CHALMERS

The Path Model of Intensional Type Theory

Master of Science Thesis in Computer Science – Algorithms, Languages and Logic

FABIAN RUCH

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
Gothenburg, Sweden, October 2015

The Path Model of Intensional Type Theory
FABIAN RUCH

Examiner: BENGT NORDSTRÖM

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
SE-412 96 Gothenburg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
Gothenburg, Sweden, October 2015

Abstract

The groupoid interpretation of Martin-Löf type theory not only shows the independence of uniqueness of identity proofs from the axioms of intensional type theory but is also constructive and validates the computation rules as definitional equalities. The groupoid semantics are very clear when interpreting dependent types and in particular the identity types but less so when defining equality preservation for terms, interpreting context extension or constructing the transport for identity proofs. The indirections stem from the fact that paths over paths is a derived notion in the groupoid interpretation. The notion is, however, a primitive in so called relational models which have been employed to prove abstraction theorems for type theories. We generalise the groupoid interpretation to a refined relational interpretation of intensional type theory and show that it is a model in the sense of categories with families. The refined relations support a concatenation operator that has identities and inverses; hence a model of paths.

Contents

Chapter 1

Introduction

Type theory is a formal system for programming, logic and mathematics. It can be used to write functional programs, prove propositions in predicate logic or construct mathematical objects. When working in type theory, one derives judgments of the form

$$\Gamma \vdash a : A$$

which is read as "a is a term of type A in context Γ". The term a can be interpreted as a *program* producing a value of the data type A or as a *proof* of the proposition A or as an *element* of the structure A. A common data type is `Bool` (the Boolean data type having two constructors `true` and `false`), a common proposition is $\top$ (the formula having truth value true) and a common structure is that of a semigroup (sets equipped with an associative binary operation).

An important logical type is the identity type Id. For instance, the semigroup structure is expressed as the type

$$\Sigma_{S:Set,\cdot:S\times S\to S}\Pi_{s,t,u:S}\mathsf{Id}_S(s\cdot(t\cdot u),(s\cdot t)\cdot u)$$

and consists not only of a set and a binary operation but also contains a proof of associativity, whose statement makes reference to equality. The other types involved here are Σ and Π which denote the types of tuples and functions but also correspond to existential and universal quantification respectively.

The crucial feature of type theory that makes it a programming language is its computational content. For instance, any term a of type `Bool` is definitionally equal (as opposed to propositionally equal via a term of type Id) to either `true` or `false`

$$\vdash a = \texttt{true} : \texttt{Bool} \quad \text{or} \quad \vdash a = \texttt{false} : \texttt{Bool},$$

even though it might be expressed using abstractions like addition and comparison on natural numbers.

We present a constructive set-theoretic model of intensional type theory (ITT) with extensional dependent function spaces and an extensional universe as well as propositional truncation. The function spaces and universe are extensional in the sense that the types

$$\Pi_{f,g:\Pi AB}(\Pi_{a:A}\mathsf{Id}_B(\mathsf{app}(f,a),\mathsf{app}(g,a)))\to\mathsf{Id}_{\Pi AB}(f,g) \tag{1.1}$$

and

$$\Pi_{A,B:U}(\Sigma_{f:El(A)\to El(B),g:El(B)\to El(A)}\mathsf{Id}(g\circ f,\mathsf{id})\times\mathsf{Id}(f\circ g,\mathsf{id}))\to\mathsf{Id}_U(A,B) \tag{1.2}$$

are inhabited. Equation (1.1) says that functions are equal if they are point-wise equal and eq. (1.2) says that isomorphisms prove small types equal. These two axioms are close to mathematical practice, although only function extensionality holds in set theory. Propositional truncation turns any type A into a type $\|A\|$ such that

$$\Pi_{x,y:\|A\|}\mathsf{Id}_{\|A\|}(x,y) \tag{1.3}$$

is inhabited. Equation (1.3) says that the truncation of a type identifies all its terms. Propositional truncation can be seen as part of an axiomatisation of the quotient $\|A\|$ of a type A by the equivalence relation that relates all terms of type A. In this sense, $\|A\|$ is comprised of a single equivalence class $\|a\|$ every $a : A$ is a representative of. A recursion principle to define functions from the quotient completes the axiomatisation of $\|A\|$

$$f : A \to P, h : \Pi_{p,q:P}\mathsf{Id}_P(p,q) \vdash \|f\| : \|A\| \to P \quad \text{and} \quad \mathsf{app}(\|f\|, \|a\|) = \mathsf{app}(f, a).$$

In order to apply the recursion principle the co-domain must be a "proposition", that is every two elements of P are equal as witnessed by the hypothesis variable h. This ensures that the definition of f maps representatives of the same equivalence class to equal elements in P and, hence, that $\|f\|$ is well-defined.

Extending the type theory by axioms of type 1.1, 1.2 and 1.3 respectively is problematic for two reasons. Firstly, such an extension introduces non-canonical constants at the types ΠAB, U and $\|A\|$, while the computation rule for identity elimination

$$J_{C,d}[a, a, \mathsf{refl}_a] = d[a]$$

only reduces on canonical ones, that is reflexivity proofs. The ensuing problem can be phrased as either computation getting stuck or proofs being based on non-evident axioms. Secondly, identity induction

$$J_C : (d : \Pi_{a:A} C[a, a, \mathsf{refl}_a]) \to (a, b : A)(p : \mathsf{Id}_A(a, b)) \to C[a, b, p]$$

only requires a proof of the reflexivity case to conclude. Then, it is not clear how the induction conclusion soundly follows in the non-canonical cases introduced by the axioms corresponding to 1.1, 1.2 and 1.3 or whether the type theory remains consistent as a logic.

The two issues are not independent. Having a computational justification of the new constants will re-establish the consistency of the theory. (Relative) consistency can also be established by devising a model. However, devising a model is weaker than giving computation rules. Still it is a step in the right direction.

The model we present is a combination of the constructive set-theoretic groupoid model [HS98] and the type-theoretic relational model [Ton13] [BJP12] [AGJ14]. The groupoid model models ITT with extensional function spaces and an extensional universe, and the relational model models ITT without the extensionality principles. We briefly recap how types are interpreted in the groupoid and the relational model respectively to motivate the interpretation of types in our model of ITT with the extensionality principles.

The groupoid model interprets contexts as groupoids and dependent types as families of groupoids and functors. More formally, a type A in context Γ consists of a groupoid $A(\gamma)$ for every $\gamma : \Gamma$ and a functor $A(g) : A(\gamma) \to A(\gamma')$ for every $g \in hom_\Gamma(\gamma, \gamma')$. In particular, the identity type $\mathsf{Id}_A(a, b)$ is interpreted by the discrete groupoid whose objects are the morphisms $hom_A(a, b)$. There are several morphisms going from a to b in general and the groupoid model thus *invalidates* uniqueness of identity proofs (UIP), that is the type

$$\Pi_{a,b:A}\Pi_{p,q:\mathsf{Id}_A(a,b)}\mathsf{Id}_{\mathsf{Id}_A(a,b)}(p, q)$$

is not inhabited for all types A. A term a of type A is a family of objects $a(\gamma) : A(\gamma)$ for every $\gamma : \Gamma$ such that equality is preserved. Since equality corresponds to morphisms there must be a morphism $a(g)$ for every $g \in hom_\Gamma(\gamma, \gamma')$. However, the objects $a(\gamma)$ and $a(\gamma')$ do not necessarily lie in the same groupoid and are thus not connected by morphisms. The solution of Hofmann and Streicher [HS98] is to define morphisms between $a(\gamma)$ and $a(\gamma')$ as the $A(\gamma')$-homset of $A(g)(a(\gamma))$ and $a(\gamma')$. However, preservation of equality also demands that the morphism $a(g \cdot g')$ is the composite of $a(g)$ and $a(g')$, which do not share co-domain and domain by the definition just given. Similar problems arise when defining component-wise composition in the groupoid corresponding to context extension of Γ by A, and when generalising the transport of equality proofs by symmetry and transitivity in the non-dependent to the dependent case.

In contrast, the equality proofs between the term components $a(\gamma) : A(\gamma)$ and $a(\gamma') : A(\gamma')$ are given as primitives by the relational interpretation of the term's type. The type A in the relational model

is interpreted by a meta-theoretic type $A(\gamma)$ for every term $\gamma : \Gamma$ of the semantic context and a type $A(g, a, a')$ for all terms $g : \Gamma(\gamma, \gamma'), a : A(\gamma), a' : A(\gamma')$. The dependent relations do not only simplify the definition of terms in the relational model but also the construction of the extended context $\Gamma.A$. The reason is that the equality proofs between terms $\langle \gamma, a \rangle : \Gamma.A$ and $\langle \gamma', a' \rangle : \Gamma.A$ can simply be taken as the pairs of proofs between γ and γ' and a and a' respectively.

Types in the relational model do not carry any of the extra structure of the groupoid model like composition of equality proofs and mapping of terms over equalities in the context. However, these operations and others inducing reflexivity and symmetry proofs are needed to interpret the identity types $\mathsf{Id}_A(a, b)$ by the semantic equality proofs or internalise the semantic equality relations by Id_A. Actually, because additional structure is required for the relations in the relational model to be equality relations it is incorrect to refer to their inhabitants as equality proofs but it suggests the connection we are about the make.

The present work combines the groupoid model with the relational model. To this end, we generalise the morphisms in the groupoid model to (dependent) paths. The notion of paths corresponds to the elements of the types $A(g, a, a')$ in the relational model. Thereby, we do not rely on the transport function to define equality preservation and composition. The model we obtain can be seen as a truncated version of the cubical set model [BCH13] [Hub15].

Organisation

The thesis is organised as follows. Chapter 2 introduces a common presentation of a specific flavour of dependent type theory as a formal system, the notion of model that will be used in the remainder of the thesis and how it relates to the syntax, because the result will not be an interpretation function. Then, chapter 3 establishes what exactly the data of a relational interpretation consists of. The main part of the thesis is a proof of the model axioms for a refined relational interpretation, which is presented in chapter 4. Before the thesis concludes, chapter 5 gives an alternative axiomatisation of the refined interpretation. Some straightforward and lengthy proofs are included in appendix A.

We apply very elementary reasoning to highlight the fact that we are working with a refinement on relations. Using the fact that we are actually working with categories and fibrations would give us a richer meta-theory and more direct proofs but also cloud the relational view, which is not restricted to the binary case.

Contributions

The contributions of this thesis are

1. a complete proof of the model axioms for the so-called groupoid interpretation of intensional type theory. Interpreting types as groupoids with transport and terms as functors gives a model of dependent type theory with extensional Π, Id, extensional U, and $\| \cdot \|$ types. The groupoid interpretation was conceived, albeit in a different formulation, more than 20 years ago and is an outstanding result still relevant today. The paper left some verifications to the reader that we have the space for to include,

2. a previously unidentified relationship between morphoid type theory and intensional type theory. Namely, that they share a common model which led to the formulation of morphoid type theory and seems to coincide with the groupoid interpretation. If paths form a morphoid, then the connected points form a groupoid and vice versa.

Notation

Throughout the thesis, the following conventions of notation are made. The capital Greek letters $\Gamma, \Delta, ...$ denote type-theoretical contexts, either semantic or syntactical depending on the context. When talking about semantic contexts, the corresponding small Greek letters denote its elements (members of the set Γ_S and the corresponding small Latin letters proofs of their relatedness (members of the set Γ_R). For instance, the elements of a semantic context Γ are denoted by $\gamma, \gamma', \gamma'', ...$ and g, g' denote proofs that

γ (γ') and γ' (γ'') are related by Γ. There are special proofs id_γ and g^{-1} which borrow their notation from category theory. The capital Latin letters $A, B, ...$ are used to refer to types, both semantic and syntactical. Semantic types are families of sets and relations indexed by the respective semantic context, which are denoted by A_γ and A_g respectively. The small Latin letters $a, b, ...$ can denote terms of the corresponding type but are usually used when talking about the elements of the corresponding semantic type. The context situation is flipped for types and the corresponding small Greek letters $\alpha, \beta, ...$ refer to relatedness proofs between elements of semantic types. For the sake of clarity or simply to increase the number of available symbols, the context symbols are used as subscripts like a_γ, α_γ or α_g to say that a type element belongs to the set A_γ and a type proof belongs to the relation A_{id_γ} or A_g. So far, we have introduced common symbols for sets and members thereof. The membership relation is usually denoted by $\in$ but can occasionally be referred to as $:$, especially when the right-hand side is a structure that involves a set and it is clear which set the left-hand side is meant to be a member of. To refer to equality in the meta-theory the symbol $\equiv$ is used, whereas the symbol $=$ refers to (definitional) equality in the object theory. The object theory also possesses an internal notion of equality (propositional equality) which is a type and will be denoted as such (Id) to distinguish it clearly from the other two.

Thanks

The author wants to thank his supervisor for the introduction to the research of categorical models of type theory as well as his examiner and opponent for their patience.

Chapter 2

Preliminaries

2.1 Dependent Type Theory

Martin-Löf type theory or more generally intensional type theory is a particular flavour of dependent type theory. Dependent type theory is a formal system, that is a relation on a set of formulas generated by a number of *schematic* inference rules. Not all formulas are meaningful and the rules define exactly when a formula is well-formed. The syntax of dependent type theory is defined in terms of the three classes of terms

1. contexts
2. types
3. terms (sub-class variables)

and the six classes of formulas or so called judgments (two for each class of terms)

1. $\vdash \Gamma$ ctx
2. $\Gamma \vdash A$ type
3. $\Gamma \vdash a : A$
4. $\vdash \Gamma = \Delta$ ctx
5. $\Gamma \vdash A = B$ type
6. $\Gamma \vdash a = b : A$

where $\Gamma, \Delta; A, B; a, b$ are non-terminals of class context, type and term respectively. What justifies the adjective dependent over simple type theory is the fact that the syntax of types refers to the syntax of terms and both must be defined simultaneously. The rules of dependent type theory are organised in groups of rules for contexts, types and terms.

$$\frac{\vdash \Gamma \text{ ctx}}{\vdash \Gamma = \Gamma \text{ ctx}} \qquad \frac{\vdash \Gamma = \Delta \text{ ctx}}{\vdash \Delta = \Gamma \text{ ctx}} \qquad \frac{\vdash \Gamma = \Delta \text{ ctx} \qquad \vdash \Delta = \Theta}{\vdash \Gamma = \Theta \text{ ctx}}$$

$$\vdash [] \text{ ctx} \ \textsc{Empty context}$$

$$\frac{\vdash \Gamma \text{ ctx} \qquad \Gamma \vdash A \text{ type}}{\vdash \Gamma, x : A \text{ ctx}} \ \textsc{Context extension} \qquad \frac{\vdash \Gamma = \Delta \text{ ctx} \qquad \Gamma \vdash A = A' \text{ type}}{\vdash \Gamma, x : A = \Delta, x : A' \text{ ctx}}$$

$$\frac{\Gamma \vdash A \text{ type}}{\Gamma \vdash A = A \text{ ctx}} \qquad \frac{\Gamma \vdash A = B \text{ type}}{\Gamma \vdash B = A \text{ type}} \qquad \frac{\Gamma \vdash A = B \text{ type} \qquad \Gamma \vdash B = C \text{ type}}{\Gamma \vdash A = C \text{ type}}$$

$$\frac{\Gamma \vdash A \text{ type} \qquad \vdash \Gamma = \Delta}{\Delta \vdash A \text{ type}}$$

$$\frac{\Gamma \vdash a : A}{\Gamma \vdash a = a : A} \qquad \frac{\Gamma \vdash a = b : A}{\Gamma \vdash b = a : A} \qquad \frac{\Gamma \vdash a = b : A \qquad \Gamma \vdash b = c : A}{\Gamma \vdash a = c : A}$$

$$\frac{\Gamma \vdash a : A \qquad \Gamma \vdash A = B}{\Gamma \vdash a : B} \ \textsc{Type conversion}$$

$$\frac{\vdash \Gamma \text{ ctx} \qquad \Gamma \vdash A \text{ type}}{\Gamma, x : A \vdash x : A} \ \textsc{Variable introduction}$$

Starting from the dependent type theory presented so far, type theories like Martin-Löf type theory extend the set of structural rules with logical rules. In addition to formation and introduction rules, the axiomatisation of a logical type also consists of elimination and computation rules. Common types include functions in the style of the lambda calculus and simple type theory

Definition 2.1.1 (Π type former)**.**

$$\frac{\Gamma \vdash A \text{ type} \qquad \Gamma, x : A \vdash B[x] \text{ type}}{\Gamma \vdash \Pi AB \text{ type}} \ \textsc{Pi} \qquad \frac{\Gamma \vdash A = A' \text{ type} \qquad \Gamma, x : A \vdash B = B' \text{ type}}{\Gamma \vdash \Pi AB = \Pi A'B' \text{ type}}$$

$$\frac{\Gamma, x : A \vdash b[x] : B[x]}{\Gamma \vdash \lambda x.b : \Pi AB} \ \textsc{Pi-Intro} \qquad \frac{\Gamma, x : A \vdash b = b' : B}{\Gamma \vdash \lambda x.b = \lambda x.b' : \Pi AB}$$

$$\frac{\Gamma \vdash f : \Pi AB \qquad \Gamma \vdash a : A}{\Gamma \vdash fa : B[a/x]} \ \textsc{Pi-Elim} \qquad \frac{\Gamma \vdash f = f' : \Pi AB \qquad \Gamma \vdash a = a' : A}{\Gamma \vdash fa = f'a' : B[a/x]}$$

$$\frac{\Gamma, x : A \vdash b[x] : B[x] \qquad \Gamma \vdash a : A}{\Gamma \vdash (\lambda x.b)a = b[a/x] : B[a/x]} \ \textsc{Pi-}\beta \qquad \frac{\Gamma \vdash f : \Pi AB}{\Gamma \vdash \lambda x.fx = f : \Pi AB} \ \textsc{Pi-}\eta$$

■

as well as proofs of identity and substitution of equals in the object language

Definition 2.1.2 (Id type former)**.**

$$\frac{\Gamma \vdash A \text{ type}}{\Gamma, x_1 : A, x_2 : A \vdash \mathsf{Id}_A \text{ type}}\ \textsc{Id} \qquad \frac{\Gamma \vdash A = A' \text{ type}}{\Gamma, x_1 : A, x_2 : A \vdash \mathsf{Id}_A = \mathsf{Id}_{A'} \text{ type}}$$

$$\frac{\Gamma \vdash A \text{ type}}{\Gamma, x : A \vdash \mathsf{refl}_A : \mathsf{Id}_A[x/x_1, x/x_2]}\ \textsc{Id-Intro} \qquad \frac{\Gamma \vdash A = A' \text{ type} \qquad \Gamma \vdash a = a' : A}{\Gamma, x : A \vdash \mathsf{refl}_A = \mathsf{refl}_{A'} : \mathsf{Id}_A}$$

$$\frac{\begin{array}{c}\Gamma \vdash A \text{ type} \\ \Gamma, x_1 : A, x_2 : A, p : \mathsf{Id}_A \vdash C[x_1, x_2, p] \text{ type} \qquad \Gamma, x : A \vdash d : C[x/x_1, x/x_2, \mathsf{refl}_A/p]\end{array}}{\Gamma, x_1 : A, x_2 : A, p : \mathsf{Id}_A \vdash J_{C,d} : C}\ \textsc{Id-Elim}$$

$$\frac{\begin{array}{c}\Gamma \vdash A \text{ type} \\ \Gamma, x_1 : A, x_2 : A, p : \mathsf{Id}_A \vdash C[x_1, x_2, p] \text{ type} \qquad \Gamma, x : A \vdash d : C[x/x_1, x/x_2, \mathsf{refl}_A/p]\end{array}}{\Gamma, x : A \vdash J_{C,d}[x/x_1, x/x_2, \mathsf{refl}_A/p] \equiv d : C[x/x_1, x/x_2, \mathsf{refl}_A/p]}\ \textsc{Id-}\eta$$

■

and a type for quantification over types and introduction of new types within the theory

Definition 2.1.3 (U type former)**.**

$$\Gamma \vdash \mathsf{U} \text{ type } \textsc{U}$$

$$\frac{\Gamma \vdash A : \mathsf{U}}{\Gamma \vdash |A| \text{ type}}\ \textsc{U-Elim} \qquad \frac{\Gamma \vdash A = A' : \mathsf{U}}{\Gamma \vdash |A| = |A'| \text{ type}}$$

■

This set of common types were introduced to have a formal intuitionistic predicate logic of types. In recent years further extensions to the novel identity types were proposed.

Per Martin-Löf's identity type is defined for every type, including the identity types themselves. Every two proofs of equality can thereby be compared for equality adding a dimension of distinguishable elements. Together with the groupoid structure induced by the identity types this suggests an infinite-dimensional groupoid structure of types in ITT. Vladimir Voevodsky proposed to organise types in ITT according to their dimension.

Types of dimension -2 are exactly the ones that are inhabited and every two elements are equal. A type is -1-dimensional if for every element pair the corresponding identity type is of dimension -2, that is every two elements are equal but the type can be the empty type. The hierarchy continues with the level 0 types whose identity types are level -1 types. The elements of level 0 types need not be equal but if they are there is only proof up to propositional equality. The hierarchy we obtain by defining $(n+1)$-types to be exactly those with n-level identity types may or may not collapse.

Voevodsky's stratification of types can be defined in MLTT internally (contractible types are defined using Σ and all other levels in terms of the previous level and Π). Imposing a specific level on a type, that is forming a new type with the same elements but additional equalities at the respective dimension, needs further axioms. [Uni13] axiomatises the propositional truncation of a type A which coerces all elements $a, b : A$ into being uniquely equal syntactically as follows.

Definition 2.1.4 ($\|\cdot\|$ type former)**.**

$$\frac{\Gamma \vdash A \text{ type}}{\Gamma \vdash \|A\| \text{ type}}\ \textsc{Trunc} \qquad \frac{\Gamma \vdash A = A' \text{ type}}{\Gamma \vdash \|A\| = \|A'\| \text{ type}}$$

$$\frac{\Gamma \vdash A \text{ type} \qquad \Gamma \vdash a : A}{\Gamma \vdash \|a\| : \|A\|}\ \textsc{Trunc-Intro} \qquad \frac{\Gamma \vdash A \text{ type} \qquad \Gamma \vdash a = a' : A}{\Gamma \vdash \|a\| = \|a'\| : \|A\|}$$

$$\frac{\Gamma \vdash A \text{ type}}{\Gamma \vdash \mathit{isProp}(\|A\|) : \Pi_{\|A\|}\Pi_{\|A\|}\mathsf{Id}_{\|A\|}}$$

$$\frac{\Gamma \vdash A \text{ type} \qquad \Gamma \vdash B \text{ type} \qquad \Gamma \vdash \mathit{isProp}(B) : \Pi_B\Pi_B\mathsf{Id}_B \qquad \Gamma \vdash f : \Pi AB}{\Gamma \vdash \|f\| : \Pi\|A\|B}$$

$$\frac{\Gamma \vdash A \text{ type} \qquad \Gamma \vdash B \text{ type} \qquad \Gamma \vdash \mathit{isProp}(B) : \Pi_B\Pi_B\mathsf{Id}_B \qquad \Gamma \vdash f = f' : \Pi AB}{\Gamma \vdash \|f\| = \|f'\| : \Pi\|A\|B}$$

$$\frac{\Gamma \vdash A \text{ type} \qquad \Gamma \vdash B \text{ type} \qquad \Gamma \vdash \mathit{isProp}(B) : \Pi_B\Pi_B\mathsf{Id}_B \qquad \Gamma \vdash f : \Pi AB \qquad \Gamma \vdash a : A}{\Gamma \vdash \|f\|\|a\| = fa : B}$$

■

The Trunc type former 2.1.4 does not follow the intro-elim pattern. Also note that the codomain type does not depend on the domain as otherwise $\Pi\|A\|B$ would not be well-typed/well-scoped.

Actually, [Uni13] also gives the following induction principle for propositional truncation

$$\frac{\begin{array}{c}\Gamma, x : \|A\| \vdash C \text{ type} \\ \Gamma, x : \|A\| \vdash \mathit{isProp}(C(x)) : \Pi_{c_1:C,c_2:C}\mathsf{Id}_C[c_1/x_1, c_2/x_2] \qquad \Gamma \vdash i : \Pi_{a:A}C[\|a\|/x]\end{array}}{\Gamma \vdash \|i\| : \Pi_{\|A\|}C}$$

but immediately shows that it is implied by the recursion principle and the fact that truncating a type forces all its elements into equality. More concretely, given a witness $a : A$ that A is non-empty, $C[x]$ can be proved inhabited by transporting $i(\|a\|)$ along the equality $\mathsf{Id}_{\|A\|}[\|a\|/x_1, x/x_2]$. Hence, we can construct a term of type $\Pi_A\Pi_{\|A\|}C$. The last step is to realise that the function space between two propositions is a proposition and apply truncation recursion. The resulting term of type $\Pi_{\|A\|}\Pi_{\|A\|}C$ then proves the conclusion.

It is a theorem that $\mathit{isProp}(\|A\|)$ implies that $\|A\|$ is a mere proposition. As is, the axiom only says that all elements are equal but not that all identity types are contractible.

Even if A is higher than -1 in the hierarchy, the proof that A implies a mere proposition B cannot depend on the extra structure but must be constant on even unrelated equality proofs. Hence, making all elements equal does not erase relevant information about the map.

Propositional truncation can be understood as a special case of the more general concept of higher inductive types. Higher inductive types are inductive types with constructors for both the type itself and its identity types. Propositional truncation not only introduces the elements $\|a\|$ but also proofs of the equalities $\mathsf{Id}_{\|A\|}[\|a\|/x_1, \|a'\|/x_2]$. In the general case, many constructors for an identity type can be defined, constructors of the identity types of identities can be defined, the identity constructors can depend on the type context or the element constructors. Other higher inductive types include propositional truncation with non-contractible identity types or truncation to higher levels in the hierarchy, for instance.

Further logical rules include

dependent sum type

$$\Gamma, A \text{ type}, B \text{ type} \vdash \Sigma AB \text{ type}$$

with introduction

$$\Gamma, a : A, b : B \vdash \langle a, b\rangle : \Sigma AB$$

and eliminations

$$\Gamma, p : \Sigma AB \vdash p_1 : A, p_2 : B[p_1/a]$$

inductive definitions empty type, void, natural numbers, vectors are axiomatised by their constructors and structural recursion principles

univalence the axioms says that the identity type of two small types is equivalent to the type of isomorphisms between the two small types

universe of propositions this universe is axiomatised with an additional axiom saying that the identity types of its small types are contractible

universes closed under type formers closure for universes allows for not eliminating of but also introducing types to the universe of small types like functions, products and sums

Some of the rules make use of capture-free substitution which can be generalised to context morphisms that replace all free variables simultaneously. Rather than concentrating on which variables to replace, we look at how the context in which the term can be typed changes. We are not interested on how the term changes syntactically but which are the free variables and how to assign a term to one. The abstract view characterises generalised substitutions as a category because they can be composed and the composition is both associative and has units. This is only a good abstraction because it subsumes capture-free substitution, weakening and the rule set can be expressed with at most one context morphism per rule.

Instead of generating substituted terms, context morphisms can be added as a fourth class of terms and further rules. The new rules axiomatise substitution as an operation adhering to the derived composition laws.

We take a leap and mention that variables can be replaced by a canonical context morphism $\mathbf{p}$ that removes the last variable from the context and a canonical term $\mathbf{q}$ to refer to the last variable in the context.

2.2 Models

The variable-free dependent type theory with context morphisms and definitional equality inherited from the meta-theory can be presented as a generalised algebraic theory.

A generalised algebraic theory (gat) consists of sort and operator symbols, introduction rules for the sorts and operators as well as equality axioms for the sorts and operators. [Car86] Operators define the elements of sorts. The introduction rules of both sorts and operators may depend on elements of other sorts. The induced equality relation is reflexive, symmetric, transitive and a congruence. What distinguishes generalised algebraic theories from dependent type theories is the lack of dependency and variables in formulas.

As already said, the theory we end up with by removing variables and adding explicit substitutions has a gat presentation. Contexts, context morphisms, types and terms are presented as the sorts and the operators are empty and extended context; composite, identity, projection ($\mathbf{p}$, into the empty context) and extension context morphism; application of context morphisms to types and terms (substitution) as well as the projection term ($\mathbf{q}$). Lastly, the equations characterise extended contexts as lists of types, context morphisms by their composition laws and substitution such that it respects the structure of context morphisms, that is composites correspond to successive substitutions and identities correspond to the do-nothing substitution.

