \circ g$ for $g \in \mathrm{Hom}(Z, Y)$. The groupoid structure is given pointwise using the fact that paths $a \mathrel{\overset{}{\rightarrow}}_{\mu \cdot \mu'} a''$ can always be factored as paths $a \mathrel{\overset{}{\rightarrow}}_{\mu} a'$ and $a' \mathrel{\overset{}{\rightarrow}}_{\mu'} a''$ for some $a' \in A(\gamma')$ and which factorization $\alpha \cdot \alpha'$ is chosen to define $(\omega \cdot \omega')(\alpha \cdot \alpha') = \omega(\alpha) \cdot \omega'(\alpha')$ is irrelevant. The lifting structure is also given pointwise using the fact that paths $\mu : \gamma \mathrel{\overset{}{\rightarrow}} \gamma'$ cannot just be lifted to paths $\mu^{\rightarrow} a : a \mathrel{\overset{}{\rightarrow}}_{\mu} \mu^{+} a$ for $a \in A(\gamma)$ but also to paths $\mu^{\leftarrow} a' : \mu^{-} a' \mathrel{\overset{}{\rightarrow}}_{\mu} a'$ for $a' \in A(\gamma')$. We refer to the construction of the groupoid model for further details. It can be checked that the substitution law $(\Pi_A B)\sigma = \Pi_{A\sigma} B\langle\sigma\rangle$ holds and that this indeed defines a type over the context Γ.

Since the index category $\mathscr{C}$ is assumed to be $\mathscr{U}$-small, the set universe $\mathscr{U}$ is closed under dependent products and sums, and the set universe $\mathscr{P}$ of propositions is closed under dependent products for $\mathscr{P}$-small families indexed by $\mathscr{U}$-small sets, it is the case that $\Pi_A B$ is small whenever A and B are.

The operations $\mathrm{app}(t, u)$ and $\lambda\, b$ for $t \in \mathrm{Tm}(\Gamma, \Pi_A B)$, $u \in \mathrm{Tm}(\Gamma, A)$, and $b \in \mathrm{Tm}(\Gamma.A, B)$ are defined by $\mathrm{app}(t, u)(\gamma) = t(\gamma, \mathrm{id}_X, u(\gamma))$ and $\mathrm{app}(t, u)(\mu) = t(\mu, \mathrm{id}_X, u(\mu))$, and $(\lambda\, b)(\gamma, f, a) = b(\gamma | f, a)$ and $(\lambda\, b)(\mu, f, \alpha) = b(\mu | f, \alpha)$. It can be checked that the substitution, computation and extensionality laws hold and that these indeed define terms of type $B[u]$ and $\Pi_A B$, respectively.

6.3.3 Identity type structure

The operation $\mathrm{Id}_A(t, u)$ for $A \in \mathrm{Ty}(\Gamma)$ and $t, u \in \mathrm{Tm}(\Gamma, A)$ is defined by taking vertical paths $\alpha : t(\gamma) \mathrel{\overset{}{\rightarrow}}_{\gamma} u(\gamma)$ as the points over $\gamma \in \Gamma(X)$, taking $0 \in \mathscr{P}$ as the unique path from α to α' over $\mu : \gamma \mathrel{\overset{}{\rightarrow}} \gamma'$ whenever $t(\mu) \cdot \alpha' = \alpha \cdot u(\mu)$, and defining the restriction of points by the restriction of paths in A. Since the path sets are subsingletons by construction, in order to define the restriction of paths it suffices to observe that $t(\mu | f) \cdot \alpha' | f = \alpha | f \cdot u(\mu | f)$ whenever there is a path $\alpha \mathrel{\overset{}{\rightarrow}}_{\mu} \alpha'$, in order to define the groupoid structure it suffices to observe that $t(\mu \cdot \mu') \cdot \alpha'' = \alpha \cdot u(\mu \cdot \mu')$ whenever there are paths $\alpha \mathrel{\overset{}{\rightarrow}}_{\mu} \alpha'$ and $\alpha' \mathrel{\overset{}{\rightarrow}}_{\mu'} \alpha''$, and in order to define the lifting structure it suffices to observe that there is a unique path $\mu^{+} \alpha$ such that $t(\mu) \cdot \mu^{+} \alpha = \alpha \cdot u(\mu)$ for each $\alpha : t(\gamma) \mathrel{\overset{}{\rightarrow}}_{\gamma} u(\gamma)$. The substitution law $\mathrm{Id}_A(t, u)\sigma = \mathrm{Id}_{A\sigma}(t\sigma, u\sigma)$ and that this indeed defines a type over the context Γ follow then directly.

The operations refl_t for $t \in \mathrm{Tm}(\Gamma, A)$, and $\mathrm{J}(d, p)$ for t and $u \in \mathrm{Tm}(\Gamma, A)$, $C \in \mathrm{Ty}(\Gamma.A.\mathrm{Id}_{A\mathrm{p}}(t\mathrm{p}, \mathrm{q}))$, $d \in \mathrm{Tm}(\Gamma, C[t, \mathrm{refl}_t])$ and $p \in \mathrm{Tm}(\Gamma, \mathrm{Id}_A(t, u))$ are defined by $\mathrm{refl}_t(\gamma) = \mathrm{id}_{t(\gamma)}$, and $\mathrm{J}(d, p)(\gamma) = \nu^{+} d(\gamma)$ and $\mathrm{J}(d, p)(\mu) =$

$\nu^+ d(\mu)$ where $\nu = (\mu, t(\mu) \cdot p(\gamma'), 0) = (\mu, p(\gamma) \cdot u(\mu), 0) : (\gamma, t(\gamma), \mathrm{id}_\gamma) \nrightarrow (\gamma', u(\gamma'), p(\gamma'))$. It can be checked that the substitution and computation laws hold and that these indeed define terms of type $\mathrm{Id}_A(t, t)$ and $C[u, p]$, respectively.

The type $\mathrm{Id}_A(t, u)$ is set-like by construction for any type A and small if A is $\mathcal{U}$-small, in particular whenever A is small.

Using the (Π and) identity type structure we can define the type $\mathrm{isProp}(A)$ expressing that A is proof-irrelevant up to path equality, also called a homotopy proposition. We observe the following sufficient condition for $\mathrm{isProp}(A)$ to be inhabited in the model for some type A:

Lemma 59. *Let $A \in \mathrm{Ty}(\Gamma)$ be codiscrete, i.e. such that $A(\gamma)$ is a propositional groupoid for each $\gamma \in \Gamma(X)$, then A is a homotopy proposition, i.e. we have a section of* $\mathrm{isProp}(A) \in \mathrm{Ty}(\Gamma)$. *The converse holds as well.*

Proof. By assumption we have a path $p(\gamma, a_1, a_2) \in a_1 \nrightarrow_\gamma a_2$ for each $\gamma \in \Gamma(X)$, $a_1, a_2 \in A(\gamma)$ such that $\alpha = p(\gamma, a_1, a_2)$ for all $\alpha : a_1 \nrightarrow_\gamma a_2$. To construct a section of $\mathrm{isProp}(A)$ (using p) it suffices to show that $p(\gamma, a_1, a_2)|f = p(\gamma|f, a_1|f, a_2|f)$, but this we have by assumption.

The converse holds as well because if there is a section of $\mathrm{isProp}(A)$ then there is also a section of $\mathrm{isSet}(A)$ and hence the path sets $A(\mu, a, a')$, which are in bijection with $A(\mathrm{id}_\gamma, a, \mu^- a') = \mathrm{Id}_A(\gamma, a, \mu^- a')$, are singletons. □

Corollary 60. *Let $A \in \mathrm{Ty}(\Gamma)$ be levelwise a proposition, i.e. such that each $A(X) \in \mathrm{Ty}(\Gamma(X))$ is a proposition, then A is a homotopy proposition, i.e. we have a section of* $\mathrm{isProp}(A) \in \mathrm{Ty}(\Gamma)$. *The converse holds as well.*

Proof. The assumption implies that A is pointwise a proposition using Lemma 54 so that we can conclude by Lemma 59. Similarly for the converse. □

6.3.4 Booleans and natural numbers type structures

The set-like constant types $\mathrm{Bool} \in \mathrm{Ty}([])$ and $\mathrm{Nat} \in \mathrm{Ty}([])$ are given by taking the sets $2 = \{0, 1\}$ of Booleans and $\mathbb{N}$ of natural numbers, respectively, as the set of points over $x \in [](X)$.

We have $\mathrm{Bool} \in \mathrm{ty}([])$ and $\mathrm{Nat} \in \mathrm{ty}([])$ because they are set-like by construction and $\mathcal{U}$-small because the sets 2 and $\mathbb{N}$ are.

The operations true and $\mathrm{false} \in \mathrm{Tm}([], \mathrm{Bool})$ as well as $\mathrm{zero} \in \mathrm{Tm}([], \mathrm{Nat})$ and $\mathrm{succ}(n) \in \mathrm{Tm}(\Gamma, \mathrm{Nat}\langle\rangle)$ for $n \in \mathrm{Tm}(\Gamma, \mathrm{Nat}\langle\rangle)$ are given levelwise by the corresponding set-theoretic operations. The operations $\mathrm{elim}(c_0, c_1, b) \in$

$\mathrm{Tm}(\Gamma, C[b])$ and $\mathrm{elim}(d_0, d_1, n) \in \mathrm{Tm}(\Gamma, D[n])$ are defined levelwise by induction on $b(\gamma) \in \mathrm{Bool}(X) = 2$ and $n(\gamma) \in \mathrm{Nat}(X) = \mathbb{N}$ for $\gamma \in \Gamma(X)$, respectively. The equations for stability under substitution and computation hold by definition.

6.3.5 Propositional truncation type structure

The operation $\|A\|$ on types $A \in \mathrm{Ty}(\Gamma)$ is given by taking the same points as A, taking $0 \in \mathscr{P}$ as the unique path from any a to any a' over any $\mu : \gamma \nrightarrow \gamma'$, and defining the restriction and transport operations on points as in A. Since the path sets are singletons by construction, the restriction, groupoid and lifting operations are trivial. The substitution law $\|A\|\sigma = \|A\sigma\|$ and this indeed defines a type follow then directly.

Since $\|A\|$ is codiscrete by construction for any type A and inhabitation of $\mathrm{isProp}(C)$ implies that C is codiscrete by Corollary 60 for any type C, in order to define the operations $\mathrm{inc}(a) \in \mathrm{Tm}(\Gamma, \|A\|)$ for $a \in \mathrm{Tm}(\Gamma, A)$ and $\mathrm{rec}(c, p, t) \in \mathrm{Tm}(\Gamma, C)$ for $c \in \mathrm{Tm}(\Gamma.A, C\mathrm{p})$, $p \in \mathrm{Tm}(\Gamma, \mathrm{isProp}(C))$, and $t \in \mathrm{Tm}(\Gamma, \|A\|)$ it suffices to give and check their actions $\mathrm{inc}(a)(\gamma) = a(\gamma)$ and $\mathrm{rec}(c, p, t)(\gamma) = c(\gamma, t(\gamma))$ on points, and the definition of the operation $\mathrm{squash}(t, u) \in \mathrm{Tm}(\Gamma, \mathrm{Id}_{\|A\|}(t, u))$ for $t, u \in \|A\|$ is trivial because all path sets are uniquely inhabited, in particular all path equations hold. The computation law for rec on the constructor inc follows directly.

Like in the groupoid model, type families $B \in \mathrm{Ty}(\Gamma.A)$ indexed by *discrete* types $A \in \mathrm{Ty}(\Gamma)$ satisfy the axiom of choice in the sense that the function types $\Pi_A \|B\| \to \left\|\Pi_A B\right\|$ are inhabited in the model for discrete A. In particular, the axiom of countable choice is validated by the model regardless of whether the axiom holds in its metatheory.

Note that, while $\|A\|$ has $\mathscr{P}$-small path sets by construction for any type A and is $\mathscr{U}$-small whenever A is, $\|A\|$ is in general *not* small regardless of whether A is because it will usually not be discrete. In particular, the universe type structure which we describe next is not closed under propositional truncation. For a similar reason the universe type structure could be shown not to be closed under the suspension used in [69, Theorem 10.1.14] to prove Diaconescu's theorem [24] that choice implies the principle of excluded middle, and in fact could not possibly be as we will see with the application of this model construction to the principle of excluded middle at the end of this chapter.

6.3.6 Universe type structure

We give a universe structure (Definition 26) that classifies small types (see Subsection 6.2.2).

The operation $\mathrm{U}(\Gamma)$ for $\Gamma \in \mathrm{Ctx}$ is defined by taking $\mathscr{U}$-small presheaves P over X, i.e. pairs of families of $\mathscr{U}$-small sets $P(f)$ for $Y \in \mathrm{Obj}$ and $f \in \mathrm{Hom}(Y, X)$ and functorial families of functions $-|g : P(f) \to P(f \circ g)$ for $g \in \mathrm{Hom}(Z, Y)$, as the points over $\gamma \in \Gamma(X)$ and natural isomorphisms $\iota : P \to Q$, i.e. natural families of bijections $\iota(f) : P(f) \to Q(f)$ for $f \in \mathrm{Hom}(Y, X)$, as the paths from P to Q over $\mu : \gamma \xrightarrow{\sim} \gamma'$. The restriction of points and paths along $f \in \mathrm{Hom}(Y, X)$ is given by restricting the underlying families to the morphisms $f \circ g$ for $g \in \mathrm{Hom}(Z, Y)$. The groupoid structure is given by the composition structure on natural transformations and the lifting structure is given by the identity. The substitution law $\mathrm{U}(\Gamma)\sigma = \mathrm{U}(\Delta)$ and that this indeed defines a type over the context Γ follow then directly.