The original theory is not a gat because the axiomatisation of well-formed terms needs to take variable bindings into consideration, f.i. given $[] \vdash A$ type

$$x : A \vdash y : A$$

is not well-formed while

$$x : A \vdash x : A$$

is, as well as capture-free substitution, f.i.

$$x : A, y : A \vdash (\lambda z.y)[x/y] = \lambda z.y[x/y]$$

is not well-formed unless z and x are distinct. Such cases are handled by side-conditions in the inference rule system but cannot be expressed in a gat directly. The switch to the variable-free formulation gives us a straightforward gat presentation (in every non-empty context $\mathbf{q}$ refers to the last variable in the context and under a binder this index gets incremented) but it is not clear whether the two presentations are equivalent.

The importance of the gat presentation for modelling type theory is that there is a mathematical notion of model that has a term model instance which is initial among all instances. Via the initiality of the term model an interpretation function of the syntax can be defined for an arbitrary instance in a uniform way. Moreover, a once and for all interpretation of variables and substitutions into combinators and context morphisms should be extendible to an interpretation function from the syntax of type theory into the instances of the gat model notion.

2.3 Categories with Families

Several equivalent notions of models of type theory as a gat exist. Categories with families [Dyb95][Hof97b] is a particular one that reformulates the gat axioms using categorical language. The operations on the collection of contexts turn it into a category with a terminal object, the equations for substitution characterise it as a functor from the category of contexts to the collection of families of sets, which together with re-indexing functions forms a category. The only equations of the gat that are not condensed into categorical language are the equations governing the combinators $\mathbf{p}$ and $\mathbf{q}$. We recall the complete definition of categories with families here.

Definition 2.3.1 (Category with families)**.** A category with families (cwf) consists of the following data.

- A category $\mathcal{C}tx$ with a terminal object $[] : \mathcal{C}tx$, that is
  - a set $Obj(\mathcal{C}tx)$ of objects (sometimes denoted as just $\mathcal{C}tx$)
  - a set $hom_{\mathcal{C}tx}(\Gamma, \Delta)$ of morphisms (sometimes written without the index and also referred to as $\Gamma \to_{\mathcal{C}tx} \Delta$ or $\Gamma \to \Delta$ respectively) for every pair of objects $\Gamma, \Delta \in Obj(\mathcal{C}tx)$
  - a member $[] \in Obj(\mathcal{C}tx)$
  - a member $!_\Gamma \in hom_{\mathcal{C}tx}(\Gamma, [])$ for every object $\Gamma \in Obj(\mathcal{C}tx)$
  - a member $\mathsf{id}_\Gamma \in hom_{\mathcal{C}tx}(\Gamma, \Gamma)$ for every object $\Gamma \in Obj(\mathcal{C}tx)$
  - a family of functions $\circ_{\mathcal{C}tx} : \{\Gamma, \Delta, \mathrm{E} \in Obj(\mathcal{C}tx)\}\, hom_{\mathcal{C}tx}(\Delta, \mathrm{E}) \to hom_{\mathcal{C}tx}(\Gamma, \Delta) \to hom_{\mathcal{C}tx}(\Gamma, \mathrm{E})$

  such that
  - $! \in hom(\Gamma, [])$ is unique for every object $\Gamma \in Obj(\mathcal{C}tx)$
  - $\mathsf{id}_\Delta \circ s \equiv s \equiv s \circ \mathsf{id}_\Gamma$ for all morphisms $s \in hom(\Gamma, \Delta)$
  - $u \circ (t \circ s) \equiv (u \circ t) \circ s$ for all morphisms $s \in hom(\Gamma, \Delta), t \in hom(\Delta, \mathrm{E}), u \in hom(\mathrm{E}, \Phi)$
- A functor $T : \mathcal{C}tx^{\mathsf{op}} \to \mathcal{F}am$, that is
  - a family of sets $\mathsf{Tm}_\Gamma(A)$ indexed by the members A of a set $\mathsf{Ty}(\Gamma)$ for every object $\Gamma \in Obj(\mathcal{C}tx)$
  - functions $\cdot\{s\} : \mathsf{Ty}(\Gamma) \to \mathsf{Ty}(\Delta)$ and $\cdot\{s\} : \mathsf{Tm}_\Gamma(A) \to \mathsf{Tm}_\Delta(A\{s\})$ for every morphism $s \in hom_{\mathcal{C}tx}(\Delta, \Gamma)$

  such that
  - $\cdot\{\mathsf{id}_\Gamma\}$ are the identity functions on $\mathsf{Ty}(\Gamma)$ and $\mathsf{Tm}_\Gamma(A)$ for every object $\Gamma \in Obj(\mathcal{C}tx)$
  - $\cdot\{s \circ t\}$ are the composites $\cdot\{t\} \circ \cdot\{s\}$ from $\mathsf{Ty}(\Gamma)$ and $\mathsf{Tm}_\Gamma(A)$ to $\mathsf{Ty}(\mathrm{E})$ and $\mathsf{Tm}_\mathrm{E}(A\{s\}\{t\})$ respectively for all morphisms $s \in hom_{\mathcal{C}tx}(\Delta, \Gamma), t \in hom_{\mathcal{C}tx}(\mathrm{E}, \Delta)$

- For every object $\Gamma \in Obj(\mathcal{C}tx)$ and element $A \in \mathsf{Ty}(\Gamma)$ an object $\Gamma.A \in Obj(\mathcal{C}tx)$ as well as a morphism $\mathbf{p}_{\Gamma.A} \in hom_{\mathcal{C}tx}(\Gamma.A, \Gamma)$ and an element $\mathbf{q}_{\Gamma.A} \in \mathsf{Tm}_{\Gamma.A}(A\{\mathbf{p}_{\Gamma.A}\})$

  such that

  for every morphism $s \in hom_{\mathcal{C}tx}(\Delta, \Gamma)$ and element $t \in \mathsf{Tm}_{\Delta}(A\{\sigma\})$ there is a *unique* morphism $\langle \sigma, t \rangle \in hom_{\mathcal{C}tx}(\Delta, \Gamma.A)$ satisfying the equations $\mathbf{p}_{\Gamma.A} \circ \langle \sigma, t \rangle \equiv \sigma$ and $\mathbf{q}_{\Gamma.A}\{\langle \sigma, t \rangle\} \equiv t$

■

The sets $\mathsf{Ty}(\Gamma)$ and $\mathsf{Tm}_{\Gamma}(A)$ correspond to the types A in context Γ and the terms of type A respectively. $\mathbf{p}$ and $\mathbf{q}$ behave like a first and second projection of $\Gamma.A$ respectively when we think of it as a Cartesian product of unnamed variables. Substitution corresponds to the operations $\cdot\{\cdot\}$ on the sets of types and terms. Categories with families are therefore close to the syntax of type theory, which will also show in the definition of type formers.

To get an idea of how to work with substitutions in cwfs, we define a shorthand for the common operation of replacing the last variable in the extended context $\Gamma.A$ with a term of type A.

Definition 2.3.2 (Variable elimination as context morphism)**.** Given a cwf, a context $\Gamma : \mathcal{C}tx$, a type $A \in \mathsf{Ty}(\Gamma)$ and a term $a \in \mathsf{Tm}_{\Gamma}(A)$, write $[a]$ for the substitution $\langle \mathsf{id}_{\Gamma}, a \rangle : \Gamma \to \Gamma.A$. ■

Every cwf is a model of the structural rules of a dependent type theory. The subsequent definitions say what structure is needed on a particular cwf such that it also models the logical rules Π, Id, U and $\|\cdot\|$.

Definition 2.3.3 (Π structure)**.** For every $\Gamma : \mathcal{C}tx$, $A \in \mathsf{Ty}(\Gamma)$ and $B \in \mathsf{Ty}(\Gamma.A)$ there is a $\Pi AB \in \mathsf{Ty}(\Gamma)$.
For every $b \in \mathsf{Tm}_{\Gamma.A}(B)$ there is $\lambda b \in \mathsf{Tm}_{\Gamma}(\Pi AB)$.
For every $\lambda \in \mathsf{Tm}_{\Gamma}(\Pi AB)$ and $a \in \mathsf{Tm}_{\Gamma}(A)$ there is $\mathsf{app}(\lambda, a) \in \mathsf{Tm}_{\Gamma}(B[a])$.
We have $\lambda\mathsf{app}(\lambda\mathbf{p}, \mathbf{q}) \equiv \lambda$ and $\mathsf{app}(\lambda b, a) \equiv b[a]$.
For every $\sigma : \Delta \to \Gamma$ we have

- $\Pi_A B\sigma \equiv \Pi_{A\sigma}(B\langle \sigma \circ \mathbf{p}, \mathbf{q} \rangle)$
- $\lambda b\sigma \equiv \lambda(b\langle \sigma \circ \mathbf{p}, \mathbf{q} \rangle)$
- $\mathsf{app}(\lambda, a)\sigma \equiv \mathsf{app}(\lambda\sigma, a\sigma)$

■

Definition 2.3.4 (Id structure)**.** For every $\Gamma : \mathcal{C}tx$ and $A \in \mathsf{Ty}(\Gamma)$ there is a $\mathsf{Id}_A \in \mathsf{Ty}(\Gamma.A.A\mathbf{p})$.
There is a $\mathsf{refl}_A \in \mathsf{Tm}_{\Gamma.A}(\mathsf{Id}_A[\mathbf{q}])$.
For every $C \in \mathsf{Ty}(\Gamma.A.A\mathbf{p}.\mathsf{Id}_A)$ and $d \in \mathsf{Tm}_{\Gamma.A}(C[\mathsf{refl}_A])$ there is $J_{C,d} \in \mathsf{Tm}_{\Gamma.A.A\mathbf{p}.\mathsf{Id}_A}(C)$.
We have $J_{C,d}[\mathsf{refl}_A\mathbf{p}] \equiv d$.
For every $\sigma : \Delta \to \Gamma$ we have

- $\mathsf{Id}_A\langle\langle \sigma \circ \mathbf{p}, \mathbf{q} \rangle \circ \mathbf{p}, \mathbf{q} \rangle \equiv \mathsf{Id}_{A\sigma}$
- $\mathsf{refl}_A\langle \sigma \circ \mathbf{p}, \mathbf{q} \rangle \equiv \mathsf{refl}_{A\sigma}$
- $J_{C,d}\langle\langle\langle \sigma \circ \mathbf{p}, \mathbf{q} \rangle \circ \mathbf{p}, \mathbf{q} \rangle \circ \mathbf{p}, \mathbf{q} \rangle \equiv J_{C\langle\langle\langle \sigma\circ\mathbf{p}, \mathbf{q}\rangle\circ\mathbf{p}, \mathbf{q}\rangle\circ\mathbf{p}, \mathbf{q}\rangle, d\langle \sigma\circ\mathbf{p}, \mathbf{q}\rangle}$

■

Definition 2.3.5 (U structure)**.** There is $\mathsf{U} \in \mathsf{Ty}([])$.
For every $A \in \mathsf{Tm}_{[]}(\mathsf{U})$ there is $|A| \in \mathsf{Ty}([])$. ■

Definition 2.3.6 ($\|\cdot\|$ structure)**.** For every $\Gamma : \mathcal{C}tx$ and $A \in \mathsf{Ty}(\Gamma)$ there is a $\|A\| \in \mathsf{Ty}(\Gamma)$.
For every $a \in \mathsf{Tm}_{\Gamma}(A)$ there is $\|a\| \in \mathsf{Tm}_{\Gamma}(\|A\|)$.
There is $prop(A) \in \mathsf{Tm}_{\Gamma}(\Pi_{\|A\|}\Pi_{\|A\|\mathbf{p}}\mathsf{Id}_{\|A\|})$.
For every $B \in \mathsf{Ty}(\Gamma)$, $prop(B) \in \mathsf{Tm}_{\Gamma}(\Pi_B\Pi_{B\mathbf{p}}\mathsf{Id}_B)$ and $f \in \mathsf{Tm}_{\Gamma}(\Pi A(B\mathbf{p}))$ there is $\|f\| \in \mathsf{Tm}_{\Gamma}(\Pi\|A\|(B\mathbf{p}))$.
We have $\mathsf{app}(\|f\|, \|a\|) \equiv \mathsf{app}(f, a)$.
For every $\sigma : \Delta \to \Gamma$ we have

- $\|A\|\sigma \equiv \|A\sigma\|$
- $\|a\|\sigma \equiv \|a\sigma\|$

- $\|f\|\sigma \equiv \|f\sigma\|$

■

This presentation of the logical cwf structure is based on [Hof97b]. In general, the cwf structure corresponding to a given type can be derived systematically from the inference rules of a dependent type theory, f.i. the one presented in section 2.1. Particular attention only needs to be paid to stability under substitution.

Chapter 3

Relational Model

This chapter presents the general relational model of dependent type theory. It is not the most general notion of relational model since the involved relations are all binary relations. Still, it is general enough to have the relational models mentioned in the introduction as refinements.

The general relational model can also be seen as a first example of a cwf and what a proof of the cwf axioms usually consists of. Hence, the basic structure of contexts, types and terms is the only part of the theory that we are concerned with here. In particular, we will not look at the interpretation of any logical rules like function spaces for the general relational model.

Let's begin with the interpretation of contexts.

Definition 3.0.1 (Relational contexts)**.** Let the contexts be sets with proof-relevant binary relations defined on them, that is tuples

$$\langle \Gamma_S : \mathcal{S}et, \Gamma_R : \Gamma_S \times \Gamma_S \to \mathcal{S}et \rangle.$$

Here, $\mathcal{S}et$ denotes the class of all sets and the type of Γ_R should be understood as the type of a mapping or family that assigns a set to every pair of members of the set Γ_S.

A morphism between two contexts Δ and Γ is defined to be a pair of a function

$$\sigma_0 : \Delta_S \to \Gamma_S$$

and a family of functions

$$\sigma_1 : \{\delta, \delta' \in \Delta_S\} \to \Delta_R(\delta, \delta') \to \Gamma_R(\sigma_0(\delta), \sigma_0(\delta')).$$

The type of σ_1 says that it denotes a family of functions which is indexed by the set comprehension in curly braces. We will often omit the indices and apply σ_1 directly as a function if the indices can be inferred from its argument.

Lastly, we pick out a particular context as interpretation for the empty context. Define the empty context as the singleton set $\{\bullet\}$ with the total relation $\langle \bullet, \bullet \rangle \mapsto \{\bullet\}$. The bullet point $\bullet$ is some arbitrary but fixed object, for instance the empty set $\emptyset$.

■

The tuples of sets and relations together with the component-wise mappings form a category (cf. lemma A.0.1) with the empty context as the terminal object (cf. lemma A.0.3), the category of contexts.

The proof-irrelevant binary relations are subsumed by this definition. One possible way of embedding the proof-irrelevant relations is by restriction to singletons in the domain of Γ_R. It is also possible to restrict ourselves to transitive relations by assuming a family of functions

$$trans : \{\gamma, \gamma', \gamma'' \in \Gamma_S\} \to \Gamma_R(\gamma, \gamma') \times \Gamma_R(\gamma', \gamma'') \to \Gamma_R(\gamma, \gamma'').$$

Similarly, the non-emptiness of specific sets can force the context relations to be reflexive

$$refl : \{\gamma \in \Gamma_S\} \to \Gamma_R(\gamma, \gamma)$$

or symmetric

$$sym : \{\gamma, \gamma' \in \Gamma_S\} \to \Gamma_R(\gamma, \gamma') \to \Gamma_R(\gamma', \gamma).$$

We continue with the interpretation of types.

Definition 3.0.2 (Relational types)**.** Let a type A in a context Γ be a pair of a family of sets

$$(A_\gamma : \mathcal{S}et)_{\gamma \in \Gamma_S}$$

indexed by the members of Γ_S and a family of relations

$$(A_g : A_\gamma \times A_{\gamma'} \to \mathcal{S}et)_{\gamma,\gamma' \in \Gamma_S, g \in \Gamma_R(\gamma,\gamma')}$$

indexed by the proofs of Γ_R.

■

The relations A_g are very general. As for contexts, transitivity can also be imposed on types by assuming a family of functions with the following signature.

$$\begin{aligned} trans : &\{\gamma, \gamma', \gamma'' \in \Gamma_S\}\{g \in \Gamma_R(\gamma,\gamma')\}\{g' \in \Gamma_R(\gamma',\gamma'')\}\{a_\gamma \in A_\gamma\}\{a_{\gamma'} \in A_{\gamma'}\}\{a_{\gamma''} \in A_{\gamma''}\} \\ &\to A_g(a_\gamma, a_{\gamma'}) \times A_{g'}(a_{\gamma'}, a_{\gamma''}) \\ &\to A_{g \cdot_\Gamma g'}(a_\gamma, a_{\gamma''}) \end{aligned}$$

The restriction to reflexive and symmetric types follows by similar generalisations of the respective restrictions on contexts.

In fact, the only feature that distinguishes types from contexts is the dependency. We remark that closed types are in 1-to-1 correspondence with contexts.

Remark 3.0.3 (Closed types are contexts)**.** A closed type is a type over the empty context, which has exactly one member and one proof that the unique member is related to itself. Hence, there is exactly one set and one relation associated with a closed semantic type.

Given a semantic context, its set and relation can be regarded as a family over the singleton set that is the empty context. This construction turns a context into a type in the empty context.

Next, we interpret terms in the general relational model.

Definition 3.0.4 (Relational terms)**.** Let a term a of type A in a context Γ be a context-indexed family of elements

$$(a_\gamma \in A_\gamma)_{\gamma \in \Gamma_S}$$

and proofs

$$(a_g \in A_g(a_\gamma, a_{\gamma'}))_{\gamma,\gamma' \in \Gamma_S, g \in \Gamma_R(\gamma,\gamma')}.$$

■

In other words, terms choose for every context member $\gamma \in \Gamma$ a member of the respective type subset A_γ such that the choices are related whenever the context members are related.

An interpretation of contexts, types and terms is not a model without a semantic substitution operation, that is a way to apply context morphisms to types and terms in the semantics. This is necessary because the definitions of types and terms are given with respect to a fixed context while dependent type theory allows context changes that retain the meaning of types and terms.

Definition 3.0.5 (Relational substitution)**.** Given a type A and a term $a : A$ in a context Γ, then applying a substitution $\sigma : \Delta \to \Gamma$ yields the type

$$\langle (A_{\sigma(\delta)})_{\delta \in \Delta_S}, (\langle a, a' \rangle \mapsto A_{\sigma(d)}(a, a'))_{\delta,\delta' \in \Delta_S, d \in \Delta_R(\delta,\delta')} \rangle$$

and the term

$$\langle (a_{\sigma(\delta)})_{\delta \in \Delta_S}, (a_{\sigma(d)})_{\delta,\delta' \in \Delta_S, d \in \Delta_R(\delta,\delta')} \rangle.$$

These are well-defined because σ preserves relatedness. Since $\sigma(d)$ is a witness that $\sigma(\delta)$ and $\sigma(\delta')$ are related, $A_{\sigma(d)}$ indeed denotes a family of sets and $a_{\sigma(d)}$ proves that $a_{\sigma(\delta)}$ is related to $a_{\sigma(\delta')}$.

■

Substitutions must be functorial with respect to composition and identity in the context category. Both facts are proved in A.0.2.

We indicated above how to axiomatise common properties of relations for contexts and types. In order to obtain closure of reflexive, symmetric or transitive relations under type substitution, the context morphisms need to preserve the respective proofs.

$$\begin{aligned}\sigma(refl(\delta)) &\equiv refl(\sigma(\delta))\\ \sigma(sym(d)) &\equiv sym(\sigma(d))\\ \sigma(trans(d,d')) &\equiv trans(\sigma(d),\sigma(d'))\end{aligned}$$

In the case of reflexivity for instance, a priori $A_{\sigma(refl(\delta))}(a,a)$ will only be inhabited if $\sigma(refl(\delta))$ is the reflexivity of some member of Γ_S. Similar assumptions must be made when proving symmetry and transitivity for the substituted type $A\{\sigma\}$.

The interpretation of context extension by types is straightforward in the relational model because the type relations are indexed over the context relations.

Definition 3.0.6 (Relational context comprehension)**.** The extension of a context Γ by a type A in context Γ is the set of tuples

$$\{\langle\gamma, a_\gamma\rangle | \gamma \in \Gamma_S, a_\gamma \in A_\gamma\}$$

together with the family of relations

$$\langle\langle\gamma, a_\gamma\rangle, \langle\gamma', a_{\gamma'}\rangle\rangle \mapsto \{\langle g, \alpha\rangle | g \in \Gamma_R(\gamma,\gamma'), \alpha \in A_g(a_\gamma, a_{\gamma'})\}$$

Moreover, the two projections are defined by projecting the tuples to Γ and A respectively.

- $\mathbf{p}_{\Gamma.A} := \langle\langle\gamma, a_\gamma\rangle \mapsto \gamma, \langle g,\alpha\rangle \mapsto g\rangle : \Gamma.A \to \Gamma$
- $\mathbf{q}_{\Gamma.A} := \langle (a_\gamma)_{\langle\gamma,a_\gamma\rangle}, (\alpha_g)_{\langle g,\alpha_g\rangle}\rangle \in \mathsf{Tm}_{\Gamma.A}(A\{\mathbf{p}\Gamma.A\})$

This is well-defined because $(A\{\mathbf{p}_{\Gamma.A}\})_{\langle\gamma,a_\gamma\rangle} \equiv A_{\mathbf{p}_{\Gamma.A}(\gamma,a_\gamma)} \equiv A_\gamma$ by definition of substitution and $\mathbf{p}_{\Gamma.A}$. Similar for the relation part.

$\mathbf{q}_{\Gamma.A}$ is not exactly the second projection because the domain A depends on the first component and, hence, is not closed but open.

Given a substitution $\sigma : \Delta \to \Gamma$, a type $A \in \mathsf{Ty}(\Gamma)$ and a term $s \in \mathsf{Tm}_\Delta(A\{\sigma\})$, we define the lifted substitution into the extended context by

$$\langle\sigma, s\rangle := \langle\delta \mapsto \langle\sigma(\delta), s_\delta\rangle, d \mapsto \langle\sigma(d), s_d\rangle\rangle : \Delta \to \Gamma.A,$$

This is well-defined because $(A\{\sigma\})_\delta \equiv A_{\sigma(\delta)}$.

■

Lemma A.0.4 proves the closure property of relational contexts under comprehension, which involves the fact that the definition of $\langle\sigma, s\rangle$ is the unique solution to $\mathbf{p}_{\Gamma.A} \circ \langle\sigma, s\rangle \equiv \sigma$ and $\mathbf{q}_{\Gamma.A}\{\langle\sigma, s\rangle\} \equiv s$.

Not only correspond closed types to contexts but also terms are represented in the category of contexts. Specifically, terms are in 1-to-1 correspondence with sections of the projections $\mathbf{p}$.

Remark 3.0.7 (Terms are sections of the extended context)**.** Every semantic term $t \in \mathsf{Tm}_\Gamma(A)$ defines a context morphism

$$\overline{t} := \begin{cases}\gamma & \mapsto \langle\gamma, t_\gamma\rangle\\ g & \mapsto \langle g, t_g\rangle\end{cases} : \Gamma \to \Gamma.A$$

such that

$$(\mathbf{p}_{\Gamma.A} \circ t)(\gamma) \equiv \gamma \quad \text{and} \quad (\mathbf{p}_{\Gamma.A} \circ t)(g) \equiv g.$$

Conversely, every section $f : \Gamma \to \Gamma.A$ of $\mathbf{p}_{\Gamma.A}$ defines a term $\overline{f} \in \mathsf{Tm}_\Gamma(A)$ because $\mathbf{p}_{\Gamma.A} \circ f \equiv \mathsf{id}_\Gamma$ implies

$$\pi_2(f(\gamma)) \in A_\gamma \quad \text{and} \quad \pi_2(f(g)) \in A_g(\pi_2(f(\gamma)), \pi_2(f(\gamma'))).$$

Lastly, applying the construction twice yields again the same term and context morphism, respectively.

Consequently, we can understand the context morphisms into Γ as the representations of the types depending on Γ. Thinking along these lines, the earlier remark about the correspondence between contexts and closed types follows immediately because from every context there is a context morphism into the empty context, which is terminal.

We end this section with the theorem stating that the relational interpretation is a model in the sense of cwfs.

Theorem 3.0.8 (Relational interpretation models dependent type theory)**.** The relational interpretation forms a cwf.

Proof. By lemmata A.0.1 through A.0.4.

□

In fact, the relational interpretation also models dependent products, dependent sums, intensional identity types, the Booleans, finite sets and the natural numbers. The result is proved in Martin-Löf type theory with inductive definitions by Tonelli [Ton13].

Chapter 4

Path Model

The model that we present in this chapter is a reformulation of the groupoid interpretation [HS98] of intensional type theory as a relational interpretation as characterised in chapter 3. There we already made some observations about how to define reflexivity, symmetry and transitivity for relational types. As we know from the setoid interpretation [Hof97a], we also need transport operations

$$\cdot^{+} : \{\gamma, \gamma' \in \Gamma_S\}\{g \in \Gamma_R(\gamma, \gamma')\} \to A_\gamma \to A_{\gamma'}$$
$$\cdot^{\uparrow} : \{\gamma, \gamma' \in \Gamma_S\}\{g \in \Gamma_R(\gamma, \gamma')\} \to (a \in A_\gamma) \to A_g(a, a^{+})$$

in order to interpret intensional identity types by the type relations A_g for all $g \in \Gamma_R$. Imposing additional coherence conditions on a proof-relevant equivalence relation turns the subsets A_γ into groupoids. From the groupoid interpretation we know that such types refute uniqueness of identity proofs and support universe extensionality. In the resulting interpretation not only the proofs of A_{id_γ} can be concatenated and inverted but all proofs, that is over any element of Γ_R. This is reminiscent of the notion of path in topology; hence the name of the model.

Before we continue with the verification of the path model, let us observe that for every type in the groupoid interpretation there is a path structure that coincides with the groupoid structure on the subsets A_γ.

Recall that a type in the groupoid interpretation is a functor A from a groupoid Γ to the category $\mathcal{GPD}$ of groupoids and functors. Then define a family of relations that relates two elements $a : A(\gamma), a' : A(\gamma')$ over $g : \gamma \to_\Gamma \gamma'$ if and only if there is morphism from $A(g)(a)$ to a' in $A(\gamma')$. More precisely, let

$$A_R \coloneqq (hom_{A(\gamma')}(A(g)(a), a'))_{\gamma, \gamma' : \Gamma, g : \gamma \to_\Gamma \gamma', a : A(\gamma), a' : A(\gamma')}.$$

Relating elements by transportation to $A(\gamma)$, that is

$$A^*_R \coloneqq (hom_{A(\gamma)}(a, A(g^{-1})(a')))_{\gamma, \gamma' : \Gamma, g : \gamma \to_\Gamma \gamma', a : A(\gamma), a' : A(\gamma')},$$

is also a natural choice of relation on the disjoint union of the sets $Obj(A(\gamma))$ for every $\gamma : \Gamma$. We remark that both relations are logically equivalent.

Remark 4.0.1 (Equivalent definitions of the equivalence relation on groupoid types)**.** For all $\gamma, \gamma' : \Gamma$, $g : \gamma \to_\Gamma \gamma'$ and $a : A(\gamma), a' : A(\gamma')$ it holds that $A_R(a, a')$ is non-empty if and only if $A^*_R(a, a')$ is non-empty. In fact, the restriction of the morphism part of $A(g) : A(\gamma) \to_{\mathcal{GPD}} A(\gamma')$ is a bijection between $A^*_R(g, a, a')$ and $A_R(g, a, a')$.

The next lemma states that the relation A_R is an equivalence relation.

Lemma 4.0.2 (The relation on groupoid types is an equivalence)**.** There are families of functions

$$\begin{aligned}
&refl : \{\gamma : \Gamma\} \to (a : A(\gamma)) \to A_R(\mathsf{id}_\gamma, a, a)\\
&sym : \{\gamma, \gamma' : \Gamma\}\{g : \gamma \to_\Gamma \gamma'\}\{a : A(\gamma)\}\{a' : A(\gamma')\} \to A_R(g, a, a') \to A_R(g^{-1}, a', a)\\
&trans : \{\gamma, \gamma', \gamma'' : \Gamma\}\{g : \gamma \to \Gamma\gamma'\}\{g' : \gamma' \to_\Gamma \gamma''\}\{a : A(\gamma)\}\{a' : A(\gamma')\}\{a'' : A(\gamma'')\}\\
&\qquad \to A_R(g, a, a') \times A_R(g', a', a'')\\
&\qquad \to A_R(g \cdot_\Gamma g', a, a'')
\end{aligned}$$

Proof. Since $A(g)$ and $A(g^{-1})$ are functors and Γ is a groupoid the following membership relations and equations hold

$$\begin{aligned}
\mathsf{id}_a &\in hom_{A(\gamma)}(a,a)\\
A(g^{-1})(\alpha^{-1}) &\in hom_{A(\gamma)}(A(g^{-1})(a'), A(g^{-1})(A(g)(a))) \equiv hom_{A(\gamma)}(A(g^{-1})(a'), a)\\
A(g')(\alpha) \cdot_{A(\gamma'')} \alpha' &\in hom_{A(\gamma'')}(A(g')(A(g)(a)), a'') \equiv hom_{A(\gamma'')}(A(g \cdot_\Gamma g')(a), a'')
\end{aligned}$$

Therefore, the following definitions are well-typed

$$\begin{aligned}
refl\ \{\gamma\}\ a &\coloneqq \mathsf{id}_a\\
sym\ \{\gamma,\gamma'\}\ \{g\}\ \{a\}\ \{a'\}\ \alpha &\coloneqq A(g^{-1})(\alpha^{-1})\\
trans\ \{\gamma,\gamma',\gamma''\}\ \{g\}\ \{g'\}\ \{a\}\ \{a'\}\ \{a''\}\ \alpha\ \alpha' &\coloneqq A(g')(\alpha) \cdot_{A(\gamma'')} \alpha'.
\end{aligned}$$

□

Besides being witnesses that A_R is an equivalence relation, the terms $refl$, sym and $trans$ along with the object parts of the functors $A(g : hom\,\Gamma)$ as transports

$$\begin{aligned}
a^+\ \{\gamma,\gamma'\}\ \{g\} &\coloneqq A(g)(a)\\
a^\uparrow\ \{\gamma,\gamma'\}\ \{g\} &\coloneqq \mathsf{id}_{A(g)(a)} \in hom_{A(\gamma')}(A(g)(a), A(g)(a)) \equiv A_R(g,a,a^+)
\end{aligned}$$

exhibit a path structure on A_R. The straightforward verifications can be found in the appendix (lemmata A.0.5 through A.0.8). For the relation associated with a closed type these observations simply correspond to the groupoid axioms but for the relations over non-identities $g \in hom\,\Gamma$ the functoriality of $A(g)$ is crucial.

Lastly, observe that the path structure A_R of a closed type A in the groupoid interpretation coincides with the homsets of the groupoid A since $A(\mathsf{id}_\gamma) \equiv \mathsf{Id}_{A(\gamma)}$ and, hence,

$$\begin{aligned}
A_R(\mathsf{id}_\gamma, a, a') &\equiv hom_{A(\gamma)}(A(\mathsf{id}_\gamma)(a), a') \equiv hom_{A(\gamma)}(a, a')\\
refl(a) &\equiv \mathsf{id}_a\\
sym(\alpha_\gamma) &\equiv A({\mathsf{id}_\gamma}^{-1})({\alpha_\gamma}^{-1}) \equiv {\alpha_\gamma}^{-1}\\
trans(\alpha_\gamma, \alpha'_\gamma) &\equiv A(\mathsf{id}_\gamma)(\alpha_\gamma) \cdot_{A(\gamma)} \alpha'_\gamma \equiv \alpha_\gamma \cdot_{A(\gamma)} \alpha'_\gamma
\end{aligned}$$

The fact that we were able to define a path structure on an arbitrary groupoid type shows that the set of path types will include at least the groupoid types. Whether this is a strict inclusion and whether the cwf structure is derivable from just the path structure axioms remains to be answered.

The following sections give the complete definition of path types and show that they model Martin-Löf type theory with function and universe extensionality as well as propositional truncation, or squashing as it is called in [Hof97a].

4.1 Category with Families

We begin with the interpretation of contexts in the path model as a refinement of the general relational model (cf. chapter 3).

Definition 4.1.1 (Path contexts)**.** A context Γ in the path model is a relational context (cf. definition 3.0.1) with the additional operations

- for every $\gamma \in \Gamma_S$ an element $\mathsf{id}_\gamma \in \Gamma_R(\gamma,\gamma)$ (reflexivity)
- for every $\gamma,\gamma' \in \Gamma_S$ and $g \in \Gamma_R(\gamma,\gamma')$ an element $g^{-1} \in \Gamma_R(\gamma',\gamma)$ (symmetry)
- for every $\gamma,\gamma',\gamma''\Gamma_S$ and $g \in \Gamma_R(\gamma,\gamma'), g' \in \Gamma_R(\gamma',\gamma'')$ an element $g \cdot g' \in \Gamma_R(\gamma,\gamma'')$ (transitivity)

for which the following equations must hold

- $g \cdot (g' \cdot g'') \equiv (g \cdot g') \cdot g''$ (associativity of $\cdot$)

- $\mathsf{id}_\gamma \cdot g \equiv g$ and $g \cdot \mathsf{id}_{\gamma'} \equiv g$ (unit laws for $\cdot$)
- $g^{-1} \cdot g \equiv \mathsf{id}_{\gamma'}$ and $g \cdot g^{-1} \equiv \mathsf{id}_\gamma$ (inverse laws for $\cdot$)
- $f_1(\mathsf{id}_\gamma) \equiv \mathsf{id}_{f_0(\gamma)}$, $f_1(g \cdot g') \equiv f_1(g) \cdot f_1(g')$ (functoriality of f)

■

The last equation says that context morphisms must commute with the context operations. Commutation with the inverse operation is implied by commutation with the other two operations.

Lemma 4.1.2 (Path context morphisms preserve inverses)**.** If a context morphism $f : \Delta \to \Gamma$ preserves path concatenation and identity paths, then it preserves inverses.

The path context axioms are exactly those of groupoids and functors between groupoids. We obtain the full subcategory $\mathcal{GPD}$ of $\mathcal{C}at$.

Since each context morphism preserves the context structure, so do their composites. The identity function and the morphisms into the singleton set trivially preserve the context structure. See lemma A.0.9 for the complete proof.

We continue with the interpretation of type in the path model. Path types are both refined relational types and setoid types.

Definition 4.1.3 (Path types)**.** A path type A in context Γ is relational type (cf. definition 3.0.2) with the additional operations

- for every $\gamma \in \Gamma$ and $a \in A_\gamma$ an element $\mathsf{id}_a \in A_{\mathsf{id}_\gamma}(a, a)$ (reflexivity)
- for every $\gamma, \gamma' \in \Gamma$, $g \in R_\Gamma(\gamma, \gamma')$, $a \in A_\gamma$, $a' \in A_{\gamma'}$ and $\alpha \in A_g(a, a')$ an element $\alpha^{-1} \in A_{g^{-1}}(a', a')$ (symmetry)
- for every $\gamma, \gamma', \gamma'' \in \Gamma$, $g \in R_\Gamma(\gamma, \gamma')$, $g' \in R_\Gamma(\gamma', \gamma'')$, $a \in A_\gamma$, $a' \in A_{\gamma'}$, $a'' \in A_{\gamma''}$, $\alpha \in A_g(a, a')$ and $\alpha' \in A_{g'}(a', a'')$ an element $g \cdot g' A_{g \cdot g'}(a, a'')$ (transitivity)
- for every $\gamma, \gamma' \in \Gamma$, $g \in R_\Gamma(\gamma, \gamma')$, $a \in A_\gamma$ elements $a^+_g \in A_{\gamma'}$ and $a^\uparrow_g \in A_g(a, a^+)$.(transport)

Up to now, the type axioms coincide with those in the setoid model [Hof97a]. However, path types must also satisfy the following coherence conditions

- $\alpha \cdot (\alpha' \cdot \alpha'') \equiv (\alpha \cdot \alpha') \cdot \alpha''$ (associativity of $\cdot$)
- $\mathsf{id}_a \cdot \alpha \equiv \alpha$ and $\alpha \cdot \mathsf{id}_{a'} \equiv \alpha$ (unit laws for $\cdot$)
- $\alpha^{-1} \cdot \alpha \equiv \mathsf{id}_{a'}$ and $\alpha \cdot \alpha^{-1} \equiv \mathsf{id}_a$ (inverse laws for $\cdot$)
- $a^+_{\mathsf{id}_\gamma} \equiv a$ and $a^\uparrow_{\mathsf{id}_\gamma} \equiv \mathsf{id}_a$ (unit preservation law for transport)
- $a^+_{g \cdot g'} \equiv a^{+^+}_{g\ g'}$ and $a^\uparrow_{g \cdot g'} \equiv a^\uparrow_g \cdot a^{+^\uparrow}_{g'}$ (concatenation preservation law for transport)

■

Since we require contexts to be groupoids, the 1-to-1 correspondence between types and contexts we proved for the general model still holds in the path model.

Lemma 4.1.4 (Closed path types are contexts)**.** Every context is a path type in the empty context and for every $\gamma \in \Gamma_S$ we have that $\langle A_\gamma, A_{\mathsf{id}_\gamma} \rangle$ is a context.

Proof. $\langle A_\gamma, A_{\mathsf{id}_\gamma} \rangle$ is a path context because it contains the id_a proofs, is closed under inverses (${\mathsf{id}_\gamma}^{-1} \equiv \mathsf{id}_\gamma$) and composites ($\mathsf{id}_\gamma \cdot \mathsf{id}_\gamma \equiv \mathsf{id}_\gamma$).

A path context Γ is a path type in the empty context because the empty context contains only the identity path at the unique element $\bullet \in []_S$.

□

Now that contexts have some extra structure we can also prove some facts about open types.

Lemma 4.1.5 (Open path types are functorial)**.** For every $g \in \Gamma_R(\gamma, \gamma'), g' \in \Gamma_R(\gamma', \gamma'')$ in the context we have $A_{g \cdot g'} \equiv A_g \cdot A_{g'}$ and $A_{g^{-1}} \equiv {A_g}^{-1}$ but *not* $A_{\mathsf{id}_\gamma} \equiv \mathsf{id}(A_\gamma)$.

Here, the operations on the sets A_g are to be understood as being applied member-wise.

Proof. The inclusion of the right-hand side of $A_{g \cdot g'} \equiv A_g \cdot A_{g'}$ holds by the type of path composition. The opposite direction follows from transport, which connects every element of A_γ to an element in $A_{\gamma'}$ over $g \in \Gamma_R(\gamma, \gamma')$. Let $\alpha \in A_{g \cdot g'}$, then

$$\alpha \equiv {a_\gamma}^{\uparrow}_{g} \cdot ({a_\gamma}^{\uparrow}_{g}{}^{-1} \cdot \alpha) \in A_g \cdot A_{g'}.$$

Both inclusions of $A_{g^{-1}} \equiv {A_g}^{-1}$ hold by the definition of path inverses.

$$A_{g^{-1}} \subseteq {A_{g^{-1}}}^{-1^{-1}} \subseteq {A_g}^{-1} \subseteq A_{g^{-1}}$$

Lastly, A_{id_γ} may contain paths besides the reflexivity paths. Thus, $A_{\mathsf{id}_\gamma} \not\equiv \mathsf{id}(A_\gamma)$.

□

Because of the equalities $A_{g \cdot g'} \equiv A_g \cdot A_{g'}$ the sets A_{id_γ} are units of the lifted concatenation. In that sense, a path type A defines a mapping that preserves composition and identities. However, the lemma also says that the co-domain of that mapping cannot be $\{A_\gamma | \gamma \in \Gamma_S\}$ with the canonical identities $\mathsf{id}(A_\gamma)$. The co-domain should be definable independent of the mapping and, hence, its objects must contain more information than their members. Otherwise, it is not clear what the identity morphism at each object should be.

Moving on to the interpretation of terms in the path model.

Definition 4.1.6 (Path terms)**.** A path term is a relational term (cf. definition 3.0.4) that preserves the extra structure of path contexts, that is

- $a_{\mathsf{id}_\gamma} \equiv \mathsf{id}_{a_\gamma}$
- $a_{g \cdot g'} \equiv a_g \cdot a_{g'}$
- $a_{g^{-1}} \equiv {a_g}^{-1}$

■

As usual, the third equation (preservation of inverses) is derivable for every path term that satisfies the first two.

We now define the substitution operation on path terms and types. It is the same re-indexing as in the general case (cf. definition 3.0.5) but formally the path operations have to be redefined as well.

Definition 4.1.7 (Path substitution)**.** For every $s : \Delta \to \Gamma$ in the category of path contexts, path type $A \in \mathsf{Ty}(\Gamma)$ and path term $t \in \mathsf{Tm}_\Gamma(A)$ define the type $A\{s\} \in \mathsf{Ty}(\Delta)$

$$\begin{aligned}
A\{s\}_\delta &:= A_{s(\delta)} \\
A\{s\}_d(a, a') &:= \{\gamma := s(\delta), \gamma' := s(\delta')\}\ A_{s(d)}(a, a') \\
\mathsf{id}_a &:= \{\gamma := s(\delta)\}\ \mathsf{id}_a \\
\alpha^{-1} &:= \{\gamma := s(\delta), \gamma' := s(\delta'), g := s(d), a := a, a' := a'\}\ \alpha^{-1} \\
\alpha \cdot \alpha' &:= \{\gamma := s(\delta), \gamma' := s(\delta'), \gamma'' := s(\delta''), g := s(d), g' := s(d'), a := a, a' := a', a'' := a''\}\ \alpha \cdot \alpha'
\end{aligned}$$

and term $t\{s\} \in \mathsf{Tm}_\Delta(A\{s\})$

$$\begin{aligned}
t\{s\}_\delta &:= t_{s(\delta)} \\
t\{s\}_d &:= \{\gamma := s(\delta), \gamma' := s(\delta')\}\ t_{s(d)},
\end{aligned}$$

for all $\delta, \delta', \delta'' \in \Delta$, $d \in \Delta(\delta, \delta')$, $d' \in \Delta(\delta', \delta'')$, $a \in A\{s\}_\delta$, $a' \in A\{s\}_{\delta'}$, $a'' \in A\{s\}_{\delta''}$, $\alpha \in A\{s\}_d(a, a')$ and $\alpha' \in A\{s\}_{d'}(a', a'')$.

The associativity, unit, inverse and preservation laws follow directly from A and t and the fact that s preserves the context structure. We conclude that indeed $A\{s\} \in \mathsf{Ty}(\Delta)$ and $t\{s\} \in \mathsf{Tm}_\Delta(A\{s\})$.

■

It is crucial for the cwf notion to work that the application of several substitutions can be merged into the application of a single substitution while maintaining the result. Definition 2.3.1 formalises this property by the functor laws and the next lemma verifies those for the path interpretation.

Lemma 4.1.8 (Path substitution is a functor)**.** The mapping of categories

$$\begin{cases} \Gamma & \mapsto (\mathsf{Tm}_\Gamma(A))_{A \in \mathsf{Ty}(\Gamma)} \\ s & \mapsto \langle A \mapsto A\{s\}, t \mapsto t\{s\} \rangle \end{cases}$$

from path contexts to path terms indexed by path types is a functor.

A proof can be found in the appendix (lemma A.0.11).

Before we conclude with the theorem that the path interpretation forms a cwf we must adapt context comprehension to path contexts.

Definition 4.1.9 (Path context comprehension)**.** Given a path context Γ and a path type A in context Γ. Define $\Gamma.A$ as in definition 3.0.6 plus the identities, inverses and composites as follows.

- $\mathsf{id}_{\langle \gamma, a \rangle} \coloneqq \langle \mathsf{id}_\gamma, \mathsf{id}_a \rangle$
- $\langle g, \alpha \rangle^{-1} \coloneqq \langle g^{-1}, \alpha^{-1} \rangle$
- $\langle g, \alpha \rangle \cdot \langle g', \alpha' \rangle \coloneqq \langle g \cdot g', \alpha \cdot \alpha' \rangle$

The projections $\mathbf{p}_{\Gamma.A}$ and $\mathbf{q}_{\Gamma.A}$ and the extensions $\langle s, t \rangle$ for substitutions $s : \Delta \to \Gamma$ and terms $t \in \mathsf{Tm}_\Delta(A\{s\})$ are defined exactly as before.

■

Lemma A.0.10 shows that this definition yields a path context and that $\mathbf{p}_{\Gamma.A}$, $\mathbf{q}_{\Gamma.A}$ and $\langle s, t \rangle$ preserve the extra structure. As a result, the category of path contexts supports context comprehension.

We conclude this section with the theorem that the path interpretation is indeed a model. The subsequent sections exhibit additional logical structure that is supported by the path model.

Theorem 4.1.10 (Path interpretation models dependent type theory)**.** The path interpretation forms a cwf.

4.2 Dependent Function Space

This section interprets the theory of dependent functions (cf. definition 2.1.1) in the path model. More specifically, it will be shown that the path model supports the Π structure (cf. definition 2.3.3) with function extensionality. The section is divided into four subsections. The first interprets the type itself, the second interprets its introduction and elimination rules and the third validates the computation rules. The last subsection interprets the extensionality axiom for Π types.

4.2.1 Sort

First, we define a type $\Pi AB \in \mathsf{Ty}(\Gamma)$ given types $A \in \mathsf{Ty}(\Gamma)$ and $B \in \mathsf{Ty}(\Gamma.A)$.

If we interpret function application in the theory as function application in the model, then the β-rule (cf. definition 2.1.1) dictates that the element $(\lambda b)_\gamma$ constructed from a term $b \in \mathsf{Tm}_{\Gamma.A}(B)$ is a functor from A_γ to $\{a \in A_\gamma\} B_{\langle \gamma, a \rangle}$, namely one that is equal to $a_\gamma \mapsto b_{\langle \gamma, a_\gamma \rangle}$. The terms of type B range over all functors and, hence, $(\Pi AB)_\gamma$ must include them all.

Definition 4.2.1 (Path Π sets)**.** For every $\gamma \in \Gamma_S$ set

$$\Pi AB_\gamma \coloneqq \{f_\gamma : \{\alpha_\gamma \in A_{\mathsf{id}_\gamma}\} \to B_{\langle \mathsf{id}_\gamma, \alpha_\gamma \rangle} | f_\gamma \text{ functor}\} \equiv \mathsf{Tm}_{A_\gamma}(B\langle K_\gamma, \mathsf{id}_{\langle A_\gamma, A_{\mathsf{id}_\gamma} \rangle} \rangle),$$

where $K_\gamma \coloneqq \langle a_\gamma \mapsto \gamma, \alpha_\gamma \mapsto \mathsf{id}_\gamma \rangle$ is the constant context morphism from $\langle A_\gamma, A_{\mathsf{id}_\gamma} \rangle$ to Γ.

■

We want the equality between functions to be extensional and, since paths interpret equality, a path between two functions must exist exactly when they map equals to equals. The coherence condition is inspired by the natural transformation law for a monoidal composition operation of morphisms between functors.

Definition 4.2.2 (Π paths)**.** For every $g \in \Gamma_R(\gamma, \gamma')$, $f_\gamma \in \Pi AB_\gamma$ and $f'_{\gamma'} \in \Pi AB_{\gamma'}$ set

$$\Pi AB_g(f_\gamma, f'_{\gamma'}) := \{\varphi : \{a_\gamma \in A_\gamma, a'_{\gamma'} \in A_{\gamma'}, \alpha \in A_g(a_\gamma, a'_{\gamma'})\} B_{\langle g,\alpha\rangle}(f_\gamma(a_\gamma), f'_{\gamma'}(a'_{\gamma'})) \\ | \varphi_{\alpha_g} \cdot f'_{\alpha'_{\gamma'}} \equiv f_{\alpha_\gamma} \cdot \varphi_{\alpha'_g} \text{ if } \alpha_g \cdot \alpha'_{\gamma'} \equiv \alpha_\gamma \cdot \alpha'_g\}$$

■

In other words, given two points $f_\gamma \in (\Pi AB)_\gamma$ and $f'_{\gamma'} \in (\Pi AB)_{\gamma'}$, the path set $(\Pi AB)_g(f_\gamma, f'_{\gamma'})$ contains exactly the families of paths $\varphi_\alpha \in B_{\langle g,\alpha\rangle}(f_\gamma(a_\gamma), f'_{\gamma'}(a'_{\gamma'}))$ indexed by the paths $\alpha \in A_g(a_\gamma, a'_{\gamma'})$ such that if $\alpha_\gamma \in A_{\mathsf{id}_\gamma}(a_\gamma, a)$ and $\alpha_g \in A_g(a, a'_{\gamma'})$ factorise α via $a \in A_\gamma$, and $\alpha'_g \in A_g(a_\gamma, a')$ and $\alpha'_{\gamma'} \in A_{\mathsf{id}_{\gamma'}}(a', a'_{\gamma'})$ constitute a factorisation of α via $a' \in A_{\gamma'}$, then both $\varphi_{\alpha'_g}$ and $f_{\alpha'_{\gamma'}}$ and f_{α_γ} and φ_{α_g} factorise φ_α

$$\varphi_{\alpha'_g} \cdot f'_{\alpha'_{\gamma'}} \equiv \varphi_\alpha \equiv f_{\alpha_\gamma} \cdot \varphi_{\alpha_g}$$

In fact, this is a generalisation of the naturality law because for two elements $f_\gamma, f'_\gamma \in \Pi AB_\gamma$ of the same fibre and a path $\alpha_\gamma \in \Pi AB_{\mathsf{id}_\gamma}(a_\gamma, a'_\gamma)$ there are several $\alpha_{a_\gamma} \in A_{\mathsf{id}_\gamma}(a_\gamma, a_\gamma), \alpha_{a'_\gamma} \in A_{\mathsf{id}_\gamma}(a'_\gamma, a'_\gamma)$ in general such that $\alpha_\gamma \cdot \alpha_{a'_\gamma} \equiv \alpha_{a_\gamma} \cdot \alpha_\gamma$, not just id_{a_γ} and $\mathsf{id}_{a'_\gamma}$. Moreover, a path $\varphi \in \Pi AB_{\mathsf{id}_\gamma}(f_\gamma, f'_\gamma)$ does not need to satisfy $\varphi_{\alpha_{a_\gamma}} \cdot f'_{\alpha_\gamma} \equiv f_{\alpha_\gamma} \cdot \varphi_{\alpha_{a'_\gamma}}$ if $\alpha_{a_\gamma} \cdot \alpha_\gamma \not\equiv \alpha_\gamma \cdot \alpha_{a'_\gamma}$. Actually, requiring this gives natural function paths even with the extra data, as the following remark shows.

Remark 4.2.3 (Characterisation of natural function paths)**.** $\varphi_{\alpha_g} \cdot f'_{\alpha_{\gamma'}} \equiv f_{\alpha_\gamma} \cdot \varphi_{\alpha'_g}$ (regardless of whether $\alpha_g \cdot \alpha_{\gamma'} \equiv \alpha_\gamma \cdot \alpha'_g$) if and only if $\varphi_{\alpha_g} \equiv \varphi_{\alpha'_g}$ for all $\alpha_g, \alpha'_g \in A_g(a_\gamma, a'_{\gamma'})$.

A yet stronger coherence condition for function paths is the following.

Remark 4.2.4 (Constant function paths are natural function paths)**.** $\varphi_{\alpha_g \cdot_A \alpha_{\gamma'}} \equiv \varphi_{\alpha_g} \equiv \varphi_{\alpha_\gamma \cdot_A \alpha_g}$ for all $\alpha_g, \alpha_\gamma, \alpha_{\gamma'}$, then $\varphi_{\alpha_g} \cdot f'_{\alpha_{\gamma'}} \equiv \varphi_{\alpha_g} \equiv f_{\alpha_\gamma} \cdot \varphi_{\alpha_g}$.

However, this also means that functions are constant on related elements.

This condition can be expressed equivalently but more concisely as a condition on the equivalence classes of a relation on A_g.

Remark 4.2.5 (Characterisation of constant function paths)**.** $\varphi_{\alpha_g \cdot_A \alpha_{\gamma'}} \equiv \varphi_{\alpha_g} \equiv \varphi_{\alpha_\gamma \cdot_A \alpha_g}$ for all $\alpha_g, \alpha_\gamma, \alpha_{\gamma'}$ if and only if $\varphi_{\alpha_g} \equiv \varphi_{\alpha'_g}$ for all α_g, α'_g such that there is $\alpha \in A_g(a'_\gamma, a_{\gamma'})$.

Before we define a concatenation operation on function paths we observe that it can be well-defined by point-wise concatenation.

Lemma 4.2.6 (Path application)**.** For all factorisations $\alpha'_g \cdot \alpha'_{g'} \equiv \alpha_g \cdot \alpha_{g'}$ of a path in $A_{g \cdot g'}(a_\gamma, a_{\gamma''})$ the paths given by $\varphi \in \Pi AB_g(f_\gamma, f'_{\gamma'}), \varphi' \in \Pi AB_{\gamma'}(f'_{\gamma'}, f''_{\gamma''})$ between $f_\gamma(a_\gamma)$ and $f''_{\gamma''}(a_{\gamma''})$ are the same.

$$\varphi_{\alpha'_g} \cdot_B \varphi'_{\alpha'_{g'}} \equiv \varphi_{\alpha_g} \cdot_B \varphi'_{\alpha_{g'}}$$

Proof. There exist factorisations $\alpha'_g \cdot_A \alpha_{\gamma'} \equiv \alpha_g$ and $\alpha_{g'} \equiv \alpha_{\gamma'} \cdot_A \alpha'_{g'}$ and, hence,

$$\varphi_{\alpha_g} \cdot_B \varphi'_{\alpha'_g} \equiv \varphi_{\alpha_g} \cdot_B {f_{\alpha_{\gamma'}}}^{-1} \cdot_B f_{\alpha_{\gamma'}} \cdot_B \varphi'_{\alpha_{g'}} \equiv \varphi_{\alpha'_g} \cdot_B \varphi'_{\alpha'_{g'}}.$$

□

We now define the composites $\varphi_g \cdot_{\Pi AB} \varphi'_{g'}$ for all $\varphi_g \in \Pi AB_g(f, f')$ and $\varphi_{g'} \in \Pi AB_{g'}(f', f'')$.

Definition 4.2.7 (Path concatenation in Π)**.**

$$\varphi_g \cdot_{\Pi AB} \varphi'_{g'} := (\varphi_{\alpha_g} \cdot_B \varphi'_{\alpha_{g'}})_{g \in \Gamma_R(\gamma,\gamma'), g' \in \Gamma_R(\gamma',\gamma''), \alpha \in A_{g \cdot g'}(a,a'')}$$

This is well-defined because every path over a composite can be factorised and for every factorisation we have 4.2.6. Moreover,

$$(\varphi \cdot \varphi')_\alpha \cdot_B f''_{\alpha_{\gamma''}} \equiv \varphi_{\alpha_g} \cdot_B \varphi'_{\alpha_{g'}} \cdot_B f''_{\alpha_{\gamma''}} \equiv \varphi_{\alpha_g} \cdot_B f'_{{\alpha_g}^{-1} \cdot \alpha_\gamma \cdot \alpha'_g} \cdot_B \varphi'_{\alpha'_{g'}} \equiv f_{\alpha_\gamma} \cdot_B \varphi_{\alpha'_g} \cdot_B \varphi'_{\alpha'_{g'}} \equiv f_{\alpha_\gamma} \cdot_B (\varphi \cdot \varphi')_{\alpha'}$$

for $\alpha \cdot \alpha_{\gamma''} \equiv \alpha_\gamma \cdot \alpha'$ and the associativity of $\cdot_{\Pi AB}$ is inherited from $\cdot_B$. ■

With the definition of function path concatenation at hand, the path application result can be expressed as

$$(\varphi \cdot \varphi')(\alpha_g \cdot \alpha_{g'}) \equiv \varphi_{\alpha_g} \cdot \varphi'_{\alpha_{g'}}.$$

Instead of relying on the fact that $(\varphi \cdot \varphi')_\alpha$ can be defined via any factorisation of α we can define concatenation using the transport paths, which give us canonical factorisations, directly.

Remark 4.2.8 (Function path concatenation using transport)**.** Given two paths $\varphi \in \Pi A B_g(f, f')$ and $\varphi' \in \Pi A B_{g'}(f', f'')$, define their composite by setting

$$(\varphi \cdot \varphi')_\alpha \coloneqq \varphi_{a_g^+} \cdot f'_{\overline{\alpha}} \cdot \varphi'_{a''^-_{g'}}$$

for each $\alpha \in A_{g \cdot g'}(a, a'')$. Alternatively, one might choose either $a_g^+ \cdot \overline{\alpha}$ or $\overline{\alpha} \cdot a''^-_{g'^{-1}}$. Here, $\overline{\alpha}$ denotes the projection of α via the transport paths to A_γ, $A_{\gamma''}$ and $A_{\gamma'}$ respectively.