The operation $\mathrm{El}(a)$ for $a \in \mathrm{Tm}(\Gamma, \mathrm{U})$ is defined by taking the evaluation of the presheaf $a(\gamma)$ at $\mathrm{id}_X \in \mathrm{Hom}(X, X)$ as the set of points over $\gamma \in \Gamma$ and taking $\{0 \mid q = a(\mu, p)\}$ as the set of paths from $p \in \mathrm{El}(a)(\gamma) = a(\gamma, \mathrm{id}_X)$ to $q \in \mathrm{El}(a)(\gamma') = a(\gamma', \mathrm{id}_X)$ over $\mu : \gamma \xrightarrow{\sim} \gamma'$. The restriction of points along $f \in \mathrm{Hom}(Y, X)$ is given by the functions $-|f : a(\gamma, \mathrm{id}_X) \to a(\gamma, f) = (a(\gamma)|f)(\mathrm{id}_Y) = a(\gamma|f, \mathrm{id}_Y)$. For the restriction of paths it suffices to check that $q|f = a(\mu|f, p|f)$ whenever $q = a(\mu, p)$, for the groupoid structure it suffices to check that $p = a(\mathrm{id}_\gamma, p)$ and $r = a(\mu \cdot \mu')(p)$ whenever $q = a(\mu, p)$ and $r = a(\mu', q)$ for some $q \in a(\gamma', \mathrm{id}_X)$, and for the lifting structure it suffices to observe that $a(\mu, p)$ is the unique $q \in a(\gamma', \mathrm{id}_X)$ such that $p \xrightarrow{\sim}_\mu q$. Note that this indeed defines a small type over the context Γ. The substitution law $\mathrm{El}(a\sigma) = \mathrm{El}(a)\sigma$ follows directly.

The operation $\mathrm{In}(A)$ for small types $A \in \mathrm{ty}(\Gamma)$ is defined by $\mathrm{In}(A)(\gamma, f) = A(\gamma|f)$ and $\mathrm{In}(A)(\mu, f) : a \mapsto (\mu|f)^+ a$. It can be checked that this indeed defines a term of type U and in fact the unique term $a \in \mathrm{Tm}(\Gamma, \mathrm{U})$ such that $\mathrm{El}(a) = \mathrm{In}(A)$. In particular, the substitution law $\mathrm{In}(A\sigma) = \mathrm{In}(A)\sigma$ holds since it holds for El.

In summary, U (together with El) classifies small types and hence defines a universe type structure by Proposition 37.

This concludes the description of the cwf structure of the naive groupoid-valued presheaf model:

Theorem 61. *Groupoid-valued presheaves form a cwf with unit, Σ, Π, identity, univalent universe, and propositional truncation type structures.*

6.4 Examples of presheaf models: The law of excluded middle

Hofmann and Streicher [41] devised the groupoid model to refute the uniqueness of identity proofs. Another logical principle one might consider is the law of excluded middle (LEM). While the groupoid model validates this principle whenever its metatheory does, there are index categories such that the (groupoid-valued) presheaf models over them refute LEM.

Before we give a concrete countermodel, let us recall that in type theory the principle can be formulated as follows. [69, Equation 3.4.1]

$$LEM :\equiv \prod_{P:\mathrm{HProp}} P + \neg P$$

Note that the quantification over mere propositions is essential because, like in any model with a univalent universe U [69, Corollary 3.2.7], the type $\prod_{A:\mathrm{U}} A + \neg A$ is empty in the groupoid model. The issue vanishes when we consider the type $\prod_{A:\mathrm{U}} \|A + \neg A\|$ instead, which is indeed equivalent to LEM.

Now, to show that LEM is independent of type theory with a univalent universe, we can consider the groupoid-valued presheaf model over the index category

$$\mathscr{C}: \quad 0 \longleftarrow 1$$

Over the preorder $\mathscr{C}$ there is even the closed proposition

$$P: \quad \mathrm{Empty} \longrightarrow \mathrm{Unit}$$

such that the proposition $P + \neg P$ is not inhabited simply because the groupoids $P(0) = \mathrm{Empty}$ and $\neg P(0) \simeq \mathrm{Empty}$ have no points at all, and since $\mathrm{isProp}(P)$ we have $\neg LEM$ in the groupoid-valued presheaf model over $\mathscr{C}$.

For another example, consider the index category

$$\mathscr{D}: \quad \bullet \circlearrowright e$$

with $e \neq \mathrm{id}$ and $e \circ e = e$. Over the monoid $\mathscr{D}$ we have $P + \neg P$ for *closed* and *strict* propositions P if our metatheory satisfies the law of excluded middle because then the groupoid $P(\bullet)$ is either empty or has a point which is necessarily fixed by the monoid action. In fact, we have $A + \neg A$ for all *closed* types A over $\mathscr{D}$ if we can decide whether arbitrary groupoids $A(\bullet)$ have a point a because then the point $a|e$ is fixed by idempotency of e. However, we

also have $\neg LEM$ in the groupoid-valued presheaf model over $\mathscr{D}$ because the proposition

$$Q : \quad \text{Empty} \longrightarrow \text{Unit} \circlearrowright$$

in the representable context Y, i.e. $Y(\bullet) = \mathrm{Hom}(\bullet, \bullet)$ with composition as the action, has no section because the groupoids $Q(\mathrm{id}) = \text{Empty}$ and $\neg Q(\mathrm{id}) \simeq \text{Empty}$ have no points.

These two counterexamples already exist in the set-valued presheaf models. As a counterexample that is a closed proposition in a situation where all set-valued propositions satisfy LEM, consider the groupoid-valued presheaf

$$R : \quad 0 \xrightarrow{\sim} 1$$

given by the interval on which the group

$$\mathscr{E} : \quad \bullet \circlearrowright i$$

with $i \neq \mathrm{id}$ and $i \circ i = \mathrm{id}$ acts by reversal. The presheaf R can also be described as the propositional truncation $\|Y\|$ of the representable presheaf Y over $\mathscr{E}$. To summarize this last counterexample, there is an index category $\mathscr{E}$ such that the groupoid-valued presheaf model over $\mathscr{E}$ refutes LEM while the corresponding set-valued model satisfies LEM if we have it in the metatheory. Another perspective on this counterexample is that in the groupoid-valued presheaf models we described in this chapter there might be more propositions (even up to equivalence) than in the corresponding set-valued models. The failure of homotopy propositions being equivalent to strict propositions in general is an instance of the failure of levelwise equivalences being invertible in general. In the next chapter we will consider a refined notion of type for groupoid-valued presheaves so that two types are equivalent if and only if they are levelwise equivalent.

Chapter 7

A refined groupoid-valued presheaf model

The goal of this chapter is to apply the theory of descent data operations from Part I to the problem of constructing a model of type theory that combines the two generalizations 1. from constant sets to variable sets, or presheaves, due to Hofmann, and 2. from sets to groupoids due to Hofmann and Streicher. Such a combination can be seen either as a generalization from variable *sets* to variable *groupoids*, or as a generalization from *constant* groupoids to *variable* groupoids. Variable groupoids occur in practice and once we show that they form a model we can use the language of type theory to carry out constructions and reason about them. The resulting model will support a *univalent* universe (of variable sets) and propositional truncation as well as refute classical principles like the law of excluded middle.

In the previous chapter (Chapter 6) we made a first attempt at constructing a model in which types are interpreted by variable groupoids. That attempt succeeded in constructing a model that supports the desired type structure and refutes the law of excluded middle. However, we also noted various related shortcomings that suggest that the attempt did not succeed in constructing a model that represents variable groupoids as they occur in practice. More specifically, the attempt resulted in a model (the *naive* groupoid-valued presheaf model) with types C that are levelwise contractible but do not satisfy $\mathrm{isContr}(C)$, maps $s : A \to B$ that are levelwise surjective but do not satisfy $\mathrm{isSurj}(s)$, as well as maps $e : A \to B$ that are levelwise invertible but do not satisfy $\mathrm{isEquiv}(e)$.

In this chapter we refine the naive groupoid-valued presheaf model from

the previous chapter in order to remedy its shortcomings. The refinement will force properties (like being contractible, surjective, or invertible) to be true whenever they are levelwise true in a way that preserves the desired type structure. We carry out this refinement by applying the theory of descent data operations from Part I. First, in Propositions 62 and 65 we construct a descent data operation $\boldsymbol{D}$ (see Definition 33 in Chapter 4) on the naive groupoid-valued presheaf model. Then, in Theorem 67 we apply Proposition 43 from Chapter 4 to obtain a new model of type theory with one univalent universe (the *refined* groupoid-valued presheaf model) in which types are interpreted by groupoid-valued presheaves that are modal with respect to the descent data operation $\boldsymbol{D}$ (see Definition 34 in Chapter 4). Lastly, in Corollaries 68 to 70 we show that types and maps in the refined groupoid-valued presheaf model are contractible, surjective, or invertible if and only if they are levelwise contractible, surjective, or invertible, respectively.

The descent data operation $\boldsymbol{D}$ we use to refine the naive groupoid-valued presheaf model arises as follows. The failure of contractibility, surjectivity, and invertibility to be true globally in general once they are true levelwise can be traced back to the existence of "virtual" elements for which no "actual" element exists. An actual element a of a groupoid-valued presheaf A in context Γ is given by a family of elements $a(\gamma) : A(\gamma)$ for each $\gamma : \Gamma(X)$ (as well as paths $a(\mu) : a(\gamma) \nrightarrow_{\mu} a(\gamma')$ over each $\mu : \gamma \nrightarrow \gamma'$, which we ignore for the moment) such that the naturality conditions $a(\gamma)|f = a(\gamma|f)$ for each $f : Y \to X$ hold *strictly*. On the other hand, for a virtual element b of A given by a family of elements $b(\gamma) : A(\gamma)$ the naturality conditions hold only *up to coherent paths* in the sense that a virtual element b comes equipped with *paths* $b(\gamma, f) : b(\gamma)|f \nrightarrow b(\gamma|f)$ such that $b(\gamma, f)|g \cdot b(\gamma|f, g) = b(\gamma, fg)$. If a groupoid-valued presheaf $\boldsymbol{C}$ is levelwise contractible then it is "inhabited" by a virtual element, if a map $s : A \to B$ of groupoid-valued is levelwise surjective then its fibres are merely inhabited virtual elements, and if a map $e : A \to B$ is levelwise invertible then its fibres are inhabited virtual elements. Therefore, when groupoid-valued presheaves are closed under virtual elements $\boldsymbol{C}$ is globally inhabited, the fibres of s are globally merely inhabited, and the fibres of e are globally inhabited. Now, the virtual elements of a groupoid-valued presheaf A form a groupoid-valued presheaf $\boldsymbol{D}(A)$ with a canonical map $\eta_A : A \to \boldsymbol{D}(A)$ that embeds actual elements as virtual elements. So, it is the groupoid-valued presheaves $\boldsymbol{D}(\boldsymbol{C})$, $\boldsymbol{D}(\|\mathrm{fib}(s, b)\|)$, and $\boldsymbol{D}(\mathrm{fib}(e, b))$ that become inhabited for a levelwise contractible $\boldsymbol{C}$, a levelwise surjective s, and a levelwise invertible e, respectively. A groupoid-valued

presheaf A is closed under virtual elements if this map η_A is invertible by a map $patch_A : D(A) \to A$ that produces for each virtual element b an actual element a such that $\eta_A(a) \simeq b$. In particular, if C, A, and B are closed under virtual elements then C is actually contractible, s is actually surjective, and e is actually invertible. This is why in the refined groupoid-valued presheaf model, where all types are modal, i.e. closed under virtual elements, levelwise properties hold globally.

7.1 Groupoid-valued cobar operation

In this section we explicitly construct the pseudoendomorphism D on the cwf of groupoid-valued presheaves from Chapter 6, and show that the lex operation induced by D is a descent data operation, in particular a lex modality. As a result, we will obtain a new cwf of groupoid-valued presheaves (the *refined* model) in Theorem 67 below. In the next section we will also show that the refined model supports Booleans, natural numbers, and propositional truncation type structures, and that in the refined model contractibility, surjectivity, and invertibility are levelwise properties.

From now on we will refer to (families of) groupoid-valued presheaves as defined in Chapter 6 simply as (families of) presheaves.

Given a presheaf Γ, define the presheaf $D(\Gamma)$ by taking as points $d \in D(\Gamma)(X)$ families of points $d(f) \in \Gamma(Y)$ and paths $d(f, f') : d(f)|f' \simeq d(ff')$ indexed by morphism $f : Y \to X$, $f' : Z \to Y$ such that $d(f, f')|f'' \cdot d(ff', f'') = d(f, f'f'')$, and as paths $\omega : d \simeq d'$ families of paths $\omega(f) : d(f) \simeq d'(f)$ indexed by $f : Y \to X$ such that $\omega(f)|f' \cdot d'(f, f') = d(f, f') \cdot \omega(ff')$. The restriction of d and $\omega : d \simeq d'$ along $f : Y \to X$ from $D(\Gamma)(X)$ to $D(\Gamma)(Y)$ is $((d(f \circ g))_g, (d(f \circ g, g'))_{g,g'})$ and $(\omega(f \circ g))_g$, respectively, which can be checked to be functorial in f. The groupoid structure on each $D(\Gamma)(X)$ is defined componentwise and it can be checked that the restriction maps preserve it.

Next, define the natural transformation $\eta_\Gamma : \Gamma \to D(\Gamma)$ by mapping points γ and paths $\mu : \gamma \simeq \gamma'$ to $((\gamma|f)_f, (\mathrm{id}_{\gamma|ff'})_{f,f'})$ and $(\mu|f)_f$, respectively.

And, define the functors $\epsilon_\Gamma(X) : D(\Gamma)(X) \to \Gamma(X)$ by mapping points d and paths $\omega : d \simeq d'$ to $d(\mathrm{id}_X)$ and $\omega(\mathrm{id}_X)$, respectively. We have $\mathrm{id}_\gamma : \gamma \simeq \epsilon_\Gamma(\eta_\Gamma(\gamma))$ natural in γ and $(d(\mathrm{id}_X, f))_f : \eta_\Gamma(\epsilon_\Gamma(d)) \simeq d$ natural in d so that $\eta_\Gamma : \Gamma \to D(\Gamma)$ is levelwise an equivalence with levelwise (homotopy) inverses $\epsilon_\Gamma(X)$.