This defines a function path because given any factorisation $\alpha'_1 \cdot \alpha'_2 \equiv \alpha$

$$\begin{aligned}
&(\varphi \cdot \varphi')_\alpha \\
\equiv&\text{Definition of concatenation} \\
&\varphi_{a_g^+} \cdot f'_{\overline{\alpha}} \cdot \varphi'_{a''^-_{g'}} \\
\equiv&a_g^+ \cdot \overline{\alpha} \cdot a''^-_{g'} \equiv \alpha \equiv \alpha_1 \cdot \alpha_2 \equiv a_g^+ \cdot \overline{\alpha_1} \cdot a'^-_{g'} \cdot \alpha_2 \text{ and factorisation of } \varphi' \\
&\varphi_{a_g^+} \cdot f'_{\overline{\alpha_1}} \cdot \varphi'_{a'^-_{g'}} \cdot f_{\alpha_2} \\
\equiv&\text{Definition of concatenation} \\
&(\varphi \cdot \varphi')_{\alpha_1} \cdot f_{\alpha_2}
\end{aligned}$$

The proof that $\varphi \cdot \varphi'$ can be factorised over factorisations via A_γ is symmetric.

For any three paths $\varphi \in \Pi A B_g(f, f'), \varphi' \in \Pi A B_{g'}(f', f''), \varphi'' \in \Pi A B_{g''}(f'', f''')$ we have that their composite can be constructed in any order (associativity).

$$\begin{aligned}
&(\varphi \cdot (\varphi' \cdot \varphi''))_\alpha \\
\equiv&\text{Definition of concatenation in } \Pi A B \\
&\varphi_{a_g^+} \cdot f'_{\overline{\alpha}} \cdot (\varphi'_{a'''^-_{g''}{}^-_{g'}} \cdot f''_{\overline{a'''^-_{g' \cdot g''}}} \cdot \varphi''_{a'''^-_{g''}}) \\
\equiv&\overline{a'''^-_{g' \cdot g''}} \equiv \mathsf{id}_{a'''^-_{g''}} \text{ and, hence, } f''_{\overline{a'''^-_{g' \cdot g''}}} \equiv \mathsf{id}_{f''_{a'''^-_{g''}}} \\
&\varphi_{a_g^+} \cdot f'_{\overline{\alpha}} \cdot (\varphi'_{a'''^-_{g''}{}^-_{g'}} \cdot \varphi''_{a'''^-_{g''}}) \\
\equiv&\text{Associativity of concatenation in } B \\
&\varphi_{a_g^+} \cdot (f'_{\overline{\alpha}} \cdot \varphi'_{a'''^-_{g''}{}^-_{g'}}) \cdot \varphi''_{a'''^-_{g''}} \\
\equiv&\overline{\alpha} \cdot a'''^-_{g''}{}^-_{g'} \equiv a_g^{+}{}^{+}_{g'} \cdot \overline{\alpha} \text{ and factorisation of } \varphi' \\
&\varphi_{a_g^+} \cdot (\varphi'_{a_g^{+}{}^{+}_{g'}} \cdot f''_{\overline{\alpha}}) \cdot \varphi''_{a'''^-_{g''}} \\
\equiv&\text{Associativity of concatenation in } B \\
&(\varphi_{a_g^+} \cdot \varphi'_{a_g^{+}{}^{+}_{g'}}) \cdot f''_{\overline{\alpha}} \cdot \varphi''_{a'''^-_{g''}} \\
\equiv&\overline{a^+_{g \cdot g'}} \equiv \mathsf{id}_{a_g^+} \text{ and, hence, } f'_{\overline{a^+_{g \cdot g'}}} \equiv \mathsf{id}_{f'_{a_g^+}} \\
&(\varphi_{a_g^+} \cdot f'_{\overline{a^+_{g \cdot g'}}} \cdot \varphi'_{a_g^{+}{}^{+}_{g'}}) \cdot f''_{\overline{\alpha}} \cdot \varphi''_{a'''^-_{g''}} \\
\equiv&\text{Definition of concatenation in } \Pi A B \\
&((\varphi \cdot \varphi') \cdot \varphi'')_\alpha
\end{aligned}$$

That we require the elements $f \in \Pi A B_\gamma$ to be functors can not only be explained by the Π structure axioms. First, each f is not just a function on elements because there are in general many paths between

$f(a_\gamma)$ and $f(a'_{\gamma'})$ whenever a_γ and $a'_{\gamma'}$ are related, none of which is the identity if $f(a_\gamma) \not\equiv f(a'_{\gamma'})$. Hence, in general, $B_{\langle g,\alpha\rangle}(f(a_\gamma), f_(a_{\gamma'}))$ contains neither a unique element nor a canonical choice and we let the elements carry that information. The functoriality axioms for elements $f \in \Pi AB_\gamma$ can be explained by the naturality requirement for identity paths, which implies

$$f(\alpha_{\gamma_1}) \cdot f(\alpha_{\gamma_2}) \equiv f(\alpha_{\gamma_1} \cdot \alpha_{\gamma_2}) \cdot f(\mathsf{id}_{a'''_\gamma}) \equiv f(\alpha_{\gamma_1} \cdot \alpha_{\gamma_2}).$$

The next two lemmata exhibit identities and inverses of $\cdot_{\Pi AB}$ in ΠAB.

Lemma 4.2.9 (Π identity paths)**.** For every $\gamma \in \Gamma_S$ and $f \in \Pi AB_\gamma$ set

$$\mathsf{id}_f := \{a_\gamma, a'_\gamma \in A_\gamma, \alpha_\gamma \in A_{\mathsf{id}_\gamma}(a_\gamma, a'_\gamma)\}\ f(\alpha_\gamma)$$

This defines a path because given factorisations $\alpha_\gamma \cdot \alpha'_\gamma \equiv \alpha''_\gamma \cdot \alpha'''_\gamma$

$$(\mathsf{id}_f)_{\alpha_\gamma} \cdot f_{\alpha'_\gamma} \equiv f_{\alpha_\gamma \cdot \alpha'_\gamma} \equiv f_{\alpha''_\gamma \cdot \alpha'''_\gamma} \equiv f_{\alpha''_\gamma} \cdot (\mathsf{id}_f)_{\alpha'''_\gamma}$$

Let $\varphi \in \Pi AB_g(f, f')$ be a path, then $\mathsf{id}_f \cdot \varphi \equiv \varphi$ and $\varphi \cdot \mathsf{id}_{f'} \equiv \varphi$.

$(\mathsf{id}_f \cdot \varphi)_\alpha$
$\equiv$Definition of concatenation in ΠAB
$\mathsf{id}_{f_{\mathsf{id}_{a_\gamma}}} \cdot \varphi_\alpha$
$\equiv$Definition of identity in ΠAB
$f_{\mathsf{id}_{a_\gamma}} \cdot \varphi_\alpha$
$\equiv f$ preserves identities
$\mathsf{id}_{f_{a_\gamma}} \cdot \varphi_\alpha$
$\equiv$Identity path in B
φ_α

The proof of the right identity law is analogous.

Lemma 4.2.10 (Π inverse paths)**.** For every $\varphi \in \Pi AB_g(f, f')$ set

$$\varphi^{-1} := \{a_\gamma \in A_\gamma, a'_{\gamma'} \in A_{\gamma'}, \alpha \in A_g(a_\gamma, a'_{\gamma'})\}\ {\varphi_{\alpha^{-1}}}^{-1}$$

This defines a path in $\Pi AB_{g^{-1}}(f', f)$ because given a composite $\alpha \cdot \alpha_\gamma$ via A_γ

$(\varphi^{-1})_{\alpha \cdot \alpha_\gamma}$
$\equiv$Definition of path inverse
${\varphi_{(\alpha \cdot \alpha_\gamma)^{-1}}}^{-1}$
$\equiv$Inverse-inverse law in A
${\varphi_{\alpha_\gamma^{-1} \cdot \alpha^{-1}}}^{-1}$
$\equiv$Factorisation of φ
$(f_{\alpha_\gamma^{-1}} \cdot \varphi_{\alpha^{-1}})^{-1}$
$\equiv$Inverse-inverse law in B
${\varphi_{\alpha^{-1}}}^{-1} \cdot {f_{\alpha_\gamma^{-1}}}^{-1}$
$\equiv f$ functor and definition of path inverse
$(\varphi^{-1})_\alpha \cdot f_{\alpha_\gamma}$

The case $A_{\gamma'}$ can be proved in an analogous way.

For every path $\varphi \in \Pi AB_g(f, f')$ the paths φ and φ^{-1} are inverses, that is $\varphi \cdot \varphi^{-1} \equiv \mathsf{id}_f$ and $\varphi^{-1} \cdot \varphi \equiv \mathsf{id}_{f'}$.

$$
\begin{aligned}
&(\varphi \cdot \varphi^{-1})_{\alpha_\gamma} \\
\equiv&\text{Definition of concatenation in } \Pi AB \\
&\varphi_{\alpha_g} \cdot \varphi^{-1}_{\alpha_g{}^{-1} \cdot \alpha_\gamma} \\
\equiv&\text{Factorisation of } \varphi^{-1}_{\alpha_g{}^{-1} \cdot \alpha_\gamma} \\
&\varphi_{\alpha_g} \cdot \varphi^{-1}_{\alpha_g{}^{-1}} \cdot f_{\alpha_\gamma} \\
\equiv&\text{Definition of inverse path in } \Pi AB \\
&\varphi_{\alpha_g} \cdot \varphi_{\alpha_g}{}^{-1} \cdot f_{\alpha_\gamma} \\
\equiv&\text{Concatenation of inverse paths at } f_a \text{ in } B \\
&\mathsf{id}_{f_a} \cdot f_{\alpha_\gamma} \\
\equiv&f \text{ functor and definition of identity path in } \Pi AB \\
&(\mathsf{id}_f)_{\mathsf{id}_a} \cdot f_{\alpha_\gamma} \\
\equiv&\text{Factorisation of } (\mathsf{id}_f)_{\alpha_\gamma} \\
&(\mathsf{id}_f)_{\alpha_\gamma}
\end{aligned}
$$

The proof of the left inverse law is analogous.

The missing ingredient for a path structure on ΠAB is transport, which we define next.

Definition 4.2.11 (Transport in Π)**.** For every path $g \in \Gamma_R(\gamma, \gamma')$ and point $f \in (\Pi AB)_\gamma$ define a functor in $(\Pi AB)_{\gamma'}$

$$
f^+_{g,\Pi AB} \coloneqq \begin{cases} a \in a & \mapsto (f_{a_g^-})^+_{\langle g, a_g^- \rangle, B} \\ \alpha \in A_{\mathsf{id}_{\gamma'}}(a, a') & \mapsto (f_{a_g^-})^\downarrow_{\langle g, a_g^\downarrow \rangle, B} \cdot f_{\alpha_g^-} \cdot (f_{a'_g{}^-})^\uparrow_{\langle g, a'^\downarrow_g \rangle, B} \end{cases}
$$

(note, $f_{\alpha_g^-} \in B_{\langle \mathsf{id}_\gamma, \alpha_g^- \rangle}(f_{a_g^-}, f_{a'_g{}^-})$) and a path from f to $f^+_{g,\Pi AB}$ over g

$$
f^\uparrow_{g,\Pi AB} \coloneqq \{a \in A_\gamma, a' \in A_{\gamma'}, \alpha \in A_g(a, a')\}\ f_{\alpha \cdot a'^\downarrow_g} \cdot (f_{a'_g{}^-})^\uparrow_{\langle g, a'^\downarrow_g \rangle, B}
$$

This indeed a path because given a composite $\alpha \cdot \alpha_{\gamma'}$ we have

$$
\begin{aligned}
&(f_{\alpha \cdot a'^\downarrow_g} \cdot (f_{a'_g{}^-})^\uparrow_{\langle g, a'^\downarrow_g \rangle, B}) \cdot (f_{a'_g{}^-})^\downarrow_{\langle g, a'^\downarrow_g \rangle, B} \cdot f_{\alpha_{\gamma'{}^\downarrow_g}} \cdot (f_{a''_g{}^-})^\uparrow_{\langle g, a''^\downarrow_g \rangle, B} \\
\equiv&f_{\alpha \cdot a'^\downarrow_g} \cdot f_{\alpha_{\gamma'{}^-_g}} \cdot (f_{a''_g{}^-})^\uparrow_{\langle g, a''^\downarrow_g \rangle, B} \\
\equiv&f_{\alpha \cdot a'^\downarrow_g \cdot \alpha_{\gamma'{}^-_g}} \cdot (f_{a''_g{}^-})^\uparrow_{\langle g, a''^\downarrow_g \rangle, B} \\
\equiv&f_{\alpha \cdot \alpha_{\gamma'} \cdot a''^\downarrow_g} \cdot (f_{a''_g{}^-})^\uparrow_{\langle g, a''^\downarrow_g \rangle, B} \\
\equiv&(f^\uparrow_g)_{\alpha \cdot \alpha_{\gamma'}}
\end{aligned}
$$

and for composites via A_γ we have

$$
\begin{aligned}
&f_{\alpha_\gamma} \cdot f_{\alpha \cdot a'^\downarrow_g} \cdot (f_{a'_g{}^-})^\uparrow_{\langle g, a'^\downarrow_g \rangle, B} \\
\equiv&f_{\alpha_\gamma \cdot \alpha \cdot a'^\downarrow_g} \cdot (f_{a'_g{}^-})^\uparrow_{\langle g, a'^\downarrow_g \rangle, B} \\
\equiv&(f^\uparrow_g)_{\alpha_\gamma \cdot \alpha}.
\end{aligned}
$$

■

Lemma A.0.12 and lemma A.0.13 in the appendix prove that the just defined transport preserves identities and commutes with path composition.

We conclude the subsection on the Π sort with the lemma that substitutions can be pushed under the type former.

Lemma 4.2.12 (Path Π types commute with substitution)**.** For every $s : \Delta \to \Gamma$

$$\Pi AB\{s\} \equiv \Pi(A\{s\})(B\{\langle s \circ \mathbf{p}, \mathbf{q}\rangle\}).$$

See the appendix (lemma A.0.16) for the proof.

4.2.2 Operations

This subsection is dedicated to the interpretation of λ-abstraction (Π introduction) and function application (Π elimination). We begin with the former. Its interpretation basically corresponds to currying in the index.

Definition 4.2.13 (Path λ terms)**.** For every term $b \in \mathsf{Tm}_{\Gamma.A}(B)$ a term $\lambda b \in \mathsf{Tm}_\Gamma(\Pi AB)$ is constructed as follows.

$$(\lambda b)_\gamma := b[\cdot] \in \mathsf{Tm}_{A_\gamma}(B[\cdot]), \text{ where } [\cdot] : A_\gamma \to \Gamma.A \text{ is the obvious substitution}$$
$$(\lambda b)_g := (b_{\langle g,\alpha\rangle})_{\alpha \in A_g(a,a')} \in \Pi AB_g((\lambda b)_\gamma, (\lambda b)_{\gamma'})$$

■

See A.0.14 in the appendix for the proof that this defines a path term. We also need to verify that substitutions can be pushed under the abstraction.

Lemma 4.2.14 (Path λ terms commute with substitution)**.** For every context morphism $s : \Delta \to \Gamma$

$$(\lambda b)\{s\} \equiv \lambda(b\langle s \circ \mathbf{p}, \mathbf{q}\rangle)$$

Again, a proof can be found in the appendix (lemma A.0.17).

We conclude this subsection with the interpretation of Π elimination or function application.

Definition 4.2.15 (Path $\mathsf{app}(\cdot,\cdot)$ terms)**.** For every two terms $\lambda \in \mathsf{Tm}_\Gamma(\Pi AB)$ and $a \in \mathsf{Tm}_\Gamma(A)$ a term $\mathsf{app}(\lambda, a) \in \mathsf{Tm}_\Gamma(B[a])$ gets constructed as follows.

$$\mathsf{app}(\lambda, a)_\gamma := (\lambda_\gamma)_{a_\gamma} \in B_{\langle \gamma, a_\gamma\rangle}$$
$$\mathsf{app}(\lambda, a)_g := (\lambda_g)_{a_g} \in B_{\langle g, a_g\rangle}(\lambda_{a_\gamma}, \lambda_{a_{\gamma'}})$$

■

See lemma A.0.15 for the proof that this defines a path term and lemma A.0.18 for the verification that a substitution is applied by applying it to both the function and the argument terms. Here, we only give the statement of the commutation with substitution.

Lemma 4.2.16 (Path $\mathsf{app}(\cdot,\cdot)$ terms commute with substitution)**.** For every context morphism $s : \Delta \to \Gamma$

$$\mathsf{app}(f, a)\{s\} \equiv \mathsf{app}(f\{s\}, a\{s\})$$

4.2.3 Equalities

The second last part of this section concerns the verification of the computation rules β and η for Π.

Lemma 4.2.17 (Path Π types support β-conversion)**.** For all terms $b \in \mathsf{Tm}_{\Gamma.A}(B)$ and $a \in \mathsf{Tm}_\Gamma(A)$

$$\mathsf{app}(\lambda b, a) \equiv b[a]$$

Proof.

$$
\begin{aligned}
&(\mathsf{app}(\lambda b, a))_\gamma \\
\equiv&\text{Definition of application} \\
&((\lambda b)_\gamma)_{a_\gamma} \\
\equiv&\text{Definition of } \lambda \text{ abstraction} \\
&b_{\langle \gamma, a_\gamma \rangle} \\
\equiv&\text{Definition of substitution} \\
&(b\langle \mathsf{id}_\Gamma, a\rangle)_\gamma \\
\equiv&[a] \equiv \langle \mathsf{id}_\Gamma, a\rangle \text{ by definition} \\
&(b[a])_\gamma
\end{aligned}
$$

The proof on paths is symmetric.

$$
\begin{aligned}
&(\mathsf{app}(\lambda b, a))_g \\
\equiv&\text{Definition of application} \\
&((\lambda b)_g)_{a_g} \\
\equiv&\text{Definition of } \lambda \text{ abstraction} \\
&b_{\langle g, a_g \rangle} \\
\equiv&\text{Definition of substitution} \\
&(b\langle \mathsf{id}_\Gamma, a\rangle)_g \\
\equiv&\text{Definition of substitution} \\
&(b[a])_g
\end{aligned}
$$

□

Lemma 4.2.18 (Path Π types support η-conversion)**.**

$$\lambda(\mathsf{app}(f\mathbf{p}, \mathbf{q})) \equiv f$$

Proof.

$$
\begin{aligned}
&(\lambda(\mathsf{app}(f\mathbf{p}, \mathbf{q})))_\gamma \\
\equiv&\text{Definition} \\
&(\mathsf{app}(f\mathbf{p}, \mathbf{q}))[\cdot] \\
\equiv&\text{Application and substitution commute, } \mathbf{p} \circ [\cdot] \equiv \mathit{const} \\
&\mathsf{app}(f_\gamma, \mathbf{q}[\cdot]) \\
\equiv&\mathbf{q}[\cdot] \in \mathsf{Tm}_{A_\gamma}(A(\mathbf{p}[\cdot])) \text{ “identity”} \\
&f_\gamma
\end{aligned}
$$

$$
\begin{aligned}
&(\lambda(\mathsf{app}(f\mathbf{p}, \mathbf{q})))_g \\
\equiv&\text{Definition of } \lambda \text{ abstraction} \\
&((\mathsf{app}(f\mathbf{p}, \mathbf{q}))_{\langle g, \alpha\rangle})_{\alpha \in A_g(a, a')} \\
\equiv&\text{Definition of application} \\
&((f_g)_\alpha)_{\alpha \in A_g(a, a')} \\
\equiv&\text{“}\eta \text{ conversion”} \\
&f_g
\end{aligned}
$$

□

4.2.4 Function extensionality

Function extensionality says that two functions are equal if they are point-wise equal. The function paths of definition 4.2.1 were defined with function extensionality in mind and the work in section 4.2.1 can be seen as establishing the fact the relation induced function extensionality soundly interprets the theory of intensional identity types.

Lemma 4.2.19 (Path Π types are extensional)**.** Given two terms $f, h \in \mathsf{Tm}_\Gamma(\Pi AB)$, then a proof

$$phi \in \mathsf{Tm}_\Gamma(\Pi_A \mathsf{Id}_B[\mathsf{app}(f\mathbf{p},\mathbf{q})\mathbf{p}][\mathsf{app}(h\mathbf{p},\mathbf{q})])$$

implies a proof $\varphi \in \mathsf{Tm}_\Gamma(\mathsf{Id}_{\Pi AB}[f\mathbf{p}][h])$.

Proof. We define function paths $\varphi_\gamma \in \Pi AB_{\mathsf{id}_\gamma}(f_\gamma, h_\gamma)$.

$$\varphi_{\gamma_\alpha} \coloneqq phi_{\gamma_a} \cdot h_{\mathsf{id}_\gamma}(\alpha) \equiv f_{\mathsf{id}_\gamma}(\alpha) \cdot phi_{\gamma'} \in B_{\langle \mathsf{id}_\gamma, \alpha\rangle}(f_\gamma(a), h_\gamma(a'))$$

and show that $\mathsf{Id}_{\Pi AB\langle g, f_g, h_g\rangle}(\varphi_\gamma, \varphi_{\gamma'})$ is inhabited. Indeed,

$$\varphi_\gamma \cdot h_g \equiv f_g \cdot \varphi_{\gamma'}$$

because

$$phi_{\gamma_{a_\gamma}} \cdot h_{\mathsf{id}_\gamma}(\alpha_\gamma) \cdot h_g(\alpha_g)$$

$\equiv$Path application

$$phi_{\gamma_{a_\gamma}} \cdot h_g(\alpha_\gamma \cdot \alpha_g)$$

$$\equiv phi_{g_{\alpha_\gamma \cdot \alpha_g}} \in \mathsf{Id}_{B\langle g, \alpha_\gamma\cdot\alpha_g, f_g(\alpha_\gamma\cdot\alpha_g), h_g(\alpha_\gamma\cdot\alpha_g)\rangle}(phi_{\gamma_{a_\gamma}}, phi_{\gamma'_{a_{\gamma'}}})$$

$$f_g(\alpha_\gamma \cdot \alpha_g) \cdot phi_{\gamma'_{a_{\gamma'}}}$$

$\equiv$Path application

$$f_g(\alpha_\gamma \cdot \alpha_g) \cdot f_{\mathsf{id}_\gamma}(\mathsf{id}_{a_{\gamma'}}) \cdot phi_{\gamma'_{a_{\gamma'}}}$$

The naturality condition for φ_γ holds for the same reasons. □

Every term of function paths defines a term of point-wise equalities. Hence, the converse direction holds as well such that

$$\mathsf{Tm}_\Gamma(\Pi_A \mathsf{Id}_B[\mathsf{app}(f\mathbf{p},\mathbf{q})\mathbf{p}][\mathsf{app}(g\mathbf{p},\mathbf{q})]) \quad \text{and} \quad \mathsf{Tm}_\Gamma(\mathsf{Id}_{\Pi AB}[f\mathbf{p}][g])$$

are logically equivalent. The question whether the types are even equivalent in the theory is not answered here.

This completes the proof of the Π structure for the path model.

Theorem 4.2.20 (The path model supports a Π structure)**.** The path interpretation models dependent type theory with Π types.

4.3 Identity Types

The structure of identity types in type theory governs how dependent types are interpreted in the groupoid and the path model. It can proved within intensional type theory that every type in the empty context has a groupoid structure (HS) and that at every dependent type C in context $\Gamma.A$ there is a conversion of terms in Ca to terms in Cb whenever $\mathsf{Id}_A(a,b)$ is inhabited (substitution of propositionally equal terms). In fact, it is known that an equivalent axiomatisation of identity types is given by requiring transport as a primitive operation and contractible identity types in the sense $\Pi_{a:A} isContr(\Sigma_{b:A} \mathsf{Id}_A(a,b))$.

This is exactly what we have done. Every dependent needs to be interpreted by an equivalence relation across its fibres and a transport operation between fibres. The equivalence relation will be used to interpret the identity sets and the transport operation to interpret the identity eliminator.

Let Γ be a context, $\gamma, \gamma' \in \Gamma_S$, $A \in \mathsf{Ty}(\Gamma)$, $a, a' \in \mathsf{Tm}_\Gamma(A)$, $\alpha \in A_g(a_\gamma, a'_\gamma)$, $\alpha' \in A_{g'}(a_{\gamma'}, a'_{\gamma'})$ and $\alpha'' \in A_{g''}(a_{\gamma''}, a'_{\gamma''})$.

4.3.1 Sort

When a model of type theory is constructed by definition of a cwf, syntactical substitution is interpreted by context morphisms. The elimination rule for identity types refers to substitution of propositionally equal terms so that type and term formers that depend on terms need to be interpreted by types and terms in contexts that include those terms. Since propositionally equality is a property between terms, the identity type at a type $A \in \mathsf{Ty}(\Gamma)$ must be interpreted in the context $\Gamma.A.A\mathbf{p}$.

The set of proofs of intensional equality between two elements of a fibre is given by their path set.

Definition 4.3.1 (Equality proofs in the path model)**.** For every $\langle \gamma, a, a' \rangle \in \Gamma.A.A\mathbf{p}$ set

$$(\mathsf{Id}_A(a,a'))_\gamma := A_{\mathsf{id}_\gamma}(a_\gamma, a'_\gamma))$$

■

There is a path between two equality proofs if and only if the path in the context transforms one into the other. In other words, two equality proofs are connected if and only if one is the transport of the other. In that sense we are defining the minimal possible path space.

Definition 4.3.2 (Equality paths)**.** If $\langle g, \alpha, \alpha' \rangle \in \Gamma.A.A\mathbf{p}(\langle \gamma, a, a' \rangle, \langle \gamma', b, b' \rangle)$ is a path, then for every two objects $\alpha_\gamma \in (\mathsf{Id}_A(a,a'))_\gamma$ and $\alpha_{\gamma'} \in (\mathsf{Id}_A(b,b'))_{\gamma'}$ define the set

$$\mathsf{Id}_{A_{\langle g,\alpha,\alpha' \rangle}}(\alpha_\gamma, \alpha_{\gamma'}) := \{\bullet | \alpha \cdot \alpha_{\gamma'} \equiv \alpha_\gamma \cdot \alpha'\}$$

■

Proofs of concatenation, inverses and identities correspond exactly to the proofs of transitivity, symmetry and reflexivity of the proof-irrelevant relation induced by the set comprehension. Associativity is then obvious. Indeed,

$$\mathsf{id}_a \cdot \alpha \equiv \alpha \cdot \mathsf{id}_{a'}$$

as well as

$$\alpha_{\gamma'} \cdot \alpha'^{-1} \equiv \alpha^{-1} \cdot \alpha_\gamma \text{ given } \mathsf{Id}_{A_{\langle g,\alpha,\alpha' \rangle}}(\alpha_\gamma, \alpha_{\gamma'})$$

and

$$\alpha_1 \cdot \alpha_2 \cdot \alpha_{\gamma''} \equiv \alpha_1 \cdot \alpha_{\gamma'} \cdot \alpha'_2 \equiv \alpha_\gamma \cdot \alpha'_1 \cdot \alpha'_2 \text{ given both } \mathsf{Id}_{A_{\langle g,\alpha_1,\alpha'_1 \rangle}}(\alpha_\gamma, \alpha_{\gamma'}) \text{ and } \mathsf{Id}_{A_{\langle g',\alpha_2,\alpha'_2 \rangle}}(\alpha_{\gamma'}, \alpha_{\gamma''}).$$

In this interpretation, the identity types are at least sets because they contain distinct elements in general. However, from the relation sets we see that identity types in the path model are also sets at most because there is at most one proof $\bullet$ of equality between two elements. Hence, the model interprets identity types as sets like the groupoid interpretation. This is a particularity of the path model and cannot be proved within the theory because intensional type theory admits higher-dimensional models [Str14a][BCH13].

It is tempting to generalise this to the following path sets.