Given a natural transformation $\sigma : \Delta \to \Gamma$, define the natural transformation $D(\sigma) : D(\Delta) \to D(\Gamma)$ by mapping points d and paths $\omega : d \nrightarrow d'$ to $((\sigma(d(f)))_f, (\sigma(d(f, f')))_{f,f'})$ and $(\sigma(\omega(f)))_f$, respectively. The mapping $\sigma \mapsto D(\sigma)$ is functorial so that we have a functor $D : \mathrm{Ctx} \to \mathrm{Ctx}$ on the category of contexts of the groupoid-valued presheaf model.

The family of natural transformations $\eta_\Gamma : \Gamma \to D(\Gamma)$ is itself natural in Γ, i.e. $D(\sigma) \circ \eta_\Delta = \eta_\Gamma \circ \sigma$, because natural transformations $\sigma : \Delta \to \Gamma$ preserve identity paths and commute with restriction; in other words, we have a natural transformation $\eta : \mathrm{Id} \to D$.

Given a family $A \in \mathrm{Ty}(\Gamma)$ of groupoid-valued presheaves, define the family $\widetilde{D}(A) \in \mathrm{Ty}(D(\Gamma))$ whose points $e \in \widetilde{D}(A)(d)$ are given by a family of points $e(f) \in A(d(f))$ and a family of paths $e(f, f') : e(f)|f' \nrightarrow_{d(f,f')} e(ff')$ indexed by morphisms $f : Y \to X$, $f' : Z \to Y$, and whose paths $\psi : e \nrightarrow_\omega e'$ are given by a family of paths $\psi(f) : e(f) \nrightarrow_{\omega(f)} e'(f)$ such that $\psi(f)|f' \cdot e'(f, f') = e(f, f') \cdot \psi(ff')$. The restriction along $f : Y \to X$ of $\psi : e \nrightarrow_\omega e'$ to $(\psi(f \circ g))_g : ((e(f \circ g))_g, (e(f \circ g, g'))_{g,g'}) \nrightarrow_{\omega|f} ((e'(f \circ g))_g, (e'(f \circ g, g'))_{g,g'})$ can be checked to be functorial in f. The groupoid and lifting structures are defined componentwise and it can be checked that they are preserved by restriction. It can be checked that we have $\widetilde{D}(A\sigma) = \widetilde{D}(A)D(\sigma)$. Moreover, it can be checked that we have $\widetilde{D}(A) \in \mathrm{ty}(D(\Gamma))$ whenever $A \in \mathrm{ty}(\Gamma)$, i.e. $\widetilde{D}$ preserves (both discreteness and) set-likeness.

Given a section $a \in \mathrm{Tm}(\Gamma, A)$ of $A \in \mathrm{Ty}(\Gamma)$, define the section $\widetilde{D}(a) \in \mathrm{Tm}(D(\Gamma), \widetilde{D}(A))$ which maps points $d \in D(\Gamma)$ to $((a(d(f)))_f, (a(d(f, f')))_{f,f'})$ and paths $\omega : d \nrightarrow d'$ to $(a(\omega(f)))_f$. It can be checked that we have $\widetilde{D}(a\sigma) = \widetilde{D}(a)D(\sigma)$.

Proposition 62. *$(D, \widetilde{D})$ is a (pointed) pseudomorphism from the cwf of groupoid-valued presheaves to itself, and hence induces a lex operation structure by Corollary 42 from Chapter 4.*

Proof. We have already seen that $D : \mathrm{Ctx} \to \mathrm{Ctx}$ is a functor and that $\widetilde{D}(\Gamma) : \mathrm{Ty}(\Gamma) \to \mathrm{Ty}(D(\Gamma))$ and $\widetilde{D}(\Gamma, A) : \mathrm{Tm}(\Gamma, A) \to \mathrm{Tm}(D(\Gamma), \widetilde{D}(A))$ commute with substitution.

The unique natural transformation $D([]) \to []$ is an isomorphism because we have $d(f) = 0$, $d(f, f') = 0$ and $\omega(f) = 0$ for all $d \in D([])(X)$, $\omega : d \nrightarrow d'$, $f : Y \to X$, $f' : Z \to Y$.

For a family $A \in \mathrm{Ty}(\Gamma)$, the canonical substitution $s_A = \langle D(\mathrm{p}_A), \widetilde{D}(\mathrm{q}_A)\rangle : D(\Gamma.A) \to D(\Gamma).\widetilde{D}(A)$ is an isomorphism whose inverse at level X is giv-

en by $s_A^{-1}(d,e)(f) = (d(f), e(f))$, $s_A^{-1}(d,e)(f,f') = (d(f,f'), e(f,f'))$ and $s_A^{-1}(\omega,\psi)(f) = (\omega(f), \psi(f))$. □

We denote the actions $A \mapsto \widetilde{D}(A)\eta_\Gamma$ on types $\mathrm{Ty}(\Gamma)$ and $m \mapsto \widetilde{D}(m)s_A\langle\eta_\Gamma\rangle$ from maps $\mathrm{Tm}(\Gamma.A, B\mathrm{p})$ to maps $\mathrm{Tm}(\Gamma.D(A), D(B)\mathrm{p})$ of the induced lex operation structure also by D, like the corresponding actions on contexts and substitutions.

Lemma 63. *If $A \in \mathrm{Ty}(\Gamma)$ is levelwise contractible, then $\widetilde{D}(A) \in \mathrm{Ty}(D(\Gamma))$ and hence $D(A) \in \mathrm{Ty}(\Gamma)$ are contractible.*

Proof. Assume $A \in \mathrm{Ty}(\Gamma)$ levelwise contractible, i.e. $A(X) \in \mathrm{Ty}(\Gamma(X))$ contractible for each X. Then, A is in particular levelwise a mere proposition and in fact a mere proposition by Corollary 60. Since D is a lex operation by Proposition 62 and lex operations preserve families of mere propositions by Corollary 19, also $\widetilde{D}(A)$ is a mere proposition. Now for $\widetilde{D}(A)$ to be contractible it remains to show that it is inhabited. Since each $A(X)$ is contractible, we have a family of points $a(\gamma) \in A(\gamma)$ for each $\gamma \in \Gamma(X)$, but not necessarily $a(\gamma|f) = a(\gamma)|f$. We define $a_0 \in \mathrm{Tm}(D(\Gamma), \widetilde{D}(A))$ by setting $a_0(d)(f) = a(d(f))$, letting $a_0(d)(f,f')$ be the unique path $a_0(d)(f)|f' \simeq a_0(d)(f\,f')$ and letting $a_0(\omega)(f)$ be the unique path $a_0(d)(f) \simeq_\omega a_0(d')(f)$. $D(A) = \widetilde{D}(A)\eta_\Gamma$ is contractible because being contractible is preserved by substitution. □

Lemma 64. *Let Γ and Δ be groupoid-valued presheaves and $\sigma : \Delta \to \Gamma$ a natural transformation that is levelwise an equivalence, i.e. $\sigma(X) : \Delta(X) \to \Gamma(X)$ is an equivalence for each level X, then the natural transformation $D(\sigma) : D(\Delta) \to D(\Gamma)$ is an equivalence.*

Proof. Assume $\sigma : \Delta \to \Gamma$ levelwise an equivalence, i.e. $\mathrm{fib}(\sigma(X)) \in \mathrm{Ty}(\Gamma(X))$ contractible for each X. Then, $\mathrm{fib}(\sigma) \in \mathrm{Ty}(\Gamma)$ is levelwise contractible because evaluation at X commutes with dependent sum and identity types and application, i.e. $\mathrm{fib}(\sigma)(X) = \mathrm{fib}(\sigma(X))$ for all X. Thus, $\widetilde{D}(\mathrm{fib}(\sigma)) \in \mathrm{Ty}(D(\Gamma))$ is contractible by Lemma 63. Since D is a lex operation by Proposition 62 and lex operations preserve terminal objects and homotopy pullbacks by Propositions 5 and 7, $\widetilde{D}(\mathrm{fib}(\sigma))$ is equivalent to $\mathrm{fib}(D(\sigma))$. It follows that also $\mathrm{fib}(D(\sigma)) \in \mathrm{Ty}(D(\Gamma))$ is contractible and hence that $D(\sigma) : D(\Delta) \to D(\Gamma)$ is an equivalence. □

Proposition 65. *The lex operation induced by the (pointed) pseudoendomorphism $(D, \widetilde{D})$ is a descent data operation.*

Proof. $D(\eta_\Gamma) : D(\Gamma) \to D(D(\Gamma))$ is an equivalence by Lemma 64 because each $\eta_\Gamma(X) : \Gamma(X) \to D(\Gamma)(X)$ gets inverted by $\epsilon_\Gamma(X)$. Thus, $\eta_{D(\Gamma)} : D(\Gamma) \to D(D(\Gamma))$ is also an equivalence because $h(d)(f)(g) = d(f,g) : d(f)|g \nrightarrow d(fg)$ defines a homotopy $h : D(\eta_\Gamma) \sim \eta_{D(\Gamma)}$. □

Now that we have constructed a descent data operation D we can apply Proposition 43 from Chapter 4 to obtain a new model of type theory. And we will do that in Theorem 67 below. The resulting *refined* groupoid-valued presheaf model supports unit, Σ, Π, identity, and univalent universe types because the naive model (the model on which the descent data operation D is defined) supports them. We construct Booleans, natural numbers, and propositional truncation type structures directly.

Let $A \in \mathrm{Ty}(\Gamma)$ be a family of presheaves. An element of $\mathrm{Tm}(\Gamma, \mathrm{isPA}(A))$ is then given by a map $patch_A : D(A) \to A$ together with a homotopy $linv_A : \prod_{a:A} \mathrm{Id}(patch_A(\eta_A(a)), a)$ witnessing that $patch_A$ is a left inverse (and in fact an inverse) of the embedding $\eta_A : A \to D(A)$. We call such an element a patching structure on A and the triple $(A, patch_A, linv_A)$ a patch algebra. Recall that having a patch structure is equivalent to A being D-modal, i.e. there being an element of $\mathrm{Tm}(\Gamma, \mathrm{isModal}_D(A))$.

7.2 Inductive type structures

Let I be a (higher) inductive type (like a coproduct $A + B$, the natural numbers Nat, or a propositional truncation $\|A\|$) in the naive presheaf model. The type $D(I)$ can be shown to support the introduction and elimination principles of I with respect to D-modal types A using the unit $I \to D(I)$ and the equivalence $\prod_{d:D(I)} A(x) \to \prod_{i:I} A(\eta(i))$. However, the elimination principle for $D(I)$ satisfies the computation rules of I not strictly but only up to homotopy.

7.2.1 Booleans and natural numbers type structures

In this subsection we show that the refined model of presheaves with patching structure supports Booleans and natural numbers type structures. Recall from Subsection 6.3.4 in Chapter 6 that the Booleans and natural numbers type structures on the naive model of arbitrary presheaves are given by the constant presheaves Bool and Nat for the sets $2 = \{0, 1\}$ and $\mathbb{N}$, respectively. Constant presheaves like these and, more generally, discrete presheaves, i.e.

presheaves $A \in \mathrm{Ty}(\Gamma)$ such that $a' = a$ and $\alpha = \mathrm{id}_a$ for all $\gamma \in \Gamma(X)$ and $\alpha : a \xrightarrow{\sim}_{\mathrm{id}_\gamma} a'$, have a patching structure:

Proposition 66. *Let $A \in \mathrm{Ty}(\Gamma)$ be a* discrete *presheaf, i.e. a presheaf such that $a' = a$ and $\alpha = \mathrm{id}_a$ for all $\alpha \in A(\mathrm{id}_\gamma, a, a')$, then A has a patching structure.*

Proof. For $\gamma \in \Gamma(X)$ and $e \in D(A)(\gamma)$ we have $e(f) = \mathrm{id}^+_{\gamma|f}(e(\mathrm{id})|f) = e(\mathrm{id})|f$ for all $f : Y \to X$ and hence $e = \eta_A(e(\mathrm{id}))$. □

7.2.2 Propositional truncation type structure

Let $A \in \mathrm{Ty}(\Gamma)$ be a family of groupoid-valued presheaves (not necessarily a patch algebra). Define a new codiscrete family $\|A\| \in \mathrm{Ty}(\Gamma)$ whose points over $\gamma \in \Gamma(X)$ are either $\mathrm{inc}\, a$ for $a \in A(\gamma)$ or $\mathrm{patch}\, \phi$ for ϕ an *arbitrary* family of points $\phi(f) \in A(\gamma|f)$ indexed by $f \in Y \to X$ (no naturality conditions, neither strict nor pseudo). The restriction of points $\mathrm{inc}\, a$ is as in A, and the restriction of points $\mathrm{patch}\, \phi$ is given by reindexing. Defining $\|A\|$ as a codiscrete family means that there is a unique path $\tau_{t,t'}$ between any pair of points $t \in \|A\|(\gamma)$ and $t' \in \|A\|(\gamma')$ over any path $\mu : \gamma \xrightarrow{\sim} \gamma'$ in $\Gamma(X)$ by setting $\|A\|(\mu, u, u') := \{0\}$. The codiscreteness of $\|A\|$ also means that the composition and restriction structures on paths is uniquely determined.

We define the term $\mathrm{inc}(a) \in \mathrm{Tm}(\Gamma, \|A\|)$ for $a \in \mathrm{Tm}(\Gamma, A)$ by $\mathrm{inc}(a)(\gamma) = \mathrm{inc}\, a(\gamma)$. The action of $\mathrm{inc}(a)$ on paths $\mu : \gamma \xrightarrow{\sim} \gamma'$ is again uniquely determined, and so is the term $\mathrm{squash}(t, u) \in \mathrm{Tm}(\Gamma, \mathrm{Id}(t, u))$ for $t, u \in \mathrm{Tm}(\Gamma, \|A\|)$.

Let $C \in \mathrm{Ty}(\Gamma)$ be a homotopy proposition and $patch_C \in \mathrm{Tm}(\Gamma.D(C), C\mathrm{p})$ a patching function for C (the proof that $patch_C$ inverts $\eta_C \in \mathrm{Tm}(\Gamma.C, D(C)\mathrm{p})$ is uniquely determined and will not be needed), and a map $i \in \mathrm{Tm}(\Gamma.A, C)$. We define $\mathrm{elim}(i, t) \in \mathrm{Tm}(\Gamma, C)$ for $t \in \mathrm{Tm}(\Gamma, \|A\|)$ by $\mathrm{elim}(i, t)(\gamma) = i(\gamma, a)$ if $t(\gamma) = \mathrm{inc}\, a)$ and $\mathrm{elim}(i, t)(\gamma) = patch_C(\gamma, d)$ if $t(\gamma) = \mathrm{patch}\, \phi$ where the descent datum d is given by $d(f) = i(\gamma|f, \phi(f))$ (the coherence paths $d(f, f')$ between $d(f)|f'$ and $d(ff')$ in $C(\gamma|ff')$ are uniquely determined). The action of $\mathrm{elim}(i, t)$ on paths $\mu : \gamma \xrightarrow{\sim} \gamma'$ is again uniquely determined.

By definition we have $\mathrm{elim}(i, \mathrm{inc}\, a) = i[a]$.

We define a patching structure on $\|A\|$, i.e. a homotopy inverse to the embedding $\eta_A : \|A\| \to D(\|A\|)$. Since $\|A\|$ and $D(\|A\|)$ are homotopy propositions, it suffices to give just a map $patch_{\|A\|} : D(\|A\|) \to \|A\|$ and it even suffices to give it on points. Indeed, $\|A\|$ is a homotopy proposition by construction and D preserves homotopy propositions by Corollary 19 from Chapter 2. Let $\gamma \in \Gamma(X)$ and $d \in D(\|A\|)(\gamma)$, i.e. $d(f) \in \|A\|(\gamma|f)$

and $d(f, f') : d(f)|f' \nRightarrow d(f\,f')$ such that $d(f, f')|f'' ; d(f\,f', f'') = d(f, f'\,f'')$. Set $patch_{\|A\|}(\gamma, d) \coloneqq \mathrm{patch}(\psi)$ where $\psi(f) = a$ if $d(f) = \mathrm{inc}\, a$ and $\psi(f) = \phi(f)(\mathrm{id})$ if $d(f) = \mathrm{patch}\, \phi$.

Theorem 67. *Groupoid-valued presheaves together with a patching structure form a cwf with unit, Σ, Π, identity, univalent universe, Booleans, natural numbers, and propositional truncation type structures.*

Proof. A presheaf with a patching structure is exactly a presheaf that is modal with respect to the descent data operation $\boldsymbol{D}$. Apply Proposition 43 from Chapter 4 to Theorem 61 from Chapter 6 (groupoid-valued presheaves form a model, namely the naive groupoid-valued presheaf model) and Proposition 65 above to obtain the cwf with unit, Σ, Π, identity, and univalent universe type structure. The Booleans, natural numbers, and propositional truncation type structures we defined in Section 7.2. □

7.3 Levelwise properties

In this section we show that in the refined groupoid-valued presheaf model contractibility, surjectivity and invertibility hold globally if and only if they hold levelwise.

The following corollary of Lemma 63 says that propositions with a patching structure are inhabited if (and only if) they are levelwise inhabited.

Corollary 68. *Let $A \in \mathrm{Ty}(\Gamma)$ be a family of groupoid-valued presheaves that are (homotopy) propositions and have patching structures, then for any family of points $a_\gamma : A(\gamma)$ indexed by $\gamma : \Gamma(X)$ there is a section $a \in \mathrm{Tm}(\Gamma, A)$. Equivalently, a family $A \in \mathrm{Ty}(\Gamma)$ of presheaves with a patching structure is contractible if (and only if) it is levelwise contractible.*

Proof. The assumption says that A is levelwise contractible, i.e. the groupoids $A(\gamma)$ are contractible for all $\gamma : \Gamma(X)$. By Lemma 63 the family $D(A) \in \mathrm{Ty}(\Gamma)$ of presheaves is contractible, and using the patching structure we get that A is contractible. □

Call a map internally surjective if the propositional truncations of its fibres are contractible. The following corollary says that maps between patch algebras are internally surjective if (and only if) they are levelwise surjective.

Corollary 69. *Let Δ and Γ be groupoid-valued presheaves with a patching structure, and $\sigma : \Delta \to \Gamma$ a map between them, then for any family of points $\delta_\gamma : \Delta(X)$ and paths $\nu_\gamma : \sigma(\delta_\gamma) \simeq \gamma$ indexed by $\gamma : \Gamma(X)$ there is a section of* $\mathrm{isSurj}(\sigma) = \prod_{\gamma:\Gamma} \left\| \sum_{\delta:\Delta} \sigma(\delta) \equiv \gamma \right\|$.

Proof. The assumption says that the groupoids $\left\| \sum_{\delta:\Delta(X)} \sigma(\delta) \equiv \gamma \right\|$ are contractible for all $\gamma : \Gamma(X)$, i.e. the presheaves $\mathrm{isSurj}(\sigma)(\gamma)$ are levelwise contractible for all $\gamma : \Gamma$. Note that Theorem 67 in particular says that presheaves with a patching structure are closed under identity, dependent sum and propositional truncation types. Hence, we can apply Corollary 68 to get a section of $\mathrm{isSurj}(\sigma)(\gamma)$ for all $\gamma : \Gamma$. We get a section of $\mathrm{isSurj}(\sigma)$ because contractible types are closed under dependent product types. □

Call a map $\sigma : \Delta \to \Gamma$ an epimorphism if maps τ and $\tau' : \Gamma \to \Theta$ are homotopic whenever their composites $\tau \circ \sigma$ and $\tau' \circ \sigma : \Delta \to \Theta$ with σ are. In a classical metatheory Corollary 69 implies that maps between patch algebras are internally surjective if and only if they are epimorphisms. Indeed, the principle of excluded middle implies that epimorphisms $\sigma : \Delta \to \Gamma$ have nonempty fibres, and the axiom of choice asserts the existence of a family of points in the fibres so that we can apply the corollary to get a section of $\mathrm{isSurj}(\sigma)$ whenever Δ and Γ have patching structures. In particular, in a classical metatheory epimorphisms between set-valued presheaves seen as discrete groupoid-valued presheaves, which have a patching structure by Proposition 66, are internally surjective.

Corollary 70. *Let Γ and Δ be groupoid-valued presheaves with patching structure, then a natural transformation $\sigma : \Delta \to \Gamma$ is an equivalence if and only if it is levelwise an equivalence.*

Proof. Assume Γ and Δ patch algebras, i.e. $\eta_\Gamma : \Gamma \to D(\Gamma)$ and $\eta_\Delta : \Delta \to D(\Delta)$ equivalences. The "only if" direction is immediate. In the other direction, if $\sigma : \Delta \to \Gamma$ is levelwise an equivalence, then $D(\sigma) : D(\Delta) \to D(\Gamma)$ is an equivalence by Lemma 64, $\eta_\Gamma \circ \sigma = D(\sigma) \circ \eta_\Delta$ is an equivalence because $\eta_\Delta : \Delta \to D(\Delta)$ is an equivalence by assumption, and finally $\sigma : \Delta \to \Gamma$ is an equivalence because also η_Γ is an equivalence by assumption. □

Corollary 71. *Let Γ be a groupoid-valued presheaf with a patching structure. In a classical metatheory, if Γ is a homotopy proposition then Γ is equivalent to a strict proposition.*

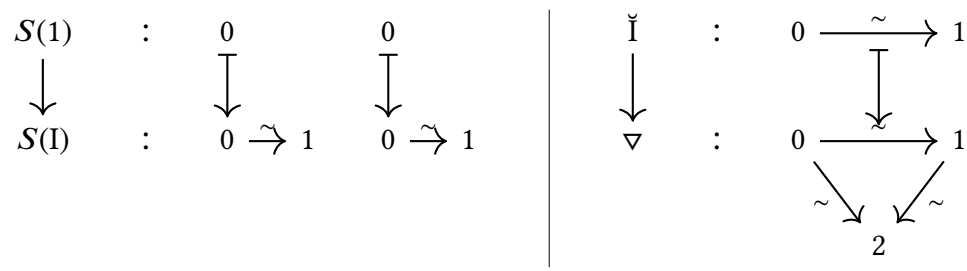
Proof. Apply Lemma 55 from Chapter 5 levelwise to obtain a family of subsingletons $\boldsymbol{C}(\Gamma(X))$ and equivalences $c(X) : \Gamma(X) \nrightarrow \boldsymbol{C}(\Gamma(X))$. The subsingletons assemble to a strict proposition $\boldsymbol{C}(\Gamma)$ and the equivalences to a natural transformation $c : \Gamma \to \boldsymbol{C}(\Gamma)$ which is levelwise an equivalence and hence an equivalence by Corollary 70. □

7.3.1 Remarks

We end with a few observations.

Firstly, we observe that (closed) $\boldsymbol{D}$-modal types $\boldsymbol{P}$ can be thought of as stacks for the trivial topology (and hence $\boldsymbol{D}$ as stackification) because the points of $\boldsymbol{D}(\boldsymbol{P})(X)$ are descent data d satisfying the cocycle condition for $\boldsymbol{P}$ on the trivial cover of X, and the left and right inverse laws for a patching structure $p : \boldsymbol{D}(\boldsymbol{P}) \to \boldsymbol{P}$ say that $p(d) \in \boldsymbol{P}(X)$ is the unique element $p_0 \in \boldsymbol{P}(X)$ up to isomorphism such that there are coherent paths $p_0|f \nrightarrow d(f)$. For more general sites we can also characterize the homotopy sheaves using a lex operation but it seems that we would need quotient inductive–inductive types to define stackification and, more generally, model higher inductive types.

Secondly, we observe that over the group $\mathbb{Z}/2\mathbb{Z}$ (with addition) the $\boldsymbol{D}$-modal types can also be thought of as fibrations in the injective model structure on groupoid-valued presheaves over $\mathbb{Z}/2\mathbb{Z}$. Bordg [12] shows that injective fibrations are another notion of type in groupoid-valued presheaves that can be used to model intensional type theory with one univalent universe. A natural transformation $\Delta \to \Gamma$ over $\mathbb{Z}/2\mathbb{Z}$ is an injective fibration if and only if it lifts against the following two trivial cofibrations [12, Proposition 5.3.5]:



The action on $S(1) \to S(\mathrm{I})$ swaps the left and right injections, and the action on $\breve{\mathrm{I}} \to \triangledown$ reverses the interval $0 \nrightarrow 1$ while keeping the point 2 fixed. Lifting problems for $S(1) \to S(\mathrm{I})$ encode a point $a \in A(\gamma)$ over the starting point of a path $\mu : \gamma \nrightarrow \gamma'$ in a context Γ and are solved by the lifting structure of any type $A \in \mathrm{Ty}(\Gamma)$ in the model presented here (modal or not). Lifting

problems for $\check{\mathbb{I}} \to \nabla$ encode points $e \in \widetilde{D}(A)$ over a point $d \in D(\Gamma)$ with a patch $\omega : d \xRightarrow{\sim} \eta_\Gamma(\gamma)$ and solutions are given by a patching structure for A. Note, however, that there are index categories $\mathscr{C}$ like the monoid $\mathbb{N}$ (with addition) such that not all D-modal types lift against all trivial cofibrations of the injective model structure on groupoid-valued presheaves over $\mathscr{C}$.

Lastly, we look at the presheaves $D(P)$ for a few concrete index categories $\mathscr{C}$.

7.4 Examples of the cobar operation

7.4.1 Over the preorder $\mathbb{N}$

In this subsection we consider $\mathbb{N}$ together with the usual order $\leqslant$ as the base category $\mathscr{C}$.

$$0 \longrightarrow 1 \longrightarrow 2 \longrightarrow \cdots$$

Diagram 6: *The preorder $\mathbb{N}$ as a category*

In this case, a presheaf A is a family of groupoids $A(n)$ with restriction functors $A(n) \to A(m), a \mapsto a|m$ for all $n \geqslant m$.

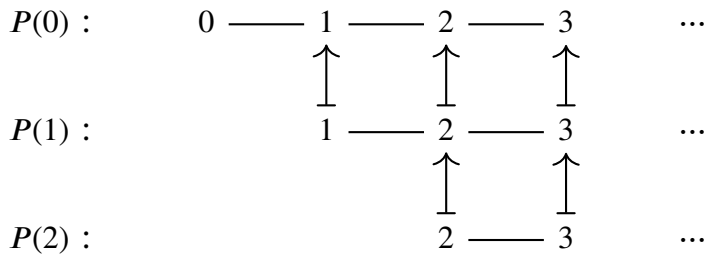
$$A(0) \longleftarrow A(1) \longleftarrow A(2) \longleftarrow \cdots$$

An element of $D(A)(n)$ is given by a family of elements $a(m) \in A(m)$ and paths $a(m,l) \in A(a(m)|l, a(l))$ such that $a(m,l)|k \cdot a(l,k) = a(m,k)$ for all $n \geqslant m \geqslant l \geqslant k$.

A global element of A is a family of elements $a(n) \in A(n)$ such that $a(n)|m = a(m)$.

A global element of $D(A)$ is given by a family of elements $a(n) \in A(n)$ and paths $a(n,m) \in A(a(n)|m, a(m))$ such that $a(n,m)|l \cdot a(m,l) = a(n,l)$.

We construct a presheaf P such that $D(P)$ has a global element while P does not. P is constructed as a family of codiscrete groupoids such that $P(n)$ is a full subgroupoid of $P(m)$ for all $n \geqslant m$. We take $P(n)$ to be the codiscrete groupoid on $\mathbb{N}_{\geqslant n} = \{n, n+1, n+2, \dots\}$ and the restriction functor from $P(n)$ to $P(m)$ the inclusion of $P(n)$ into $P(m)$.