Definition 4.3.3 (Generalised equality paths)**.** If $\langle g, \alpha, \alpha' \rangle \in \Gamma.A.A\mathbf{p}(\langle \gamma, a, a' \rangle, \langle \gamma', b, b' \rangle)$ is a path, then for every two objects $\alpha_\gamma \in (\mathsf{Id}_A(a,a'))_\gamma$ and $\alpha_{\gamma'} \in (\mathsf{Id}_A(b,b'))_{\gamma'}$ define the set

$$\mathsf{Id}_{A_{\langle g,\alpha,\alpha' \rangle}}(\alpha_\gamma, \alpha_{\gamma'}) := \{\langle \alpha_g, \alpha'_g \rangle | \alpha_g \in A_g(a,b), \alpha'_g \in A_g(a',b'), \alpha_g \cdot \alpha_{\gamma'} \equiv \alpha_\gamma \cdot \alpha'_g\}$$

■

Definition 4.3.4 (Generalised equality path concatenation)**.** For every two paths $\langle \mathrm{A}_1, \mathrm{A}_2 \rangle := \mathrm{A} : \alpha \to_g \alpha'$ and $\langle \mathrm{B}_1, \mathrm{B}_2 \rangle := \mathrm{B} : \alpha' \to_{g'} \alpha''$ define the path

$$\mathrm{A} \cdot \mathrm{B} := \langle \mathrm{A}_1 \cdot \mathrm{B}_1, \mathrm{A}_2 \cdot \mathrm{B}_2 \rangle$$

This defines a path between α and α' over g because $\mathrm{A}_1 \cdot \mathrm{B}_1 \cdot \alpha'' \equiv \mathrm{A}_1 \cdot \alpha' \cdot \mathrm{B}_2 \equiv \alpha \cdot \mathrm{A}_2 \cdot \mathrm{B}_2$ (concatenation is associative and A, B are paths). ■

Definition 4.3.5 (Generalised equality path identities)**.** For every object $\alpha \in (\mathsf{Id}_A(a, a'))_\gamma$ define the path

$$\mathsf{id}_\alpha \coloneqq \langle \mathsf{id}_{a_\gamma}, \mathsf{id}_{a'_\gamma} \rangle$$

This defines a path at α because $\mathsf{id}_a \cdot \alpha \equiv \alpha \cdot \mathsf{id}_{a'}$. Moreover, it defines an identity element with respect to path concatenation in identity types since given any $\langle \mathrm{A}_1, \mathrm{A}_2 \rangle : \alpha \to_g \alpha'$, $\mathrm{A}_1 \cdot \mathsf{id}_{a_{\gamma'}} \equiv \mathrm{A}_1$ and $\mathrm{A}_2 \cdot \mathsf{id}_{a'_{\gamma'}} \equiv \mathrm{A}_2$, $\mathsf{id}_{a_\gamma} \cdot \mathrm{A}_1 \equiv \mathrm{A}_1$ and $\mathsf{id}_{a'_\gamma} \cdot \mathrm{A}_2 \equiv \mathrm{A}_2$ (concatenation in A). ■

Definition 4.3.6 (Generalised equality path inverses)**.** For every path $\langle \mathrm{A}_1, \mathrm{A}_2 \rangle \coloneqq \mathrm{A} : \alpha \to_g \alpha'$ define the path

$$\mathrm{A}^{-1} \coloneqq \langle {\mathrm{A}_1}^{-1}, {\mathrm{A}_2}^{-1} \rangle$$

This defines a path between α' and α over g because ${\mathrm{A}_1}^{-1} \cdot \alpha \equiv {\mathrm{A}_1}^{-1} \cdot \mathrm{A}_1 \cdot \alpha' \cdot {\mathrm{A}_2}^{-1} \equiv \alpha' \cdot {\mathrm{A}_2}^{-1}$ (concatenation in A is associative and has inverses). Moreover, it inverts A because ${\mathrm{A}_1}^{-1} \cdot \mathrm{A}_1 \equiv \mathsf{id}_{a_{\gamma'}}$ and $\mathrm{A}_1 \cdot {\mathrm{A}_1}^{-1} \equiv \mathsf{id}_{a_\gamma}$ (the same for the second component). ■

Definition 4.3.7 (Generalised equality transport)**.** If $g : \langle \gamma, a, a' \rangle \to \langle \gamma', b, b' \rangle$ is a path, then for every object $\alpha \in (\mathsf{Id}_A(a, a'))_\gamma$ define the object

$$\alpha^+_{g, \mathsf{Id}_A} \coloneqq {g_2}^{-1} \cdot \alpha \cdot g_3$$

and the path

$$\alpha^+_{g, \mathsf{Id}_A} \coloneqq \langle g_2, g_3 \rangle$$

This defines a path between α and α^+_g over g because $g_2 \cdot ({g_2}^{-1} \cdot \alpha \cdot g_3) \equiv \alpha \cdot g_3$ (concatenation in A is associative and has inverses).

The definition above yields the identity map over identity morphisms

$$\alpha^+_{\mathsf{id}_{\langle \gamma, a, a' \rangle}} \equiv {\mathsf{id}_a}^{-1} \cdot \alpha \cdot \mathsf{id}_{a'} \equiv \alpha$$

$$\alpha^+_{\mathsf{id}_{\langle \gamma, a, a' \rangle}} \equiv \langle \mathsf{id}_a, \mathsf{id}_{a'} \rangle \equiv \mathsf{id}_\alpha$$

and commutes with concatenation

$$\alpha^+_{g \cdot g'} \equiv (g \cdot g')_2{}^{-1} \cdot \alpha \cdot (g \cdot g')_3 \equiv {g'_2}^{-1} \cdot {g_2}^{-1} \cdot \alpha \cdot g_3 \cdot g'_3 \equiv \alpha^{+^+}_{g\, g'}$$

$$\alpha^+_{g \cdot g'} \equiv \langle (g \cdot g')_2, (g \cdot g')_3 \rangle \equiv \langle g_2, g_3 \rangle \cdot \langle g'_2, g'_3 \rangle \equiv \alpha^+_g \cdot \alpha^{+^+}_{g\, g'}$$

■

However, then we will not be able to validate identity elimination.

Lemma 4.3.8 (Path Id types commute with substitution)**.** For every substitution $s : \Delta \to \Gamma$, the types $\mathsf{Id}_A\{\langle s \circ \mathbf{p}, \mathbf{q} \rangle \circ \mathbf{p}, \mathbf{q}\rangle\}$ and $\mathsf{Id}_{A\{s\}}$ are the same.

Proof. See A.0.19. □

4.3.2 Operations

Definition 4.3.9 (Reflexivity proof in the path model)**.** Let $A \in \mathsf{Ty}(\Gamma)$ be a semantic type in context $\Gamma : \mathcal{C}tx$. Define a semantic term $\mathsf{refl}_A \in \mathsf{Tm}_{\Gamma.A}(\mathsf{Id}_A[\mathbf{q}])$

$$\mathsf{refl}_A \coloneqq \begin{cases} \langle \gamma, a \rangle & \mapsto \mathsf{id}_a \\ \langle g, \alpha \rangle & \mapsto \langle \alpha, \alpha \rangle \end{cases}$$

This is well-defined because

$$\mathsf{id}_a \in A_{\mathsf{id}_\gamma}(a, a) \equiv \mathsf{Id}_{A\, \langle \gamma, a, a \rangle} \equiv \mathsf{Id}_{A\, \mathbf{q}(\langle \gamma, a \rangle)}$$

and

$$\mathsf{id}_a \cdot \alpha \equiv \alpha \equiv \alpha \cdot \mathsf{id}_{a'}$$

And it is a term because

$$\langle \mathsf{id}_a, \mathsf{id}_a \rangle \equiv \mathsf{id}_{\langle a, a \rangle}$$

and

$$\langle \alpha \cdot \alpha', \alpha \cdot \alpha' \rangle \equiv \langle \alpha, \alpha \rangle \cdot \langle \alpha', \alpha' \rangle$$

■

Lemma 4.3.10 (Path reflexivity proofs commute with substitution)**.** For every substitution $s : \Delta \to \Gamma$, the terms $\mathsf{refl}_A\{s\}$ and $\mathsf{refl}_{A\{s\}}$ are the same.

Proof. See A.0.20. □

Definition 4.3.11 (Equality elimination in the path model)**.** For every semantic type $C \in \mathsf{Ty}(\Gamma.A.A\mathbf{p}.\mathsf{Id}_A)$ and semantic term $d \in \mathsf{Tm}_{\Gamma.A}(C[\mathsf{refl}_A])$ define a semantic term $J_{C,d} \in \mathsf{Tm}_{\Gamma.A.A\mathbf{p}.\mathsf{Id}_A}(C)$

$$J_{C,d} \coloneqq \begin{cases} \langle \gamma, a, a', \alpha_\gamma \rangle & \mapsto d_{\langle \gamma, a \rangle}{}^{+}_{\langle \mathsf{id}_\gamma, \mathsf{id}_a, \alpha_\gamma, \bullet \rangle, C} \\ \langle g, \alpha, \alpha', \bullet \rangle & \mapsto d_{\langle \gamma, a \rangle}{}^{\downarrow}_{\langle \mathsf{id}_\gamma, \mathsf{id}_a, \alpha_\gamma, \bullet \rangle, C} \cdot d_{\langle g, \alpha \rangle} \cdot d_{\langle \gamma', b \rangle}{}^{\uparrow}_{\langle \mathsf{id}_{\gamma'}, \mathsf{id}_b, \alpha_{\gamma'}, \bullet \rangle, C} \end{cases}$$

This is well-defined because

$$\alpha_\gamma \in A_{\mathsf{id}_\gamma}(a, a') \quad \text{and} \quad \mathsf{id}_a, \alpha_\gamma \in A_{\mathsf{id}_\gamma}$$

and, hence,

$$\langle \mathsf{id}_\gamma, \mathsf{id}_a, \alpha_\gamma, \langle \alpha_\gamma, \mathsf{id}_a \rangle \in \Gamma.A.A\mathbf{p}.\mathsf{Id}_{A\,R}(\langle \gamma, a, a, \mathsf{id}_a \rangle, \langle \gamma, a, a', \alpha_\gamma \rangle)$$

Also,

$$d_{\langle g, \alpha \rangle} \in C_{\langle g, \alpha, \alpha, \bullet \rangle}(d_{\langle \gamma, a \rangle}, d_{\langle \gamma', b \rangle})$$

and, hence,

$$J_{C,d}(\langle g, \alpha, \alpha', \bullet \rangle) \in C_{\langle \mathsf{id}_\gamma, \mathsf{id}_a, \alpha_\gamma, \bullet \rangle^{-1} \cdot \langle g, \alpha, \alpha, \bullet \rangle \cdot \langle \mathsf{id}_{\gamma'}, \mathsf{id}_b, \alpha_{\gamma'}, \bullet \rangle} \equiv C_{\langle g, \alpha, {\alpha_\gamma}^{-1} \cdot \alpha \cdot \alpha_{\gamma'}, \bullet \rangle} \equiv C_{\langle g, \alpha, \alpha', \bullet \rangle}$$

because $\bullet \in \mathsf{Id}_{A\langle g, \alpha, \alpha' \rangle}(\alpha_\gamma, \alpha, \alpha_{\gamma'})$ if and only if ${\alpha_\gamma}^{+} \equiv \alpha_{\gamma'}$, that is $\alpha_\gamma \cdot \alpha' \equiv \alpha \cdot \alpha_{\gamma'}$.
Indeed, it defines a term because

$${\mathsf{id}_{d_{\langle \gamma, a \rangle}}}^{-1} \cdot d_{\langle \mathsf{id}_\gamma, \mathsf{id}_a \rangle} \cdot \mathsf{id}_{d_{\langle \gamma', a' \rangle}} \equiv \mathsf{id}_{d_{\langle \gamma, a \rangle}} \equiv \mathsf{id}_{J_{C, d_{\langle \gamma, a, a', \alpha_\gamma \rangle}}}$$

by identity preservation of transport and terms and

$$d_{\langle \gamma, a \rangle}{}^{\downarrow}_{\langle \mathsf{id}_\gamma, \mathsf{id}_a, \alpha_\gamma, \bullet \rangle, C} \cdot d_{\langle g \cdot g', \alpha \cdot \alpha'' \rangle} \cdot d_{\langle \gamma'', c \rangle}{}^{\uparrow}_{\langle \mathsf{id}_{\gamma''}, \mathsf{id}_c, \alpha_{\gamma''}, \bullet \rangle, C} \equiv J_{C, d_{\langle g, \alpha, \alpha', \bullet \rangle}} \cdot J_{C, d_{\langle g', \alpha'', \alpha''', \bullet \rangle}}$$

by composition preservation of transport and terms as well as

$$d_{\langle \gamma', b \rangle}{}^{\uparrow}_{\langle \mathsf{id}_{\gamma'}, \mathsf{id}_b, \alpha_{\gamma'}, \bullet \rangle, C} \cdot d_{\langle \gamma', b \rangle}{}^{\downarrow}_{\langle \mathsf{id}_{\gamma'}, \mathsf{id}_b, \alpha_{\gamma'}, \bullet \rangle, C} \equiv \mathsf{id}_{d_{\langle \gamma', b \rangle}}$$

■

Lemma 4.3.12 (Path equality elimination commutes with substitution)**.** For every substitution $s : \Delta \to \Gamma$, the terms $J_{C,d}\{\langle\langle\langle s \circ \mathbf{p}, \mathbf{q} \rangle \circ \mathbf{p}, \mathbf{q} \rangle \circ \mathbf{p}, \mathbf{q} \rangle\}$ and $J_{C\langle\langle\langle s \circ \mathbf{p}, \mathbf{q} \rangle \circ \mathbf{p}, \mathbf{q} \rangle \circ \mathbf{p}, \mathbf{q} \rangle, d\{\langle s \circ \mathbf{p}, \mathbf{q} \rangle\}}$ are the same.

Proof. See A.0.21. □

4.3.3 Equations

Lemma 4.3.13 (Path equality elimination supports Id computation rule)**.**

$$J_{C,d}[\mathsf{refl}_A\mathbf{p}][\mathbf{q}] \equiv d$$

Proof. We establish the equality for the element-indexed family

$$\begin{aligned}
&J_{C,d}[\mathsf{refl}_A\mathbf{p}][\mathbf{q}]_{\langle\gamma,a\rangle}\\
\equiv&J_{C,d_{\langle\gamma,a,a,\mathsf{id}_A\rangle}}\\
\equiv&d_{\langle\gamma,a\rangle}
\end{aligned}$$

and the path-indexed family

$$\begin{aligned}
&J_{C,d}[\mathsf{refl}_A\mathbf{p}][\mathbf{q}]_{\langle g,\alpha\rangle}\\
\equiv&J_{C,d_{\langle g,\alpha,\alpha,\bullet\rangle}}\\
\equiv&d_{\langle\gamma,a\rangle^{\downarrow}_{\langle\mathsf{id}_\gamma,\mathsf{id}_a,\mathsf{id}_a,\bullet\rangle,C}}\cdot d_{\langle g,\alpha\rangle}\cdot d_{\langle\gamma',b\rangle^{\uparrow}_{\langle\mathsf{id}_{\gamma'},\mathsf{id}_b,\mathsf{id}_b,\bullet\rangle,C}}\\
\equiv&d_{\langle g,\alpha\rangle}
\end{aligned}$$

The last steps are possible because transport preserves identities, respectively. □

4.3.4 Heterogeneous equality

The primitive relations A_g indexed by paths $g \in \Gamma_R$ in the context have mainly been useful in defining the equivalence relation on dependent sums (f. i., context comprehension). For interpreting the intensional identity types we require only the A_{id_γ} parts (for $\gamma \in \Gamma_S$). The question is whether the A_g parts interpret families of identity types indexed not only by γ but by g.

Given types $A \in \mathsf{Ty}(\Gamma)$ and $B \in \mathsf{Ty}(\Gamma.A)$ assume a type

$$\mathsf{ID}_B \in \mathsf{Ty}(\Gamma.A.A\mathbf{p}.\mathsf{Id}_A.B\mathbf{pp}.B\langle\mathbf{pp},\mathbf{q}\rangle\mathbf{pp})$$

with the introduction rule

$$\mathsf{refl}_B \in \mathsf{Tm}_{\Gamma.A.B}(\mathsf{ID}_B[\mathbf{q}]\langle\langle[\mathsf{refl}_A]\mathbf{p},\mathbf{q}\rangle\mathbf{p},\mathbf{q}\rangle\langle[\mathbf{q}]\mathbf{p},\mathbf{q}\rangle)$$

and the elimination rule

$$J_{C,d} \in \mathsf{Tm}_{\Gamma.A.A\mathbf{p}.\mathsf{Id}_A.B\mathbf{pp}.B\langle\mathbf{pp},\mathbf{q}\rangle\mathbf{pp}.\mathsf{ID}_B}(C)$$

for

$$C \in \mathsf{Ty}(\Gamma.A.A\mathbf{p}.\mathsf{Id}_A.B\mathbf{pp}.B\langle\mathbf{pp},\mathbf{q}\rangle\mathbf{pp}.\mathsf{ID}_B)$$

and

$$d \in \mathsf{Tm}_{\Gamma.A.B}(C\langle[\mathbf{q}]\langle\langle[\mathsf{refl}_A]\mathbf{p},\mathbf{q}\rangle\mathbf{p},\mathbf{q}\rangle\langle[\mathbf{q}]\mathbf{p},\mathbf{q}\rangle,\mathsf{refl}_B\rangle).$$

These generalised identity types make it possible to speak about the equality of the images under dependent functions and the dependent components of tuples. They also enable the formalisation of an extensionality principle for dependent sums, namely that two tuples are equal if and only if their components can be proved equal.

We now give the interpretation of the sets

$$\mathsf{ID}_{B\langle\gamma,a,a',\alpha,b,b'\rangle} := B_{\langle\mathsf{id}_\gamma,\alpha\rangle}(b,b')$$

and the paths

$$\mathsf{ID}_{B\langle g,\alpha,\alpha',\bullet,\beta,\beta'\rangle}(\mathrm{B},\mathrm{B}') := \{\bullet|\mathrm{B}\cdot\beta' \equiv \beta\cdot\mathrm{B}'\}$$

as well as reflexivity

$$\mathsf{refl}_B := \begin{cases} \langle \gamma, a, b \rangle & \mapsto \mathsf{id}_b \\ \langle g, \alpha, \beta \rangle & \mapsto \bullet \end{cases}$$

The dependent J will be definable using the transport of C (exactly as in the non-dependent case) if all proofs $\mathrm{B} \in B_{\langle \mathsf{id}_\gamma, \alpha \rangle}(b, b')$ are related to $\mathsf{id}_b \in B_{\langle \mathsf{id}_\gamma, \mathsf{id}_a \rangle}(b, b)$.

So, id_b must be equal to B over $\langle \mathsf{id}_\gamma, \mathsf{id}_a, \alpha, \bullet, \mathsf{id}_b, \mathrm{B} \rangle$, that is $\mathsf{id}_b \cdot \mathrm{B} \equiv \mathsf{id}_b \cdot \mathrm{B}$. This is the case indeed.

Except for the difficulties in expressing the context of ID_B, the interpretation of ID_B is analogous to the one of the non-dependent identity types. We omitted the verification that ID_B has transports, identities and inverses, and commutes with substitution.

4.4 Universe

Assume a set-theoretic universe $\mathcal{U}$. Specifically, assume a Grothendieck universe as in [HS14]. This entails that $\mathcal{U}$ is a set such that all transitive members $e \in s \in \mathcal{U}$ are elements of $\mathcal{U}$ and $\mathcal{U}$ together with the restricted membership relation is a model of set theory.

We define a type U whose points are small sets M_S endowed with a (small) path structure M_R connected by isomorphisms.

Definition 4.4.1 (Small types in the path model)**.** Set

$$\begin{aligned} U_S := \{ \langle M_S, M_R \rangle | M_S \in \mathcal{U}, M_R : M_S \times M_S \to \mathcal{U} \\ , \cdot : \{ \mu, \mu', \mu'' \in M_S \} M_R(\mu, \mu') \to M_R(\mu', \mu'') \to M_R(\mu, \mu'') \} \end{aligned}$$

Additionally, require that the operation is associative, that is

$$m \cdot (m' \cdot m'') \equiv (m \cdot m') \cdot m'',$$

has neutral elements in $M_R(\mu, \mu)$, that is for every $\mu \in M_S$ an element

$$\mathsf{id}_\mu \in M_R(\mu, \mu) \quad \text{s.t.} \quad m \cdot \mathsf{id}_{\mu'} \equiv m \equiv \mathsf{id}_\mu \cdot m$$

and inverses in the symmetric relations, that is for every $m \in M_R(\mu, \mu')$ an element

$$m^{-1} \in M_R(\mu', \mu) \quad \text{s.t.} \quad m \cdot m^{-1} \equiv \mathsf{id}_\mu \quad \text{and} \quad \mathsf{id}_{\mu'} \equiv m^{-1} \cdot m$$

■

Definition 4.4.2 (Paths between small types)**.** For every two elements $M, N \in U_S$ set

$$U_R(M, N) := \{ \langle f : M \to N, g : N \to M \rangle | f, g \text{ functors}, f \circ g \equiv \mathsf{id}_{N_S}, g \circ f \equiv \mathsf{id}_{M_S} \}$$

■

We define path composition in U component-wise. The composite of two functors is indeed again a functor. And, associativity of function composition not only implies associativity for this definition of path composition but is also needed to verify the two identity conditions for the composites. Given $\langle f, g \rangle \in U_R(M, N)$ and $\langle f', g' \rangle \in U_R(N, O)$, we calculate

$$(f' \circ f) \circ (g \circ g') \equiv f' \circ (f \circ g) \circ g' \equiv f' \circ \mathsf{id}_{N_S} \circ g' \equiv f' \circ g' \equiv \mathsf{id}_{O_S}$$

The second condition $((g \circ g') \circ (f' \circ f) \equiv \mathsf{id}_{M_S})$ can be shown to hold analogously.

Since paths are isomorphisms and identity functions are isomorphisms, it is straightforward to define and verify inverse and identity paths.

Definition 4.4.3 (Identity paths at small types)**.** Given a small type $M \in U_S$ define the path

$$\mathsf{id}_M := \langle \mathsf{id}_{M_S}, \mathsf{id}_{M_S} \rangle \in U_R(M, M).$$

■

Definition 4.4.4 (Inverse paths between small types)**.** For every path $\langle f, g\rangle \in U_R(M, N)$ between small types $M \in U_S$ and $N \in U_S$ we define the path

$$M^{-1} \coloneqq \langle g, f\rangle \in U_R(N, M).$$

■

Definition 4.4.5 (Small type elimination in the path model)**.** For every semantic term $M \in \mathsf{Tm}_{[]}(U)$ define the semantic type in context []

$$El(M)_\bullet \coloneqq M_S \quad \text{and} \quad El(M)_\bullet(\mu, \mu') \coloneqq M_R(\mu, \mu')$$

This is indeed an element of $\mathsf{Ty}([])$ because M_R is defined as a semantic type relation.

The dependent case taking $M \in \mathsf{Tm}_\Gamma(U\mathbf{p})$ to $El(M) \in \mathsf{Ty}(\Gamma)$ is more involved since the components $M(g)$ are isomorphisms and the components $El(M)(g)$ must be path sets. To construct the latter from the former we apply the same construction that we used to transform a type in the groupoid model into a path type.

$$\begin{aligned} El(M)_g(\mu, \mu') &\coloneqq M_{\gamma'}(M(g)(\mu), \mu') \\ \mathsf{id}_\mu &\coloneqq \mathsf{id}_\mu \\ m^{-1} &\coloneqq M_{g^{-1}}(m^{-1}) \\ m \cdot m' &\coloneqq M_g(m) \cdot m' \\ \mu_g^+ &\coloneqq M_g(\mu) \\ \mu_g^\uparrow &\coloneqq \mathsf{id}_{M_g(\mu)} \end{aligned}$$

id_μ, m^{-1} and $\cdot$ on the right-hand sides refer to the neutral elements, inverses and composites in $M_{\gamma'}$ respectively. The proofs A.0.5 through A.0.8 can easily be adapted to show that the thus defined $El(M)$ is a type in context Γ. Lemma A.0.22 proves that $El(M)$ is stable under substitution. ■

The type U is basically a small groupoid version of $\mathcal{C}tx$. A discrete version is possible as well but does not support universe extensionality. Neither does our universe with the strict path condition $f \circ g \equiv \mathsf{id}_N$, unless U is discrete. If U is discrete, that is all $M_R(\mu, \mu')$ are empty except for $M_R(\mu, \mu) \equiv \{\mathsf{id}_\mu\}$, then the following extensionality principle holds.

Remark 4.4.6 (Path U type is extensional)**.** If there are semantic functions $f \in \mathsf{Tm}_\Gamma(\Pi_{El(M)} El(N))$ and $g \in \mathsf{Tm}_\Gamma(\Pi_{El(N)} El(M))$ such that there are proofs

$$\mathsf{Tm}_\Gamma(\Pi_{El(M)} \mathsf{Id}_{El(N)}[\mathsf{app}(g, \mathsf{app}(f, \mathbf{q}))])$$

and

$$\mathsf{Tm}_\Gamma(\Pi_{El(N)} \mathsf{Id}_{El(M)}[\mathsf{app}(f, \mathsf{app}(g, \mathbf{q}))]),$$

then there is a proof $u \in \mathsf{Tm}_\Gamma(\mathsf{Id}_{U\mathbf{p}}[N\mathbf{p}][M])$.

Proof. First, we consider the case where Γ is empty. The functors $f : M \to N$ and $g : N \to M$ are weak inverses, that is

$$M_R(g(f(\mu)), \mu) \quad \text{and} \quad N_R(f(g(\nu)), \nu)$$

are non-empty as witnessed by the assumptions. In fact, they are strict inverses because M and N are discrete as elements of the universe. We conclude

$$\langle f, g\rangle \in U_R(M, N) \equiv \mathsf{Id}_{U\langle M, N\rangle}.$$

This allows us to set $u \coloneqq \langle f, g\rangle$. Then, to generalise this definition to the dependent case with $M, N \in \mathsf{Tm}_\Gamma(U\mathbf{p})$ we need to verify

$$u_\gamma \cdot N_g \equiv M_g \cdot u_{\gamma'}$$

for every $g \in \Gamma_R(\gamma, \gamma')$ in order to obtain the element $u_g \in \mathsf{Id}_{U\mathbf{p}\langle g, M_g, N_g\rangle}(u_\gamma, u_{\gamma'})$. Indeed,

$$
\begin{aligned}
& N_g(u_\gamma(\mu)) \\
\equiv & u_\gamma(\mu) \equiv f_\gamma(\mu) \\
& N_g(f_\gamma(\mu)) \\
\equiv & \mathsf{id}_\mu \in El(M)_g(\mu, M_g(\mu)) \text{ and } (f_g)_{\mathsf{id}_\mu} : El(N)_g(f_\gamma(\mu), f_{\gamma'}(M_g(\mu))) \Leftrightarrow N_g(f_\gamma(\mu)) \equiv f_{\gamma'}(M_g(\mu)) \\
& f_{\gamma'}(M_g(\mu)) \\
\equiv & u_{\gamma'}(M_g(\mu)) \equiv f_{\gamma'}(M_g(\mu)) \\
& u_{\gamma'}(M_g(\mu))
\end{aligned}
$$

Given that $\mathsf{Id}_{U\mathbf{p}\langle g, M_g, N_g\rangle}(u_\gamma, u_{\gamma'})$ are singletons, we conclude $u \in \mathsf{Tm}_\Gamma(\mathsf{Id}_{U\mathbf{p}}[N\mathbf{p}][M])$. □

It is straightforward to show that the converse statement holds because that is how the universe path sets are constructed. Whether the equivalence also holds within the theory is left unanswered here.

The universe is closed under function spaces if it is discrete.

Remark 4.4.7 (Path U type is closed under function spaces)**.** For all semantic terms $M \in \mathsf{Tm}_\Gamma(U), N \in \mathsf{Tm}_{\Gamma.El(M)}(U)$ there is a semantic term $\pi MN \in \mathsf{Tm}_\Gamma(U\mathbf{p})$ such that $El(\pi MN) \equiv \Pi_{El(M)} El(N)$.