A global element of P would be given by an $x \in \mathbb{N}$ such that x is an element in $P(n)$ for all $n \in \mathbb{N}$. However, no such x exists because for any x there is an index $n_x \in \mathbb{N}$ such that $x \notin P(n_x)$, in fact $x \notin P(m)$ for all $m \geqslant n_x$. Therefore, P does not have a global element. A global element of $D(P)$ on the other hand is given by any family of elements $x(n) \in \mathbb{N}$ because for any $n \geqslant m$ there is a unique path from $x(n)|m = x(n)$ to $x(m)$ in the codiscrete groupoid $P(m)$ on $\mathbb{N}_{\geqslant m}$. We get the following result.

Proposition 72. *There is an* empty *presheaf P such that $D(P)$ is inhabited.*

Definition 41. An *empty* presheaf is a presheaf that is not inhabited by a global element. ■

From this result it follows that P is an example of a presheaf such that η is not an equivalence and a presheaf such that $P + \neg P$ is not inhabited because $A + B$ is inhabited only if either A or B is. In fact, neither $P \vee \neg P$ is inhabited because in groupoid-valued presheaves $\|A\|$ is inhabited only if A is already.

Another observation we can make about P is that it is pointwise contractible but not as a presheaf in the sense of there being an equivalence between 1 and P, which we summarize as follows.

Proposition 73. *Over the preorder $\mathbb{N}$, being contractible is not given pointwise.*

More generally, we can say the following.

Proposition 74. *Over the preorder $\mathbb{N}$, being an equivalence is not given pointwise.*

Proof. The terminal map $P \to 1$ is pointwise an equivalence because each $P(n) \to 1$ is an equivalence because each $P(n)$ is codiscrete. However, the terminal map $P \to 1$ is not an equivalence because there is not even any map $1 \to P$. □

7.4.2 Over the group $\mathbb{Z}/2\mathbb{Z}$

In this subsection we consider $\mathbb{Z}/2\mathbb{Z}$ together with the usual addition $+$ as the base category $\mathscr{C}$ with a single object.

Diagram 7: *The group $\mathbb{Z}/2\mathbb{Z}$ as a category, the walking involution category*

Consider the presheaf $\boldsymbol{P}$ given by the walking isomorphism

$$0 \xrightarrow{\ p\ } 1$$

with a single path p from 0 to 1 and the swap action

$$\boldsymbol{P} \xrightarrow{-\cdot i} \boldsymbol{P}$$

$$\begin{array}{ccc} 0 & \longmapsto & 1 \\ {\scriptstyle p}\,\big| & \longmapsto & \big|\,{\scriptstyle p^{-1}} \\ 1 & \longmapsto & 0 \end{array}$$

that inverts p; the points 0 and 1 can be seen as the walking "homotopy" fixed point in the sense that they are fixed by the action *up to a path*; $\boldsymbol{P}$ is "represented" by $\bullet$ in the sense that it is the groupoid of elements of the representable presheaf together with the restriction action:

$$\begin{array}{ccc} \mathrm{id} & \longmapsto & i \\ {\scriptstyle i}\,\big| & \longmapsto & \big|\,{\scriptstyle i^{-1}\circ i\circ i = i} \\ i & \longmapsto & \mathrm{id} \end{array}$$

In general, a presheaf over $\mathbb{Z}/2\mathbb{Z}$ is given by a groupoid G with an isomorphism $G \to G, g \mapsto g \cdot i$ that is idempotent: $(g \cdot i) \cdot i = g$ (functoriality). A map from $(G, \cdot)$ to $(H, \cdot)$ is given by a functor $f : G \to H$ that respects these actions: $f(g \cdot i) = f(g) \cdot i$ (equivariance).

The presheaf P is *pointwise* codiscrete in the sense that the underlying functor of the terminal map

$$P \xrightarrow{!} 1$$

is an equivalence (of groupoids). However, the terminal map is not an equivalence because there are no candidate maps for an inverse map in the opposite direction. In other words, P has no global elements $1 \xrightarrow{x} P$ despite the underlying groupoid of P being inhabited. Indeed, the only possible images of the point $\bullet \in 1$ are $x(\bullet) = 0$ and $x(\bullet) = 1$ but neither choice is equivariant:

$$x(\bullet \cdot i) = x(\bullet) = 0 \neq 1 = 0 \cdot i = x(\bullet) \cdot i$$
$$x(\bullet \cdot i) = x(\bullet) = 1 \neq 0 = 1 \cdot i = x(\bullet) \cdot i$$

(The action on the terminal category 1 is the trivial one.) In fact, the global elements of any presheaf A are exactly the elements $a \in A$ that are fixed with respect to the action on A because the point $\bullet \in 1$ is a fixed point and fixed points necessarily need to be preserved by maps. Actually, pointwise codiscrete presheaves are codiscrete not just only if they are globally inhabited but exactly when they are globally inhabited. This is the case because any functor between codiscrete groupoids is an equivalence with any functor in the opposite direction as an inverse, i. e. the underlying functor of the terminal map from a pointwise codiscrete presheaf is an equivalence with any element as an inverse, and pointwise inverse maps between presheaves are inverses.

We record the observations about P as the following two facts.

Proposition 75. *Over $\mathbb{Z}/2\mathbb{Z}$ pointwise codiscrete presheaves are not necessarily codiscrete.*

Corollary 76. *Over $\mathbb{Z}/2\mathbb{Z}$ maps that are pointwise equivalences are not necessarily equivalences.*

The presheaf P is also considered by Bordg [12] as an example of a presheaf that is *not* injectively fibrant; in other words, the terminal map $P \to 1$

is an example of a pointwise fibrant map that is *not* injectively fibrant [12, Remark 3.2].

Another way to analyze the situation is by looking at the presheaves $D(A)$ whose elements are triples $(a, a', \alpha : a \cdot i \cong a')$ and paths are pairs $(\omega_1 : a \cong b, \omega_2 : a' \cong b')$ such that $\omega_1 i \cdot \beta = \alpha \cdot \omega_2$ which are taken to $(a', a, \alpha^{-1} \cdot i)$ and (ω_2, ω_1), respectively, by the action, and the maps $\eta_A : A \to D(A), a \mapsto (a, a \cdot i, \mathrm{id})$.

The elements of $D(D(A))$ are given by elements $d = (a, a', \alpha)$ and $d' = (b, b', \beta)$ of $D(A)$ and paths $\omega_1 : a' \cong b$ and $\omega_2 : a \cong b'$ in A such that $\omega_1 i \cdot \beta = \alpha^{-1} i \cdot \omega_2$, that is $\omega = (\omega_1, \omega_2) : di \cong d'$. The map $\mu_A : D(D(A)) \to D(A)$ is given by $(d, d', \omega) \mapsto (a, b, \alpha \cdot \omega_1)$ $(\alpha \cdot \omega_1 = \omega_2 i \cdot \beta^{-1} i)$.

In the concrete case of P we have

$$\begin{array}{ccc} D(P) & \xrightarrow{\quad -\cdot i \quad} & D(P) \end{array}$$

$$\begin{array}{ccc} (0,1,\mathrm{id}) & \xrightarrow{(p,\mathrm{id})} & (1,1,p) \\ {\scriptstyle (p,p^{-1})}\downarrow & & \downarrow{\scriptstyle (p^{-1},p^{-1})} \\ (1,0,\mathrm{id}) & \xrightarrow{(p^{-1},\mathrm{id})} & (0,0,p^{-1}) \end{array} \quad \longmapsto \quad \begin{array}{ccc} (1,0,\mathrm{id}) & \xrightarrow{(\mathrm{id},p)} & (1,1,p) \\ {\scriptstyle (p^{-1},p)}\downarrow & & \downarrow{\scriptstyle (p^{-1},p^{-1})} \\ (0,1,\mathrm{id}) & \xrightarrow{(\mathrm{id},p^{-1})} & (0,0,p^{-1}) \end{array}$$

with two fixed points $(0, 0, p^{-1})$ and $(1, 1, p)$ so that there is no map in the opposite direction $D(P) \to P$ because P does not have any. Also, if there was a map $D(P) \to P$ then P would be codiscrete, which it is not, because $D(P)$ is globally inhabited. In general, $D(A)$ is globally inhabited exactly if A has an element a with a path $\alpha : a \cdot i \cong a$ such that $\alpha \cdot i = \alpha^{-1}$ and this happens if A is pointwise codiscrete because then A is locally inhabited and for any element $a \in A$ the unique path $\bullet : a \cdot i \cong a$ satisfies $\bullet \cdot i = \bullet^{-1}$ by uniqueness.

Proposition 77. *Over $\mathbb{Z}/2\mathbb{Z}$ a pointwise codiscrete presheaf A is codiscrete if and only if η_A is an equivalence, for which it is enough to have a map $D(A) \to A$.*

7.4.3 Over the walking arrow

In this subsection we consider 2 together with the usual order $\leqslant$ as the base category $\mathscr{C}$.

$$0 \xrightarrow{!} 1$$

Diagram 8: *The walking arrow category*

In this case, a presheaf is given by a (vertical) groupoid functor $A(1) \to A(0), a \mapsto a0$, a map is given by a pair of (horizontal) groupoid functors $\alpha(1) : A(1) \to B(1)$ and $\alpha(0) : A(0) \to B(0)$ such that $\alpha(0)(a0) = \alpha(1)(a)0$, and a cell is given by a pair of natural transformations $n(1) : \alpha(1) \to \beta(1)$ and $n(0) : \alpha(0) \to \beta(0)$ such that $n(0)(a0) = n(1)(a)0$.

A presheaf is globally inhabited if and only if it is locally inhabited at the terminal object of the base category. In particular, pointwise codiscrete objects are globally inhabited and hence codiscrete because any pair of maps between pointwise codiscrete objects is an equivalence.

Proposition 78. *Over the walking arrow, pointwise contractible* closed *types are contractible.*

However, this does *not* remain true for *bundles* over arbitrary objects: The (discrete) groupoid bundle

$$\begin{array}{ccc} A(1) & \longrightarrow & \Gamma(1) \\ a_{\gamma_1} & & a_{\gamma_2} \longmapsto \gamma_1 \qquad \gamma_2 \end{array}$$

is inhabited but the presheaf bundle

$$\begin{array}{ccc} A & \longrightarrow & \Gamma \\ 0 & & a_{\gamma_1} \quad a_{\gamma_2} \longmapsto \gamma \quad = \quad \gamma \\ \downarrow & & \uparrow \quad \uparrow \quad \uparrow \quad \uparrow \\ 1 & & a_{\gamma_1} \quad a_{\gamma_2} \longmapsto \gamma_1 \quad \gamma_2 \end{array}$$

is not inhabited because the unique section of $A(1) \to \Gamma(1)$ cannot be extended to a map $\Gamma \to A$ because either choice at $\gamma \in \Gamma(0)$ will break naturality. In fact, the presheaf bundle $A \to \Gamma$ is even a pointwise contractible type (the path lifting is trivial because Γ has no non-trivial paths) so that we can make the following observation.

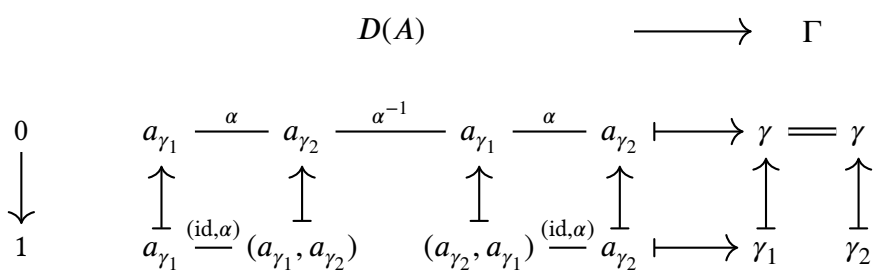
Proposition 79. *Over the walking arrow, pointwise contractible types are not necessarily contractible.*

Another way to look at pointwise contractible types is as maps of closed types that are pointwise equivalences. So, the map $\Gamma.A \to \Gamma$ is pointwise an equivalence without having an inverse $\Gamma \to \Gamma.A$ (note that inverses need not be sections in general but in this particular case the groupoids at 1 are discrete).

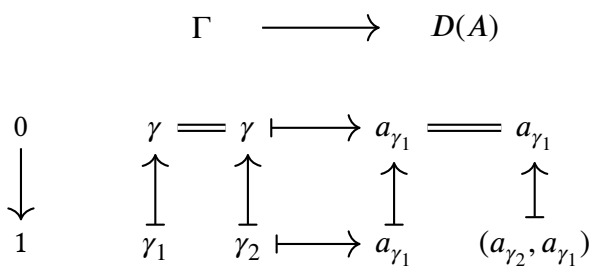
Proposition 80. *Over the walking arrow, pointwise invertible maps are not necessarily equivalences.*

What happens in this particular example is that η_A is not an equivalence. An element of $D(A)$ over $\gamma_1 \in \Gamma(1)$ is given by a triple $(a(1) \in A_{\gamma_1}, a(0) \in A_{\gamma_1 0}, \alpha : a(1)0 \cong a(0))$, and over $\gamma_0 \in \Gamma(0)$ by a single element $a(0) \in A_{\gamma_0}$; a path over $g_1 : \gamma_1 \cong \gamma_1'$ is given by a pair $(\alpha(1) : a(1) \cong_{g_1} a'(1), \alpha(0) : a(0) \cong_{g_1 0} a(0))$ such that $\alpha(1)0 \cdot \alpha' = \alpha \cdot \alpha(0)$, and over $g_0 : \gamma_0 \cong \gamma_0'$ by a single path $\alpha(0) : a(0) \cong_{g_0} a'(0)$.

In the particular example above $D(A)$ looks as follows.



Thus, $D(A)$ is globally inhabited by, for instance,



so there cannot be a map from $D(A)$ to A because, as we saw, the latter is not globally inhabited. In fact, if there was a map then A would be contractible:

Proposition 81. *Over the walking arrow, if A is a pointwise contractible type over Γ and there is a map $p_A : D(A) \to A$ then A is contractible.*

Proof. If A is pointwise contractible then for each $\gamma \in \Gamma$ there is an element $a(\gamma) \in A(\gamma)$ and for each $\gamma_1 \in \Gamma(1)$ there is a path $\alpha(\gamma_1) : a(\gamma_1)0 \cong a(\gamma_1 0)$ so that we can define a section d of $D(A)$ by $d(\gamma_1) = (a(\gamma_1), a(\gamma_1!), \alpha(\gamma_1))$ and $d(\gamma_0) = a(\gamma_0)$ (the action $d(g)$ on paths $g : \gamma \cong \gamma'$ is trivial because $D(A)_g(d(\gamma), d(\gamma'))$ is a singleton).

Now, if there is a map $p_A : D(A) \to A$, then we have the section $p_A \circ d : \Gamma \to A$ and a homotopy to the section $\Gamma.A \to A\mathrm{p}$ because the path sets of A are singletons, that is A is contractible. □

and, indeed, η_A would be an equivalence:

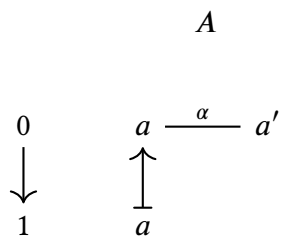
Proposition 82. *Over the walking arrow, if A is a contractible type over Γ then $\eta_A : A \to D(A)$ is an equivalence.*

Proof. Since A is contractible, any map $p_A : D(A) \to A$ will be a left inverse, whose existence is sufficient for η_A to be an equivalence, and such a map exists. □

Note that $\eta_\Gamma : \Gamma \to D(\Gamma)$ is an equivalence because (strictly) $D(\Gamma) \cong \Gamma$ (also by an application of Proposition 84 below) and that if $\eta_A : A \to D(A)$ was an equivalence then $A \to \Gamma$ would be an equivalence, which, as observed earlier, it is not.

Proposition 83. *Over the walking arrow, if $\eta_A : A \to D(A)$ and $\eta_B : B \to D(B)$ are equivalences and $f : A \to B$ is pointwise an equivalence then f is invertible.*

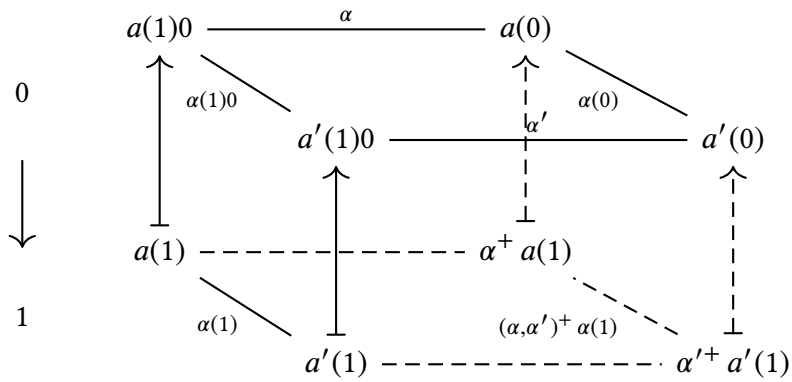
Proposition 78 says that the following presheaf A is codiscrete and hence Proposition 82 says that η_A is an equivalence.



The underlying groupoid functor $A(1) \to A(0)$ is not a fibration because a cannot be lifted along α simply because the fibre over a' is empty. However, the converse holds.

Proposition 84. *Any cloven fibration $A(1) \to A(0), a \mapsto a0$ is the underlying functor of a presheaf A such that $\eta_A : A \to D(A)$ is an equivalence.*

Proof. Assume a (possibly non-strict) lifting structure on $A(1) \to A(0)$ and construct a map $p_A : D(A) \to A$ by mapping elements $(a(1), a(0), \alpha)$ and paths $(\alpha(1), \alpha(0))$ at 1 to the chosen solutions $\alpha^+ a(1)$ and $(\alpha, \alpha')^+ \alpha(1)$ of their lifting problems



and elements $a(0)$ and paths $\alpha(0)$ at 0 to themselves. This mapping is functorial because it can be checked that $\mathrm{id}_{\alpha^+ a(1)}$ and $(\alpha, \alpha')^+ \alpha(1) \circ (\alpha', \alpha'')^+ \alpha'(1)$ are solutions and it is both natural and a left inverse of $\eta_A : A \to D(A)$ by construction. □

The last two observations suggest that there are two different model structures on the arrow category $\overrightarrow{\mathrm{Gpd}}$ both in which equivalences are given pointwise.

7.4.4 Over the walking idempotent

In this subsection we consider 2 together with the usual multiplication $\cdot$ as the base category $\mathscr{C}$ with a single object.

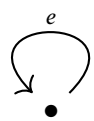


Diagram 9: *The walking idempotent category*

In this case, a presheaf A is given by an idempotent groupoid action $A \to A, a \mapsto ae$: $aee = ae$; a map $f : A \to B$ is given by an equivariant groupoid functor: $f(ae) = f(a)e$.

An element $(a(f), a(f,g))_{f,g}$ of $D(A)$ is uniquely determined by the two elements $a(\mathrm{id})$ and $a(e)$ and the path $a(\mathrm{id}, e) \in A(a(\mathrm{id})e, a(e))$ in A subject to no conditions.

1. $a(f, \mathrm{id}) = a(f, \mathrm{idid}) = a(f, \mathrm{id})\mathrm{id} \cdot a(f\mathrm{id}, \mathrm{id}) = a(f, \mathrm{id}) \cdot a(f, \mathrm{id})$, and
2. $a(\mathrm{id}, e)e \cdot a(e, e) = a(\mathrm{id}, e)e \cdot a(\mathrm{id}e, e) = a(\mathrm{id}, ee) = a(\mathrm{id}, e)$.

Pointwise codiscrete groupoid-valued presheaves are codiscrete if and only if they are globally inhabited. Over the base category under consideration, any non-empty object A is globally inhabited because the global inhabitants of presheaves indexed by a monoid are exactly the fixed points of the monoid action on A and for any element $x \in A$ its restriction $xe \in A$ is fixed by the action because e is the only non-trivial monoid element and an idempotent in the case under consideration. However, this does *not* remain true in the general case of types in context. The following type (with trivial lifting because its context Γ has no non-trivial paths) is pointwise codiscrete but does not have a section

$$\begin{array}{ccc} A & \longrightarrow & \Gamma \end{array}$$

$$\begin{array}{ccccc} a_{\gamma_1} & a_{\gamma_2} & \longmapsto & \gamma_1 & \gamma_2 \\ \downarrow & \downarrow & & \downarrow & \downarrow \\ a'_{\gamma_1} & a'_{\gamma_2} & \longmapsto & \gamma & \gamma \\ \circlearrowleft & \circlearrowleft & & \circlearrowleft & \circlearrowleft \end{array}$$

because any section $t : \Gamma \to A$ of $A \to \Gamma$ would need to map γ_1 to $t(\gamma_1) = a_{\gamma_1}$ and γ_2 to $t(\gamma_2) = a_{\gamma_2}$ but then $a'_{\gamma_1} = t(\gamma_1)e = t(\gamma) = t(\gamma_2)e = a'_{\gamma_2}$, which is a contradiction. In conclusion,

Proposition 85. *Over the walking idempotent, pointwise contractible types are not necessarily contractible.*

Corollary 86. *Over the walking idempotent, pointwise invertible maps are not necessarily equivalences.*

Proof. Consider the map $A \to 1\langle\rangle$ for $\Gamma \vdash A$ the counterexample considered in the proof of Proposition 85. □

Proposition 87. *Over the walking idempotent,* closed *types whose underlying groupoid is codiscrete are contractible.*

Conclusion

In this thesis we have constructed two groupoid-valued presheaf models (over an arbitrary index category $\mathscr{C}$) of univalent type theory with a single universe.

The first model is called the naive groupoid-valued presheaf model. It is constructed by translating univalent type theory into extensional type theory extended with a universe of (strict) propositions. This translation is essentially the groupoid model construction by Hofmann and Streicher [42, 41].

The second model is called the refined groupoid-valued presheaf model. It is constructed by translating univalent type theory into univalent type theory extended with a descent data operation. This translation, which uses ideas from Quirin [70], Coquand, Mannaa and Ruch [20] and Rijke, Shulman and Spitters [74], is presented in this thesis.

In the naive groupoid-valued presheaf model types are interpreted by presheaf groupoids. A presheaf groupoid P is a family of groupoids $P(X)$ with restriction functors $P(f) : P(X) \to P(Y)$ indexed by objects $X, Y : \mathscr{C}$ and morphisms $f : Y \to X$ such that the equations $P(\mathrm{id}_X) = \mathrm{Id}_{P(X)}$ and $P(f \circ g) = P(g) \circ P(f)$ hold strictly. The naive groupoid-valued presheaf model is obtained by constructing the groupoid model internal to the set-valued presheaf model (over $\mathscr{C}$) of extensional type theory. This model is deficient because over $\mathscr{C} = \mathbb{Z}/2\mathbb{Z}$, for instance, we do not get the expected notion of surjective maps between presheaves: For the unique map s_0 from the presheaf $P_0 = \{0, 1\}$ with the "swap" action into the terminal presheaf 1, which we expect to be surjective because the unique element of 1 has both elements of P_0 in its preimage, the type $\mathrm{isSurj}(s_0) = \prod_{x:1} \left\| \mathrm{fib}(s_0, x) \right\|$ is not inhabited. In other words, the naive groupoid-valued presheaf model is deficient because *levelwise* surjective maps $s : P \to Q$ (like $s_0 : P_0 \to 1$) might not be (internally) surjective in the sense of $\mathrm{isSurj}(s)$. Levelwise surjectivity of a map s gives us that the type $\mathrm{isSurj}(s)$ is levelwise inhabited and since this type is a homotopy proposition levelwise surjectivity gives us a *virtual* element of $\mathrm{isSurj}(s)$. A virtual element p of a presheaf P is like a strict element $(p(f) \in P(Y))_{f:Y \to X}$

except that the strict equalities $p|(fg) = (p|f)|g$ are replaced by (coherent) paths $p|(fg) \nrightarrow (p|f)|g$. In these terms, the deficiency of the naive groupoid-valued presheaf model can be formulated as virtual elements of presheaf groupoids possibly not corresponding to strict elements. It is resolved by the refined groupoid-valued presheaf model.

In the refined groupoid-valued presheaf model types are interpreted by presheaf groupoids equipped with a patching structure. A patching structure allows virtual elements to be strictified. Having a patching structure can be expressed as being modal with respect to a so-called descent data operation that sends a presheaf groupoid to its presheaf of virtual elements. The refined groupoid-valued presheaf model is then obtained by constructing the submodel of modal types with respect to this descent data operation. Note that the construction of the submodel of modal types is completely general in that it can be carried out for an arbitrary model of univalent type theory equipped with a descent data operation.

A natural next goal is to construct groupoid-valued *sheaf* models (over arbitrary sites, i.e. index categories equipped with a Grothendieck topology) of univalent type theory with a single universe. They can be used [20] to refute Markov's principle and the axiom of countable choice.

Generalizing the models of Coquand, Mannaa and Ruch [20] to arbitrary sites does not seem possible because over an arbitrary site the set-valued sheaf of Booleans, for instance, does not eliminate into arbitrary groupoid-valued sheaves in such a way that the computation rules hold strictly. A type of Booleans that eliminates strictly into groupoid-valued sheaves would need to have elements that are identified by paths with but not strictly equal to canonical elements. Those elements are precisely the patch elements that would be identified strictly with canonical elements in a set-valued sheaf. What should be possible, however, is to construct a model where the universe does not contain a code for the type of Booleans but only for an equivalent type. The reason why the universe in a groupoid-valued sheaf model would not contain a code for the type of Booleans is the same why the universe in the groupoid-valued presheaf model does not contain codes for higher inductive types, for instance it is not closed under propositional truncation. Another issue with *constructive* groupoid-valued sheaf models over arbitrary sites is that quotient inductive types seem to be required in the metatheory to interpret W-types.

The issues of universes not being closed under inductive types and quotient inductive types being needed in the metatheory go away [22] when considering cubical- instead of groupoid-valued models. The models of Coquand, Ruch

and Sattler [22] moreover support a hierarchy of univalent universes instead of just a single one. The cubical-valued sheaf model is constructed using a variation of the technique used for the refined groupoid-valued presheaf model presented in this thesis: First a constructive model of univalent type theory (a cubical model in this case) is relativized to an index category, then the cubical-valued presheaf model is constructed as the submodel of modal types with respect to a descent data operation, and finally the cubical-valued sheaf model is constructed as the submodel of modal types with respect to a family of descent data operations.

The crucial step in the construction of the cubical-valued sheaf model is the second one, that is the construction of the cubical-valued presheaf model from the descent data operation D that sends a presheaf cubical set to its presheaf of virtual elements. Once the presheaf model has been constructed, the sheaf model for a Grothendieck topology on the index category can be constructed internal to it. The presheaf model comes equipped with a family of (strict) propositions $[c]$ indexed by the preorder J of covers $c : \mathrm{J}$ that the Grothendieck topology specifies. A descent datum on a cover c for a presheaf A, i.e. a D-modal presheaf cubical set A, can then be expressed as a function $[c] \to A$. For each $c : \mathrm{J}$, the operation D_c that sends a presheaf A to its presheaf of descent data $[c] \to A$ on c is a descent data operation because exponentiation by any proposition is. Therefore, as we have seen, the unit type is D_c-modal for each $c : \mathrm{J}$, and the types that are modal for all covers are closed under dependent product, dependent sum, and identity types. Call a type J-modal if it is D_c-modal for all $c : \mathrm{J}$. To show that a subuniverse U_J of J-modal types is itself J-modal Coquand, Ruch and Sattler [22] use a slightly different argument than for the subuniverses of D-modal types. Instead of defining a descent data operation whose modal types are the J-modal ones, the construction proceeds by showing that each D_c preserves[1] J-modal types. Patching structures on U_J for each D_c can then be defined in essentially the same way as for a subuniverse of modal types for a single descent data operation. Up to this point, the present construction could also be carried out to obtain a groupoid-valued sheaf model of univalent type theory with a single univalent universe. However, the higher inductive types in the relativized cubical model that Coquand, Ruch and Sattler [22] use to model inductive types in the sheaf model would not be contained in the universe and might not exist constructively in the groupoid-valued case, as mentioned above.