Proof. It is clear that the set of terms $\mathsf{Tm}_{El(M)_\gamma}(El(N)_\gamma)$ together with identity paths is a discrete small type, just as every small set is. We set

$$\pi MN_\gamma \coloneqq \mathsf{Tm}_{El(M)_\gamma}(El(N)_\gamma)$$

and define πMN_g like the Π transport

$$\pi MN_g(f) \coloneqq \mu' \mapsto N_{\langle g, \mathsf{id}_{\mu'}\rangle}(f_{M_{g^{-1}}(\mu'), \gamma}).$$

Then, πMN_g is an isomorphism because M_g and N_g are. Therefore, $\pi MN \in \mathsf{Tm}_\Gamma(U\mathbf{p})$.
$El(\pi MN)$ and $\Pi_{El(M} El(N)$ agree on their set components

$$El(\pi MN)_\gamma \equiv \pi MN_\gamma \mathsf{Tm}_{El(M)_\gamma}(El(N)_\gamma) \equiv \Pi_{El(M)} El(N)_\gamma.$$

We show that $\Pi_{El(M)} El(N) \in \mathsf{Ty}(\Gamma)$ does not contain any non-trivial paths. For every path

$$\varphi \in \Pi_{El(M)} El(N)_g(f, f')$$

we have

$$\varphi_m \in El(N)_{\langle g, m\rangle}(f(\mu), f'(\mu'))$$

for $m \in El(M)_g(\mu, \mu')$. Because M and N are discrete

$$M_g(\mu) \equiv \mu' \quad \text{and} \quad N_{\langle g, \mathsf{id}_{M_g(\mu)}\rangle}(f(\mu)) \equiv f'(M_g(\mu))$$

and therefore

$$f' \equiv \pi MN_g(f) \quad \text{and} \quad \varphi_m \equiv \mathsf{id}_{\pi MN_g(f)}.$$

We conclude that $\Pi_{El(M)} El(N) \equiv El(\pi MN)$. □

4.5 Propositional Truncation

For types in the empty context, that is types interpreted by a groupoid, it is clear how to obtain the truncated type. Truncation depends on the same context (Sort) and, hence, the truncated type is also interpreted by a groupoid. The axioms governing the recursion principle for truncated types suggest that the two groupoids have the same objects and the fact that the truncation is a level -1 type implies that each homset consists of exactly one element. Such data trivially constitutes a unique groupoid up to isomorphism of categories.

Given a level -1 type P, the function space $\Pi_A P$ (P doesn't depend on A) is interpreted by the groupoid of functors from A to P and the to be constructed functor $\|f\|$ for an arbitrary functor f is uniquely defined on objects by the equation law for propositional truncation. What is missing for a complete interpretation is the mapping of morphisms in $\|A\|$ to morphisms in P. Since both types live on level -1 there is no room for interpretation and the unique definition yields trivially a functor.

Moving to the general case of a dependent type, truncating each fibre doesn't yield a dependent type because the concatenation of a path in A over a path in Γ with the formal paths added by the construction is undefined a priori and it's not clear how to define it. Instead, we apply the same construction to the path sets over paths in Γ as we did to the fibres, that is make them all singleton sets. The erasure of multiple paths and addition of further connections is not an issue because the only axiom that relates the paths in the type and its truncation is the computational rule for truncation recursion. And, the terms compared there are terms of the co-domain, which is a proposition by assumption and is thus interpreted by a semantic type with unique paths.

Let $A \in \mathsf{Ty}(\Gamma)$ be a type, then the propositional truncation $\|A\| \in \mathsf{Ty}(\Gamma)$ of A is defined as follows.

4.5.1 Sort

Definition 4.5.1 (Path $\|\cdot\|$ sets)**.** For every $\gamma \in \Gamma_S$ define the set

$$\|A\|_\gamma := A_\gamma$$

■

Next, we define the relations such that every two elements are related. We do this using singletons sets and because of the following result this is not an oversimplification if *isProp* should be provable internally.

Lemma 4.5.2 (Syntactic proofs of propositionality imply singleton path sets (I))**.** Let $P \in \mathsf{Ty}(\Gamma)$ be a semantic type, then the relations $P_{\mathsf{id}_\gamma}(\pi_\gamma, \pi'_\gamma)$ are singletons if and only if there is a term $isProp(P) \in \mathsf{Tm}_\Gamma(\Pi_P \Pi_{P\mathbf{p}} \mathsf{Id}_P)$.

Proof. Fix arbitrary $p, p' \in P_\gamma$ and $\pi, \pi' \in P_{\mathsf{id}_\gamma}(p, p')$, then

$$\mathsf{Id}_{P\langle \mathsf{id}_p, \pi\rangle}(isProp(P)_\gamma(p,p), isProp(P)_\gamma(p,p'))$$

and, hence,

$$\mathsf{Id}_{P\langle \mathsf{id}_\gamma, \mathsf{id}_p, \mathsf{id}_{p'}\rangle}(isProp(P)_\gamma(p,p)^+_{\langle \mathsf{id}_p, \pi\rangle}, isProp(P)_\gamma(p,p'))$$

because

$$isProp(P)_\gamma(p,p)\cdot\pi \equiv \mathsf{id}_p^{-1}\cdot isProp(P)_\gamma(p,p)\cdot\pi \equiv isProp(P)_\gamma(p,p)^+_{\langle \mathsf{id}_p, \pi\rangle}, isProp(P)_\gamma(p,p'))$$

which, since $\mathsf{Id}_{P\langle \gamma, p, p'\rangle}$ is discrete, entails

$$isProp(P)_\gamma(p,p)\cdot\pi \equiv isProp(P)_\gamma(p,p')$$

and by composition with the inverse in P_{id_γ}

$$\pi \equiv isProp(P)_\gamma(p,p)^{-1}\cdot isProp(P)_\gamma(p,p')$$

We observe that the right-hand side doesn't depend on π and that we can repeat the argument with π' substituted for π, so

$$\pi \equiv \pi'$$

In conclusion, each P_{id_γ} is a graph. Now make this an assumption and show the existence of a function that witnesses the equality of every pair of terms of type P. Define a term of type $\Gamma.P.P\mathbf{p} \vdash \mathsf{Id}_P$ as follows

$$\begin{cases} \langle \gamma, p, p' \rangle & \mapsto \star_{\mathsf{id}_\gamma, p, p'} \\ \langle g, \pi, \pi' \rangle & \mapsto \bullet \end{cases}$$

This is well-defined because $\star_{\mathsf{id}_\gamma, p, p'} \cdot \pi' \equiv \pi \cdot \star_{\mathsf{id}_{\gamma'}, q, q'}$.

That this mapping commutes with composition in $\Gamma.P.P\mathbf{p}$ and preserves identities follows directly from the transport properties.

□

Lemma 4.5.3 (Syntactic proofs of propositionality imply singleton path sets (II))**.** Let $P \in \mathsf{Ty}(\Gamma)$ be a semantic type, then the relations $P_g(p_\gamma, p_{\gamma'})$ are singletons if the relations $P_{\mathsf{id}_\gamma}(p_\gamma, p'_\gamma)$ are singletons.

Proof. By assumption, $P_{\mathsf{id}_{\gamma'}}({p_\gamma}^+, p_{\gamma'})$ is inhabited by a unique element $\star$ and, hence, ${p_\gamma}^\uparrow \cdot \star \in P_g(p_\gamma, p_{\gamma'})$. Moreover, this is the only such element because for any two elements $\pi, \pi' \in P_g(p_\gamma, p_{\gamma'})$ we have $\pi \cdot {\pi'}^{-1} \in P_{\mathsf{id}_\gamma}(p_\gamma, p_\gamma)$ with the sole inhabitant id_{p_γ}.

□

Definition 4.5.4 (Path $\|\cdot\|$ relations)**.** For every path $g \in \Gamma_R(\gamma, \gamma')$ and two elements $a \in \|A\|_\gamma, a' \in \|A\|_{\gamma'}$ define the set

$$\|A\|_g(a, a') \coloneqq \{\star\},$$

where $\star$ is a fresh so that all relations are distinct.

■

The singleton relations force the meaning of path concatenation, identity and inverse paths.

Definition 4.5.5 (Path concatenation, identities, inverses in the $\|\cdot\|$ types)**.** For every $g \in \Gamma_R(\gamma, \gamma'), g' \in \Gamma_R(\gamma', \gamma'')$ and $a \in \|A\|_\gamma, a' \in \|A\|_{\gamma'}, c \in \|A\|_{\gamma''}$ set

$$\star_{g,a,a'} \cdot \star_{g',a',a''} \coloneqq \star_{g \cdot g', a, a''}$$

and

$$\mathsf{id}_a \coloneqq \star_{\mathsf{id}_\gamma, a, a}$$

as well as

$$\star_{g,a,a'}{}^{-1} \coloneqq \star_{g^{-1}, a', a}.$$

Concatenation is obviously associative and the definition of id_a and $\star^{-1}$ are obviously neutral and inverses, respectively.

$$\star_{g,a,a'} \cdot (\star_{g',a',a''} \cdot \star_{g'',a'',a'''}) \equiv \star_{g \cdot g' \cdot g'', a, a'''} \equiv (\star_{g,a,a'} \cdot \star_{g',a',a''}) \cdot \star_{g'',a'',a'''}$$

$$\mathsf{id}_a \cdot \star_{g,a,a'} \equiv \star_{\mathsf{id}_a \cdot g, a, a'} \equiv \star_{g,a,a'} \equiv \star_{g,a,a'} \cdot \mathsf{id}_{a'}$$

$$\star_{g,a,a'}{}^{-1} \cdot \star_{g,a,a'} \equiv \star_{g^{-1} \cdot g, a', a'} \equiv \mathsf{refl}_{a'} \quad \text{and} \quad \star_{g,a,a'} \cdot \star_{g,a,a'}{}^{-1} \equiv \star_{g \cdot g^{-1}, a, a} \equiv \mathsf{refl}_a$$

■

With singleton relations there is no lack of possible transport either but a constructive choice is given by the type that gets truncated.

Definition 4.5.6 (Transport in $\|\cdot\|$ types)**.** For every path $g \in \Gamma_R(\gamma, \gamma')$ and element $a \in \|A\|_\gamma$ define the element

$$a^+_{g,\|A\|} := a^+_{g,A}$$

and the connecting path

$$a^\uparrow_{g,\|A\|} := \star.$$

These are well-defined because $\|A\|$ and A share the same sets and there is exactly one path to choose to connect any two elements in $\|A\|$. For the exact same reason the transport path laws follow

$$a^\uparrow_{\mathsf{id}_\gamma,\|A\|} \equiv \star \equiv \mathsf{id}_a$$
$$a^\uparrow_{g\cdot g',\|A\|} \equiv \star \equiv \star\cdot\star \equiv a^\uparrow_{g,\|A\|} \cdot a^{+\uparrow}_{g',\|A\|}$$

and remaining two transport laws for elements follow from the laws for the transport in A

$$a^+_{\mathsf{id}_\gamma,\|A\|} \equiv a^+_{\mathsf{id}_\gamma,A} \equiv a$$
$$a^+_{g\cdot g',\|A\|} \equiv a^+_{g\cdot g',A} \equiv a^+_{g,A}{}^+_{g',A} \equiv a^+_{g,\|A\|}{}^+_{g',\|A\|}$$

■

Lemma 4.5.7 (Path $\|\cdot\|$ types commute with substitution)**.** For every context morphism $s : \Delta \to \Gamma$ the types $\|A\|\{s\}$ and $\|A\{s\}\|$ are the same.

Proof. See A.0.23. □

4.5.2 Operations

A type and its truncation have the same elements both from a theory and model point of view.

Definition 4.5.8 (Path $\|\cdot\|$ terms)**.** For every term $a \in \mathsf{Tm}_\Gamma(A)$ define a term $\|a\| \in \mathsf{Tm}_\Gamma(\|A\|)$ by

$$\begin{cases} \gamma & \mapsto a_\gamma \\ g & \mapsto \star_{g,a_\gamma,a_{\gamma'}} \end{cases}$$

This is obviously a well-defined term. ■

Lemma 4.5.9 (Path $\|\cdot\|$ terms commute with substitution)**.** For every context morphism $s : \Delta \to \Gamma$ the terms $\|a\|\{s\}$ and $\|a\{s\}\|$ are the same.

Proof. See A.0.24. □

Lemma 4.5.10 (Path $\|\cdot\|$ are propositions)**.** There is a semantic term in $\mathsf{Tm}_\Gamma(\Pi_{\|A\|}\Pi_{\|A\|\mathbf{p}}\mathsf{Id}_{\|A\|})$

Proof. We define a term $isProp(\|A\|) \in \mathsf{Tm}_{\Gamma.\|A\|.\|A\|\mathbf{p}}(\mathsf{Id}_{\|A\|})$.

The object part is simply the unique elements of the relation sets because $\mathsf{Id}_{\|A\|}{}_{\langle\gamma,a,a'\rangle} \equiv \|A\|_{\mathsf{id}_\gamma}(a,a')$

$$isProp(\|A\|) : \langle\gamma, a, a'\rangle \mapsto \star_{\mathsf{id}_\gamma,a,a'}$$

For the path part we note that

$$\star_{g,a,b} \cdot \star_{\mathsf{id}_\gamma,b,b'} \equiv \star_{\mathsf{id}_\gamma,a,a'} \cdot \star_{g,a',b'}$$

and set

$$isProp(\|A\|) : \langle g, \star_{g,a,b}, \star_{g,a',b'}\rangle \mapsto \bullet$$

$\mathsf{Tm}_\Gamma(\Pi_{\|A\|}\Pi_{\|A\|\mathbf{p}}\mathsf{Id}_{\|A\|})$ is then inhabited by $\lambda(\lambda isProp(\|A\|))$.

□

As a result of 4.5.3, any non-dependent map into a propositional type is necessarily constant on all paths connecting a pair of points.

Lemma 4.5.11 (Functions into propositions are constant on paths)**.** Let $P \in \mathsf{Ty}(\Gamma)$ with $isProp(P) \in \mathsf{Tm}_\Gamma(\Pi_P \Pi_{P\mathbf{p}} \mathsf{Id}_P)$, then for every $f \in \mathsf{Tm}_\Gamma(A \to P)$ and $\alpha, \alpha' \in A_g(a_\gamma, a_{\gamma'})$

$$f_g(\alpha) \equiv f_g(\alpha')$$

Proof. Since f is non-dependent the paths α and α' are mapped into the same relation.

$$P_{\mathbf{p}(\langle g, \alpha \rangle)}(f_\gamma(a), f_{\gamma'}(a')) \equiv P_g(f_\gamma(a), f_{\gamma'}(a')) \equiv P_{\mathbf{p}(\langle g, \alpha' \rangle)}(f_\gamma(a), f_{\gamma'}(a'))$$

By assumption, $P_g(f_\gamma(a), f_{\gamma'}(a'))$ is a singleton. We conclude

$$f_g(\alpha) \equiv f_g(\alpha')$$

□

Definition 4.5.12 (Path $\| \cdot \|$ recursion)**.** Let $P \in \mathsf{Ty}(\Gamma)$ with $isProp(P) \in \mathsf{Tm}_\Gamma(\Pi_P \Pi_{P\mathbf{p}} \mathsf{Id}_P)$, then every $f \in \mathsf{Tm}_\Gamma(\Pi_A P\mathbf{p})$ from a type $A \in \mathsf{Ty}(\Gamma)$ defines a pointwise equal function $\|f\| \in \mathsf{Tm}_\Gamma(\Pi_{\|A\|} P\mathbf{p})$ on its truncation.

For every $\gamma \in \Gamma$ we define the functor $\|f\|_\gamma : \|A\|_\gamma \to P_\gamma$. Since this is going to be pointwise equal to f_γ, the object part is straightforward

$$\|f\|_\gamma : a \in \|A\|_\gamma \mapsto f_\gamma(a)$$

This is well-defined because $\|A\|_\gamma \equiv A_\gamma$. The path part cannot be the same as f_γ because $\|A\|$ do not share any. However, from 4.5.11 we know that the end points of a path $\alpha_\gamma \in A_{\mathsf{id}_\gamma}(a_\gamma, a'_\gamma)$ already determine its image under f_γ, namely the single element of $P_{\mathsf{id}_\gamma}(f_\gamma(a_\gamma), f_\gamma(a'_\gamma))$.

$$\|f\|_\gamma : \alpha_\gamma \in \|A\|_{\mathsf{id}_\gamma}(a_\gamma, a'_\gamma) \mapsto isProp(P)_\gamma(f_\gamma(a_\gamma), f_\gamma(a'_\gamma))$$

What is left are the paths connecting $\|f\|_\gamma(a_\gamma)$ and $\|f\|_{\gamma'}(a_{\gamma'})$ over $g \in \Gamma_R(\gamma, \gamma')$ and $\alpha \in A_g(a_\gamma, a_{\gamma'})$.

$$\|f\|_g : \alpha \in \|A\|_g(a_\gamma, a_{\gamma'}) \mapsto f_\gamma(a_\gamma)^\uparrow_g \cdot isProp(P)_{\gamma'}(f_\gamma(a_\gamma)^+_g, f_{\gamma'}(a_{\gamma'}))$$

We cannot just use f_g to construct the image because we are not given a path between a and a' in A, neither does $isProp(P)_g$ help us because the paths in Id_P carry no information about the paths in P that connect two equality proofs.

Lastly, given factorisations $\alpha \cdot \alpha_{\gamma'} \equiv \alpha_\gamma \cdot \alpha'$ we have that both composites $\|f\|_g(\alpha) \cdot \|f\|_{\gamma'}(\alpha_{\gamma'})$ and $\|f\|_\gamma(\alpha) \cdot \|f\|_g(\alpha')$ yield the single path in $P_g(\|f\|_\gamma(a_\gamma), \|f\|_{\gamma'}(a_{\gamma'}))$ and, hence, are equal.

We conclude that $\|f\| \in \mathsf{Tm}_\Gamma(\Pi_{\|A\|} P\mathbf{p})$.

■

Lemma 4.5.13 (Path $\| \cdot \|$ recursion commutates with substitution)**.** For every context morphism $s : \Delta \to \Gamma$ the terms $\|f\|\{s\}$ and $\|f\{s\}\|$ are the same.

Proof. See A.0.25. □

4.5.3 Equations

Lemma 4.5.14 (Path $\| \cdot \|$ recursion supports computation rule)**.** Let $P \in \mathsf{Ty}(\Gamma)$ with $isProp(P) \in \mathsf{Tm}_\Gamma(\Pi_P \Pi_{P\mathbf{p}} \mathsf{Id}_P)$, then for every $f \in \mathsf{Tm}_\Gamma(\Pi_A P\mathbf{p})$ and $a \in \mathsf{Tm}_\Gamma(A)$

$$\mathsf{app}(\|f\|, \|a\|) \equiv \mathsf{app}(f, a)$$

Proof.

$$
\begin{aligned}
&\mathsf{app}(\|f\|, \|a\|)_\gamma \\
\equiv &\|f\|_\gamma(\|a\|_\gamma) \\
\equiv &f_\gamma(a_\gamma) \\
\equiv &\mathsf{app}(f, a)_\gamma
\end{aligned}
$$

$$
\begin{aligned}
&\mathsf{app}(\|f\|, \|a\|)_g \\
\equiv &\|f\|_{\langle g, \|a\|_g \rangle} \\
\equiv &f_\gamma(a_\gamma)^\uparrow_g \cdot isProp(P)_{\gamma'}(f_\gamma(a_\gamma)^+_g, f_{\gamma'}(a_{\gamma'})) \\
\equiv &f_{\langle g, a_g \rangle} \\
\equiv &\mathsf{app}(f, a)_g
\end{aligned}
$$

The crucial step in the proof of the path case is sound because by 4.5.3 every two elements of $P_g(f_\gamma(a_\gamma), f_{\gamma'}(a_{\gamma'}))$ are equal. □

In general, the propositional truncation is not as straightforward to interpret. We make heavy use of discreteness for our identity types.

Chapter 5

Morphoid Model

The morphoid [McA14] model provides an alternative axiomatisation of the relations in the path model. As we will see, it is a refinement of the general relation model where both the contexts and the types are groupoids. The inverse and units for the relations of a dependent type will be definable from a local condition, the morphoid condition. Likewise, non-functorial transports are not primitives of the definition of types but can be defined using the axiom of choice.

5.0.1 Morphoids

We recall the definition of a morphoid here.

Definition 5.0.1 (Morphoid)**.** Let Γ be a groupoid. A morphoid $\mathrm{M} \subseteq \Gamma$ is a subset of $hom(\Gamma)$ with a restriction of $\cdot_\Gamma$ closed under composites of the form $m \cdot_\mathrm{M} n^{-1} \cdot_\mathrm{M} o \in \mathrm{M}_R(\gamma, \gamma''')$ for $m \in \mathrm{M}_R(\gamma, \gamma')$, $n \in \mathrm{M}_R(\gamma'', \gamma')$ and $o \in \mathrm{M}_R(\gamma'', \gamma''')$. ■

Since every category is closed under composition and groupoids are closed under inverses, we note the following fact.

Remark 5.0.2 (Every groupoid is a morphoid)**.** Given a groupoid G, then the set of all morphisms $hom(G)$ is a morphoid.

Morphoids are not groupoids but there are canonical groupoids associated with every morphoid.

Definition 5.0.3 (Left and right groupoids of a morphpid)**.** Let $\mathrm{M} \subseteq \Gamma$ be a morphoid. We define the groupoids $Left(\mathrm{M}) \coloneqq \mathrm{M} \cdot \mathrm{M}^{-1}$ and $Right(\mathrm{M}) \coloneqq \mathrm{M}^{-1} \cdot \mathrm{M}$.

- $Obj(Left(\mathrm{M})) \coloneqq \{\mu \in \mathrm{M} | \mu' \in \mathrm{M}, m \in \mathrm{M}_R(\mu, \mu')\}$
- $hom_{Left(\mathrm{M})}(\mu, \mu') \coloneqq \{m \cdot_\Gamma m'^{-1} | \mu'' \in \mathrm{M}, m \in \mathrm{M}_R(\mu, \mu''), m' \in \mathrm{M}_R(\mu', \mu'')\}$
- $m \cdot_{Left(\mathrm{M})} n \coloneqq m \cdot_\Gamma m'^{-1} \cdot_\Gamma n \cdot_\Gamma n'^{-1}$

The composite is well-defined because $m \in \mathrm{M}_R(\mu, \mu'')$ and $n' \cdot n^{-1} \cdot m' \in \mathrm{M}_R(\mu'''', \mu'')$.
The definition of $Right(\mathrm{M})$ follows the same pattern.
We verify that $Left(\mathrm{M})$ and $Right(\mathrm{M})$ are indeed groupoids.

Proof. For every object $\mu \in Left(\mathrm{M})$ there is a $m \in \mathrm{M}_R(\mu, \mu')$ and, hence, $\mathsf{id}_\mu \in hom_{Left(\mathrm{M})}(\mu, \mu)$. The unit laws in $Left(\mathrm{M})$ follow directly from the unit laws in Γ.

For every morphism $\langle m, m' \rangle \in hom_{Left(\mathrm{M})}(\mu, \mu')$ we have $m' \cdot m^{-1} \in hom_{Left(\mathrm{M})}(\mu', \mu)$ because the relation induced by $hom(Left(\mathrm{M}))$ is symmetric. The inverse laws follow from the corresponding laws in Γ.

The proof for $Right(\mathrm{M})$ follows the same pattern. ■

■

However, if a morphoid $\mathrm{M} \subseteq \Gamma$ contains all identity morphisms from Γ, then it is already a groupoid.

Lemma 5.0.4 (Morphoids closed under identity morphisms (reflexive morphoids) are groupoids)**.** Closure under identity morphisms implies closure under composition and inverses.

Proof. Let $\mathrm{M} \subseteq \Gamma$ be a reflexive morphoid. Then $\cdot_{\mathrm{M}}$ is total because for every $m \in \mathrm{M}_R(\mu, \mu')$ and $n \in \mathrm{M}_R(\mu', \mu'')$

$$m \cdot_\Gamma n \equiv m \cdot_\Gamma \mathsf{id}_{\mu'}^{-1} \cdot_\Gamma n \in \mathrm{M}_R(\mu, \mu').$$

Therefore, M is a category. Moreover, we have that M is closed under inverses because

$$m^{-1} \equiv \mathsf{id}_{\mu'} \cdot_\Gamma m^{-1} \cdot_\Gamma \mathsf{id}_{\mu} \in \mathrm{M}_R(\mu', \mu).$$

We conclude that the reflexive morphoid M is a groupoid. □

Morphoids form a category where they are morphisms between their left and right groupoid and the composite morphoid is defined to consist of the element-wise composites.

Lemma 5.0.5 (Morphoids are closed under point-wise composition)**.** Let M, N be morphoids with $\mathit{Right}(\mathrm{M}) \equiv \mathit{Left}(\mathrm{N})$, then $\{m \cdot n| \in \mathit{Left}(\mathrm{M}), \mu' \in \mathit{Right}(\mathrm{M}), \mu'' \in \mathit{Right}(\mathrm{N}), m \in \mathrm{M}_R(\mu, \mu'), n \in \mathrm{N}_R(\mu', \mu'')\}$ is a morphoid.

Proof. Given

$$m \cdot n \cdot (m' \cdot n')^{-1} \cdot m'' \cdot n''$$

there exist some $m_{n'}, m_n \in \mathrm{M}_R$ and $n_{m''}, n_{m'} \in \mathrm{N}_R$ such that

$${m_{n'}}^{-1} \cdot m_n \equiv n \cdot {n'}^{-1}$$

and

$$n_{m'} \cdot {n_{m''}}^{-1} \equiv {m'}^{-1} \cdot m''$$

because the left and right groupoids coincide. Thus, we conclude by application of the morphoid condition

$$m \cdot {m_{n'}}^{-1} \cdot m_n \cdot n_{m'} \cdot {n_{m''}}^{-1} \cdot n'' \in \{m \cdot n | m \in \mathrm{M}_R, n \in \mathrm{N}_R\}.$$

□

This composition is associative because the groupoid composition is. Moreover, the left and right groupoids, which are morphoids in particular, satisfy the unit laws with respect to point-wise morphoid composition.

Lemma 5.0.6 (Point-wise morphoid composition has identities)**.** $\mathit{Left}(\mathrm{M})$ ($\mathit{Right}(\mathrm{M})$) is a left (right) identity of M.

Proof. By the morphoid axiom. □

We conclude this exhibition into morphoids with the fact that the category of morphoids is closed under inverses.

Remark 5.0.7 (Morphoids form a groupoid)**.** The morphoid category is closed under inverses and hence a groupoid.

Proof. Let $\mathrm{M} \subseteq \Gamma$ be a morphoid, then the set of inverses $\mathrm{M}^{-1} \equiv \{m^{-1} | m \in \mathrm{M}_R\}$ forms a morphoid because

$$m^{-1} \cdot {m'}^{-1^{-1}} \cdot {m''}^{-1} \equiv m^{-1} \cdot m' \cdot {m''}^{-1} \equiv (m'' \cdot {m'}^{-1} \cdot m)^{-1},$$

which is an element of M^{-1} by the morphoid property for M. By the definition of morphoid identity it follows directly that M and M^{-1} are inverses. □

5.0.2 Morphoid types

We recall the definition of morphoid types here.

Definition 5.0.8 (Morphoid type)**.** Let Γ and A_{sp} be groupoids. A dependent type A in the morphoid model is a mapping from Γ to the category of morphoids or subspaces of A_{sp} such that

$$Left(A(g)) \equiv A(\gamma) \qquad Right(A(g)) \equiv A(\gamma')$$

and

$$A(g \cdot g') \equiv A(g) \cdot A(g')$$

The composite on the right-hand side is defined because $Right(A(g)) \equiv Left(A(g'))$. ■

$A_{g \cdot_\Gamma g'} \equiv A_g \cdot_{A_{sp}} A_{g'}$ is not axiom in [McA14] but a property of all types they construct within the model. Recall that the property also holds for the path interpretation but can proved directly from the type axioms. We assume it from the beginning for our exposition of the morphoid model.

The crucial ingredient for the proof of $A_{g \cdot_\Gamma g'} \equiv A_g \cdot_{A_{sp}} A_{g'}$ in the path model is the existence of transports, which we derive for the morphoid model now.

Lemma 5.0.9 (Elements can be transported over paths in the context)**.** Given a morphoid type A and $g \in \Gamma_R(\gamma, \gamma')$, then for every $a \in A(\gamma)$ and $a' \in A(\gamma')$ there exist morphisms $a^\uparrow \in A_g(a, a^+)$ and $a'^\downarrow \in A_g(a'^-, a')$.