[1] To show that each D_c preserves J-modal types it is crucial that $c_0 \leq c_1$ in the preorder J of covers implies $[c_0] \to [c_1]$, and that for any two covers c_1 and c_2 there exists $c_0 \leq c_1, c_2$.

Bibliography

[1] Peter Aczel. 'On Relating Type Theories and Set Theories'. In: *Types for Proofs and Programs, International Workshop TYPES '98, Kloster Irsee, Germany, March 27-31, 1998, Selected Papers*. Ed. by Thorsten Altenkirch, Wolfgang Naraschewski and Bernhard Reus. Vol. 1657. Lecture Notes in Computer Science. Springer, 1998, pp. 1–18. DOI: 10.1007/3-540-48167-2_1 (cit. on pp. 68, 75, 76, 85, 88).

[2] Peter Aczel. 'The type theoretic interpretation of constructive set theory'. In: *Logic Colloquium '77 (Proc. Conf., Wrocław, 1977)*. Vol. 96. Studies in Logic and the Foundations of Mathematics. North-Holland, Amsterdam-New York, 1978, pp. 55–66 (cit. on p. 46).

[3] Benedikt Ahrens and Peter LeFanu Lumsdaine. 'Displayed Categories'. In: *Log. Methods Comput. Sci.* 15.1 (2019). DOI: 10.23638/LMCS-15(1:20)2019 (cit. on pp. 68, 71).

[4] Thorsten Altenkirch, Martin Hofmann and Thomas Streicher. 'Categorical Reconstruction of a Reduction Free Normalization Proof'. In: *Category Theory and Computer Science, 6th International Conference, CTCS '95, Cambridge, UK, August 7-11, 1995, Proceedings*. Ed. by David H. Pitt, David E. Rydeheard and Peter T. Johnstone. Vol. 953. Lecture Notes in Computer Science. Springer, 1995, pp. 182–199. DOI: 10.1007/3-540-60164-3_27 (cit. on p. 5).

[5] Thorsten Altenkirch, Martin Hofmann and Thomas Streicher. 'Reduction-Free Normalisation for a Polymorphic System'. In: *Proceedings, 11th Annual IEEE Symposium on Logic in Computer Science, New Brunswick, New Jersey, USA, July 27-30, 1996*. IEEE Computer Society, 1996, pp. 98–106. DOI: 10.1109/LICS.1996.561309 (cit. on p. 5).

[6] Jean-Marc Andreoli. 'Logic Programming with Focusing Proofs in Linear Logic'. In: *J. Log. Comput.* 2.3 (1992), pp. 297–347. DOI: 10.1093/logcom/2.3.297 (cit. on pp. 52, 55).

[7] Robert Atkey, Neil Ghani and Patricia Johann. 'A relationally parametric model of dependent type theory'. In: *The 41st Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL '14, San Diego, CA, USA, January 20-21, 2014*. Ed. by Suresh Jagannathan and Peter Sewell. ACM, 2014, pp. 503–516. DOI: 10.1145/2535838.2535852 (cit. on p. 5).

[8] Jeremy Avigad, Krzysztof Kapulkin and Peter LeFanu Lumsdaine. 'Homotopy limits in type theory'. In: *Math. Struct. Comput. Sci.* 25.5 (2015), pp. 1040–1070. DOI: 10.1017/S0960129514000498 (cit. on pp. 19, 20, 24).

[9] Steve Awodey. 'Natural models of homotopy type theory'. In: *Math. Struct. Comput. Sci.* 28.2 (2018), pp. 241–286. DOI: 10.1017/S0960129516000268 (cit. on p. 51).

[10] Jean-Philippe Bernardy, Thierry Coquand and Guilhem Moulin. 'A Presheaf Model of Parametric Type Theory'. In: *The 31st Conference on the Mathematical Foundations of Programming Semantics, MFPS 2015, Nijmegen, The Netherlands, June 22-25, 2015*. Ed. by Dan R. Ghica. Vol. 319. Electronic Notes in Theoretical Computer Science. Elsevier, 2015, pp. 67–82. DOI: 10.1016/j.entcs.2015.12.006 (cit. on p. 5).

[11] Lars Birkedal et al. 'Guarded Cubical Type Theory'. In: *J. Autom. Reason.* 63.2 (2019), pp. 211–253. DOI: 10.1007/s10817-018-9471-7 (cit. on p. 5).

[12] Anthony Bordg. 'On a Model Invariance Problem in Homotopy Type Theory'. In: *Appl. Categorical Struct.* 27.3 (2019), pp. 311–322. DOI: 10.1007/s10485-019-09558-w (cit. on pp. 8, 108, 112, 113).

[13] Kenneth S. Brown. 'Abstract homotopy theory and generalized sheaf cohomology'. In: *Transactions of the American Mathematical Society* 186 (1973), pp. 419–458. ISSN: 0002-9947. DOI: 10.2307/1996573 (cit. on pp. 19, 24).

[14] John Cartmell. 'Generalised algebraic theories and contextual categories'. PhD thesis. University of Oxford, UK, 1978 (cit. on p. 51).

[15] John Cartmell. 'Generalised algebraic theories and contextual categories'. In: *Annals of Pure and Applied Logic* 32.3 (1986), pp. 209–243. ISSN: 0168-0072. DOI: 10.1016/0168-0072(86)90053-9 (cit. on pp. 9, 45, 46).

[16] Pierre Clairambault and Peter Dybjer. 'The biequivalence of locally cartesian closed categories and Martin-Löf type theories'. In: *Math. Struct. Comput. Sci.* 24.6 (2014). DOI: 10.1017/S0960129513000881 (cit. on p. 63).

[17] Cyril Cohen et al. 'Cubical Type Theory: A Constructive Interpretation of the Univalence Axiom'. In: *21st International Conference on Types for Proofs and Programs, TYPES 2015, May 18-21, 2015, Tallinn, Estonia*. Ed. by Tarmo Uustalu. Vol. 69. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2015, 5:1–5:34. DOI: 10.4230/LIPIcs.TYPES.2015.5 (cit. on p. 5).

[18] François Conduché. 'Au sujet de l'existence d'adjoints à droite aux foncteurs "image réciproque" dans la catégorie des catégories'. In: *Comptes Rendus Hebdomadaires des Séances de l'Académie des Sciences. Séries A et B* 275 (1972), A891–A894. ISSN: 0151-0509 (cit. on p. 74).

[19] Thierry Coquand. 'An algorithm for testing conversion in type theory'. In: *Logical frameworks (Sophia-Antipolis, 1990)*. Cambridge Univ. Press, Cambridge, 1991, pp. 255–279. DOI: 10.1017/CBO9780511569807.011 (cit. on p. 5).

[20] Thierry Coquand, Bassel Mannaa and Fabian Ruch. 'Stack semantics of type theory'. In: *32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavik, Iceland, June 20-23, 2017*. IEEE Computer Society, 2017, pp. 1–11. DOI: 10.1109/LICS.2017.8005130 (cit. on pp. 8, 10, 121, 122).

[21] Thierry Coquand and Christine Paulin. 'Inductively defined types'. In: *COLOG-88, International Conference on Computer Logic, Tallinn, USSR, December 1988, Proceedings*. Ed. by Per Martin-Löf and Grigori Mints. Vol. 417. Lecture Notes in Computer Science. Springer, 1988, pp. 50–66. DOI: 10.1007/3-540-52335-9_47 (cit. on p. 22).

[22] Thierry Coquand, Fabian Ruch and Christian Sattler. 'Constructive sheaf models of type theory'. In: *Math. Struct. Comput. Sci.* 31.9 (2021), pp. 979–1002. DOI: 10.1017/S0960129521000359 (cit. on pp. 8, 10, 122, 123).

[23] Djordje Cubric, Peter Dybjer and Philip J. Scott. 'Normalization and the Yoneda Embedding'. In: *Math. Struct. Comput. Sci.* 8.2 (1998), pp. 153–192. URL: http://journals.cambridge.org/action/displayAbstract?aid=44745 (cit. on p. 5).

[24] Radu Diaconescu. 'Axiom of choice and complementation'. In: *Proceedings of the American Mathematical Society* 51 (1975), pp. 176–178. ISSN: 0002-9939. DOI: 10.2307/2039868 (cit. on pp. 83, 95).

[25] Peter Dybjer. 'Internal Type Theory'. In: *Types for Proofs and Programs, International Workshop TYPES'95, Torino, Italy, June 5-8, 1995, Selected Papers*. Ed. by Stefano Berardi and Mario Coppo. Vol. 1158. Lecture Notes in Computer Science. Springer, 1995, pp. 120–134. DOI: 10.1007/3-540-61780-9_66 (cit. on pp. 9, 45, 46, 49).

[26] Thomas Ehrhard. 'Une sémantique catégorique des types dépendants. Application au Calcul des Constructions.' PhD thesis. Université Paris VII, 1988 (cit. on p. 52).

[27] Samuel Eilenberg and J. A. Zilber. 'Semi-simplicial complexes and singular homology'. In: *Annals of Mathematics. Second Series* 51 (1950), pp. 499–513. ISSN: 0003-486X. DOI: 10.2307/1969364 (cit. on p. 2).

[28] Marcelo Fiore and Dmitrij Szamozvancev. 'Formal metatheory of second-order abstract syntax'. In: *Proc. ACM Program. Lang.* 6.POPL (2022), pp. 1–29. DOI: 10.1145/3498715 (cit. on p. 5).

[29] Marcelo P. Fiore. 'Discrete Generalised Polynomial Functors - (Extended Abstract)'. In: *Automata, Languages, and Programming - 39th International Colloquium, ICALP 2012, Warwick, UK, July 9-13, 2012, Proceedings, Part II*. Ed. by Artur Czumaj et al. Vol. 7392. Lecture Notes in Computer Science. Springer, 2012, pp. 214–226. DOI: 10.1007/978-3-642-31585-5_22 (cit. on p. 51).

[30] Marcelo P. Fiore, Gordon D. Plotkin and Daniele Turi. 'Abstract Syntax and Variable Binding'. In: *14th Annual IEEE Symposium on Logic in Computer Science, Trento, Italy, July 2-5, 1999*. IEEE Computer Society, 1999, pp. 193–202. DOI: 10.1109/LICS.1999.782615 (cit. on p. 5).

[31] Nicola Gambino and Richard Garner. 'The identity type weak factorisation system'. In: *Theor. Comput. Sci.* 409.1 (2008), pp. 94–109. DOI: 10.1016/j.tcs.2008.08.030 (cit. on pp. 19, 23).

[32] Jean-Yves Girard. 'Interprétation fonctionnelle et Élimination des coupures de l'arithmétique d'ordre supérieur'. PhD thesis. Université Paris VII, 1972 (cit. on p. 58).

[33] Jean-Yves Girard. 'On the Unity of Logic'. In: *Ann. Pure Appl. Log.* 59.3 (1993), pp. 201–217. DOI: 10.1016/0168-0072(93)90093-S (cit. on pp. 52, 55).

[34] Jean Giraud. 'Méthode de la descente'. In: *Société Mathématique de France. Bulletin. Mémoire* 2 (1964), pp. viii+150. ISSN: 0583-8665 (cit. on p. 74).

[35] Joseph A. Goguen and José Meseguer. 'Completeness of many-sorted equational logic'. In: *ACM SIGPLAN Notices* 17.1 (1982), pp. 9–17. DOI: 10.1145/947886.947887 (cit. on p. 46).

[36] Robert Goldblatt. *Topoi - the categorial analysis of logic, Second rev. Edition*. Vol. 98. Studies in logic and the foundations of mathematics. North-Holland, 1984. ISBN: 978-0-444-86711-7. URL: https://www.sciencedirect.com/bookseries/studies-in-logic-and-the-foundations-of-mathematics/vol/98 (cit. on p. 71).

[37] Martin Hofmann. *Extensional constructs in intensional type theory*. CPHC/BCS distinguished dissertations. Springer, 1997. ISBN: 978-3-540-76121-1 (cit. on pp. 9, 68, 84).

[38] Martin Hofmann. 'Semantical Analysis of Higher-Order Abstract Syntax'. In: *14th Annual IEEE Symposium on Logic in Computer Science, Trento, Italy, July 2-5, 1999*. IEEE Computer Society, 1999, pp. 204–213. DOI: 10.1109/LICS.1999.782616 (cit. on p. 5).

[39] Martin Hofmann. 'Syntax and semantics of dependent types'. In: *Semantics and logics of computation (Cambridge, 1995)*. Vol. 14. Publ. Newton Inst. Cambridge Univ. Press, Cambridge, 1997, pp. 79–130. DOI: 10.1017/CBO9780511526619.004 (cit. on pp. 4, 9, 16, 45, 56, 85, 87, 99).

[40] Martin Hofmann and Thomas Streicher. 'Lifting Grothendieck Universes'. 1997. URL: https://www2.mathematik.tu-darmstadt.de/~streicher/NOTES/lift.pdf (cit. on pp. 4, 9, 85, 88).

[41] Martin Hofmann and Thomas Streicher. 'The groupoid interpretation of type theory'. In: *Twenty-five years of constructive type theory (Venice, 1995)*. Vol. 36. Oxford Logic Guides. Oxford Univ. Press, New York, 1998, pp. 83–111 (cit. on pp. 5, 6, 67, 73, 84, 87, 97, 99, 121).