Proof. By $A(\gamma) \equiv Left(A(g))$ and $A(\gamma') \equiv Right(A(g))$ we have

$$Obj(A(\gamma)) \equiv \{dom(m \cdot n^{-1}), cod(m \cdot n^{-1}) | m, n \in A(g)\} \equiv \{dom(m) | m, n \in A(g)\}$$
$$Obj(A(\gamma')) \equiv \{dom(m^{-1} \cdot n), cod(m^{-1} \cdot n) | m, n \in A(g)\} \equiv \{cod(m) | m, n \in A(g)\}$$

Therefore, there must exist morphisms $a^\uparrow, a'^\downarrow \in A(g)$ such that $dom(a^\uparrow) \equiv a$ and $cod(a'^\downarrow) \equiv a'$, which we can select using the axiom of choice. □

Lemma 5.0.10 (Paths can be transported over morphoids)**.** Given a morphoid type A, $g \in \Gamma_R(\gamma, \gamma')$ and morphisms $\alpha_\gamma \in A_{\mathsf{id}_\gamma}(a_\gamma, a'_\gamma), \alpha_{\gamma'} \in A_{\mathsf{id}_\gamma}(a_{\gamma'}, a'_{\gamma'})$, define the morphisms

$$\alpha_\gamma{}^+ := a_\gamma{}^{\uparrow -1} \cdot_A \alpha \cdot_A a_{\gamma'}{}^\uparrow \in A_{\mathsf{id}_{\gamma'}}(a_\gamma{}^+, a'_\gamma{}^+)$$

and

$$\alpha_{\gamma'}{}^- := a_{\gamma'}{}^\downarrow \cdot_A \alpha_{\gamma'} \cdot_A (a'_{\gamma'}{}^\downarrow)^{-1} \in A_{\mathsf{id}_\gamma}(a_{\gamma'}{}^-, a'_{\gamma'}{}^-).$$

Then, we have

$$\mathsf{id}_{a_\gamma}{}^+ \equiv \mathsf{id}_{a_\gamma{}^+} \quad (\mathsf{id}_{a_{\gamma'}}{}^- \equiv \mathsf{id}_{a_{\gamma'}{}^-})$$

and

$$(\alpha_\gamma \cdot \alpha'_\gamma)^+ \equiv \alpha_\gamma{}^+ \cdot \alpha'_\gamma{}^+ \quad ((\alpha_{\gamma'} \cdot \alpha'_{\gamma'})^- \equiv \alpha_{\gamma'}{}^- \cdot \alpha'_{\gamma'}{}^-).$$

Morphoid types not only commute with composition but also preserve identities.

Lemma 5.0.11 (Morphoid types are functors)**.** The mapping associated with a morphoid type preserves identities.

Proof.

$A(\mathsf{id}_\gamma)$

$\equiv$Right groupoid is a right identity

$A(\mathsf{id}_\gamma) \cdot \mathit{Right}(A(\mathsf{id}_\gamma))$

$\equiv A(\mathsf{id}_\gamma)$ is a morphism from $A(\gamma) \equiv \mathit{Left}(A(\mathsf{id}_\gamma))$ to $A(\gamma) \equiv \mathit{Right}(A(\mathsf{id}_\gamma))$

$A(\mathsf{id}_\gamma) \cdot \mathit{Left}(A(\mathsf{id}_\gamma))$

$\equiv$Definition of the left groupoid

$A(\mathsf{id}_\gamma) \cdot A(\mathsf{id}_\gamma) \cdot A(\mathsf{id}_\gamma)^{-1}$

$\equiv$A commutes with composition

$A(\mathsf{id}_\gamma \cdot \mathsf{id}_\gamma) \cdot A(\mathsf{id}_\gamma)^{-1}$

$\equiv \mathsf{id}_\gamma$ is an identity

$A(\mathsf{id}_\gamma) \cdot A(\mathsf{id}_\gamma)^{-1}$

$\equiv$Definition of the left groupoid

$\mathit{Left}(A(\mathsf{id}_\gamma))$

$\equiv$Definition of the left identity

$\mathsf{id}_{A(\gamma)}$

□

Since morphoid types are groupoid functors, they preserve inverses. Therefore, morphoid types are closed under inverses in the sense of path types.

$$\forall \alpha \in A_g(a, a').\alpha^{-1} \in A_{g^{-1}}(a', a)$$

Because of the way the subspaces that a morphoid connects are defined, elements can be transported along morphoids. However, it is not clear how to choose the transported elements for an arbitrary morphoid type such that it can be equipped with a functorial transport operation in the sense of path types. In fact, it is impossible in general. Consider the morphoid type

$$A(\bullet) \coloneqq \{\circ\}$$
$$A_{\mathbf{0}}(\circ, \circ) \coloneqq \{\mathbf{n} | n \bmod 2 \equiv 0\} \quad A_{\mathbf{1}}(\circ, \circ) \coloneqq \{\mathbf{n} | n \bmod 2 \equiv 1\}$$
$$\mathsf{id}_\circ \coloneqq \mathbf{0}$$
$$\mathbf{n}^{-1} \coloneqq -\mathbf{n}$$
$$a \cdot_A a' \coloneqq a +_{\mathbb{Z}} a'$$

in the context

$$\Gamma_S \coloneqq \{\bullet\}$$
$$\Gamma_R(\bullet, \bullet) \coloneqq \{\mathbf{0}, \mathbf{1}\}$$
$$\mathsf{id}_\bullet \coloneqq \mathbf{0}$$
$$\mathbf{0}^{-1} \coloneqq \mathbf{0} \quad \mathbf{1}^{-1} \coloneqq \mathbf{1}$$
$$\gamma \cdot_\Gamma \gamma' \coloneqq \gamma +_{\mathbb{Z}} \gamma' \bmod 2$$

A functorial transport must satisfy

$$\circ^{\uparrow}_{\mathbf{1}} \cdot_A \circ^{\uparrow}_{\mathbf{1}} \cdot_A \circ^{\uparrow}_{\mathbf{1}} \equiv \circ^{\uparrow}_{\mathbf{1} \cdot_\Gamma \mathbf{1} \cdot_\Gamma \mathbf{1}} \equiv \circ^{\uparrow}_{\mathbf{1}},$$

that is choose an integer $n \not\equiv 0$ such that $n + n + n \equiv n$. This is impossible and hence an example of a morphoid type for which we cannot possibly define a functorial transport for. The non-example is taken from [Str14b].

In the language of fibrations (cf. [Str14b]), we are faced with the difference between a cleavage and a splitting. Morphoid types are only cleavaged while path types are split. We have seen that the path interpretation models intensional type theory as presented in section 2.1 and we used the functoriality of transport to interpret the term J. Because terms in both the path and the morphoid model are functorial, it is unclear how one would interpret J for paths in the morphoid model. If we eliminate this difference and simply require a functorial transport, we conjecture that the resulting set of types forms a cwf that supports the same logical structure as the path model.

This closes the section on morphoids, which imply closure under inverse paths for dependent types without requiring their existence as an axiom.

Chapter 6

Conclusion

We presented a model of intensional type theory with extensional dependent function spaces and an extensional universe (plus propositional truncation) in the sense of categories with families. The reason for this particular choice of logical rules is that it comprises Martin-Löf's Logical Framework (LF) [NPS90] with intensional identity types, which has been used as the basis for proof assistants or to encode predicate logic. This does not mean that the presented model automatically extends to logics defined within LF. In particular, common additions to dependent type theories are inductive families, a hierarchy of universes or dependent sum types and none of these was considered in this thesis. However, the model supports function and universe extensionality, which are non-standard logical rules.

The interpretation underlying the path model takes a type to be a path space and a family of subspaces such that points can be transported between the subspaces that correspond to connected contexts. Then, terms are interpreted by families of connected points indexed by the type context. More generally, the path model can be seen as an interpretation of type theory into proof-relevant equivalence relations with coherence conditions. A possible generalisation to n-ary relations comes to mind but was not considered in the present work.

The path model in its current form is one-dimensional in the sense of Voevodsky, that is types can be seen as categories with discrete homsets. A generalisation to homcategories was not part of the work conducted for this thesis. It is worth noting that restricting or truncating the relations to mere propositions seems to give an interpretation logically equivalent to a setoid interpretation of type theory. Applying the truncation keeps the information about which pairs of points are connected in the model but throws away multiple paths. This then validates the uniqueness principle of identity proofs and invalidates the extensionality principle for the universe.

While looking at the morphoid model of type theory, we found that the crucial difference is the missing transport in the morphoid interpretation; closed types and contexts coincide in both models. The lack of transport has implications on the theory that is modelled. The morphoid model validates a meta-theoretical abstraction theorem akin to [Rey83], whereas in intensional type theory this gets internalised by identity elimination.

The primitive operations on paths allowed us to reason constructively but whether the construction can actually be carried out in a type-theoretic meta-theory like Martin-Löf type theory itself is left unanswered. In particular, the strict equalities under substitution or β-/η-conversion seem to suggest that the meta-theory needs to support at least function extensionality, which is not available in Martin-Löf type theory. Moreover, the coherence conditions on paths are formulated using definitional equality and can thus not be included in a meta-theoretic type of semantic contexts, types or terms.

Interesting directions for future work seem to be a formalisation of the truncation relation with a setoid and the cubical sets interpretation as cwf morphisms for instance, the connection between type theory with parametricity and intensional type theory given they both have relational models and an investigation into how the interpretation of the extensionality principles in the path model can help to justify them computationally.

Bibliography

[AGJ14] R. Atkey, N. Ghani, and P. 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 S. Jagannathan and P. Sewell. ACM, 2014, pp. 503–516. ISBN: 978-1-4503-2544-8. DOI: 10.1145/2535838.2535852. URL: http://doi.acm.org/10.1145/2535838.2535852.

[BCH13] M. Bezem, T. Coquand, and S. Huber. "A Model of Type Theory in Cubical Sets". In: *19th International Conference on Types for Proofs and Programs, TYPES 2013, April 22-26, 2013, Toulouse, France*. Ed. by R. Matthes and A. Schubert. Vol. 26. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2013, pp. 107–128. ISBN: 978-3-939897-72-9. DOI: 10.4230/LIPIcs.TYPES.2013.107. URL: http://dx.doi.org/10.4230/LIPIcs.TYPES.2013.107.

[BJP12] J. Bernardy, P. Jansson, and R. Paterson. "Proofs for free - Parametricity for dependent types". In: *J. Funct. Program.* 22.2 (2012), pp. 107–152. DOI: 10.1017/S0956796812000056. URL: http://dx.doi.org/10.1017/S0956796812000056.

[Car86] J. Cartmell. "Generalised algebraic theories and contextual categories". In: *Ann. Pure Appl. Logic* 32 (1986), pp. 209–243. DOI: 10.1016/0168-0072(86)90053-9. URL: http://dx.doi.org/10.1016/0168-0072(86)90053-9.

[Dyb95] P. Dybjer. "Internal Type Theory". In: *Types for Proofs and Programs, International Workshop TYPES'95, Torino, Italy, June 5-8, 1995, Selected Papers*. Ed. by S. Berardi and M. Coppo. Vol. 1158. Lecture Notes in Computer Science. Springer, 1995, pp. 120–134. ISBN: 3-540-61780-9. DOI: 10.1007/3-540-61780-9_66. URL: http://dx.doi.org/10.1007/3-540-61780-9_66.

[Hof97a] M. Hofmann. *Extensional constructs in intensional type theory*. CPHC/BCS distinguished dissertations. Springer, 1997. ISBN: 978-3-540-76121-1.

[Hof97b] M. Hofmann. "Syntax and Semantics of Dependent Types". In: *Semantics and Logics of Computation*. Cambridge University Press, 1997, pp. 79–130.

[HS14] M. Hofmann and T. Streicher. "Lifting Grothendieck Universes". Unpublished notes available at the author's home page. Dec. 2014. URL: http://www.mathematik.tu-darmstadt.de/~streicher/NOTES/lift.pdf.

[HS98] M. Hofmann and T. Streicher. "The groupoid interpretation of type theory". In: *Twenty-five years of constructive type theory (Venice, 1995)*. Vol. 36. Oxford Logic Guides. New York: Oxford Univ. Press, 1998, pp. 83–111.

[Hub15] S. Huber. *A Model of Type Theory in Cubical Sets*. Author homepage. 2015. URL: http://www.cse.chalmers.se/~simonhu/misc/lic.pdf.

[McA14] D. McAllester. "Implementation and Abstraction in Mathematics". In: *CoRR* abs/1407.7274 (2014). URL: http://arxiv.org/abs/1407.7274.

[NPS90] B. Nordström, K. Petersson, and J. M. Smith. *Programming in Martin-Löf's Type Theory, An Introduction*. Oxford University Press, 1990. URL: http://www.cse.chalmers.se/research/group/logic/book/.

[Rey83] J. C. Reynolds. "Types, Abstraction and Parametric Polymorphism". In: *IFIP Congress*. 1983, pp. 513–523.

[Str14a] T. Streicher. "A model of type theory in simplicial sets: A brief introduction to Voevodsky's homotopy type theory". In: *J. Applied Logic* 12.1 (2014), pp. 45–49. DOI: 10.1016/j.jal.2013.04.001. URL: http://dx.doi.org/10.1016/j.jal.2013.04.001.

[Str14b] T. Streicher. "Fibred Categories à la Jean Bénabou". Unpublished notes available at the author's home page. Dec. 2014. URL: http://www.mathematik.tu-darmstadt.de/~streicher/FIBR/FibLec.pdf.

[Ton13] S. Tonelli. "Investigations into a model of type theory based on the concept of basic pair". MA thesis. Stockholms Universitet, 2013.

[Uni13] T. Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. Institute for Advanced Study: http://homotopytypetheory.org/book, 2013.

Appendix A

Proofs

Relational interpretation forms a cwf

Lemma A.0.1 (Relational contexts form a category)**.**

$$\begin{aligned} Obj_{2Rel} &:= \{\langle \Gamma_S : \mathcal{S}et, \Gamma_R : \Gamma \to \Gamma \to \mathcal{S}et\rangle\} \\ hom_{2Rel}(\langle \Gamma_S, \Gamma_R\rangle, \langle \Delta_S, \Delta_R\rangle) &:= \{\langle f_0 : \Gamma_S \to_{\mathcal{S}et} \Delta_S, f_1 : \{\gamma, \gamma' \in \Gamma_S\}\Gamma_R(\gamma,\gamma') \to_{\mathcal{S}et} \Delta_R(f_0(\gamma), f_0(\gamma'))\rangle\} \\ g \circ f &:= \langle g_0 \circ_{\mathcal{S}et} f_0, g_1 \circ_{\mathcal{S}et} f_1\rangle \\ \mathsf{id}_{\langle \Gamma, R_\Gamma\rangle} &:= \langle \mathsf{id}_{\mathcal{S}et\,\Gamma}, \mathsf{id}_{\mathcal{S}et\,R_\Gamma}\rangle \end{aligned}$$

is a category.

Proof. Composition is well-typed because

$$g_0(f_0(\gamma)) \equiv (g_0 \circ f_0)(\gamma)$$

and identity is because

$$\mathsf{id}_{\langle \Gamma_S, \Gamma_R\rangle_0}(\gamma) \equiv \gamma.$$

Associativity of composition and neutrality of identity are inherited from $\mathcal{S}et$. □

Lemma A.0.2 (Relational substitution forms a functor)**.**

$$\begin{aligned} \mathsf{Ty}(\langle \Gamma_S, \Gamma_R\rangle) &:= \{\langle A_S : \Gamma_S \to \mathcal{S}et, A_R : \gamma, \gamma' \in \Gamma_S \Gamma_R(\gamma,\gamma') \to A_S(\gamma) \to A_S(\gamma') \to \mathcal{S}et\rangle\} \\ \mathsf{Tm}_{\langle \Gamma_S, \Gamma_R\rangle}(\langle A_S, A_R\rangle) &:= \{\langle t_s : \{\gamma \in \Gamma_S\}A_S(\gamma), t_r : \{\gamma, \gamma' \in \Gamma_S, g \in \Gamma_R(\gamma,\gamma') \to A_{R_{\gamma,\gamma'}}(g)(t_{s\,\gamma}, t_{s\,\gamma'})\rangle\} \\ \langle A_S, A_R\rangle\{\langle f_0, f_1\rangle\} &:= \langle A_S \circ f_0, A_R \circ f_1\rangle \\ \langle t_s, t_r\rangle\{\langle f_0, f_1\rangle\} &:= \langle t_s \circ f_0, t_r \circ f_1\rangle \\ T &:= \begin{cases} \Gamma & \mapsto (\mathsf{Tm}_\Gamma(A))_{A \in \mathsf{Ty}(\Gamma)} \\ f & \mapsto \langle (\cdot)\{f\}, (\cdot)\{f\}\rangle \end{cases} \end{aligned}$$

(substitution) is a functor.

Proof. Substitution on types and terms is well-defined because

$$A_S(f_0(\delta)) \equiv A\{f\}_S(\delta) \quad \text{and} \quad A_R(f_0(\delta), f_0(\delta')) \equiv A\{f\}_R(\delta).$$

T preserves identities because the components of id_Γ are identities (in $\mathcal{S}et$). Lastly, application of T commutes with composition because composition (in $\mathcal{S}et$) is associative. □

Lemma A.0.3 (Relational contexts support an empty context)**.** $[] \coloneqq \langle \{\bullet\}, \langle \bullet, \bullet \rangle \mapsto \{\bullet\} \rangle$ is a terminal object.

Proof. Let $\langle \Gamma_S, \Gamma_R \rangle$ be any other object, then $! \coloneqq \langle \gamma \mapsto \bullet, g \mapsto \bullet \rangle$ is the unique morphism from Γ into $[]$. □

Lemma A.0.4 (Relational contexts support context comprehension)**.** $2Rel$ is closed under dependent sum. For every $\langle \Gamma_S, \Gamma_R \rangle : 2Rel$ and $\langle A_S, A_R \rangle \in \mathsf{Ty}(\Gamma)$

$$\Gamma.A_S \coloneqq \{\langle \gamma, a \rangle | \gamma \in \Gamma, a \in A_\gamma\}$$
$$\Gamma.A_R(\langle \gamma, a \rangle, \langle \gamma', a' \rangle) \coloneqq \{\langle g, \alpha \rangle | g \in \Gamma_R(\gamma, \gamma'), \alpha \in A_{R_{\gamma,\gamma'}}(g, a, a')\}$$

is an object of $2Rel$ with two morphisms

$$\mathbf{p}_{\Gamma.A} \coloneqq \begin{cases} \langle \gamma, a \rangle & \mapsto \gamma \\ \langle g, \alpha \rangle & \mapsto g \end{cases} : \Gamma.A \to \Gamma$$

$$\mathbf{q}_{\Gamma.A} \coloneqq \begin{cases} \langle \gamma, a \rangle & \mapsto \langle \gamma, \langle a, a \rangle \rangle \\ \langle g, \alpha \rangle & \mapsto \langle g, \langle \alpha, \alpha \rangle \rangle \end{cases} : \Gamma.A \to \Gamma.A.A\{\mathbf{p}_{\Gamma.A}\}$$

such that $\mathbf{p}_{\Gamma.A.A\{\mathbf{p}_{\Gamma.A}\}} \circ \mathbf{q}_{\Gamma.A} \equiv \mathsf{id}_{\Gamma.A}$ and for every $\sigma : \Delta \to \Gamma$ and $s : \Delta \to \Delta.A\{\sigma\}$ with $\mathbf{p}_{\Delta.A\{\sigma\}} \circ s \equiv \mathsf{id}_\Delta$ a unique $\langle \sigma, s \rangle : \Delta \to \Gamma.A$ such that

$$\mathbf{p}_{\Gamma.A} \circ \langle \sigma, s \rangle \equiv \sigma \quad \text{and} \quad \mathbf{q}_{\Gamma.A}\{\langle \sigma, s \rangle\} \equiv s$$

Proof. $\mathbf{q}_{\Gamma.A}$ is well-defined because

$$A_{S_\gamma} \equiv A\{\mathbf{p}_{\Gamma.A}\}_{S_{\langle \gamma, a \rangle}} \quad \text{and} \quad A_{R_{\gamma,\gamma'}}(g, a, a') \equiv A\{\mathbf{p}_{\Gamma.A}\}_{R_{\langle \gamma, a \rangle, \langle \gamma', a' \rangle}}(\langle g, \alpha \rangle, \langle a, a \rangle, \langle a', a' \rangle).$$

And, indeed, the equations

$$\mathbf{p}_{\Gamma.A} \circ \langle \sigma, s \rangle \equiv \sigma \quad \text{and} \quad \mathbf{q}_{\Gamma.A}\{\langle \sigma, s \rangle\} \equiv s$$

completely determine the morphism

$$\langle \sigma, s \rangle(\delta) \equiv \langle \sigma(\delta), s(\delta) \rangle.$$

□

Groupoid interpretation supports a path structure

Lemma A.0.5 (*trans* is an associative operator)**.** For all $\alpha \in A_R(a, a')$, $\alpha' \in A_R(a', a'')$ and $\alpha'' \in A_R(a'', a''')$

$$trans(\alpha, trans(\alpha', \alpha'')) \equiv trans(trans(\alpha, \alpha'), \alpha'')$$

Proof.

$$\begin{aligned}
&trans(\alpha, trans(\alpha', \alpha'')) \\
\equiv &A(g' \cdot_\Gamma g'')(\alpha) \cdot_{A(\gamma''')} trans(\alpha', \alpha'') \\
\equiv &A(g' \cdot_\Gamma g'')(\alpha) \cdot_{A(\gamma''')} (A(g'')(\alpha') \cdot_{A(\gamma''')} \alpha'') \\
\equiv &(A(g' \cdot_\Gamma g'')(\alpha) \cdot_{A(\gamma''')} A(g'')(\alpha')) \cdot_{A(\gamma''')} \alpha'' \\
\equiv &A(g'')(A(g')(\alpha) \cdot_{A(\gamma'')} \alpha') \cdot_{A(\gamma''')} \alpha'' \\
\equiv &trans(A(g')(\alpha) \cdot_{A(\gamma'')} \alpha', \alpha'') \\
\equiv &trans(trans(\alpha, \alpha'), \alpha'')
\end{aligned}$$

□

Lemma A.0.6 (The morphisms $refl(a)$ are neutral elements)**.** For all $\alpha \in A_R(a, a')$

$$trans(\alpha, refl(a')) \equiv \alpha \equiv trans(refl(a), \alpha)$$

Proof.

$$\begin{aligned}
&trans(\alpha, refl(a')) \\
\equiv &A(\mathsf{id}_{\gamma'})(\alpha) \cdot_{A(\gamma')} \mathsf{id}_{a'} \\
\equiv &A(\mathsf{id}_{\gamma'})(\alpha) \\
\equiv &\alpha
\end{aligned}$$

and

$$\begin{aligned}
&trans(refl(a), \alpha) \\
\equiv &A(g)(\mathsf{id}_a) \cdot_{A(\gamma')} \alpha \\
\equiv &\mathsf{id}_{A(g)(a)} \cdot_{A(\gamma')} \alpha \\
\equiv &\alpha
\end{aligned}$$

□

Lemma A.0.7 (The morphisms $sym(\alpha)$ are inverse elements)**.** For all $\alpha \in A_R(a, a')$

$$trans(sym(\alpha), \alpha) \equiv refl(a') \qquad trans(\alpha, sym(\alpha)) \equiv refl(a)$$

Proof.

$$\begin{aligned}
&trans(sym(\alpha), \alpha) \\
\equiv &A(g)(A(g^{-1})(\alpha^{-1})) \cdot_{A(\gamma')} \alpha \\
\equiv &\alpha^{-1} \cdot_{A(\gamma')} \alpha \\
\equiv &\mathsf{id}_{a'} \\
\equiv &refl(a')
\end{aligned}$$

and

$$\begin{aligned}
&trans(\alpha, sym(\alpha)) \\
\equiv &A(g^{-1})(\alpha) \cdot_{A(\gamma)} A(g^{-1})(\alpha^{-1}) \\
\equiv &A(g^{-1})(\alpha \cdot_{A(\gamma')} \alpha^{-1}) \\
\equiv &A(g^{-1})(\mathsf{id}_{A(g)(a)}) \\
\equiv &\mathsf{id}_{A(g^{-1})(A(g)(a))} \\
\equiv &\mathsf{id}_a \\
\equiv &refl(a)
\end{aligned}$$

□

Lemma A.0.8 (The morphisms $a^{\uparrow}$ commute with composition)**.** For all $a : A(\gamma)$ and $g : \gamma \to_\Gamma \gamma'$, $g' : \gamma' \to_\Gamma \gamma''$

$$a^{\uparrow}(g \cdot_\Gamma g') \equiv trans(a^{\uparrow}(g), (a^{+}(g))^{\uparrow}(g'))$$

Proof.

$$
\begin{aligned}
&trans(a^{\uparrow}(g), (a^{+}(g))^{\uparrow}(g')) \\
\equiv &trans(\mathsf{id}_{A(g)(a)}, \mathsf{id}_{A(g')(A(g)(a))}) \\
\equiv &A(g')(\mathsf{id}_{A(g)(a)}) \cdot_{A(\gamma'')} \mathsf{id}_{A(g')(A(g)(a))} \\
\equiv &\mathsf{id}_{A(g')(A(g)(a))} \cdot_{A(\gamma'')} \mathsf{id}_{A(g')(A(g)(a))} \\
\equiv &\mathsf{id}_{A(g')(A(g)(a))} \\
\equiv &\mathsf{id}_{A(g \cdot_\Gamma g)(a)} \\
\equiv &a^{\uparrow}(g \cdot_\Gamma g')
\end{aligned}
$$

□

Path interpretation forms a cwf

Lemma A.0.9 (Path contexts form a category with terminal object.)**.** The identity function is a context morphism.

- $\mathsf{id}_{\Gamma_1}(\mathsf{id}_\gamma) \equiv \mathsf{id}_\gamma$
- $\mathsf{id}_{\Gamma_1}(g \cdot g') \equiv g \cdot g'$
- $\mathsf{id}_{\Gamma_1}(\alpha^{-1}) \equiv \alpha^{-1}$

Context morphisms are closed under function composition.

- $(g \circ f)_1(g \cdot g') \equiv (g \circ f)_1(g) \cdot (g \circ f)_1(g')$
- $(g \circ f)_1(\mathsf{id}_\gamma) \equiv \mathsf{id}_{(g \circ f)_0(\gamma)}$
- $(g \circ f)_1(\alpha^{-1}) \equiv (g \circ f)_1(\alpha)^{-1}$

The empty context is a path context.

- $\bullet \cdot (\bullet \cdot \bullet) \equiv \bullet \equiv (\bullet \cdot \bullet) \cdot \bullet$
- $\mathsf{id}_\bullet := \bullet, \bullet \cdot \mathsf{id}_\bullet \equiv \bullet \equiv \mathsf{id}_\bullet \cdot \bullet$
- $\bullet^{-1} := \bullet, \bullet \cdot \bullet^{-1} \equiv \mathsf{id}_\bullet \equiv \bullet^{-1} \cdot \bullet$

The empty context is terminal.

- $!_1\,(g \cdot g') \equiv \bullet \equiv \bullet \cdot \bullet \equiv !_1\,(g) \cdot !_1\,(g')$
- $!_1\,(\mathsf{id}_\gamma) \equiv \bullet \equiv \mathsf{id}_\bullet$
- $!_1\,(g^{-1}) \equiv !_1\,(g)^{-1}$
- $!_1 \equiv s$ for every $s : \Delta \to []$

Proof. Straightforward. □

Lemma A.0.10 (The category of path contexts supports context comprehension)**.** Composition is associative.

$$\langle g, \alpha\rangle \cdot (\langle g', \alpha'\rangle \cdot \langle g'', \alpha''\rangle) \equiv \langle g \cdot (g' \cdot g''), \alpha \cdot (\alpha' \cdot \alpha')\rangle \equiv \langle (g \cdot g') \cdot g'', (\alpha \cdot \alpha') \cdot \alpha'\rangle \equiv (\langle g, \alpha\rangle \cdot \langle g', \alpha'\rangle) \cdot \langle g'', \alpha''\rangle$$

Has identities.

$$\langle g, \alpha\rangle \cdot \mathsf{id}_{\langle \gamma', a'\rangle} \equiv \langle g \cdot \mathsf{id}_{\gamma'}, \alpha \cdot \mathsf{id}_{a'}\rangle \equiv \langle g, \alpha\rangle \equiv \langle \mathsf{id}_{\gamma'} \cdot g, \mathsf{id}_{a'} \cdot \alpha\rangle \equiv \mathsf{id}_{\langle \gamma', a'\rangle} \cdot \langle g, \alpha\rangle$$

Has inverses.

$$\langle g, \alpha\rangle \cdot \langle g, \alpha\rangle^{-1} \equiv \langle g \cdot g^{-1}, \alpha \cdot \alpha^{-1}\rangle \equiv \mathsf{id}_{\langle \gamma, \alpha\rangle} \equiv \langle g^{-1} \cdot g, \alpha^{-1} \cdot \alpha\rangle \equiv \langle g, \alpha\rangle^{-1} \cdot \langle g, \alpha\rangle$$

$\mathbf{p}$ and $\mathbf{q}$ preserve concatenation.

$$\mathbf{p}_{\Gamma.A_1}(\langle g, \alpha\rangle \cdot \langle g', \alpha'\rangle) \equiv g \cdot g' \equiv \mathbf{p}_{\Gamma.A_1}(\langle g, \alpha\rangle) \cdot \mathbf{p}_{\Gamma.A_1}(\langle g', \alpha'\rangle)$$
$$\mathbf{q}_{\Gamma.A_1}(\langle g, \alpha\rangle \cdot \langle g', \alpha'\rangle) \equiv \langle\langle g \cdot g', \alpha \cdot \alpha'\rangle, \alpha \cdot \alpha'\rangle \equiv \mathbf{q}_{\Gamma.A_1}(\langle g, \alpha\rangle) \cdot \mathbf{q}_{\Gamma.A_1}(\langle g', \alpha'\rangle)$$

$\mathbf{p}$ and $\mathbf{q}$ preserve identities.

$$\mathbf{p}_{\Gamma.A_1}(\mathsf{id}_{\langle \gamma, a\rangle}) \equiv \mathsf{id}_\gamma \equiv \mathsf{id}_{\mathbf{p}_{\Gamma.A_0}(\langle \gamma, a\rangle)}$$
$$\mathbf{q}_{\Gamma.A_1}(\mathsf{id}_{\langle \gamma, a\rangle}) \equiv \langle\langle \mathsf{id}_\gamma, \mathsf{id}_a\rangle, \mathsf{id}_a\rangle \equiv \mathsf{id}_{\mathbf{q}_{\Gamma.A_0}(\langle \gamma, a\rangle)}$$

$\langle s, t\rangle$ preserves concatenation.

$$\langle s, t\rangle_1(d \cdot d') \equiv \langle s(d \cdot d'), t_d \cdot t_{d'}\rangle \equiv \langle s(d), t_d\rangle \cdot \langle s(d'), t_{d'}\rangle \equiv \langle s, t\rangle_1(d) \cdot \langle s, t\rangle_1(d')$$

$\langle s, t\rangle$ preserves identities.

$$\langle s, t\rangle_1(\mathsf{id}_\delta) \equiv \langle s(\mathsf{id}_\delta), t_{\mathsf{id}_\delta}\rangle \equiv \langle \mathsf{id}_{s(\delta)}, \mathsf{id}_{t_\delta}\rangle \equiv \mathsf{id}_{\langle s(\delta), t_\delta\rangle} \equiv \mathsf{id}_{\langle s, t\rangle_0(\delta)}$$

The fact that $\langle s, t\rangle$, $\mathbf{p}$ and $\mathbf{q}$ preserve inverses is implied by the fact that they preserve identities and commute with composition.