[42] Martin Hofmann and Thomas Streicher. 'The Groupoid Model Refutes Uniqueness of Identity Proofs'. In: *Proceedings of the Ninth Annual Symposium on Logic in Computer Science (LICS '94), Paris, France, July 4-7, 1994*. IEEE Computer Society, 1994, pp. 208–212. DOI: 10.1109/LICS.1994.316071 (cit. on pp. 5, 67, 121).

[43] Bart P. F. Jacobs. *Categorical Logic and Type Theory*. Vol. 141. Studies in logic and the foundations of mathematics. North-Holland, 2001. ISBN: 978-0-444-50853-9. URL: http://www.elsevierdirect.com/product.jsp?isbn=9780444508539 (cit. on p. 19).

[44] Peter T. Johnstone. *Sketches of an elephant: a topos theory compendium. Vol. 1*. Vol. 43. Oxford Logic Guides. The Clarendon Press, Oxford University Press, New York, 2002, pp. xxii+468+71. ISBN: 0-19-853425-6 (cit. on p. 13).

[45] Mark P. Jones. 'Functional Programming with Overloading and Higher-Order Polymorphism'. In: *Advanced Functional Programming, First International Spring School on Advanced Functional Programming Techniques, Båstad, Sweden, May 24-30, 1995, Tutorial Text*. Ed. by Johan Jeuring and Erik Meijer. Vol. 925. Lecture Notes in Computer Science. Springer, 1995, pp. 97–136. DOI: 10.1007/3-540-59451-5_4 (cit. on p. 16).

[46] André Joyal. 'Notes on Clans and Tribes'. In: *CoRR* abs/1710.10238 (2017). arXiv: 1710.10238. URL: https://arxiv.org/abs/1710.10238 (cit. on pp. 19, 22).

[47] Daniel M. Kan. 'Abstract homotopy. I'. In: *Proceedings of the National Academy of Sciences of the United States of America* 41 (1955), pp. 1092–1096. ISSN: 0027-8424. DOI: 10.1073/pnas.41.12.1092 (cit. on p. 2).

[48] Ambrus Kaposi, Simon Huber and Christian Sattler. 'Gluing for Type Theory'. In: *4th International Conference on Formal Structures for Computation and Deduction, FSCD 2019, June 24-30, 2019, Dortmund, Germany*. Ed. by Herman Geuvers. Vol. 131. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019, 25:1–25:19. DOI: 10.4230/LIPIcs.FSCD.2019.25 (cit. on pp. 9, 45, 60).

[49] Krzysztof Kapulkin and Peter LeFanu Lumsdaine. 'The simplicial model of univalent foundations (after Voevodsky)'. In: *Journal of the European Mathematical Society (JEMS)* 23.6 (2021), pp. 2071–2126. ISSN: 1435-9855. DOI: 10.4171/JEMS/1050 (cit. on p. 5).

[50] A. Kock and G. C. Wraith. *Elementary toposes*. Lecture Notes Series, No. 30. Aarhus Universitet, Matematisk Institut, Aarhus, 1971, pp. i+118 (cit. on p. 13).

[51] Saul A. Kripke. 'Semantical analysis of intuitionistic logic. I'. In: *Formal Systems and Recursive Functions (Proc. Eighth Logic Colloq., Oxford, 1963)*. North-Holland, Amsterdam, 1965, pp. 92–130 (cit. on pp. 1, 3).

[52] Magnus Baunsgaard Kristensen, Rasmus Ejlers Møgelberg and Andrea Vezzosi. *A model of Clocked Cubical Type Theory*. 2021. arXiv: 2102.01969 (cit. on p. 5).

[53] Marie La Palme Reyes, Gonzalo E. Reyes and Houman Zolfaghari. *Generic figures and their glueings*. Polimetrica, Monza, 2004 (cit. on p. 2).

[54] J. Lambek and P. J. Scott. *Introduction to higher order categorical logic*. Vol. 7. Cambridge Studies in Advanced Mathematics. Cambridge University Press, Cambridge, 1986, pp. x+293. ISBN: 0-521-24665-2 (cit. on p. 4).

[55] F. William Lawvere. 'Equality in hyperdoctrines and comprehension schema as an adjoint functor'. In: *Applications of Categorical Algebra (Proc. Sympos. Pure Math., Vol. XVII, New York, 1968)*. Amer. Math. Soc., Providence, R.I., 1970, pp. 1–14 (cit. on p. 51).

[56] Daniel R. Licata et al. 'Internal Universes in Models of Homotopy Type Theory'. In: *3rd International Conference on Formal Structures for Computation and Deduction, FSCD 2018, July 9-12, 2018, Oxford, UK*. Ed. by Hélène Kirchner. Vol. 108. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018, 22:1–22:17. DOI: 10.4230/LIPIcs.FSCD.2018.22 (cit. on p. 85).

[57] Saunders Mac Lane and Ieke Moerdijk. *Sheaves in geometry and logic*. Universitext. A first introduction to topos theory, Corrected reprint of the 1992 edition. Springer-Verlag, New York, 1994, pp. xii+629. ISBN: 0-387-97710-4 (cit. on p. 13).

[58] Saunders MacLane. *Categories for the working mathematician*. Graduate Texts in Mathematics, Vol. 5. Springer-Verlag, New York-Berlin, 1971, pp. ix+262 (cit. on p. 46).

[59] Bassel Mannaa, Rasmus Ejlers Møgelberg and Niccolò Veltri. 'Ticking clocks as dependent right adjoints: Denotational semantics for clocked type theory'. In: *Log. Methods Comput. Sci.* 16.4 (2020). URL: https://lmcs.episciences.org/6980 (cit. on p. 5).

[60] Per Martin-Löf. '100 years of Zermelo's axiom of choice: what was the problem with it?' In: *Comput. J.* 49.3 (2006), pp. 345–350. DOI: 10.1093/comjnl/bxh162 (cit. on p. 18).

[61] Per Martin-Löf. 'An intuitionistic theory of types'. In: *Twenty-five years of constructive type theory (Venice, 1995)*. Vol. 36. Oxford Logic Guides. Oxford Univ. Press, New York, 1998, pp. 127–172 (cit. on pp. 16, 52, 56, 57).

[62] Per Martin-Löf. 'An intuitionistic theory of types: predicative part'. In: *Logic Colloquium '73 (Bristol, 1973)*. 1975, 73–118. Studies in Logic and the Foundations of Mathematics, Vol. 80 (cit. on pp. 52, 57, 58).

[63] Per Martin-Löf. *Intuitionistic type theory*. Vol. 1. Studies in proof theory. Bibliopolis, 1984. ISBN: 978-88-7088-228-5 (cit. on pp. 16, 18, 22, 56).

[64] Per Martin-Löf. *Invariance under isomorphism and definability*. Dec. 2012. URL: https://video.ias.edu/file/33014 (cit. on pp. 9, 68).

[65] Per Martin-Löf. *Substitution calculus*. 1992. URL: https://archive-pml.github.io/martin-lof/pdfs/Substitution-calculu%20%3E%20s-1992.pdf (cit. on p. 46).

[66] Eugenio Moggi. 'A Category-theoretic Account of Program Modules'. In: *Category Theory and Computer Science, Manchester, UK, September 5-8, 1989, Proceedings*. Ed. by David H. Pitt et al. Vol. 389. Lecture Notes in Computer Science. Springer, 1989, pp. 101–117. DOI: 10.1007/BFb0018347 (cit. on p. 51).

[67] Andreas Nuyts, Andrea Vezzosi and Dominique Devriese. 'Parametric quantifiers for dependent type theory'. In: *Proc. ACM Program. Lang.* 1.ICFP (2017), 32:1–32:29. DOI: 10.1145/3110276 (cit. on p. 5).

[68] Andrew M. Pitts. 'Categorical logic'. In: *Handbook of logic in computer science, Vol. 5*. Vol. 5. Handb. Log. Comput. Sci. Oxford Univ. Press, New York, 2000, pp. 39–128 (cit. on pp. 19, 51).

[69] The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. Institute for Advanced Study, 2013. URL: https://homotopytypetheory.org/book/ (cit. on pp. 6, 23–25, 29, 33–35, 52, 81–83, 95, 97).

[70] Kevin Quirin. 'Lawvere-Tierney sheafification in Homotopy Type Theory. (Faisceautisation de Lawvere-Tierney en théorie des types homotopiques)'. PhD thesis. École des mines de Nantes, France, 2016. URL: https://tel.archives-ouvertes.fr/tel-01486550 (cit. on pp. 7–10, 14, 31, 38, 121).

[71] Michael Rathjen, Edward R. Griffor and Erik Palmgren. 'Inaccessibility in Constructive Set Theory and Type Theory'. In: *Ann. Pure Appl. Log.* 94.1-3 (1998), pp. 181–200. DOI: 10.1016/S0168-0072(97)00072-9 (cit. on p. 46).

[72] *Revêtements étales et groupe fondamental (SGA 1)*. Vol. 3. Documents Mathématiques (Paris) [Mathematical Documents (Paris)]. Séminaire de géométrie algébrique du Bois Marie 1960–61. [Algebraic Geometry Seminar of Bois Marie 1960-61], Directed by A. Grothendieck, With two papers by M. Raynaud, Updated and annotated reprint of the 1971 original [Lecture Notes in Math., 224, Springer, Berlin; MR0354651 (50 #7129)]. Société Mathématique de France, Paris, 2003, pp. xviii+327. ISBN: 2-85629-141-4 (cit. on p. 73).

[73] Egbert Rijke. 'Introduction to Homotopy Type Theory'. May 2018. URL: https://www.andrew.cmu.edu/user/erijke/hott/hott_intro.pdf (cit. on p. 28).

[74] Egbert Rijke, Michael Shulman and Bas Spitters. 'Modalities in homotopy type theory'. In: *Log. Methods Comput. Sci.* 16.1 (2020). DOI: 10.23638/LMCS-16(1:2)2020 (cit. on pp. 7, 14, 31, 36, 38, 121).

[75] Dana S. Scott. 'Relating theories of the λ-calculus'. In: *To H. B. Curry: essays on combinatory logic, lambda calculus and formalism*. Academic Press, London-New York, 1980, pp. 403–450 (cit. on p. 4).

[76] Thomas Streicher. 'Fibred Categories à la Jean Bénabou'. In: *CoRR* abs/1801.02927 (2018). arXiv: 1801.02927. URL: https://arxiv.org/abs/1801.02927 (cit. on pp. 71, 73).

[77] Paul Taylor. *Practical Foundations of Mathematics*. Vol. 59. Cambridge studies in advanced mathematics. Cambridge University Press, 1999. ISBN: 978-0-521-63107-5 (cit. on p. 19).

[78] *Théorie des topos et cohomologie étale des schémas. Tome 1: Théorie des topos*. Lecture Notes in Mathematics, Vol. 269. Séminaire de Géométrie Algébrique du Bois-Marie 1963–1964 (SGA 4), Dirigé par M. Artin, A. Grothendieck, et J. L. Verdier. Avec la collaboration de N. Bourbaki, P. Deligne et B. Saint-Donat. Springer-Verlag, Berlin-New York, 1972, pp. xix+525 (cit. on p. 46).

[79] Simone Tonelli. 'Investigations into a model of type theory based on the concept of basic pair'. MA thesis. Stockholms Universitet, Sweden, 2013. URL: https://kurser.math.su.se/pluginfile.php/16103/mod_folder/content/0/2013/2013_08_report.pdf (cit. on pp. 9, 68).

[80] A. S. Troelstra and D. van Dalen. *Constructivism in mathematics. Vol. I*. Vol. 121. Studies in Logic and the Foundations of Mathematics. An introduction. North-Holland Publishing Co., Amsterdam, 1988, xx+342+XIV. ISBN: 0-444-70266-0; 0-444-70506-6 (cit. on p. 87).

[81] A. S. Troelstra and D. van Dalen. *Constructivism in mathematics. Vol. II*. Vol. 123. Studies in Logic and the Foundations of Mathematics. An introduction. North-Holland Publishing Co., Amsterdam, 1988, i–xviii and 345–880 and I–LII. ISBN: 0-444-70358-6 (cit. on p. 87).

[82] Angelo Vistoli. 'Notes on Grothendieck topologies, fibered categories and descent theory'. In: *CoRR* abs/math/0412512 (2004). arXiv: math/0412512. URL: https://arxiv.org/abs/math/0412512 (cit. on p. 73).

[83] Vladimir Voevodsky. 'A universe polymorphic type system'. An unfinished unreleased manuscript. Oct. 2014. URL: https://www.math.ias.edu/Voevodsky/files/files-annotated/Dropbox/Unfinished_papers/Type_systems/UPTS_current/Universe_polymorphic_type_sytem.pdf (cit. on p. 57).

[84] Matthew Z. Weaver and Daniel R. Licata. 'A Constructive Model of Directed Univalence in Bicubical Sets'. In: *LICS '20: 35th Annual ACM/IEEE Symposium on Logic in Computer Science, Saarbrücken, Germany, July 8-11, 2020*. Ed. by Holger Hermanns et al. ACM, 2020, pp. 915–928. DOI: 10.1145/3373718.3394794 (cit. on p. 5).

[85] Felix Wellen. 'Formalizing Cartan Geometry in Modal Homotopy Type Theory'. PhD thesis. Karlsruher Institut für Technologie, Germany, 2017. DOI: 10.5445/IR/1000073164 (cit. on pp. 25, 27).

[86] G. C. Wraith. 'Lectures on elementary topoi'. In: *Model theory and topoi (Conf., Bangor, 1973)*. 1975, 114–206. Lecture Notes in Math., Vol. 445 (cit. on p. 13).

[87] Gavin Wraith. 'Artin glueing'. In: *Journal of Pure and Applied Algebra* 4 (1974), pp. 345–348. ISSN: 0022-4049. DOI: 10.1016/0022-4049(74)90014-0 (cit. on p. 13).

[88] Noam Zeilberger. 'On the unity of duality'. In: *Ann. Pure Appl. Log.* 153.1-3 (2008), pp. 66–96. DOI: 10.1016/j.apal.2008.01.001 (cit. on pp. 52, 55).