Lemma A.0.11 (Path substitution is a functor)**.** The mapping of categories

$$\begin{cases} \Gamma & \mapsto (\mathsf{Tm}_\Gamma(A))_{A \in \mathsf{Ty}(\Gamma)} \\ s & \mapsto \langle A \mapsto A\{s\}, t \mapsto t\{s\}\rangle \end{cases}$$

from path contexts to path terms indexed by path types is a functor.

Proof. Because composition in the category of contexts is function composition we have

$$(s \circ r)(\delta) \equiv s(r(\delta)) \text{ and } (s \circ r)(d) \equiv s(r(d))$$

in both the set and function family indices of definition 4.1.7 such that

$$A\{s \circ r\} \equiv A\{s\}\{r\} \text{ and } t\{s \circ r\} \equiv t\{s\}\{r\}.$$

Moreover, the identity morphisms are the identity functions, which implies

$$A\{\mathsf{id}_\Gamma\} \equiv A \text{ and } t\{\mathsf{id}_\Gamma\} \equiv t.$$

□

Path interpretation supports Π types

Lemma A.0.12 (Π transport preserves identities)**.** Transport along an identity path yields the same element and the identity path at the element.

Proof.

$f^{+}_{\mathsf{id}_\gamma}(a)$

$\equiv$Definition of transport in ΠAB

$f_{a^{+}_{\mathsf{id}_\gamma{}^{-1}}}{}^{+}_{\mathsf{id}_\gamma, a^{+}_{\mathsf{id}_\gamma{}^{-1}, A}, B}$

$\equiv$Transport in A preserves identities

$f_{a}{}^{+}_{\mathsf{id}_\gamma, \mathsf{id}_a, B}$

$\equiv$Definition identity path in $\Gamma.A$

$f_{a}{}^{+}_{\mathsf{id}_{\langle \gamma, a\rangle}, B}$

$\equiv$Transport in B preserves identities

f_a

$f^{+}_{\mathsf{id}_\gamma}(\alpha)$

$\equiv$Definition of transport in ΠAB

$f_{a^{-}_{\mathsf{id}_\gamma{}^{-1}}}{}^{-}_{\langle \mathsf{id}_\gamma, a^{+}_{\mathsf{id}_\gamma{}^{-1}}\rangle, B} \cdot f_{\alpha^{+}_{\mathsf{id}_\gamma{}^{-1}}} \cdot f_{a'^{+}_{\mathsf{id}_\gamma{}^{-1}}}{}^{+}_{\mathsf{id}_\gamma, a'^{+}_{\mathsf{id}_\gamma{}^{-1}}, B}$

$\equiv$Transport in A and B preserves identities

$f_{a^{-}_{\mathsf{id}_\gamma{}^{-1}}} \cdot f_{\alpha^{+}_{\mathsf{id}_\gamma{}^{-1}}} \cdot f_{a'^{+}_{\mathsf{id}_\gamma{}^{-1}}}$

$\equiv$Transport in A preserves identities

$f_{\mathsf{id}_a} \cdot f_\alpha \cdot f_{\mathsf{id}_{a'}}$

$\equiv f$ preserves identities

$\mathsf{id}_{f_a} \cdot f_\alpha \cdot \mathsf{id}_{f_{a'}}$

$\equiv$Identity of path composition

f_α

$f^{\uparrow}_{\mathsf{id}_\gamma}(\alpha)$

$\equiv$Definition of transport in ΠAB

$f_{\alpha \cdot a'^{\uparrow}_{\mathsf{id}_\gamma{}^{-1}, A}} \cdot f_{a'^{+}_{\mathsf{id}_\gamma{}^{-1}, A}}{}^{+}_{\mathsf{id}_\gamma, a'^{\uparrow}_{\mathsf{id}_\gamma{}^{-1}, A}, B}$

$\equiv$Transport in A and B preserves identities

$f_\alpha \cdot \mathsf{id}_{f_{a'}}$

$\equiv$Identity of path composition

f_α

$\equiv$Definition of identity paths in ΠAB

$(\mathsf{id}_f)_\alpha$

□

Lemma A.0.13 (Π transport commutes with path composition)**.** Transporting along a composed path is the same as first transporting along the first path and then along the second path.

Proof. We begin with the verification of the point component of $f^+_{g\cdot\rho}$.

$$
\begin{aligned}
& f^+_{g\cdot\rho}(a) \\
\equiv & \text{Definition of transport in } \Pi AB \\
& f(a^-_{g\cdot\rho})^+_{\langle g\cdot\rho, a^\downarrow_{g\cdot\rho}\rangle, B} \\
\equiv & a^\downarrow_{g\cdot\rho} \equiv (a^-_\rho)^\downarrow_g \cdot a^\downarrow_\rho \\
& f(a^-_{g\cdot\rho})^+_{\langle g, (a^-_\rho)^\downarrow_g\rangle\cdot\langle\rho, a^\downarrow_\rho\rangle, B} \\
\equiv & \text{Transport in } B \text{ commutes with path composition} \\
& (f(a^-_{g\cdot\rho})^+_{\langle g, a^{-\downarrow}_{\rho\,g}\rangle, B})^+_{\langle\rho, a^\downarrow_\rho\rangle, B} \\
\equiv & a^-_{g\cdot\rho} \equiv (a^-_\rho)^-_g \\
& \\
& (f((a^-_\rho)^-_g)^+_{\langle g, (a^-_\rho)^\downarrow_g\rangle, B})^+_{\langle\rho, a^\downarrow_\rho\rangle, B} \\
\equiv & \text{Definition of transport in } \Pi AB \\
& f^+_g(a^-_\rho)^+_{\langle\rho, a^\downarrow_\rho\rangle, B} \\
\equiv & \text{Definition of transport in } \Pi AB \\
& (f^+_g)^+_\rho(a)
\end{aligned}
$$

The same applies to the path component of $f^+_{g\cdot\rho}$.

$$
\begin{aligned}
& f^+_{g\cdot\rho}(\alpha) \\
\equiv & \text{Definition of transport in } \Pi AB \\
& f(a^-_{g\cdot\rho})^\downarrow_{\langle g\cdot\rho, a^\downarrow_{g\cdot\rho}\rangle, B} \cdot f(\alpha^-_{g\cdot\rho}) \cdot f(a'^-_{g\cdot\rho})^\uparrow_{\langle g\cdot\rho, a'^\downarrow_{g\cdot\rho}\rangle, B} \\
\equiv & \text{Transport composition in } B \\
& (f(a^-_{g\cdot\rho}))^{+\downarrow}_{g\,\langle\rho, a^\downarrow_\rho\rangle, B} \cdot f(a^-_{g\cdot\rho})^\downarrow_{\langle g, (a^-_\rho)^\downarrow_g\rangle, B} \cdot f(\alpha^-_{g\cdot\rho}) \cdot f(a'^-_{g\cdot\rho})^\uparrow_{\langle g, a'^{-\downarrow}_{\rho\,g}\rangle, B} \cdot (f(a'^-_{g\cdot\rho}))^{+\uparrow}_{g\,\langle\rho, a'^\downarrow_\rho\rangle, B} \\
\equiv & \text{Transport composition in } A \\
& (f((a^-_\rho)^-_g))^{+\downarrow}_{g\,\langle\rho, a^\downarrow_\rho\rangle, B} \cdot f((a^-_\rho)^-_g)^\downarrow_{\langle g, (a^-_\rho)^\downarrow_g\rangle, B} \cdot f((\alpha^-_\rho)^-_g) \cdot f((a'^-_\rho)^-_g)^\uparrow_{\langle g, a'^{-\downarrow}_{\rho\,g}\rangle, B} \cdot (f((a'^-_\rho)^-_g))^{+\uparrow}_{g\,\langle\rho, a'^\downarrow_\rho\rangle, B} \\
\equiv & \text{Definition of transport in } \Pi AB \\
& f^+_g(a^-_\rho)^\downarrow_{\langle\rho, a^\downarrow_\rho\rangle, B} \cdot f^+_g(\alpha^-_\rho) \cdot f^+_g(a'^-_\rho)^\uparrow_{\langle\rho, a'^\downarrow_\rho\rangle, B} \\
\equiv & \text{Definition of transport in } \Pi AB \\
& (f^+_g)^+_\rho(\alpha)
\end{aligned}
$$

We conclude $f^+_{g\cdot\rho} \equiv (f^+_g)^+_\rho$ by family extensionality in the meta-theory. Lastly, the path connecting

f and $f^{+}_{g\cdot\rho}$.

$$
\begin{aligned}
&\left(f^{\uparrow}_{g\cdot\rho}\right)_{\alpha}\\
\equiv&\text{Definition of transport in } \Pi AB\\
&f(\alpha\cdot {a''}^{\downarrow}_{g\cdot\rho})\cdot f({a''}^{-}_{g\cdot\rho})^{\uparrow}_{\langle g\cdot\rho,{a''}^{\downarrow}_{g\cdot\rho}\rangle}\\
\equiv&\text{Transport composition in } B\\
&f(\alpha\cdot {a''}^{\downarrow}_{g\cdot\rho})\cdot f({a''}^{-}_{g\cdot\rho})^{\uparrow}_{\langle g,({a''}^{-}_{\rho})^{\downarrow}_{g}\rangle}\cdot \left(f({a''}^{-}_{g\cdot\rho})^{+}_{g}\right)^{\uparrow}_{\langle\rho,{a''}^{\downarrow}_{\rho}\rangle}\\
\equiv&\text{Transport composition in } A\\
&f(\alpha\cdot {a''}^{\uparrow}_{\rho}\cdot({a''}^{-}_{\rho})^{\uparrow}_{g})\cdot f(({a''}^{-}_{\rho})^{-}_{g})^{\uparrow}_{\langle g,({a''}^{-}_{\rho})^{\downarrow}_{g}\rangle}\cdot \left(f({a''}^{-}_{g\cdot\rho})^{+}_{g}\right)^{\uparrow}_{\langle\rho,{a''}^{\downarrow}_{\rho}\rangle}\\
\equiv&\text{Definition of transport in } \Pi AB\\
&\left(f^{\uparrow}_{g}\right)_{\alpha\cdot{a''}^{\downarrow}_{\rho}}\cdot f^{+}_{g}({a''}^{-}_{g\cdot\rho})^{\uparrow}_{\langle\rho,{a''}^{\downarrow}_{\rho}\rangle}\\
\equiv&{a''}^{\uparrow}_{\rho}\cdot{a''}^{\downarrow}_{\rho}\equiv\mathsf{id}_{a''}\text{ and unit law in } B\\
&\left(f^{\uparrow}_{g}\right)_{\alpha\cdot{a''}^{\downarrow}_{\rho}}\cdot f^{+}_{g}({a''}^{\uparrow}_{\rho}\cdot{a''}^{\downarrow}_{\rho})\cdot f^{+}_{g}({a''}^{-}_{g\cdot\rho})^{\uparrow}_{\langle\rho,{a''}^{\uparrow}_{\rho}\rangle}\\
\equiv&\text{Definition of transport in } \Pi AB\\
&\left(f^{\uparrow}_{g}\right)_{\alpha\cdot{a''}^{\downarrow}_{\rho}}\cdot\left((f^{+}_{g})^{\uparrow}_{\rho}\right)_{{a''}^{\uparrow}_{\rho}}\\
\equiv&\text{Definition of composition in } \Pi AB \text{ and } {a''}^{\uparrow}_{\rho}\cdot{a''}^{\downarrow}_{\rho}\equiv\mathsf{id}_{a''}\\
&\left(f^{\uparrow}_{g}\cdot(f^{+}_{g})^{\uparrow}_{\rho}\right)_{\alpha}
\end{aligned}
$$

We conclude $f^{\uparrow}_{g\cdot\rho}\equiv f^{\uparrow}_{g}\cdot(f^{+}_{g})^{\uparrow}_{\rho}$. □

Lemma A.0.14 (Path λ terms are functorial)**.** The definition preserves identities

$$((\lambda b)_{\mathsf{id}_{\gamma}})_{\alpha}\equiv b_{\langle\mathsf{id}_{\gamma},\alpha\rangle}\equiv((\lambda b)_{\gamma})_{\alpha}\equiv(\mathsf{id}_{(\lambda b)_{\gamma}})_{\alpha}$$

and is functorial

$$
\begin{aligned}
&((\lambda b)_{g\cdot\rho})_{\alpha}\\
\equiv&\text{Definition}\\
&b_{\langle g\cdot\rho,\alpha\rangle}\\
\equiv&g^{+}(a)\cdot\overline{\alpha}\cdot(\rho^{-1})^{-}(a')\\
&b_{\langle g,g^{+}(a)\rangle}\cdot b_{\langle\mathsf{id}_{\gamma'},\overline{\alpha}\rangle}\cdot b_{\langle\rho,(\rho^{-1})^{-}(a')\rangle}\\
\equiv&\text{Definition}\\
&((\lambda b)_{g})_{g^{+}(a)}\cdot((\lambda b)_{\gamma'})_{\overline{\alpha}}\cdot((\lambda b)_{\rho})_{(\rho^{-1})^{-}(a')}\\
\equiv&\text{Definition}\\
&((\lambda b)_{g}\cdot(\lambda b)_{\rho})_{\alpha}
\end{aligned}
$$

Lemma A.0.15 (Path $\mathsf{app}(\cdot,\cdot)$ terms are functorial)**.** Indeed, this defines a term since the definition preserves identities

$$(\mathsf{app}(\lambda,a))_{\mathsf{id}_{\gamma}}\equiv\lambda_{\mathsf{id}_{\gamma}}\alpha_{\mathsf{id}_{\gamma}}\equiv\mathsf{id}_{\lambda_{\gamma}}\mathsf{id}_{a_{\gamma}}\equiv(\mathsf{id}_{\lambda_{\gamma}})_{\mathsf{id}_{a_{\gamma}}}\equiv\mathsf{id}_{(\lambda_{\gamma})_{a_{\gamma}}}\equiv\mathsf{id}_{\mathsf{app}(\lambda,a)_{\gamma}}$$

and is functorial

$$(\mathsf{app}(\lambda,a))_{g\cdot\rho}\equiv\lambda_{g\cdot\rho}a_{g\cdot\rho}\equiv(\lambda_{g}\cdot\lambda_{\rho})(a_{g}\cdot a_{\rho})\equiv\lambda_{g}a_{g}\cdot\lambda_{\rho}a_{\rho}\equiv\mathsf{app}(\lambda,a)_{g}\cdot\mathsf{app}(\lambda,a)_{\rho}$$

Lemma A.0.16 (Path Π types commute with substitution)**.**

$$\Pi AB\{s\}_\delta \equiv \Pi AB_{s(\delta)} \equiv \mathsf{Tm}_{A_{s(\delta)}}(B\{a \mapsto \langle s(\delta), a\rangle\}) \equiv \Pi(A\{s\})(B\{\langle s \circ \mathbf{p}, \mathbf{q}\rangle\})_\delta$$

$$\begin{aligned}
&\Pi AB\{s\}_d(f, f') \equiv \Pi AB_{s(d)}(f, f') \\
\equiv\ &\{(\varphi_\alpha \in B_{\langle s(d),\alpha\rangle}(f(a), f'(a')))_{\alpha \in A_{s(d)}(a,a')}\} \\
\equiv\ &A\{s\}_d(a,a') \equiv A_{s(d)}(a,a') \text{ and } B\langle s\circ\mathbf{p},\mathbf{q}\rangle_{\langle d,\alpha\rangle}(b,b') \equiv B_{\langle s\circ\mathbf{p},\mathbf{q}\rangle(\langle d,\alpha\rangle)}(b,b') \equiv B_{\langle s(d),\alpha\rangle}(b,b') \\
&\{(\varphi_\alpha \in B\{\langle s \circ \mathbf{p}, \mathbf{q}\rangle\}_{\langle d,\alpha\rangle}(f(a), f'(a')))_{\alpha \in A\{s\}_d(a,a')}\} \\
\equiv\ &\Pi(As)(B\langle s \circ \mathbf{p}, \mathbf{q}\rangle)_d(f, f')
\end{aligned}$$

$$\begin{aligned}
&(\varphi \cdot \varphi')_\alpha \equiv \varphi_{\alpha_g} \cdot \varphi_{\alpha_{g'}} \equiv (\varphi \cdot \varphi')_\alpha \\
&\mathsf{id}_{f\,\alpha} \equiv f_\alpha \equiv \mathsf{id}_{f\,\alpha} \\
&\varphi^{-1}{}_\alpha \equiv \varphi_{\alpha^{-1}}{}^{-1} \equiv \varphi^{-1}{}_\alpha
\end{aligned}$$

Lemma A.0.17 (Path λ terms commute with substitution)**.**

$$\begin{aligned}
&((\lambda b)s)_\delta \\
\equiv&\text{Definition of substitution} \\
&(\lambda b)_{s_\delta} \\
\equiv&\text{Definition of } \lambda \text{ abstraction} \\
&b[\cdot]_{A_{s_\delta}} \\
\equiv&\langle s\mathbf{p}, \mathbf{q}\rangle[\cdot]_{(As)_\delta} \equiv \langle s\mathbf{p}, \mathbf{q}\rangle\langle\Delta(\delta),\rangle \equiv [\cdot]_{A_{s_\delta}} \text{ and functoriality of substitution} \\
&(b\langle s\mathbf{p}, \mathbf{q}\rangle)[\cdot]_{A_\delta} \\
\equiv&\text{Definition of } \lambda \text{ abstraction} \\
&\lambda(b\langle s\mathbf{p}, \mathbf{q}\rangle)_\delta
\end{aligned}$$

$$\begin{aligned}
&((\lambda b)s)_\rho \\
\equiv&\text{Definition of substitution} \\
&(\lambda b)_{s_\rho} \\
\equiv&\text{Definition of } \lambda \text{ abstraction} \\
&b[\cdot]_{A_{s_\rho}} \\
\equiv&[\cdot]_{A_{s_\delta}} \equiv \langle s, \mathbf{q}\rangle[\cdot]_{(As)_\delta} \text{ (!)} \\
&(b\langle s, \mathbf{q}\rangle)[\cdot]_{(As)_\rho} \\
\equiv&\text{Definition of } \lambda \text{ abstraction} \\
&\lambda(b\langle s, \mathbf{p}\rangle)\rho
\end{aligned}$$

Lemma A.0.18 (Path $\mathsf{app}(\cdot,\cdot)$ terms commute with substitution)**.**

$$
\begin{aligned}
&((\mathsf{app}(f,a))s)_\gamma \\
\equiv&\text{Definition of substitution and application} \\
&(f_{s_\gamma})_{a_{s_\gamma}} \\
\equiv&\text{Definition of substitution} \\
&((fs)_\gamma)_{(a\,s)_\gamma} \\
\equiv&\text{Definition of application} \\
&(\mathsf{app}(fs,as))_\gamma
\end{aligned}
$$

$$
\begin{aligned}
&((\mathsf{app}(f,a))s)_g \\
\equiv&\text{Definition of substitution} \\
&\mathsf{app}(f,a)_{s_g} \\
\equiv&\text{Definition of application} \\
&f_{s_g}a_{s_g} \\
\equiv&\text{Definition of substitution} \\
&(fs)_g(as)_g \\
\equiv&\text{Definition of application} \\
&(\mathsf{app}(fs,as))_g
\end{aligned}
$$

Path interpretation supports Id types

Lemma A.0.19 (Path Id types commute with substitution)**.** For every substitution $s : \Delta \to \Gamma$, the types $\mathsf{Id}_A\{\langle\langle s \circ \mathbf{p}, \mathbf{q}\rangle \circ \mathbf{p}, \mathbf{q}\rangle\}$ and $\mathsf{Id}_{A\{s\}}$ are the same.

Proof.

$$
\begin{aligned}
&\mathsf{Id}_A\{\langle\langle s \circ \mathbf{p}, \mathbf{q}\rangle \circ \mathbf{p}, \mathbf{q}\rangle\}_{\langle\delta,a,a'\rangle} \\
\equiv&\mathsf{Id}_{A_{\langle s(\delta),a,a'\rangle}} \\
\equiv&A_{\mathsf{id}_{s(\delta)}}(a,a') \\
\equiv&A\{s\}_{\mathsf{id}_\delta}(a,a') \\
\equiv&\mathsf{Id}_{A\{s\}_{\langle\delta,a,a'\rangle}}
\end{aligned}
$$

and

$$
\begin{aligned}
&\mathsf{Id}_A\{\langle\langle s \circ \mathbf{p}, \mathbf{q}\rangle \circ \mathbf{p}, \mathbf{q}\rangle\}_{\langle d,\alpha,\alpha'\rangle}(\alpha_\delta,\alpha_{\delta'}) \\
\equiv&\mathsf{Id}_{A_{\langle s(d),\alpha,\alpha'\rangle}}(\alpha_\delta,\alpha_{\delta'}) \\
\equiv&\{\bullet | \alpha_\delta \cdot \alpha \equiv \alpha' \cdot \alpha_{\delta'}\} \\
\equiv&\mathsf{Id}_{A\{s\}_{\langle d,a,a'\rangle}}(\alpha_\delta,\alpha_{\delta'})
\end{aligned}
$$

because $A_{s(\delta)} \equiv A\{s\}_\delta$ and $A_{s(d)} \equiv A\{s\}_d$ by definition of substitution.
The equalities for transport, concatenation, inverses and identities are also straightforward. □

Lemma A.0.20 (Path reflexivity proofs commute with substitution)**.** For every substitution $s : \Delta \to \Gamma$, the terms $\mathsf{refl}_A\{s\}$ and $\mathsf{refl}_{A\{s\}}$ are the same.

Proof.

$$\mathsf{refl}_A\{s\}_{\langle \delta, a\rangle} \equiv \mathsf{id}_a \equiv \mathsf{refl}_{A\{s\}_{\langle \delta, a\rangle}}$$
$$\mathsf{refl}_A\{s\}_{\langle d, \alpha\rangle} \equiv \bullet \equiv \mathsf{refl}_{A\{s\}_{\langle d, \alpha\rangle}}$$

in $A_{\mathsf{id}_{s(\delta)}} \equiv A\{s\}_{\mathsf{id}_\delta}$ and $A_{s(d)}(a, a') \equiv A\{s\}_d(a, a')$, respectively. □

Lemma A.0.21 (Path equality elimination commutes with substitution)**.** For every substitution $s : \Delta \to \Gamma$, the terms $J_{C,d}\{\langle\langle\langle s \circ \mathbf{p}, \mathbf{q}\rangle \circ \mathbf{p}, \mathbf{q}\rangle \circ \mathbf{p}, \mathbf{q}\rangle\}$ and $J_{C\langle\langle\langle s\circ\mathbf{p},\mathbf{q}\rangle\circ\mathbf{p},\mathbf{q}\rangle\circ\mathbf{p},\mathbf{q}\rangle, d\{\langle s\circ\mathbf{p},\mathbf{q}\rangle\}}$ are the same.

Proof. This is straightforward because

$$C\langle\langle\langle s \circ \mathbf{p}, \mathbf{q}\rangle \circ \mathbf{p}, \mathbf{q}\rangle \circ \mathbf{p}, \mathbf{q}\rangle_{\langle \delta, a, a', \alpha\rangle \equiv C_{\langle s(\delta), a, a', \alpha\rangle}} \quad \text{and} \quad C\langle\langle\langle s \circ \mathbf{p}, \mathbf{q}\rangle \circ \mathbf{p}, \mathbf{q}\rangle \circ \mathbf{p}, \mathbf{q}\rangle_{\langle d, \alpha, \alpha', \bullet\rangle \equiv C_{\langle s(d), \alpha, \alpha', \bullet\rangle}}$$

and

$$d\{\langle s \circ \mathbf{p}, \mathbf{q}\rangle\}_{\langle \delta, a\rangle} \equiv d_{s(\delta), a} \quad \text{and} \quad d\{\langle s \circ \mathbf{p}, \mathbf{q}\rangle\}_{\langle d, \alpha\rangle} \equiv d_{s(d), \alpha}$$

by definition of substitution. □

Path interpretation supports a U type

Lemma A.0.22 (Small type elimination commutes with substitution)**.** For every substitution $s : \Delta \to \Gamma$ and semantic term $M \in \mathsf{Tm}_\Gamma(U)$ we have $El(M\{s\}) \equiv El(M)\{s\}$.

Proof. By definition of term and type substitution as well as structure preservation for context morphisms.

$$\begin{aligned}
El(M\{s\})_\gamma &\equiv M\{s\}_\gamma \equiv M_{s(\gamma)} \equiv El(M)_{s(\gamma)} \equiv El(M)\{s\}_\gamma \\
El(M\{s\})_g &\equiv M\{s\}_g \equiv M_{s(g)} \equiv El(M)_{s(g)} \equiv El(M)\{g\}_\gamma \\
M\{s\}_{\mathsf{id}_\gamma} &\equiv M_{s(\mathsf{id}_\gamma)} \equiv M_{\mathsf{id}_{s(\gamma)}} \\
M\{s\}_{g^{-1}} &\equiv M_{s(g^{-1})} \equiv M_{s(g)^{-1}} \\
M\{s\}_{g\cdot g'} &\equiv M_{s(g\cdot g')} \equiv M_{s(g)\cdot s(g')}
\end{aligned}$$

□

Path interpretation supports $\|\cdot\|$ types

Lemma A.0.23 (Path $\|\cdot\|$ types commute with substitution)**.** For every context morphism $s : \Delta \to \Gamma$ the types $\|A\|\{s\}$ and $\|A\{s\}\|$ are the same.

Proof. The sets $\|A\|\{s\}_\gamma$ and $\|A\{s\}\|_\gamma$ are both $A_{s(\gamma)}$ and the relations are the same because two paths are the same if they connect the same endpoints.

The transport, concatenation, inverses, identities parts are straightforward as well. □

Lemma A.0.24 (Path $\|\cdot\|$ terms commute with substitution)**.** For every context morphism $s : \Delta \to \Gamma$ the terms $\|a\|\{s\}$ and $\|a\{s\}\|$ are the same.

Proof. Directly because on the object part they are the same by definition of $\|a\|$ and on the morphism parts both terms are necessarily the same by definition of $\|A\|$. □

Lemma A.0.25 (Path $\|\cdot\|$ recursion commutates with substitution)**.** For every context morphism $s : \Delta \to \Gamma$ the terms $\|f\|\{s\}$ and $\|f\{s\}\|$ are the same.

Proof. Similar to the reasoning for $\|a\|\{s\} \equiv \|a\{s\}\|$ (cf. A.0.24). □